Programming Your Own Computer

Alan Kay

Alan Kay's article for the
1979 World Book Science Annual

VPRI Paper for Historical Context

Viewpoints Research Institute, 1025 Westwood Blvd 2nd flr, Los Angeles, CA 90024 t: (310) 208-0524

A Review of Science and Technology During the 1978 School Year

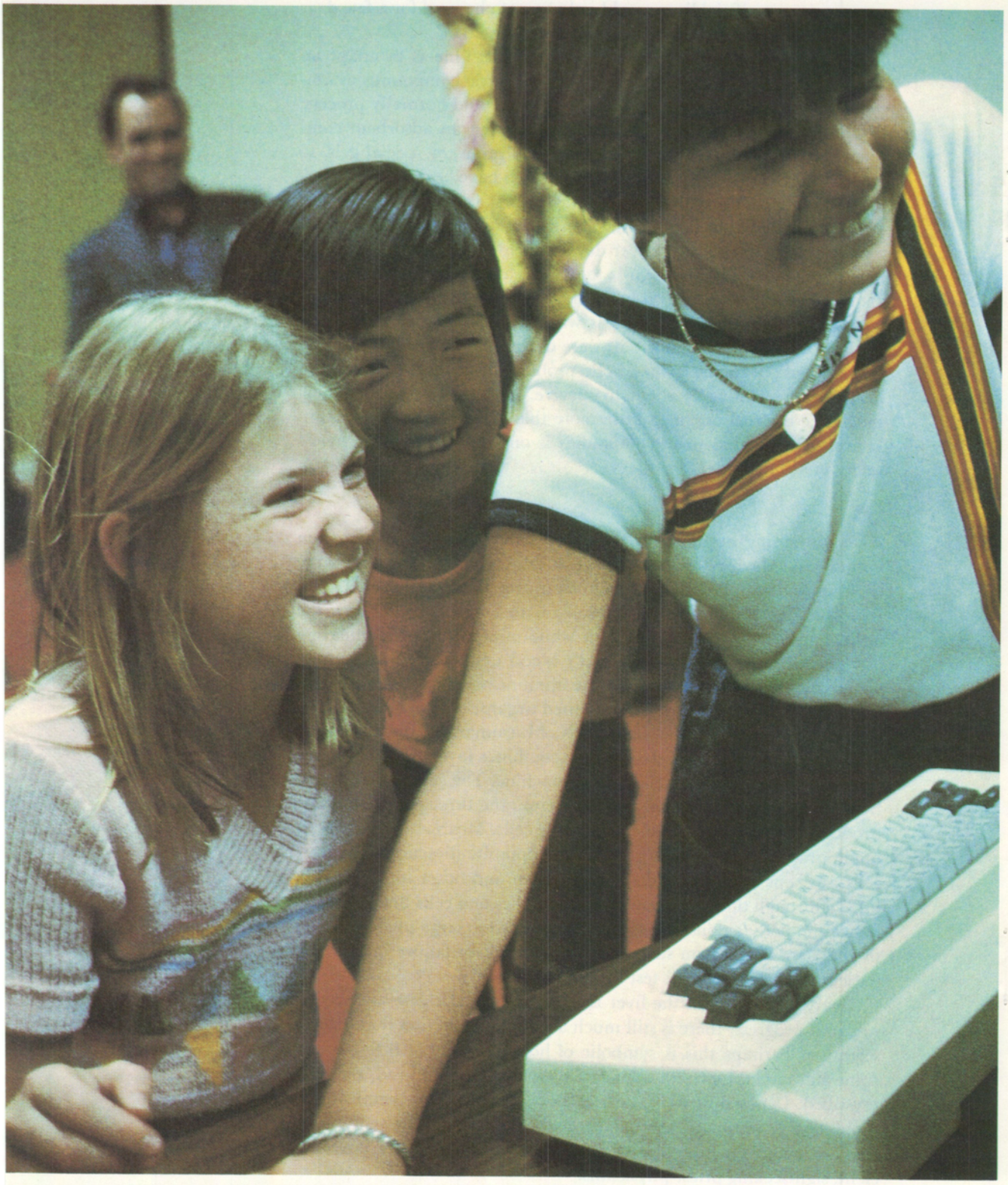
Science Year

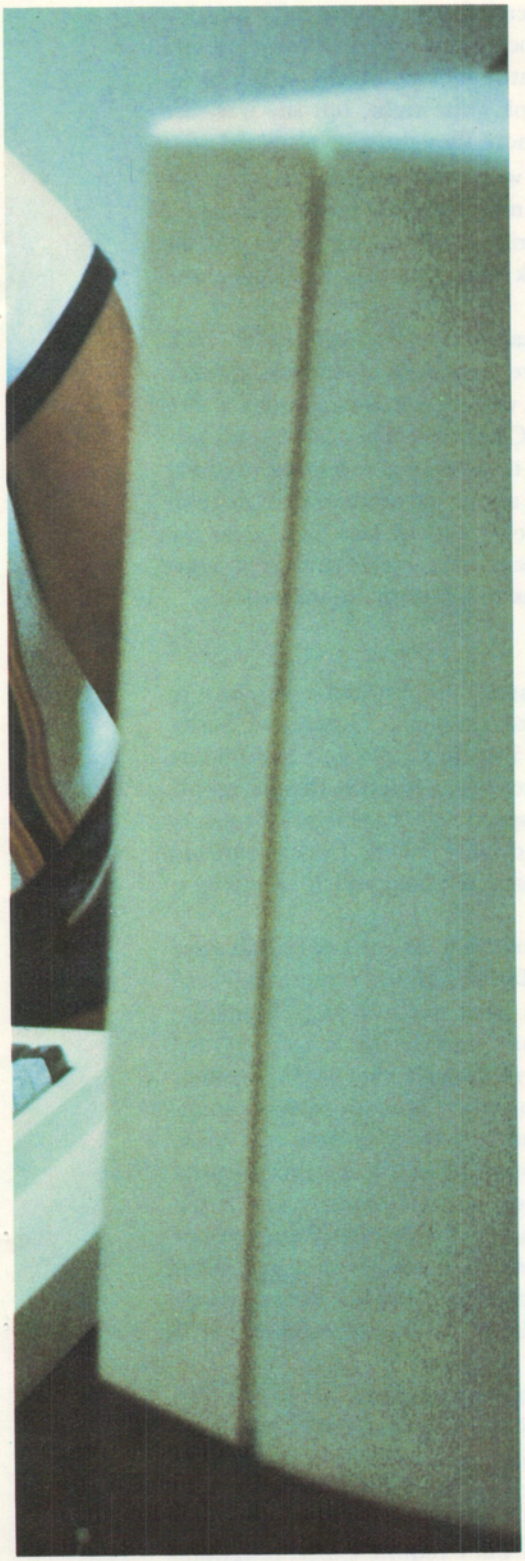
The World Book Science Annual

1979

World Book–Childcraft International, Inc.

Chicago London Paris Rome Sydney Tokyo Toronto





Programming Your Own Computer

By Alan C. Kay

The computers of the 1980s will be easy to use and inexpensive, yet powerful enough to let us do things we have yet to imagine

Beth, a 14-year-old junior high student, lounged in the grass under a tree, enjoying the sunshine. She was watching pictures move about on the televisionlike display screen of a curious gadget she held in her lap. Her mother called from the house, "Beth, time to do your homework."

"I'm doing it, mother," Beth answered.

She began striking keys on the typewriterlike keyboard below the display screen. Part of a composition about space travel that she had begun the previous day appeared on the screen, along with a half-finished illustration and some references.

An awkward sentence caught her eye. She pointed a pencillike stylus at the sentence, typed a revision, and watched the sentence change to her new phrasing. Then she finished the illustration with a drawing tool, filling in tone and drawing straight lines. When she made a mistake, she typed out instructions to undo what she had done, and then tried again.

Another of Beth's illustrations was a shuttle docking to a space station. She decided to animate it

