

# The Smalltalk Graphics Kernel

Daniel H H Ingalls  
Learning Research Group  
Xerox Palo Alto Research Center  
3333 Coyote Hill Rd  
Palo Alto CA 94304

Graphics are essential to the quality of an interactive programming system and to the interactive applications that go along with such a system. Qualitatively, people think with images, and any system that is incapable of manipulating images is incapable of augmenting such thought. Quantitatively, a person can visually absorb information equivalent to millions of characters a second, while the normal rate for reading text is less than 100 characters a second.

For the graphical interaction cycle to be complete, a computer system must provide a channel for input in the visual domain as well. While the projection of images from the realm of thought into the space of electronic information seems an impossible task, a well-designed pointing device can effectively harness the computer's graphical output capability to express graphical input from the user. Given such a pointing device, the process of selecting from graphical objects, such as text displayed on the screen, is natural and rapid. By tracking the pointer with a program that simulates a pen or paintbrush, the visual input channel can be extended to include line drawing and freehand sketches.

The purpose of graphics in the Smalltalk system is to support the *reactive principle*:

*Any object accessible to the user should be able to present itself in a meaningful way for observation and manipulation.*

Meaningful presentation of any object in the system demands maximum control over the display medium, and

many technologies fall short in this respect. One approach that provides the necessary flexibility is to allow the brightness of every discernible point in the displayed image to be independently controlled. The simplest implementation of this approach is a contiguous block of storage in which the setting of each *bit* (1 or 0) is mapped into dark or light illumina-

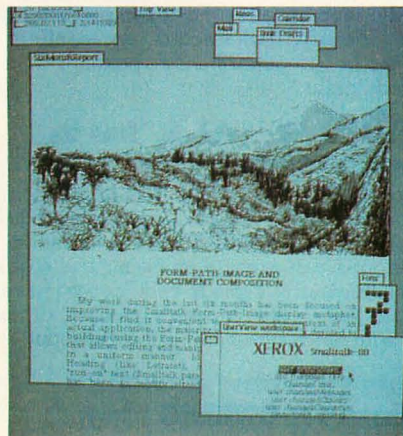


Photo 1: An example of a Smalltalk-80 video display. Note the multiple windows, the combinations of text and graphics, and the pointer in the window marked "UserView workspace."

tion of the corresponding picture element, or *pixel*, when displaying or combining with other images. The block of storage is thus referred to as a *bitmap*, and this type of display is called a *bitmap display*. The simplest form of bitmap allows only two brightness levels, white and black. The Smalltalk-80 graphics system is built around this model.

Photo 1 shows a typical view of the Smalltalk-80 system, and it illustrates the wide range of graphical idiom

implied by the reactive principle. Rectangular areas of arbitrary size are filled with white, black, and various halftone patterns. Text, in various typefaces, is placed on the screen from stored images of the individual characters. Halftone shades are "brushed" by the user to create freehand paintings. Moreover, although not shown on the printed page, images on the display may be moved or sequenced in time to provide animation.

## Graphical Storage—Forms

Simple images are represented by instances of class *Form*. A *Form* has height and width and a bitmap that indicates the white and black regions of the particular image being represented. Consider, for example, the arrow-shaped *Form* that appears in the lower-right window of the screen image in photo 1. The internal representation of this *Form* is depicted in figure 1. Its height is 16, its width is 8, and its appearance is described by the pattern of ones and zeros (shown as light and dark squares) in its bitmap. The height and width of the *Form* serve to impose the

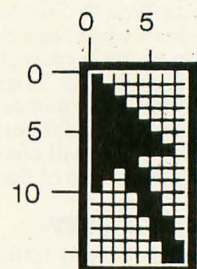


Figure 1: A simple *Form* representing the cursor in photo 1.



