

# Microelectronics and the Personal Computer

*Rates of progress in microelectronics suggest that in about a decade many people will possess a notebook-size computer with the capacity of a large computer of today. What might such a system do for them?*

by Alan C. Kay

The future increase in capacity and decrease in cost of microelectronic devices will not only give rise to compact and powerful hardware but also bring qualitative changes in the way human beings and computers interact. In the 1980's both adults and children will be able to have as a personal possession a computer about the size of a large notebook with the power to handle virtually all their information-related needs. Computing and storage capacity will be many times that of current microcomputers: tens of millions of basic operations per second will manipulate the equivalent of several thousand printed pages of information.

The personal computer can be regarded as the newest example of human mediums of communication. Various means of storing, retrieving and manipulating information have been in existence since human beings began to talk. External mediums serve to capture internal thoughts for communication and, through feedback processes, to form the paths that thinking follows. Although digital computers were originally designed to do arithmetic operations, their ability to simulate the details of any descriptive model means that the computer, viewed as a medium, can simulate any other medium if the methods of simulation are sufficiently well described. Moreover, unlike conventional mediums, which are passive in the sense that marks on paper, paint on canvas and television images do not change in

response to the viewer's wishes, the computer medium is active: it can respond to queries and experiments and can even engage the user in a two-way conversation.

The evolution of the personal computer has followed a path similar to that of the printed book, but in 40 years rather than 600. Like the handmade books of the Middle Ages, the massive computers built in the two decades before 1960 were scarce, expensive and available to only a few. Just as the invention of printing led to the community use of books chained in a library, the introduction of computer time-sharing in the 1960's partitioned the capacity of expensive computers in order to lower their access cost and allow community use. And just as the Industrial Revolution made possible the personal book by providing inexpensive paper and mechanized printing and binding, the microelectronic revolution of the 1970's will bring about the personal computer of the 1980's, with sufficient storage and speed to support high-level computer languages and interactive graphic displays.

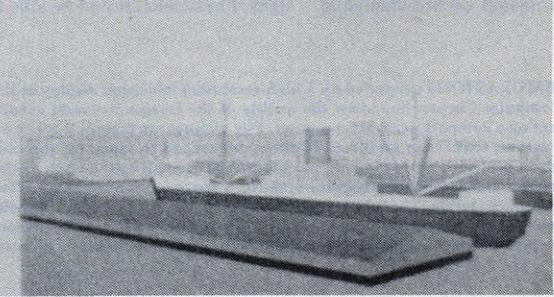
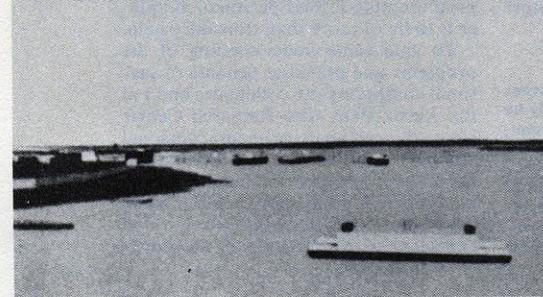
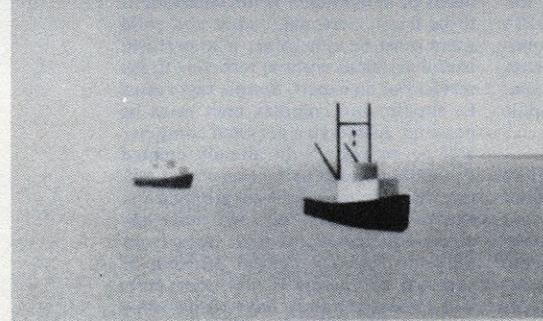
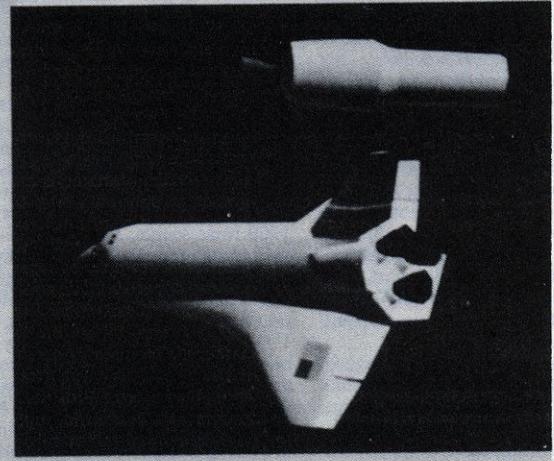
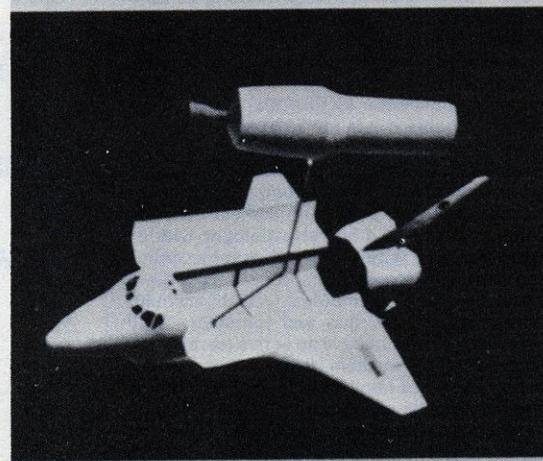
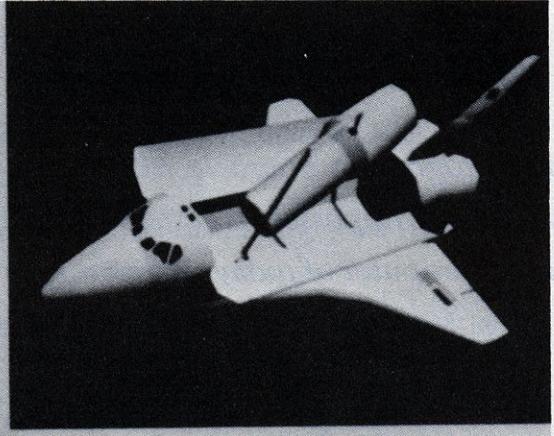
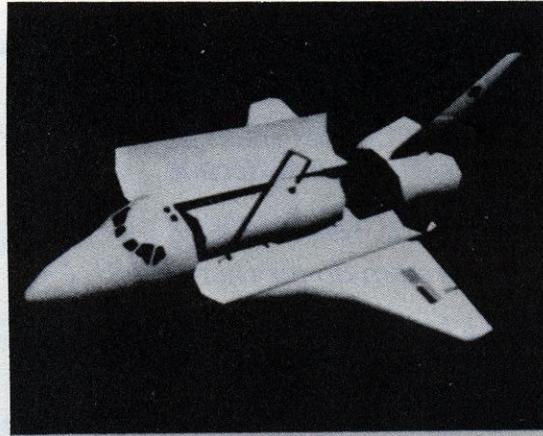
Ideally the personal computer will be designed in such a way that people of all ages and walks of life can mold and channel its power to their own needs. Architects should be able to simulate three-dimensional space in order to reflect on and modify their current designs. Physicians should be able to store

and organize a large quantity of information about their patients, enabling them to perceive significant relations that would otherwise be imperceptible. Composers should be able to hear a composition as they are composing it, notably if it is too complex for them to play. Businessmen should have an active briefcase that contains a working simulation of their company. Educators should be able to implement their own version of a Socratic dialogue with dynamic simulation and graphic animation. Homemakers should be able to store and manipulate records, accounts, budgets, recipes and reminders. Children should have an active learning tool that gives them ready access to large stores of knowledge in ways that are not possible with mediums such as books.