and began to grapple with the problem of how a spacecraft's position can be controlled if the simulated rocket can only be turned on and off. Soon, she had an image of a shuttle moving on the screen in response to her controls. It did not look quite right, but she was sure that she could correct it with a bit more thought.

When Beth was satisfied with her work, she filed the composition away by pressing a key and turned to more personal pursuits. For fun, she had been working on an elaborate horse-racing animation for some time. It was now time to think about making betting on the horses a part of the game.

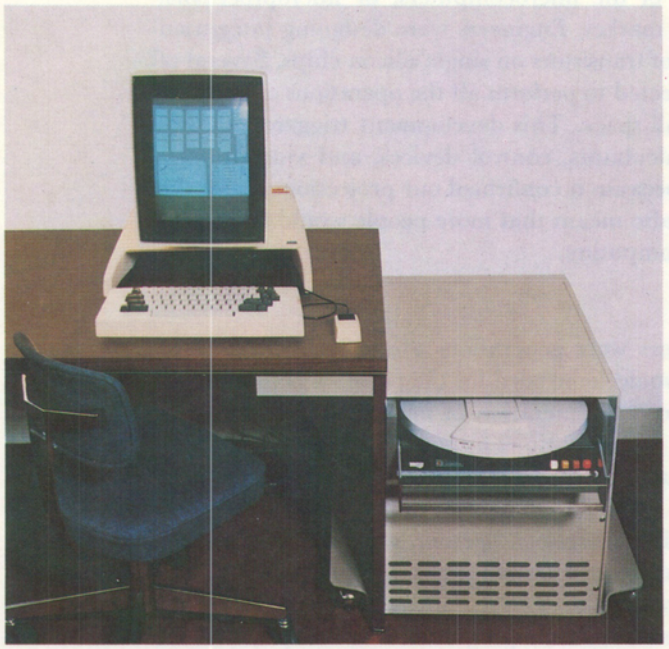
Beth was working with her personal computer. Although she is a fictitious teen-ager living in the future, hundreds of students like her have already had similar experiences with personal computers at the Xerox Palo Alto Research Center in California. They have been part of a 10-year program, begun in 1971, to develop and test the personal computers of the 1980s. By mid-1978, about 100 adults and more than 300 young people between the ages of 6 and 15 had learned to use these computers to write and illustrate reports, create drawings, compose music, and bring their own ideas to life through animation.

Each of them has learned how to instruct, or program, computers to create new tools and new effects. For example, 12-year-old Susan designed a drawing tool that would help her form and manipulate geometric shapes and textures on the television display. Dennis, an 11-year-old, created his own version of a space-war game in which spaceship fleets he designed battled on the display screen. Twelve-year-old Joan adapted a tool built by professional animators to produce a beautifully animated horse race.