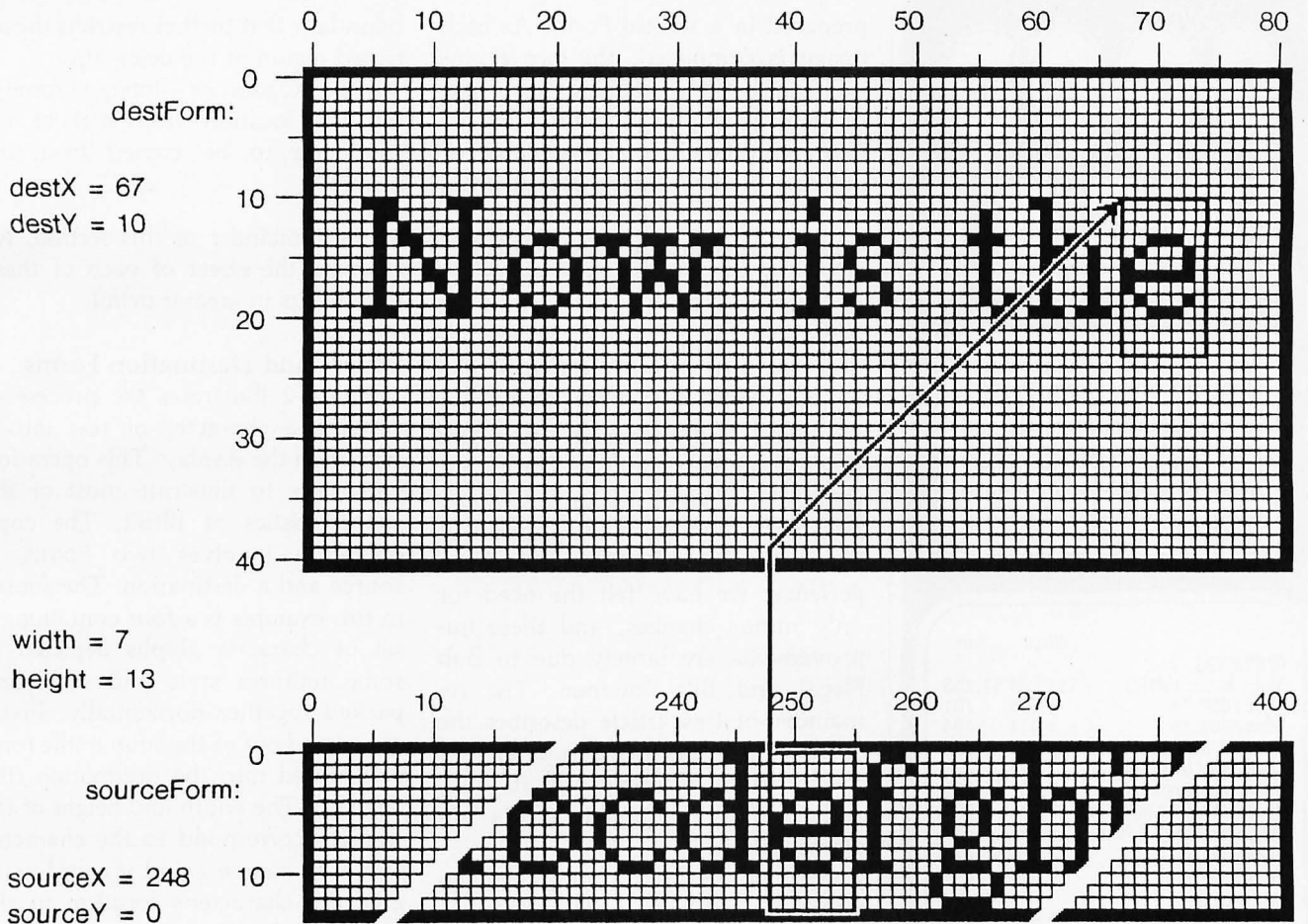


Figure 2: Copying a character of text from a source Form (bottom) to a destination Form (top).

appropriate two-dimensional ordering on the otherwise unstructured data in the bitmap. We will return to the representation of Forms in more detail later in this article.

A complex image can be represented in either of two ways: by a very large Form, or by a structure that includes many Forms and rules for combining and repeating them in

order to produce the desired image. The freehand drawing in the center of photo 1 is an example of the former, and the text below it is an example of the latter.

The large unstructured Form has an additional use of great importance: it can be presented to the display hardware as a buffer in memory of the actual data to be shown on the

display terminal. We refer to the Form which is so used as the displayForm. Since the interface to the hardware is through a Form, there is no difference between combining images internally and displaying them on the screen. Animation can be done simply in this manner: one Form serves as the displayForm while the next image to be displayed is



## Now! An Acoustic Coupler for \$125.

The Omnitec Model 715 (originate only) has the necessary features, needed to expand any business or home computer systems capability: 0-450 baud data rate, RS232 compatibility and carrier detect at a low factory-direct price of \$125. Or, select the Model 770 at \$179.00 for originate or answer communications. Both models carry a one year warranty.



**OMNITEC DATA**

2405 South 20th Street • Phoenix, Arizona 85034

PRICES INCLUDE SHIPPING & HANDLING  
WHEN PRE-PAID

AVAILABLE FACTORY-DIRECT ONLY!

Model 715: \_\_\_\_\_ Model 770: \_\_\_\_\_

Send Me: \_\_\_\_\_

Enclosed is \$ \_\_\_\_\_

NAME \_\_\_\_\_

ADDRESS \_\_\_\_\_

CITY \_\_\_\_\_

STATE \_\_\_\_\_

ZIP \_\_\_\_\_

CHECK OR MONEY ORDER

# GOOD IDEA!

You can save buying wholesale through our firm. As your agent we will buy computers on the wholesale market for you. Our fee is one fourth of what we save you off list price. Access to over 500 manufacturers. Minimum fee of \$75 per order. Call for other prices.