How can communication with computers be enriched to meet the diverse needs of individuals? If the computer is to be truly "personal," adult and child users must be able to get it to perform useful activities without resorting to the services of an expert. Simple tasks must be simple, and complex ones must be possible. Although a personal computer will be supplied with already created simulations, such as a general text editor, the wide range of backgrounds and ages of its potential users will make any direct anticipation of their needs very difficult. Thus the central problem of personal computing is that nonexperts will almost certainly have to do some programming if their personal computer is to be of more than transitory help.

To gain some understanding of the problems and potential benefits of personal computing my colleagues and I at the Xerox Palo Alto Research Center have designed an experimental personal computing system. We have had a number of these systems built and have studied how both adults and children make use of them. The hardware is faithful in capacity to the envisioned notebook-

**COMPUTER SIMULATIONS** generated on a high-resolution television display at the Evans & Sutherland Computer Corporation show the quality of the images it should eventually be possible to present on a compact personal computer. The pictures are frames from two dynamic-simulation programs that revise an image 30 times per second to represent the continuous motion of objects in projected three-dimensional space. The sequence at the top, made for the National Aeronautics and Space Administration, shows a space laboratory being lifted out of the interior of the space shuttle. The sequence at the bottom, made for the U.S. Maritime Administration, shows the movement of tankers in New York harbor. Ability of the personal computer to simulate real or imagined phenomena will make it a new medium of communication.



size computer of the 1980's, although it is necessarily larger. The software is a new interactive computer-language system called **SIMPLYTALK**.

In the design of our personal computing system we were influenced by research done in the late 1960's. At that time Edward Cheadle and I, working at the University of Utah, designed **FLEX**, the first personal computer to directly support a graphics- and simulation-oriented language. Although the **FLEX** design was encouraging, it was not comprehensive enough to be useful to a wide variety of nonexpert users. We then became interested in the efforts of Seymour A. Papert, Wallace Feurzeig and others working at the Massachusetts Institute of Technology and at Bolt, Beranek and Newman, Inc., to develop a computer-based learning environment in which children would find learning both fun and rewarding. Working with a

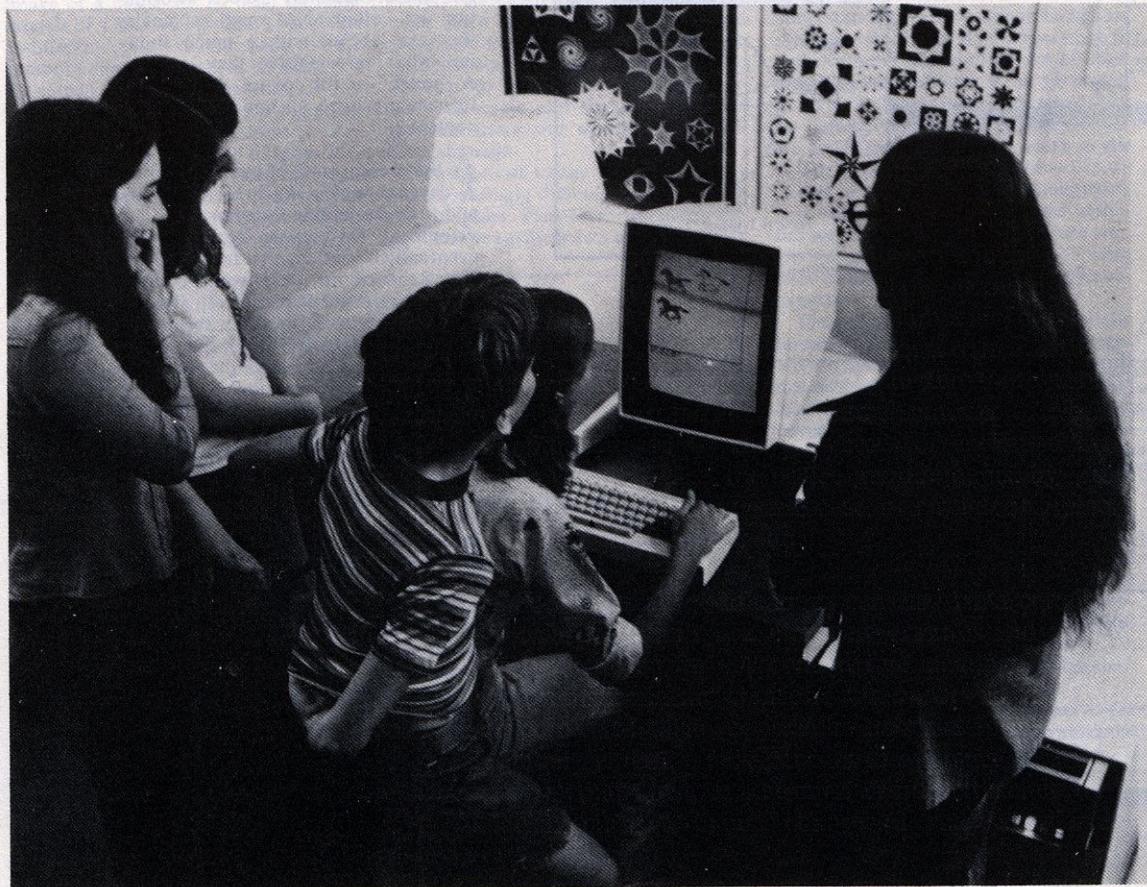
large time-shared computer, Papert and Feurzeig devised a simple but powerful computer language called **LOGO**. With this language children (ranging in age from eight to 12) could write programs to control a simple music generator, a robot turtle that could crawl around the floor and draw lines, and a television image of the turtle that could do the same things.

After observing this project we came to realize that many of the problems involved in the design of the personal computer, particularly those having to do with expressive communication, were brought strongly into focus when children down to the age of six were seriously considered as users. We also realized that children require more computer power than an adult is willing to settle for in a time-sharing system. The best outputs that time-sharing can provide are crude green-tinted line

drawings and square-wave musical tones. Children, however, are used to finger paints, color television and stereophonic records, and they usually find the things that can be accomplished with a low-capacity time-sharing system insufficiently stimulating to maintain their interest.

Since **LOGO** was not designed with all the people and uses we had in mind, we decided not to copy it but to devise a new kind of programming system that would attempt to combine simplicity and ease of access with a qualitative improvement in expert-level adult programming. In this effort we were guided, as we had been with the **FLEX** system, by the central ideas of the programming language **SIMULA**, which was developed in the mid-1960's by Ole-Johan Dahl and Kristen Nygaard at the Norwegian Computing Center in Oslo.