My interest in personal computers follows a dream I have had since I was a graduate student in the 1960s. Thanks to some far-sighted professors, I realized that computers were not just huge, clanking, million-dollar arithmetic gadgets that had to be constantly fed punched cards by highly trained experts. Instead, they had the potential to be a fantastic tool with which ordinary humans could communicate. Moreover, the room-sized equipment of that time was being replaced by much smaller and faster models. In a few years, it would be possible to package the computer power–represented by the amount of information per unit time that it can handle–of the million-dollar machines of the 1960s in inexpensive containers as small as an ordinary three-ring notebook. The technology that has already made possible devices such as pocket calculators or digital watches would bring forth the personal computer.

This meant that everyone could soon have a personal computer. But what could they do with it? It is not enough to have a machine with all its circuits, storage, and memory facilities–the hardware–compacted into a small but mighty computer. We also need a way to communicate with and control it–the software. In present computers, software systems are unwieldy, to say the least. They require expert program-

The author:
Alan C. Kay is principal scientist and head of the Learning Research Group at the Xerox Palo Alto Research Center.



The ENIAC, a 1946 vintage computer, *top left,* has given way to increasingly smaller and more powerful machines such as the PDP 11/45 text and picture layout minicomputer, *top right,* and the Alto, *above left,* used to develop programs for personal computers. Such power will soon be contained in the notebook-sized Dynabook, *above.*

mers to bludgeon them into more-or-less working order. Obviously, personal computing in the 1980s would be impossible if softwares were not drastically improved.

In 1971, my company provided the kind of long-term research support that the problem demanded, and a group of us interested in personal computers began to work on them. We visualized the personal computer of the 1980s as a notebook-sized package whose front side was a flat-screen reflective display, like a liquid-crystal watch face. The surface of the screen would be sensitive to the touch of a finger or a stylus. It would contain miniature computer circuits that could carry out millions of instructions per second, and there would be enough storage capacity to retain several months worth of projects. The machine would be completely self-contained. However, it also could be connected to hi-fi sets to create music, to other personal

computers for group projects, and to external information sources such as the computerized libraries of the future. We called this visionary computer "Dynabook."

To work on the software of the 1980s, we had to design and build a computer with the hardware of the 1970s. Miniaturization of computer components had just begun, so we had to make our computer the size of a desk to be powerful enough for our purpose. It also had to have a high-resolution television screen instead of a flat reflective display which does not yet exist. In addition to a typewriter keyboard, we provided a movable device–we call it a "mouse"–that controls an electronic pointer on the screen. It also has push buttons to indicate commands to be carried out.

Meanwhile, the first of the microcomputers, or microprocessors, began to appear on the market. Engineers were designing integrated circuits with thousands of transistors on single silicon chips. Several of these chips can be connected to perform all the operations of a digital computer in a very small space. This development triggered a commercial explosion in calculators, control devices, and video games. This was encouraging because it confirmed our projections about the future of computers. It also meant that more people would be getting interested in personal computing.

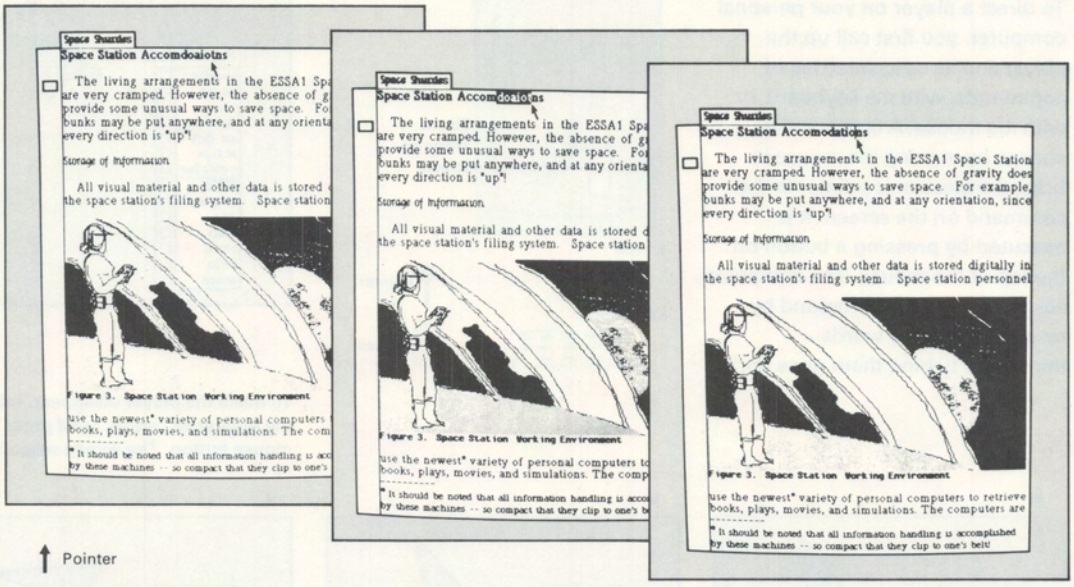
The first microprocessors were generations away from the kind of computing power that would be needed for the personal computers of the 1980s. By 1978, a second generation of microprocessors had become available that are about 10 times as powerful as their predecessors. If the trend continues, the next generation of microprocessors, in the early 1980s, will be powerful enough for our personal computer. By packaging several microprocessors together, we can get the tens of millions of executions per second Dynabook will require.

A set of instructions on how to perform a process that can be recalled and carried out later is called a program. Movie scripts, football plays, and musical scores are all programs written by humans to be followed by humans. Only in the past few decades have machines been constructed that can also store instructions and later recall and follow them, although these programs are much simpler than the programs that humans can handle.

Most computer programming is done using lists of commands in a sequence. This is often described as using ingredients in a recipe. For example, to make cookie batter, you follow a recipe that tells you first to mix the ingredients flour and water. If the batter is too sticky, add more water. If the batter is too thin, add more flour. If the batter is just right, do not do anything. The task is ended.