	Whse.	Fee
<b>COMPUTERS</b>		
Alpha Micro 10MEG	\$11,204	\$1,243
Altos 8000-10	5,695	701
Altos 8000-15	4,014	494
Archives 64K QD	4,450	512
Compustar Model 30	3,595	225
Cromemco System 3	5,357	659
Dynabyte 64K 2 MEG	5,929	691
Calif. Comp. 64K 1 MEG	3,987	427
Ithaca System 2A	2,520	270
Televideo Sys. I	2,600	349
<b>DISK DRIVES</b>		
Corvus 5 MEG Hard	2,588	290
Corvus 20 MEG Hard	4,450	500
Morrow 26 MEG Hard	3,596	349
<b>CRT'S</b>		
ADDS View Point	505	36
DEC VT 100	1,305	130
Hazeltine Esprit	565	45
IBM 3101 Model 10	1,140	60
Lear Siegler ADM 3A +	710	58
Leadex 13" Color Monitor	350	25
NEC 12" Monitor	190	18
Televideo TVI 910	545	39
Televideo 950	835	90
Visual 200	765	107
<b>PRINTERS</b>		
Anadex 9501	1,150	125
Centronics 737	650	86
Diablo 630 R/O	1,795	200
Epson MX80	410	58
IDS Paper Tiger 460 G	1,020	94
IDS 560/g	1,220	119
NEC 5510 w/Tractor	2,360	164
C. Itoh 25 cps	1,200	175
TI 810 Basic	1,355	135
<b>SOFTWARE</b>		
Word Star	249	63
Spell Guard	200	—

Prices subject to change without notice.

We are buying agents for overseas computer dealers. Export services available.

International Telex 470851

**The Purchasing Agent**  
1635 School Street, Suite 101  
Moraga, CA 94556  
(415) 376-9020

prepared in a second Form. As each image is completed, the two Forms exchange roles, causing the new image to be displayed and making the Form with the old image available for building the next image in sequence.

## Graphical Manipulation—BitBlt

To support a wide range of graphical presentation, we have specified a kernel operation on Forms that we call *BitBlt*. All text and graphic objects in Smalltalk are displayed and modified using this single graphical primitive. The author wrote the original design in October 1975 with the advice and support of Diana Merry. After five years' experience, we have felt the need for only minor changes, and these improvements are largely due to Bob Flegal and Bill Bowman. The remainder of this article describes the current BitBlt primitive in detail—its specification, examples of its use, and, finally, the details of its implementation.

One of the first computers on which a Smalltalk system was implemented had an instruction called BLT for *block transfer* of 16-bit words. The name BitBlt derives from the generalization of data transfer to arbitrary bit locations, or pixels. BitBlt is intentionally a very general operation, although most applications of it are graphically simple, such as "move this rectangle of pixels from here to there."

A specific application of BitBlt is governed by a list of parameters that includes:

- *destForm*—a Form into which pixels will be stored by BitBlt
- *sourceForm*—a Form from which pixels may be copied
- *halftoneForm*—a Form containing a spatial halftone pattern
- *combinationRule*—an Integer specifying the rule for combining corresponding pixels of the *sourceForm* and *destForm*
- *destX*, *destY*, *width*, *height*—Integers specifying the rectangular subregion to be filled in the destination
- *clipX*, *clipY*, *clipWidth*, *clipHeight*—Integers specifying a rectangular

boundary that further restricts the affected region of the destination

- *sourceX*, *sourceY*—Integers specifying the location (top left) of the subregion to be copied from the source

In the remainder of this section, we examine the effect of each of these parameters in greater detail.

## Source and Destination Forms

Figure 2 illustrates the process of copying a character of text into a region on the display. This operation will serve to illustrate most of the characteristics of BitBlt. The copy operation involves two Forms, a source and a destination. The source in this example is a *font* containing a set of character glyphs depicted in some uniform style and scale and packed together horizontally. Pixels are copied out of the source (the font) and stored into the destination (the display). The width and height of the transfer correspond to the character size. The source *x* and *y* coordinates give the character's location in the font, and the destination coordinates specify the position on the display where its copy will appear.

## Clipping Rectangle

In its specification, BitBlt includes a rectangle that limits the region of the destination that can be affected by its operation, independent of the other destination parameters. We call this rectangle the *clipping rectangle*. Often it is desirable to display a partial *window* onto larger scenes, and the clipping rectangle ensures that all picture elements fall inside the bounds of the window. By its inclusion in the BitBlt primitive, the clipping function can be done efficiently and in one place, rather than being replicated in all application programs. Figure 3 illustrates the result of imposing a clipping rectangle on the example of figure 2. Pixels that would have been placed outside the clipping rectangle (the left edge of the "N" and half of the word "the") have not been transferred. If other characters had fallen above or below this rectangle, they would have been clipped similarly.