Our experimental personal computer



**EXPERIMENTAL PERSONAL COMPUTER** was built at the Xerox Palo Alto Research Center in part to develop a high-level programming language that would enable nonexperts to write sophisticated programs. The author and his colleagues were also interested in using the experimental computer to study the effects of personal

computing on learning. The machine is completely self-contained, consisting of a keyboard, a pointing device, a high-resolution picture display and a sound system, all connected to a small processing unit and a removable disk-file memory. Display can present thousands of characters approaching the quality of those in printed material.

is self-contained and fits comfortably into a desk. Long-term storage is provided by removable disk memories that can hold the equivalent of 1,500 printed pages of information (about three million characters). Although image displays in the 1980's will probably be flat-screened mosaics that reflect light as liquid-crystal watch displays do, visual output is best supplied today by a high-resolution black-and-white or color television picture tube. High-fidelity sound output is produced by a built-in conversion from discrete digital signals to continuous waveforms, which are then sent to a conventional audio amplifier and speakers. The user makes his primary input through a typewriterlike keyboard and a pointing device called a

mouse, which controls the position of an arrow on the screen as it is pushed about on the table beside the display. Other input systems include an organ-like keyboard for playing music, a pen-like pointer, a joystick, a microphone and a television camera.

The commonest activity on our personal computer is the manipulation of simulations already supplied by the SMALLTALK system or created by the user. The dynamic state of a simulation is shown on the display, and its general course is modified as the user changes the displayed images by typing commands on the keyboard or pointing with the mouse. For example, formatted textual documents with multiple typefaces are simulated so that an image of the

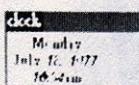
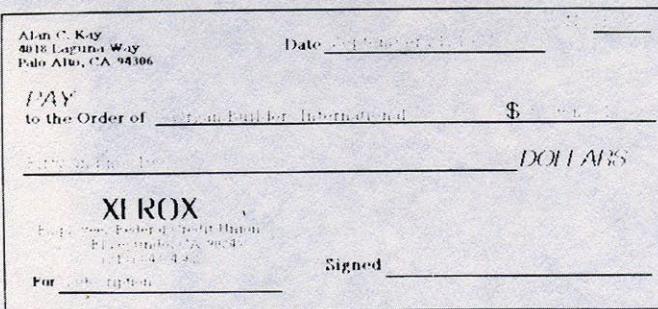
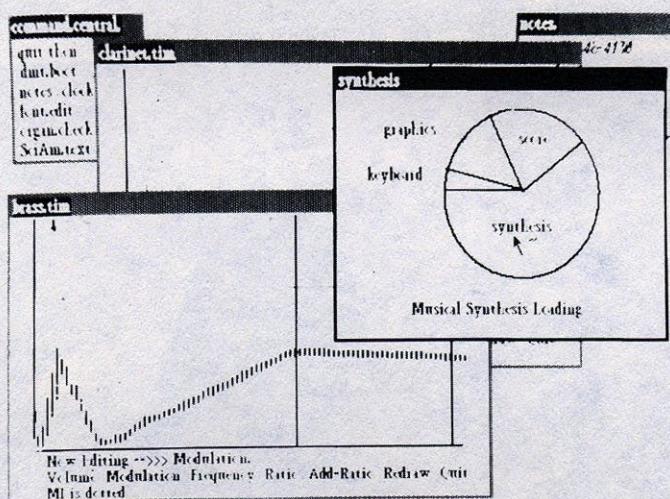
finished document is shown on the screen. The document is edited by pointing at characters and paragraphs with the mouse and then deleting, adding and restructuring the document's parts. Each change is instantly reflected in the document's image.

In many instances the display screen is too small to hold all the information a user may wish to consult at one time, and so we have developed "windows," or simulated display frames within the larger physical display. Windows organize simulations for editing and display, allowing a document composed of text, pictures, musical notation, dynamic animations and so on to be created and viewed at several levels of refinement. Once the windows have been created they overlap on the screen like sheets of paper; when the mouse is pointed at a partially covered window, the window is redisplayed to overlap the other windows. Those windows containing useful but not immediately needed information are collapsed to small rectangles that are labeled with a name showing what information they contain. A "touch" of the mouse causes them to instantly open up and display their contents.

In the present state of the art software development is much more difficult and time-consuming than hardware development. The personal computer will eventually be put together from more or less standard microelectronic components, but the software that will give life to the user's ideas must go through a long and arduous process of refinement if it is to aid and not hinder the goals of a personal dynamic medium.

For this reason we have over the past four years invited some 250 children (aged six to 15) and 50 adults to try versions of SMALLTALK and to suggest ways of improving it. Their creations, as imaginative and diverse as they themselves, include programs for home accounts, information storage and retrieval, teaching, drawing, painting, music synthesis, writing and games. Subsequent designs of SMALLTALK have been greatly influenced and improved by our visitors' projects.

When children or adults first encounter a personal computer, most of them are already involved in pursuits of their own choosing. Their initial impulse is to exploit the system to do things they are already doing: a home or office manager will automate paperwork and accounts, a teacher will portray dynamic and pictorial aspects of a curriculum, a child will work on ways to create pictures and games. The fact is that people naturally start to conceive and build personal tools. Although man has been characterized as the toolmaking species, toolmaking itself has historically been the



"WINDOWS," display frames within the larger display screen, enable the user to organize and edit information at several levels of refinement. Once the windows are created they overlap on the screen like sheets of paper. When a partially covered window is selected with the pointing device, the window is redisplayed to overlap the other windows. Images with various degrees of symbolic content can be displayed simultaneously. Such images include detailed halftone drawings, analogical images such as graphs and symbolic images such as numbers or words.

province of technological specialists. One reason is that technologies frequently require special techniques, materials, tools and physical conditions. An important property of computers, however, is that very general tools for using them can be built by anyone. These tools are made from the same materials and with the same effort as more specific creations.

Initially the children interact with our computer by "painting" pictures and drawing straight lines on the display screen with the pencillike pointer. The children then discover that programs can create structures more complex than any they can create by hand. They learn that a picture has several representations, of which only the most obvious—the image—appears on the screen. The most important representation is the editable symbolic model of the picture stored in the memory of the computer. For example, in the computer an image of a truck can be built up from models of wheels, a cab and a bed, each a different color. As the parts of the symbolic model are edited its image on the screen will change accordingly.

Adults also learn to exploit the properties of the computer medium. A professional artist who visited us spent several months building various tools that resembled those he had worked with to create images on paper. Eventually he discovered that the mosaic screen—the

indelible but instantly erasable storage of the medium—and his new ability to program could be combined to create rich textures of a kind that could not be created with ink or paint. From the use of the computer for the impoverished simulation of an already existing medium he had progressed to the discovery of the computer's unique properties for human expression.

**O**ne of the best ways to teach nonexperts to communicate with computers is to have them explore the levels of abstraction at which images can be manipulated. The manipulation of images follows the general stages of intellectual growth. For a young child an image is something to make: a free mixture of forms and colors unconnected with the real world. Older children create images that directly represent concepts such as people, pets and houses. Later analogical images appear whose form is closely related to their meaning and purpose, such as geometric figures and graphs. In the end symbolic images are used that stand for concepts that are too abstract to analogize, such as numbers, algebraic and logical terms and the characters and words that constitute language.