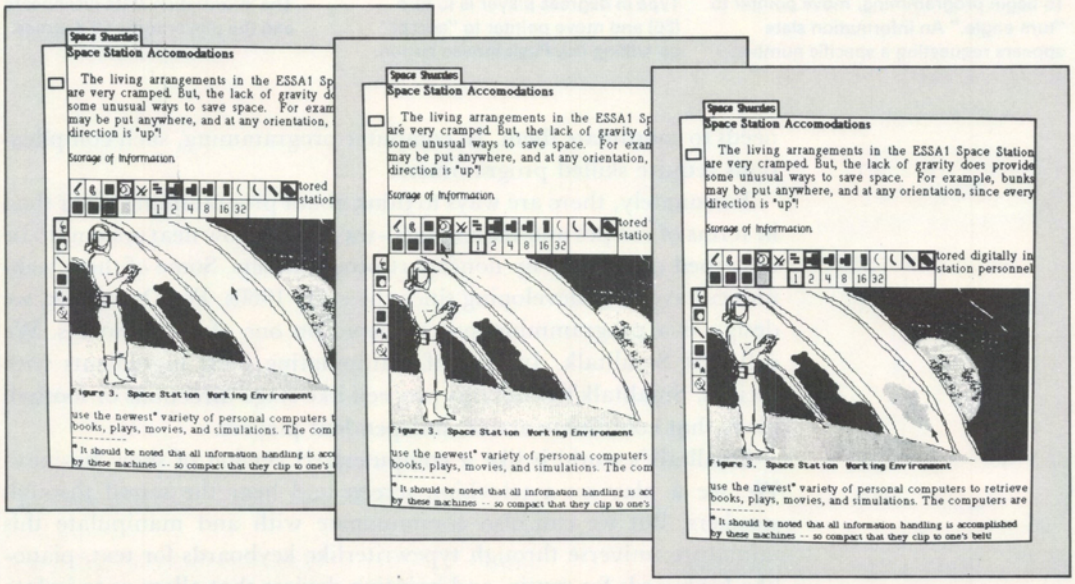
Recipes get very complicated when many things have to be coordinated simultaneously, such as in planning a banquet. A banquet "programmer" is responsible for hiring the hall, the cook, and the waiters; ordering the table settings; planning the menu; and inviting the guests. Since many of these activities happen at the same time, he

Editing a Composition



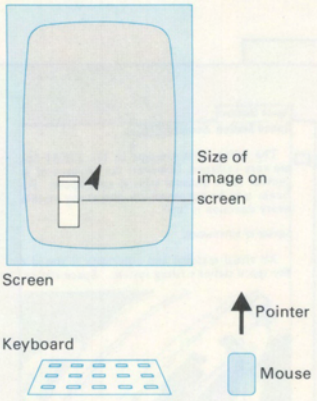
↑ Pointer

You can write and edit an illustrated composition on your personal computer. After writing and storing a story on the space shuttle, for example, you can type instructions to call up any part of the story for checking. If you discover a typographical error, you can use the "mouse" control device to correct it. When you guide the electronic pointer under the error, the screen behind this section goes dark. You type the correction, push a button on the mouse, and it appears in place. To edit the illustration, push another button on the mouse and a "painting palette" will appear on the screen. After selecting a "paint" with the pointer, you take it to the area to be painted. After pushing a button on the mouse, you can color in the desired area by moving the mouse.

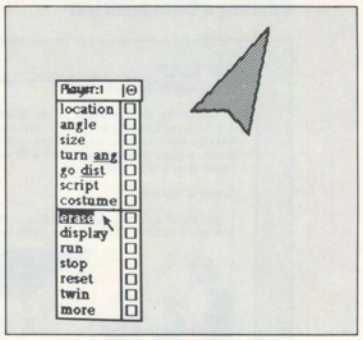


Programming a Player

To direct a player on your personal computer, you first call up the player and its associated list of commands, with the keyboard, or with the mouse. A command is chosen by moving the mouse to bring the electronic pointer to the command on the screen. It is executed by pressing a button on the top of the mouse. The computer acknowledges the command by responding to the words—the screen behind them goes dark.

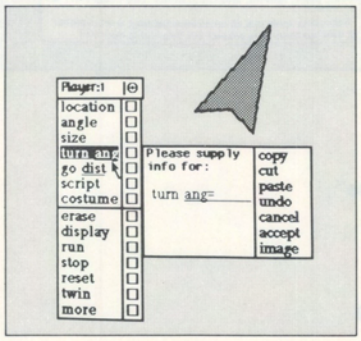


1



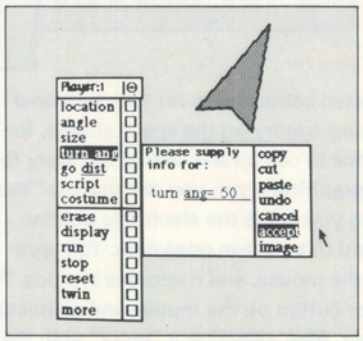
To make the player disappear, bring the pointer to "erase," and press mouse button. The screen responds.

5



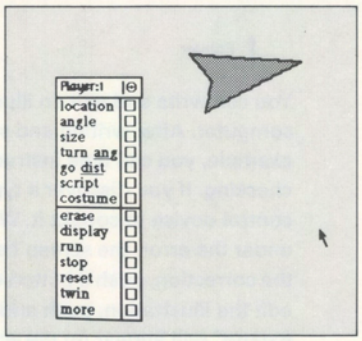
To begin programming, move pointer to "turn angle." An information slate appears requesting a specific number.

6



Type in degrees player is to turn (50) and move pointer to "accept" on editing list. Push mouse button.

7

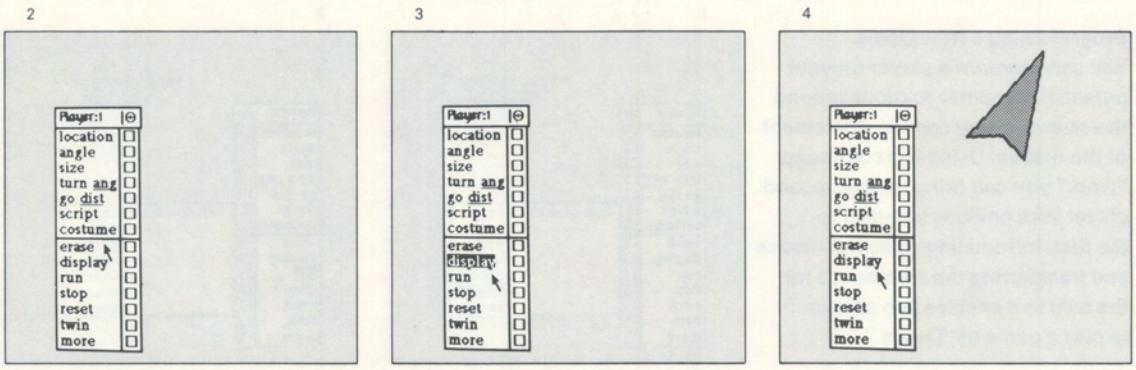


The information slate disappears and the player turns 50 degrees.

needs to coordinate them. In computer programming, such complications require skilled programmers.

Fortunately, there are ways to think about programming other than in terms of recipes and ingredients—ways that allow neat systems to be described compactly by nonexpert programmers. Some of these techniques have been developing since the early 1960s. For Dynabook, we designed a programming system based on one of these models. We called it Smalltalk. Instead of manipulating inert ingredients with recipes, Smalltalk's programs proceed like stage directions or football plays that coordinate active, independent players.

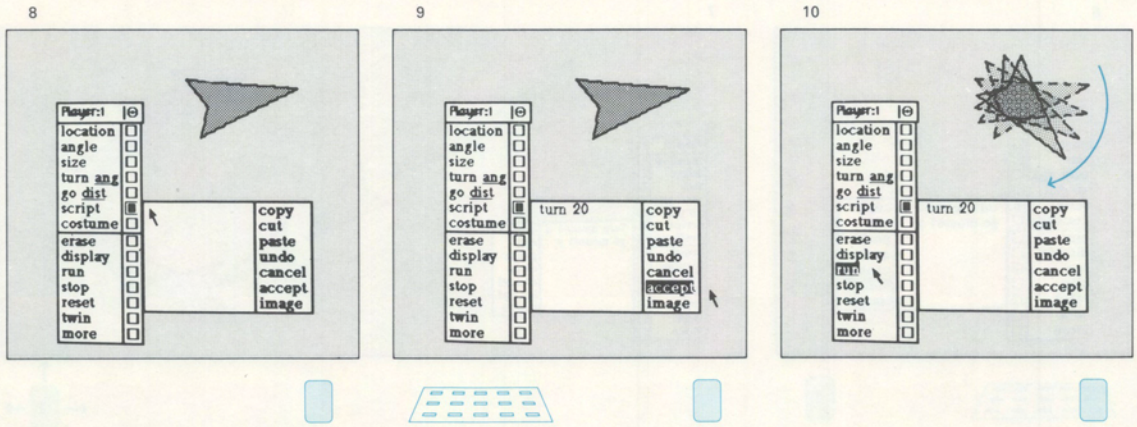
Smalltalk's world is like a computerized version of show business. We see a play on a television screen and hear the sound through speakers. But we can also communicate with and manipulate this miniature universe through typewriterlike keyboards for text, pianolike keyboards for music, and pointing devices that allow us to isolate



The player disappears and the command clears.

Move pointer to "display," and press the mouse button. The screen responds.

The player reappears and the command clears.



To program a general task, move the pointer to "script." A blank information slate will appear.

Type the command "turn 20" and move the pointer to "accept."

Move pointer to "run" and press the mouse button. The player will turn in 20-degree steps.

and work with pictures on the television screen. People learn Smalltalk by first directing an already written play. Next, they add their own modifications to the play. Finally, they are ready to write and produce a play of their own. They are then programming.

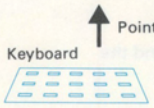
The production of a Smalltalk play begins with the author–the programmer–visualizing various scenes, then writing them down for the players. Rehearsals follow. "Bugs" are eliminated. Then the costumes and sets are added and the show is ready to be performed.

The display screen is the theater. The play consists of the intentions of the programmer. The show is how any one performance turns out. For example, Beth's horse-racing theater could produce many different races from the same play and players.

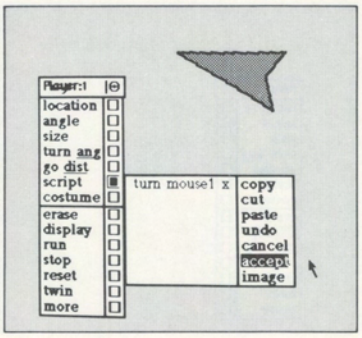
Smalltalk's program is unique in that each player can produce many offspring, or new players who inherit their characteristics from their parent. The characteristics can then be modified by program-

Programming a New Game

You can program a player on your personal computer to move around the screen in response to movement of the mouse. Using the command "twin," you can bring forth a second player with abilities identical to the first. Introducing a second mouse and transferring the commands for the twin to it enables two people to play a game of "Chase."