The types of image in this hierarchy are increasingly difficult to represent on the computer. Free-form and literal images can be easily drawn or painted with

lines and halftones in the dot matrix of the display screen with the aid of the mouse or in conjunction with programs that draw curves, fill in areas with tone or show perspectives of three-dimensional models. Analogical images can also be generated, such as a model of a simulated musical instrument: a time-sequenced graph representing the dynamic evolution of amplitude, pitch variation and tonal range.

Symbolic representations are particularly useful because they provide a means of handling concepts that are difficult to portray directly, such as generalizations and abstract relations. Moreover, as an image gets increasingly complex its most important property, the property of making local relations instantly clear, becomes less useful. Communication with computers based on symbols as they routinely occur in natural language, however, has proved to be far more difficult than many had supposed. The reason lies in our lack of understanding of how human beings exploit the context of their experience to make sense of the ambiguities of common discourse. Since it is not yet understood how human beings do what they do, getting computers to engage in similar activities is still many years in the future. It is quite possible, however, to invent artificial computer languages that can represent concepts and activities we do understand and that are simple enough in basic structure for them to be easily learned and utilized by non-experts.

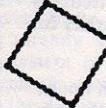
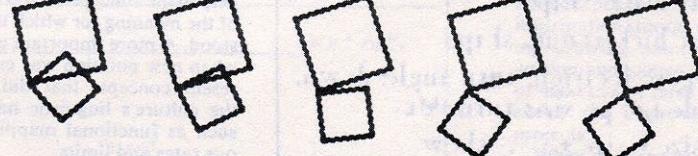
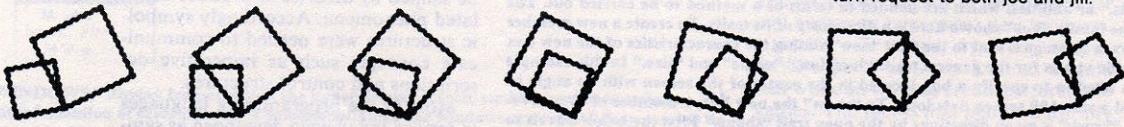
The particular structure of a symbolic language is important because it provides a context in which some concepts are easier to think about and express than others. For example, mathematical notation first arose to abbreviate concepts that could be expressed only as ungainly circumlocutions in natural language. Gradually it was realized that the form of an expression could be of great help in the conception and manipulation of the meaning for which the expression stood. A more important advance came when new notation was created to represent concepts that did not fit into the culture's linguistic heritage at all, such as functional mappings, continuous rates and limits.

The computer created new needs for language by inverting the traditional process of scientific investigation. It made new universes available that could be shaped by theories to produce simulated phenomena. Accordingly symbolic structures were needed to communicate concepts such as imperative descriptions and control structures.

Most of the programming languages in service today were developed as symbolic ways to deal with the hardware-level concepts of the 1950's. This approach led to two kinds of passive building blocks: data structures, or inert con-

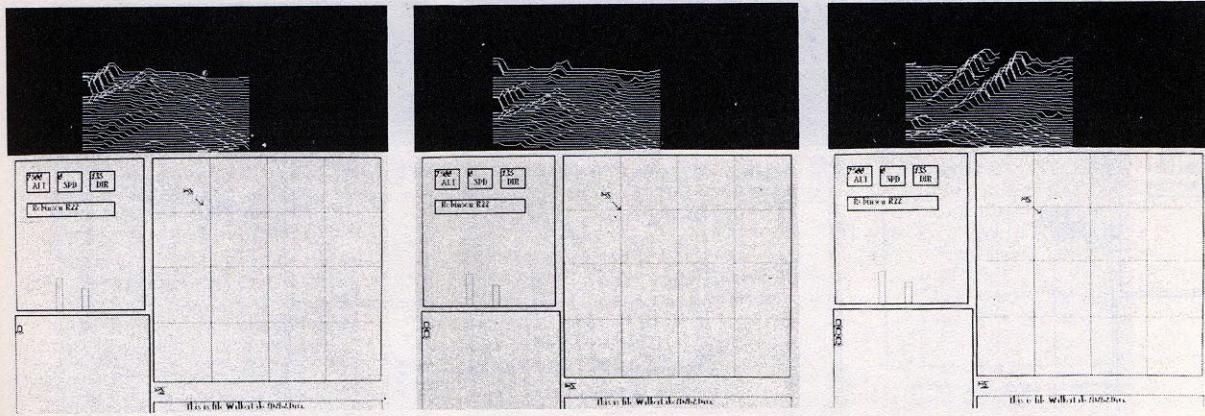
trait name	description
name	box; picture; activity
location	<input type="checkbox"/>
angle	<input type="checkbox"/>
size	<input type="checkbox"/>
new	location ← center; angle ← 0; size ← 100;
skew	& print black shape
size	& print background shape
shape	& use gete location; turn angle down 1 to & do (& go size; turn 90)
grow	size ← size + 1; skew

**SMALLTALK** is a new programming language developed at the Xerox Palo Alto Research Center for use on the experimental personal computer. It is made up of "activities," computer-like entities that can perform a specific set of tasks and can also communicate with other activities in the system. New activities are created by enriching existing families of activities with additional "traits," or abilities, which are defined in terms of a method to be carried out. The description of the family "box" shown here is a dictionary of its traits. To create a new member of the family box, a message is sent to the trait "new" stating the characteristics of the new box in terms of specific values for the general traits "location," "angle" and "size." In this example "new" has been filled in to specify a box located in the center of the screen with an angle of zero degrees and a side 100 screen dots long. To "show" the new box, a member of the curve-drawing family "brush" is given directions by the open trait "shape." First the brush travels to the specified location, turns in the proper direction and appears on the screen. Then it draws a square by traveling the distance given by "size," turning 90 degrees and repeating these actions three more times. The last trait on the list is open, indicating that a numerical value is to be supplied by the user when the trait is invoked by a message. A box is "grown" by first erasing it, increasing (or decreasing) its size by the value supplied in the message and redisplaying it.

Message Interaction	Pictorial Effect	Commentary
<input type="checkbox"/> box new named "joe"! box:joe		An offspring of the family "box" is created and is named "joe."
<input type="checkbox"/> joe turn 30! ok		The box joe receives the message and turns 30 degrees.
<input type="checkbox"/> joe grow -15! ok		joe becomes smaller by 15 units.
<input type="checkbox"/> joe erase! ok		joe disappears from the screen.
<input type="checkbox"/> joe show! ok		joe reappears.
<input type="checkbox"/> box new named "jill"! box:jill		A new box appears.
<input type="checkbox"/> jill turn -10! ok		Only jill turns. Joe and jill are independent activities.
<input type="checkbox"/> 1 to 10! interval:1 2 3 4 5 6 7 8 9 10		An interval stands for a sequence of numbers.
<input type="checkbox"/> forever! interval:1 2 3 4 5 6 7 8 9 10 11...		Forever is the infinite interval. It must be terminated by hitting an escape key.
<input type="checkbox"/> 1 to 10 do (joe turn,20)! ok		Joe spins.
<input type="checkbox"/> forever do (joe turn 11. jill turn -13)! ok		A simple parallel movie of joe and jill spinning in opposite directions is created by combining forever with a turn request to both joe and jill.