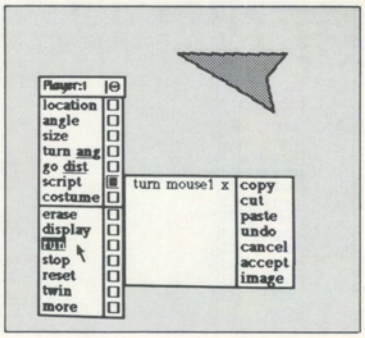


1



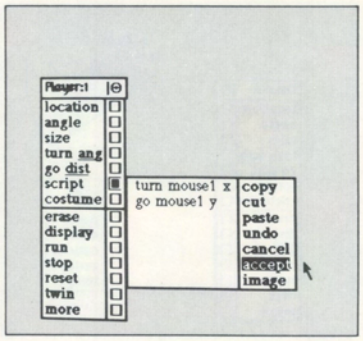
Move pointer to "script." Type "turn mouse1 x." Execute "accept."

2



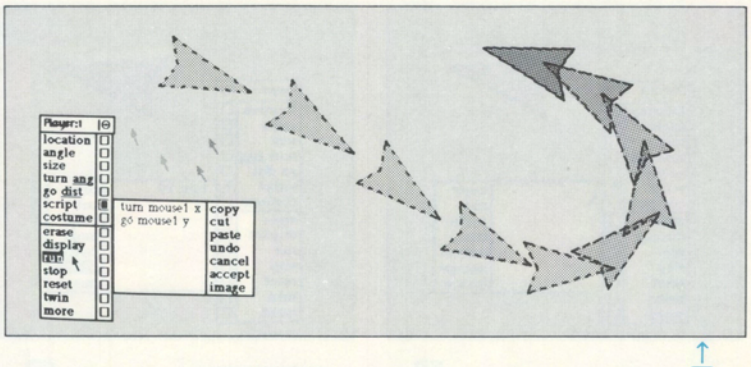
Move pointer to "run."

6



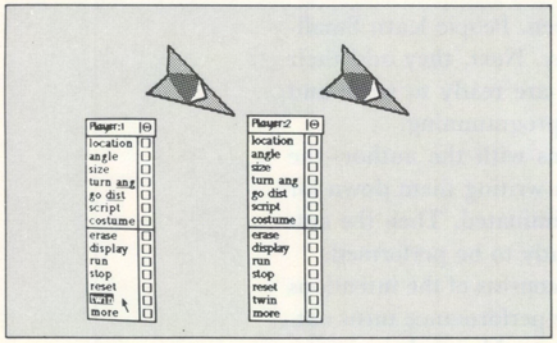
Type in "go mouse1 y" on the slate and move the pointer to "accept."

7



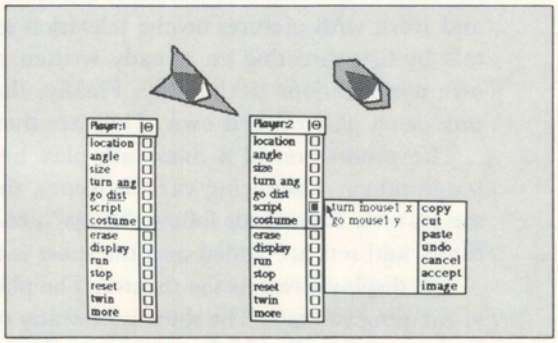
Move the pointer to "run" and move the mouse in vertical and horizontal directions. The player will then travel about the screen like a car.

10



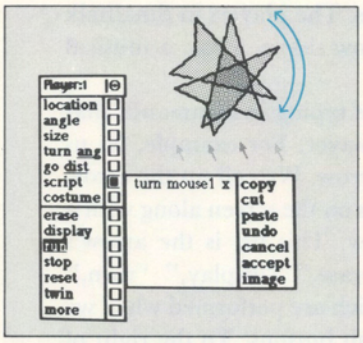
Commanding and executing "twin" on original player list causes identical player, with its identical list, to appear on the screen. The palette disappears.

11



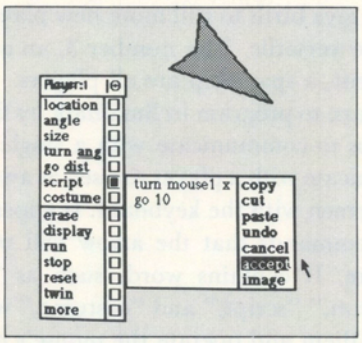
Redesign player 2 with the costume routine. Next, call up "script." Mouse 1 instructions will appear on player 2's slate.

3



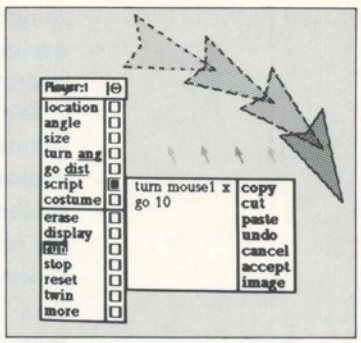
Control the player's rate of turn by moving the mouse horizontally.

4



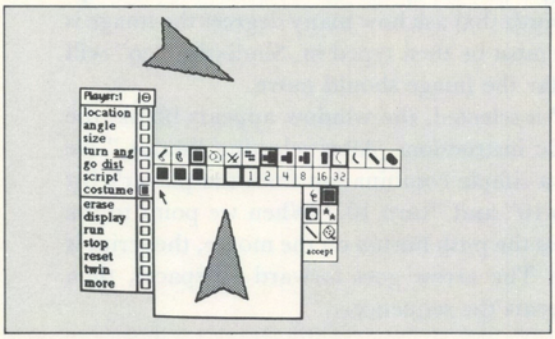
Point to "stop," type "go 10," and move pointer to "accept." The player can then move forward in 10-unit steps.

5



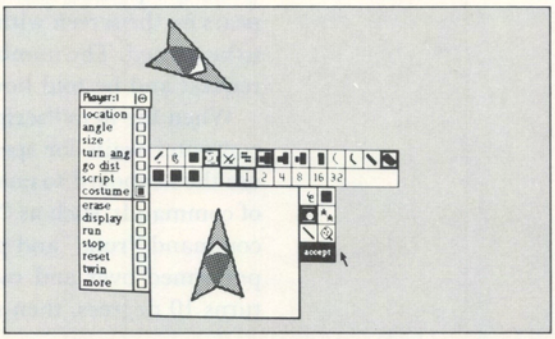
Execute "run" and move mouse to the left. The player will move forward, turning right gradually.

8



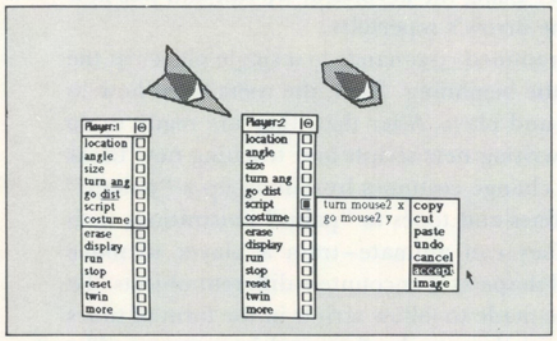
To change the appearance of the player, bring the pointer to "costume." The painting palette and the master image of the player will appear on the slate.

9



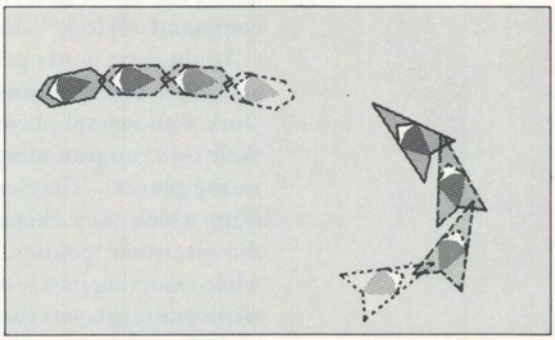
Use the pointer to pick lines and tones from palette with which to paint the player image. "Accept" transfers the costume to the player.

12



Edit the text on player 2's slate, changing 1 x and 1y to 2x and 2y. Execute "accept."

13



Execute "run" on both player lists and key instructions to remove the lists. You and a friend now have the total screen to play "Chase."

ming to give birth to still more new players. The players in Smalltalk are quite versatile. The number 3, an arrow shape, a car, a musical instrument, a spaceship are all players.

We learn to program in Smalltalk by first typing in commands that permit us to communicate with a single player. For example, let us communicate with a player dressed as an arrow. We call up the arrow on the screen with the keyboard. It appears on the screen along with a list of commands that the arrow will obey. This list is the arrow's repertoire. It contains words such as "erase," "display," "turn," "go," "run," "script," and "costume," which are performed when we point at them and operate the mouse's push buttons. To the right of the repertoire is a "how" column that is used when we want the computer to report in detail how it will follow the commands.

For example, the command "erase" makes the arrow's image disappear and "display" makes it reappear. With other commands, such as "turn" and "grow," the computer must ask questions. For example, when the pointer is brought to "turn," an information window appears on the screen with words that ask how many degrees the image is to be turned. The number must be then typed in. Similarly, "go" will request and be told how far the image should move.

When the item "script" is selected, the window appears blank; the computer waits for specific instructions. After selecting "script," we use the keyboard to enter a simple combination from the player's list of commands, such as "go 10" and "turn 10." When we point to the command "run," and press the push button on the mouse, the script is performed over and over. The arrow goes forward 10 spaces, then turns 10 degrees, then repeats the sequence.

The command "stop" is selected to make the player stop performing the script. Thereafter, we can do many things. The script can be modified, for example, by changing "go 10" to "go 15." Or we can devise an entirely new command. For example, the program we wrote drives the arrow in a circle. Using the keyboard, we can add this new command–"circle"–to the arrow's repertoire.

Issuing previously programmed commands to a single player in the computer theater is only the beginning. Next, the users learn how to work with several players and plays. After that, they are ready to do their own programming, writing new scripts and teaching new tricks to the players. They can change costumes by calling up a "palette" from which they choose lines and tones to "paint" illustrations with the electronic pointer. They can animate–train a player to move while changing its size and shape as it encounters different objects. To write music, players can be made to follow scripts in the form of tunes and given costumes that are the sounds of musical instruments. The users can simulate systems that exist in the real world such as drawbridges, electrical circuits, or rides at the amusement park.