**SMALLTALK LEARNING SEQUENCE** teaches students the basic concepts of the language by having them interact with an already defined family of activities. First, offspring of the family box are cre-

ated, named and manipulated, and a second family of activities called "interval" is introduced. Offspring of the interval and box families are then combined to generate an animation of two spinning boxes.



**HELICOPTER SIMULATION** was developed by a 15-year-old student. The user directs the helicopter where to go with the pointing

device, which controls the position of the black arrow on the screen. The window at the top shows the changing topography of the terrain

struction materials, and procedures, or step-by-step recipes for manipulating data. The languages based on these concepts (such as BASIC, FORTRAN, ALGOL and APL) follow their descriptions in a strictly sequential manner. Because a piece of data may be changed by any procedure that can find it the programmer must be very careful to choose only those procedures that are appropriate. As ever more complex systems are attempted, requiring elaborate combinations of procedures, the difficulty of getting the entire system to work increases geometrically. Although most programmers are still taught data-procedure languages, there is now a widespread recognition of their inadequacy.

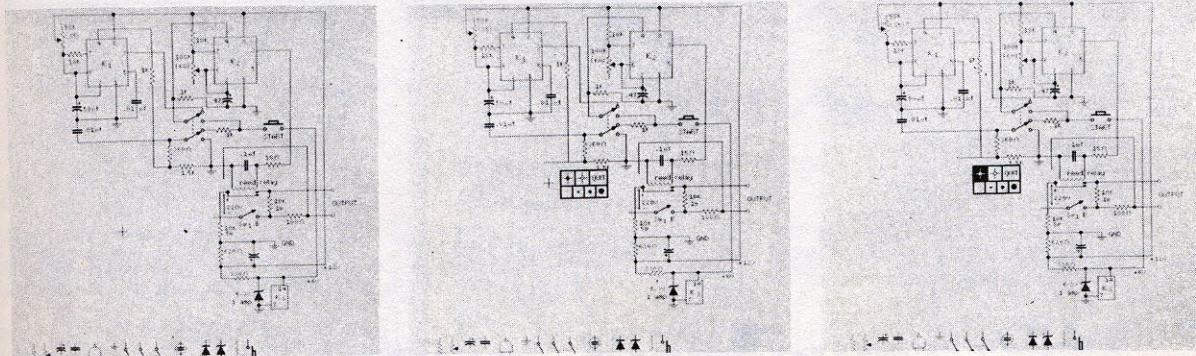
A more promising approach is to devise building blocks of greater generality. Both data and procedures can be replaced by the single idea of "activities," computerlike entities that exhibit behavior when they are sent an appropri-

ate message. There are no nouns and verbs in such a language, only dynamically communicating activities. Every transaction, description and control process is thought of as sending messages to and receiving messages from activities in the system. Moreover, each activity belongs to a family of similar activities, all of which have the ability to recognize and reply to messages directed to them and to perform specific acts such as drawing pictures, making sounds or adding numbers. New families are created by combining and enriching "traits," or properties inherited from existing families.

A message-activity system is inherently parallel: every activity is constantly ready to send and receive messages, so that the host computer is in effect divided into thousands of computers, each with the capabilities of the whole. The message-activity approach therefore enables one to dynamically represent a

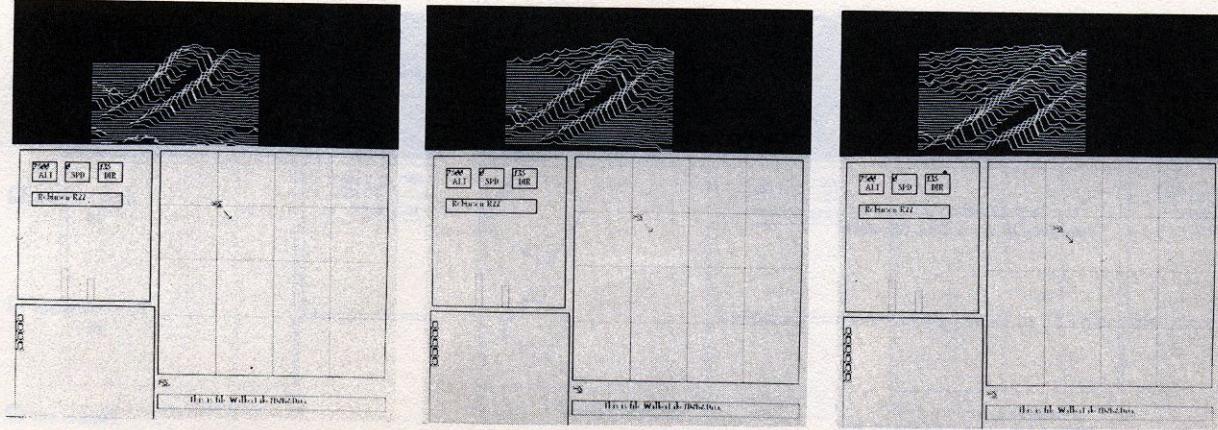
system at many levels of organization from the atomic to the macroscopic, but with a "skin" of protection at each qualitative level of detail through which negotiative messages must be sent and checked. This level of complexity can be safely handled because the language severely limits the kinds of interactions between activities, allowing only those that are appropriate, much as a hormone is allowed to interact with only a few specifically responsive target cells. SMALLTALK, the programming system of our personal computer, was the first computer language to be based entirely on the structural concepts of messages and activities.

The third and newest framework for high-level communication is the observer language. Although message-activity languages are an advance over the data-procedure framework, the relations among the various activities are somewhat independent and analytic. Many



**CIRCUIT-DRAWING PROGRAM** that was developed by a 15-year-old boy enables a user to construct a complex circuit diagram by

selecting components from a "menu" displayed at the bottom of the screen. The components are then positioned and connected with the



below as the helicopter flies over it. (Actual terrains were obtained from Landsat maps.) A third window keeps track of the helicopter's

altitude, direction and speed. The variety of events that can be simulated at the same time demonstrates the power of parallel processing.

concepts, however, are so richly interwoven that analysis causes them virtually to disappear. For example, 20th-century physics assigns equal importance to a phenomenon and its context, since observers with different vantage points perceive the world differently. In an observer language, activities are replaced by "viewpoints" that become attached to one another to form correspondences between concepts. For example, a dog can be viewed abstractly (as an animal), analytically (as being composed of organs, cells and molecules), pragmatically (as a vehicle by a child), allegorically (as a human being in a fairy tale) and contextually (as a bone's way to fertilize a lawn). Observer languages are just now being formulated. They and their successors will be the communication vehicles of the 1980's.

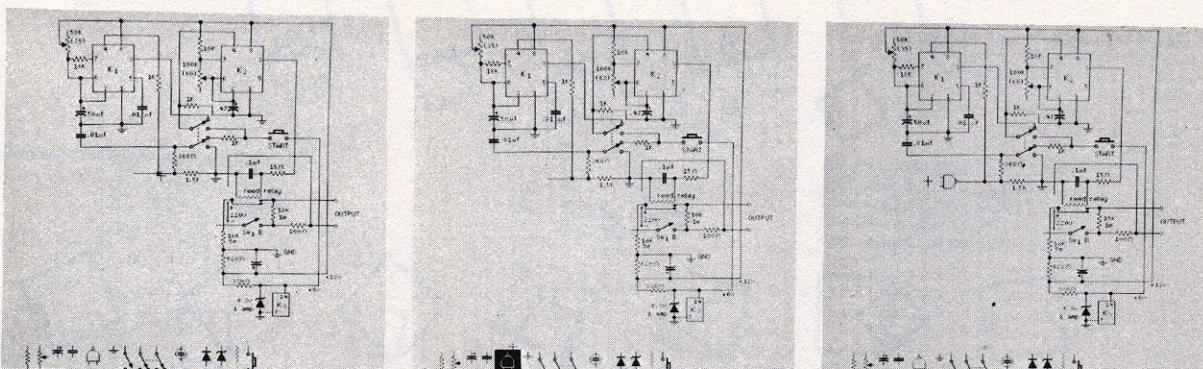
Our experience, and that of others who teach programming, is that a first computer language's particular style

and its main concepts not only have a strong influence on what a new programmer can accomplish but also leave an impression about programming and computers that can last for years. The process of learning to program a computer can impose such a particular point of view that alternative ways of perceiving and solving problems can become extremely frustrating for new programmers to learn.

At the beginning of our study we first timidly considered simulating features of data-procedure languages that children had been able to learn, such as BASIC and LOGO. Then, worried that the imprinting process would prevent stronger ideas from being absorbed, we decided to find a way to present the message-activity ideas of SMALLTALK in concrete terms without dilution. We did so by starting with simple situations that embodied a concept and then gradually increasing the complexity of the examples

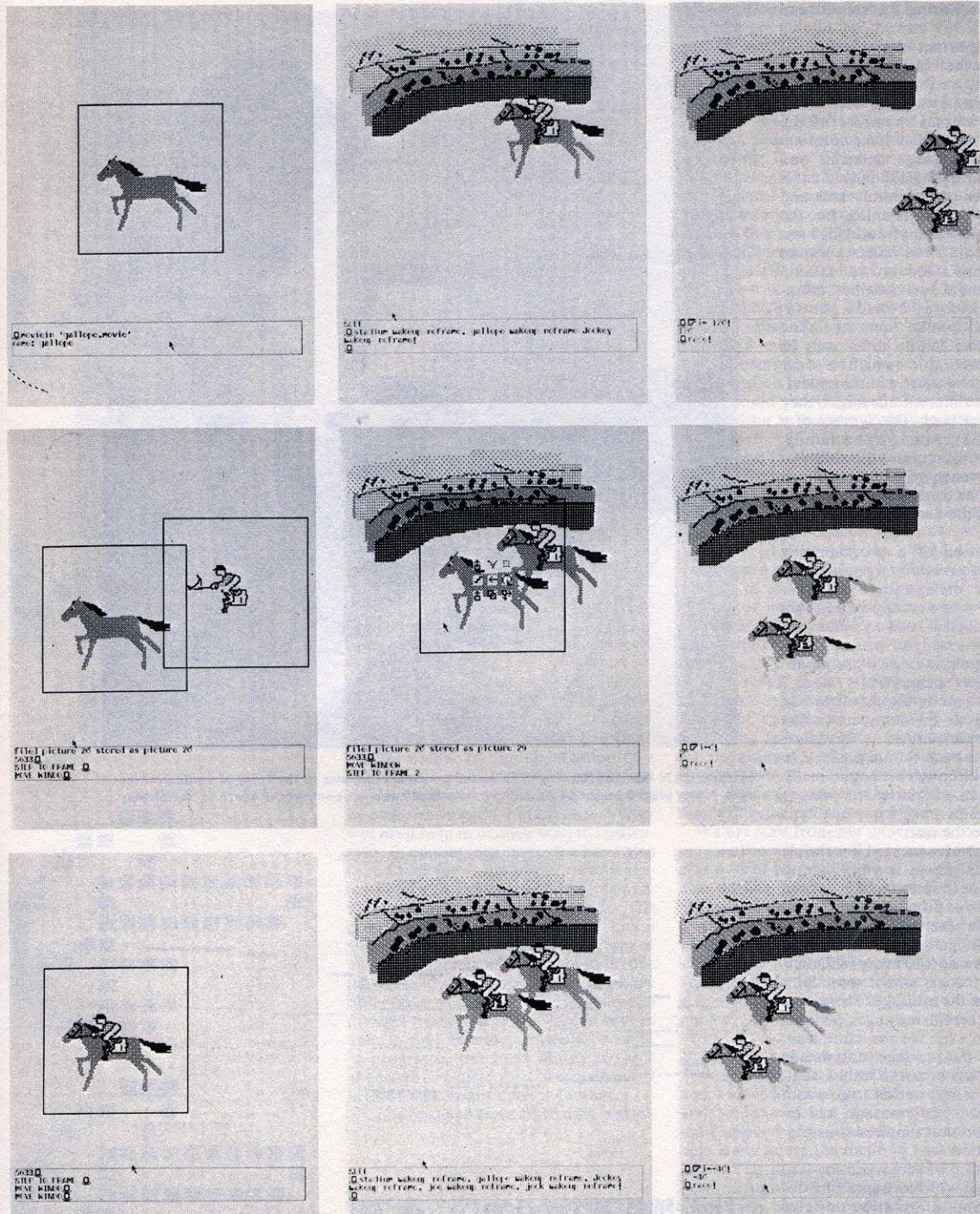
to flesh out the concept to its full generality. Although the communicationlike model of SMALLTALK is a rather abstract way to represent descriptions, to our surprise the first group and succeeding groups of children who tried it appeared to find the ideas as easy to learn as those of more concrete languages.

For example, most programming languages can deal with only one thing at a time, so that it is difficult to represent with them even such simple situations as children in a school, spacecraft in the sky or bouncing balls in free space. In SMALLTALK parallel models are dealt with from the start, and the children seem to have little difficulty in handling them. Actually parallel processing is remarkably similar to the way people think. When you are walking along a street, one part of your brain may be thinking about the route you are taking, another part may be thinking about the dinner you are going to eat, a third



pointing device. An additional menu can be generated on the screen by pushing a button on the pointing device; this menu supplies solid

and open dots and lines of various widths. In the sequence shown here two components are selected and added to a circuit diagram.



**HORSE-RACE ANIMATION** shows the capabilities of the experimental personal computer for creating dynamic halftone images. The possible range of such simulations is limited only by the versatility of the programming language and the imagination of the child or adult

user. In this sequence, images of horses, riders and background are called up independently from the storage files and arranged for the racing simulation with the pointing device. A single typed command then causes the two horses and riders to race each other across screen.

part may be admiring the sunset, and so forth.