Animated games are particularly instructive and fun to do. For example, a game in which one player chases another around the

screen can be programmed in a few minutes. We write a script that hooks up the "turn" direction to the horizontal axis of our mouse and the "go" distance to the vertical axis of the mouse. When "run" is selected and executed, the player can be driven like a car by moving the mouse forward and to the sides. The next step is to select the list item "twin" in order to produce another player with the same script. The new player's script can be modified to respond to a second mouse. Now two people can drive independent players about the screen. We can change each player's costume into an individual car shape with Smalltalk's painting capability, and the simplest version of the game is ready to be played.

The important difference between being able to program one's own video game and obtaining a canned version from a game company is that new variations can be added to the homemade program as the participants think them up. For instance, what can be done about obstacles and collisions? How about putting the cars on a track? Besides being more fun, programmable games also help the programmer learn how complex situations can be modeled.

How can all these things be done with something as small as the proposed Dynabook? How are the electronics arranged inside the computer so that you can control the images on the screen with the keyboard and the mouse?

To learn something about this, we must first realize that computers deal only in symbols–a pattern, or structure–with which a sender (the programmer) communicates with a receiver (the program). A concept may be represented by many symbols. For example, a device used to close off an opening is called "door" in English, "porte" in French, and "Tür" in German. These words are meaningless, however, to anybody who does not understand the particular language, or symbol system, to which the words belong.

There is a simplest way to represent symbols. German philosopher Gottfried Wilhelm Leibniz in the 1700s discovered that all conceivable symbols could be formed from combinations of just two indivisible marks. The form of these marks is unimportant. Leibniz used "God" and "void." Morse code uses "dot" and "dash." In computer terms they become "on" and "off," or sometimes 0 and 1. These are the familiar "bits" of computer terminology.

Leibniz discovered that any number of symbols can be formed from patterns of these two marks. The greater the number of bits in a pattern, the more symbols can be formed. For example, a two-place pattern can form four different symbols. A three-place pattern can form eight symbols. A pattern 20 places long can form more than 1 million symbols.

Computers can hold millions of bits that can be used by a programming system to represent anything a programmer understands. A computer's power depends on how many bits it can store and how fast it can manipulate them. The Smalltalk system, for example, has a

The color and detail of a computer-drawn animated film suggest how powerful computers have already become.

storage capacity of about 21 million bits. One million of these are fast and used for the display, and as a scratch pad. The rest are used for long-term storage.

The simplest use of bits to represent information is to store display images for the television screen. About half the primary memory, some 500,000 bits, is used for this purpose. From the point of view of the display programs, the screen is a configuration of 800 rows of 600 bits each. As the electron beam of the television tube scans the face of the tube, the display programs reach into this memory and pick up the bits row by row, painting an image. The image changes whenever Smalltalk's graphics programs change the bits in the display memory. Then the next time the beam sweeps through the changed area, a new picture appears on the screen. It looks somewhat like a high resolution version of the electronic signs that display time and weather and advertising messages on top of buildings or beside highways.

This manipulation of bits is the bottom layer of many levels of complexity in programming a computer. These levels are like those required in designing, building, and using a theater. A construction architect designs the building. He is responsible for its structural integrity and aesthetics. A theater architect is concerned with the design as it supports different kinds of artistic productions. Together they will work out the interior design–such as the stage, lighting, and seats. In the computer world, the construction architect is the chief hardware designer; the theater architect is the chief software designer.

The theater manager, who handles stagehands, scenery designers, costume makers, and the lighting and maintenance people, is comparable to a computer's operating and programming system. Smalltalk is, in effect, the theater manager.

A new show is put together by a production team composed of the playwright, composer, producer, director, and people in charge of the

music, sets, costumes, lighting, choreography, and properties. In the computer world, the production team is the one or more programmers who build environments for document and image handling, animation, music, information retrieval, and simulation.

Finally, the stage performers–the actors and musicians–are assembled and rehearsed. In Smalltalk, these are the players.

Every major new technology has changed people's lives. What is the onrush of computerization doing to us in 1978? There is no doubt that it has already changed our lives. The personal computers of the next decade will change them even more.

I think that the impact on human life will resemble what happened after the invention of movable printer's type in the 1400s, but in a much shorter time. When books became readily available, the concept of literacy arose and people who could not use books were labeled "illiterate." The illiterate was soon left behind by those who could read and write. The primary function of tutors, and later of schools, was to "teach" literacy.

To meet the challenge of the new technology, many schools are now conducting courses in "computer literacy." In addition, young people visit research centers like ours to see what computers can do. As they learn by doing, they also have a lot of fun. Computer literacy will become more widespread as more schools add computer classes to their curriculums. This will be an absolute necessity if people are going to understand their technology.

Today's students have not progressed much beyond computer kindergarten. They are still primarily involved with video games. The usage that will probably follow video games is both exciting and disturbing. It is exciting because of the artistic possibilities. It is disturbing because what may develop could be more mind-numbing than most of today's commercial television.

The next step in personal computing will probably be do-it-yourself cartoons. It is already possible to represent and animate stylized color images, applying techniques similar to those employed in Smalltalk. Soon after that, the first participatory drama systems will appear. Imagine a full-color James Bond movie in which you can insert a character representing yourself. You can be the hero, the villain, or one of the minor characters, with the plot controlled by decisions you make during the course of the action. Your family or friends can also assume roles. The story will be the result of your combined decisions.

The artistic possibilities are tremendous. Instead of a James Bond story, imagine manipulating a Shakespearean play. Or, working with your personal computer and a music synthesizer, you could create your own sonata, or the next rock-and-roll hit. If your taste runs to art, your computer can become a canvas on which you can experiment with a realistic sunset, or the most inventive abstract painting.

With your personal computer, you can do as much as your imagination will allow.