Another important characteristic of SMALLTALK is the classification of objects into families that are generalizations of their properties. Children readily see themselves as members of the family "kids," since they have common traits such as language, interests and physical appearance. Each individual is both a member of the family kids and has his or her own meaning for the shared traits. For example, all kids have the trait eye color, but Sam's eyes are blue and Bertha's are brown. SMALLTALK is built out of such families. Number symbols, such as 2 or 17, are instances of the family "number." The members of this family differ only in their numerical value (which is their sole property) and share a common definition of the different messages they can receive and send. The symbol of a "brush" in SMALLTALK is also a family. All the brush symbols have the ability to draw lines, but each symbol has its own knowledge of its orientation and where it is located in the drawing area.

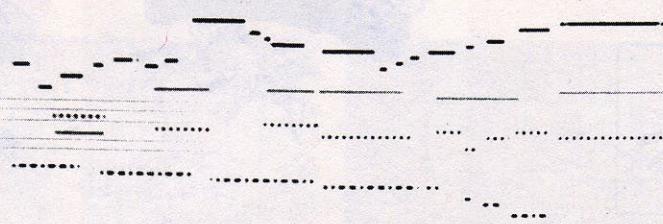
The description of a programming language is generally given in terms of its grammar: the meaning each grammatical construction is supposed to convey and the method used to obtain the meaning. For example, various programming languages employ grammatical constructions such as (PLUS 3 4) or 3 ENTER 4 + to specify the intent to add the number 3 to the number 4. The meaning of these phrases is the same. In the computer each should give rise to the number 7, although the actual methods followed in obtaining the answer can differ considerably from one type of computer to the next.

The grammar of SMALLTALK is simple and fixed. Each phrase is a message to an activity. A description of the desired activity is followed by a message that selects a trait of the activity to be performed. The designated activity will decide whether it wants to accept the message (it usually does) and at some later time will act on the message. There may be many concurrent messages pending for an activity, even for the same trait. The sender of the message may decide to wait for a reply or not to wait. Usually it waits, but it may decide to go about other business if the message has invoked a method that requires considerable computation.

The integration of programming-language concepts with concepts of editing, graphics and information retrieval makes available a wide range of useful activities that the user can invoke with little or no knowledge of programming. Learners are introduced to SMALLTALK by getting them to send messages to already existing families of activities, such

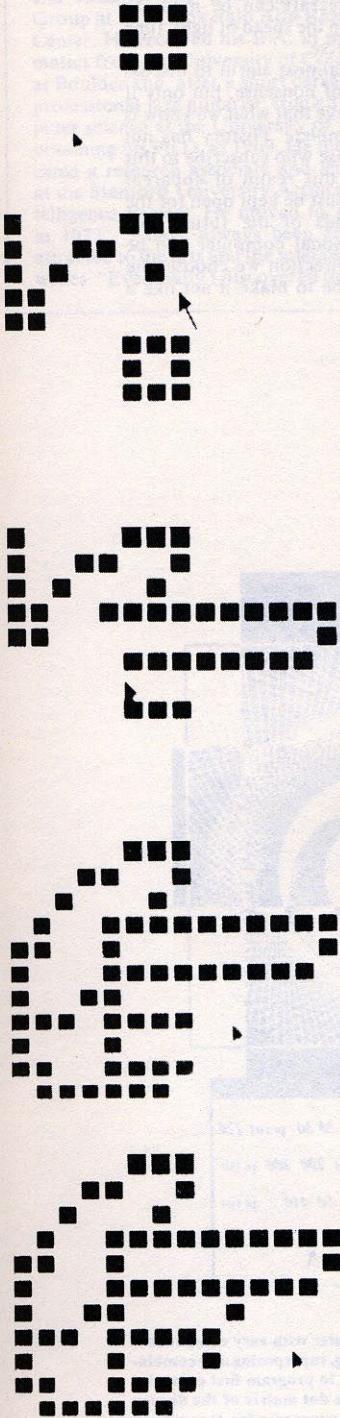


MUSIC CAN BE REPRESENTED on the personal computer in the form of analogical images. Notes played on the keyboard are "captured" as a time-sequenced score on the display.



Pitch Duration Stretch Break Sync Add  
Hear Backup Beginning Quit Copy Shift ev

MUSICAL SCORE shown here was generated as music was played on the keyboard. The simplified notation represents pitch by vertical placement and duration by horizontal length. Notes can be shortened, lengthened or changed and the modified piece then played back as music.



as the family "box," whose members show themselves on the screen as squares. A box can individually change its size, location, rotation and shape. After some experience with sending messages to cause effects on the display screen the learner may take a look at the definition of the box family. Each family in SMALLTALK is described with a dictionary of traits, which are defined in terms of a method to be carried out. For example, the message phrase "joe grow 50" says: Find the activity named "joe," find its general trait called "grow \_\_\_" and fill in its open part with the specific value 50. A new trait analogous to those already present in the family definition (such as "grow" or "turn") can easily be added by the learner. The next phase of learning involves elaboration of this basic theme by creating games such as space war and tools for drawing and painting.

**T**here are two basic approaches to personal computing. The first one, which is analogous to musical improvisation, is exploratory: effects are caused in order to see what they are like and errors are tracked down, understood and fixed. The second, which resembles musical composition, calls for a great deal more planning, generality and structure. The same language is used for both methods but the framework is quite different.

From our study we have learned the importance of a balance between free exploration and a developed curriculum. The personal computing experience is similar to the introduction of a piano into a third-grade classroom. The children will make noise and even music by experimentation, but eventually they will need help in dealing with the instrument in nonobvious ways. We have also found that for children the various levels of abstraction supplied by SMALLTALK are not equally accessible. The central idea of symbolization is to give a simple name to a complex collection of ideas, and then later to be able to invoke the ideas through the name. We have observed a number of children between the ages of six and seven who have been able to take this step in their computer programs, but their ability to look ahead, to visualize the consequences of actions they might take, is limited.

Children aged eight to 10 have a grad-

ually developing ability to visualize and plan and are able to use the concept of families and a subtler form of naming: the use of traits such as size, which can stand for different numerical values at different times. For most children, however, the real implications of further symbolic generality are not at all obvious. By age 11 or 12 we see a considerable improvement in a child's ability to plan general structures and to devise comprehensive computer tools. Adults advance through the stages more quickly than children, and usually they create tools after a few weeks of practice. It is not known whether the stages of intellectual development observed in children are absolutely or only relatively correlated with age, but it is possible that exposure to a realm in which symbolic creation is rewarded by wonderful effects could shorten the time required for children to mature from one stage to the next.

The most important limitation on personal computing for nonexperts appears when they conceive of a project that, although it is easy to do in the language, calls for design concepts they have not yet absorbed. For example, it is easy to build a span with bricks if one knows the concept of the arch, but otherwise it is difficult or impossible. Clearly as complexity increases "architecture" dominates "material." The need for ways to characterize and communicate architectural concepts in developing programs has been a long-standing problem in the design of computing systems. A programming language provides a context for developing strategies, and it must supply both the ability to make tools and a style suggesting useful approaches that will bring concepts to life.

We are sure from our experience that personal computers will become an integral part of peoples' lives in the 1980's. The editing, saving and sifting of all manner of information will be of value to virtually everyone. More sophisticated forms of computing may be like music in that most people will come to know of them and enjoy them but only a few will actually become directly involved.

**H**ow will personal computers affect society? The interaction of society and a new medium of communication and self-expression can be disturbing even when most of the society's members learn to use the medium routinely. The social and personal effects of the new medium are subtle and not easy for the society and the individual to perceive. To use writing as a metaphor, there are three reactions to the introduction of a new medium: illiteracy, literacy and artistic creation. After reading material became available the illiterate were those who were left behind by the

**DISPLAY FONTS** can be designed on personal computer by constructing them from a matrix of black-and-white squares. When the fonts are reduced, they approach the quality of those in printed material. The image of a pointing hand shown here is a symbol in SMALLTALK representing the concept of a literal word, such as the name associated with an activity.

new medium. It was inevitable that a few creative individuals would use the written word to express inner thoughts and ideas. The most profound changes were brought about in the literate. They did not necessarily become better people or better members of society, but they came to view the world in a way quite different from the way they had viewed it before, with consequences that were difficult to predict or control.

We may expect that the changes resulting from computer literacy will be as far-reaching as those that came from literacy in reading and writing, but for most people the changes will be subtle and not necessarily in the direction of their idealized expectations. For example, we should not predict or expect that the personal computer will foster a new revolution in education just because it could. Every new communication medium of this century—the telephone, the motion picture, radio and television—has elicited similar predictions that did not come to pass. Millions of uneducated people in the world have ready access to the accumulated culture of the centuries in public libraries, but they do not avail themselves of it. Once an individual or a society decides that education is essential, however, the book, and now the personal computer, can be among the society's main vehicles for the transmission of knowledge.

The social impact of simulation—the central property of computing—must also be considered. First, as with language, the computer user has a strong motivation to emphasize the similarity between simulation and experience and to ignore the great distances that symbols interpose between models and the real world. Feelings of power and a narcissistic fascination with the image of oneself reflected back from the machine are common. Additional tendencies are to employ the computer trivially (simulating what paper, paints and file cabinets can do), as a crutch (using the computer to remember things that we can perfectly well remember ourselves) or as an excuse (blaming the computer for human failings). More serious is the human propensity to place faith in and assign higher powers to an agency that is not completely understood. The fact that many organizations actually base their decisions on—worse, take their decisions from—computer models is profoundly disturbing given the current state of the computer art. Similar feelings about the written word persist to this day; if something is "in black and white," it must somehow be true.

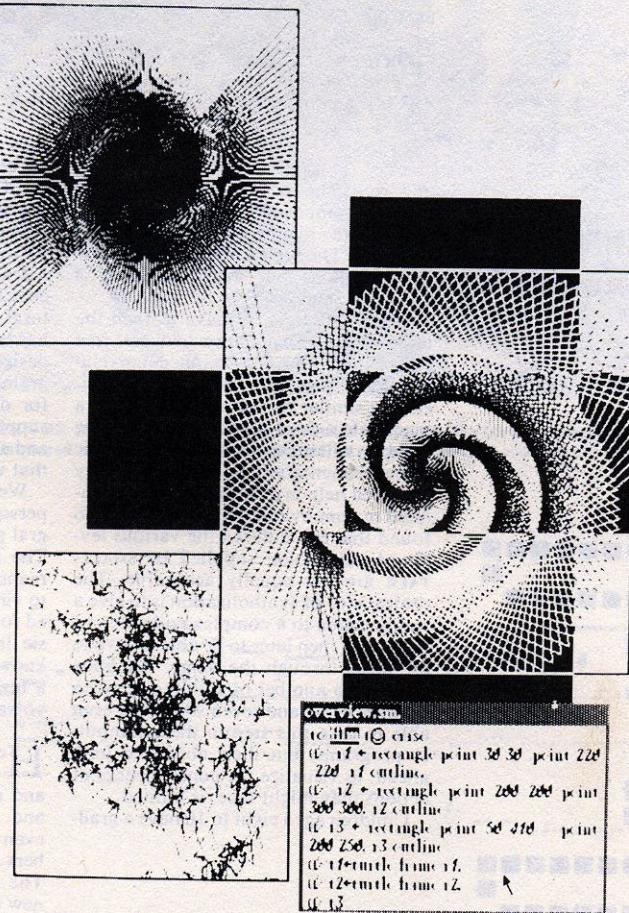
**C**hildren who have not yet lost much of their sense of wonder and fun have helped us to find an ethic about computing: Do not automate the work you are engaged in, only the materials.

If you like to draw, do not automate drawing; rather, program your personal computer to give you a new set of paints. If you like to play music, do not build a "player piano"; instead program yourself a new kind of instrument.

A popular misconception about computers is that they are logical. Forthright is a better term. Since computers can contain arbitrary descriptions, any conceivable collection of rules, consistent or not, can be carried out. Moreover, computers' use of symbols, like the use of symbols in language and mathematics, is sufficiently disconnected from the real world to enable them to create splendid nonsense. Although the hardware of the computer is subject to natural laws (electrons can move through the

circuits only in certain physically defined ways), the range of simulations the computer can perform is bounded only by the limits of human imagination. In a computer, spacecraft can be made to travel faster than the speed of light, time to travel in reverse.

It may seem almost sinful to discuss the simulation of nonsense, but only if we want to believe that what we know is correct and complete. History has not been kind to those who subscribe to this view. It is just this realm of apparent nonsense that must be kept open for the developing minds of the future. Although the personal computer can be guided in any direction we choose, the real sin would be to make it act like a machine!



**INTRICATE PATTERNS** can be generated on the personal computer with very compact descriptions in SMALLTALK. They are made by repeating, rotating, scaling, superposing and combining drawings of simple geometric shapes. Students who are learning to program first create interesting free-form or literal images by drawing them directly in the dot matrix of the display screen. Eventually they learn to employ the symbolic images in the programming language to direct the computer to generate more complex imagery than they could easily create by hand.

## The Author

ALAN C. KAY is a principal scientist and head of the Learning Research Group at the Xerox Palo Alto Research Center. He received his B.A. in mathematics from the University of Colorado at Boulder and, after a short career as a professional jazz guitarist, studied computer science at the University of Utah, obtaining his Ph.D. in 1969. He then became a research associate and lecturer at the Stanford University Artificial Intelligence Project. He moved to Xerox in 1971. "I have always been equally attracted to the arts and the sciences," he writes. "Eventually I discovered that the

world of computers provides a satisfying environment for my blend of interests."

## Bibliography

- TOWARDS A THEORY OF INSTRUCTION. Jerome S. Bruner. Belknap Press of Harvard University Press, 1966.
- ARTIFICIAL INTELLIGENCE. Seymour A. Papert and Marvin Minsky. Condon Lectures, Oregon State System of Higher Education, 1974.
- PERSONAL DYNAMIC MEDIA. Alan C. Kay and Adele Goldberg in *Computer*, Vol. 10, No. 3, pages 31-41; March, 1977.