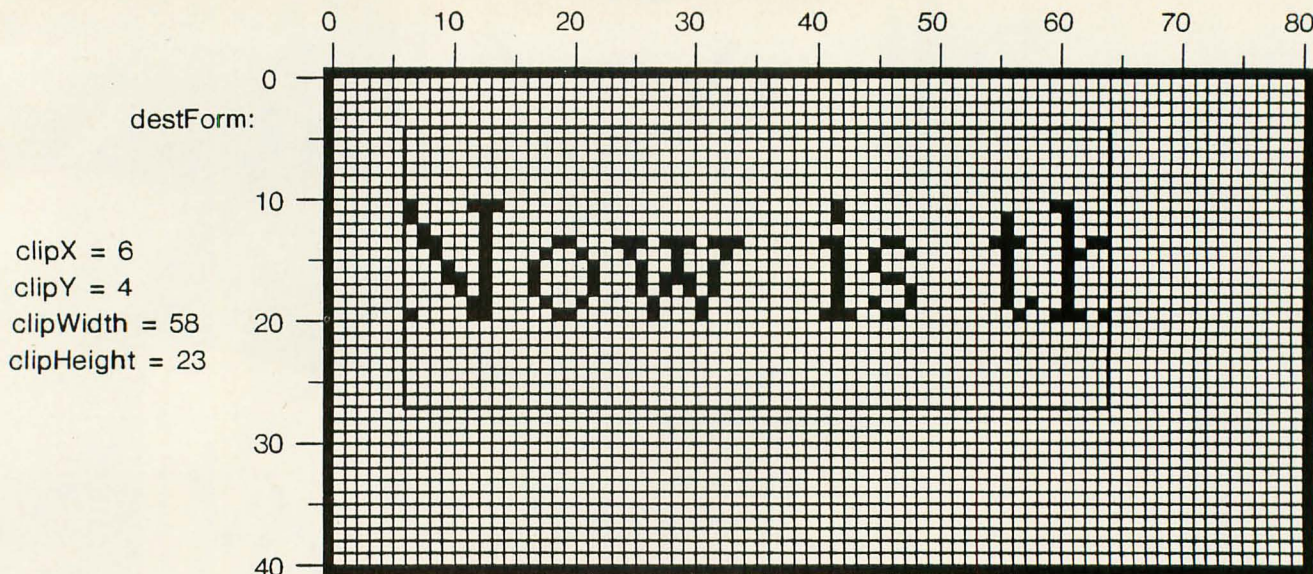


Figure 3: An example of using a clipping window on the illustration in figure 2.

### Halftone Form

It is often desirable to fill areas with a regular pattern that gives the effect of gray shading or texture. To this end, BitBlt provides for reference to a third Form (halftoneForm) containing the desired pattern. This Form is

restricted to a height and width of 16. When halftoning is specified, this pattern is effectively repeated every 16 units horizontally and vertically over the entire destination. There are four "modes" of supplying pixels from the source and halftone controlled by

eliding (supplying nil for) sourceForm or halftoneForm:

- Mode 0—No source, no halftone (supplies solid black)
- Mode 1—Halftone only (supplies halftone pattern)

## Peachtree Software™ Announces

# SALES TRACKER™

**S**olved! The case of ineffective business control. Now Sales Tracker microcomputer software gives business managers unprecedented control over lifestream activities.

You get ANSI COBOL, floppy and hard disk support, and a modular format that lets you install packages as you need them. And look at these outstanding features.

**Accounts Receivable:** Multiple companies on one disk. Nine selectable reports including G/L sales and receivables, customer classes, and sales tax routines.

**Sales Analysis:** Reports by salesman (with commission), product class, inventory item, customer, item by customer, and customer by item.



**Order Processing:** Handles multiple shipping locations, automatic discounting, partial shipments, back orders, demand invoicing, full order status and maintenance.

**Inventory:** Multiple companies on one disk; reports by class, warehouse, price level and quantity break; automatic price updating; maintains items optionally by case, lot or serial number.

**C**lues to why Sales Tracker is so easy to use? Excellent documentation, sample data files, conversant menu-driven programs, and standardized formats. For information call 800-835-2245 ext. 35 (Kansas, 800-362-2421, ext. 35)



The Very Best "Off-The-Shelf" Software Is "Off-The-Tree."



PEACHTREE SOFTWARE, 3 Corporate Square, Suite 700, Atlanta, Georgia 30329 (404) 325-8533  
Telex II: 810-751-0273 PEACHTREE ATL.

Peachtree Software and Sales Tracker are trademarks of Peachtree Software Incorporated.



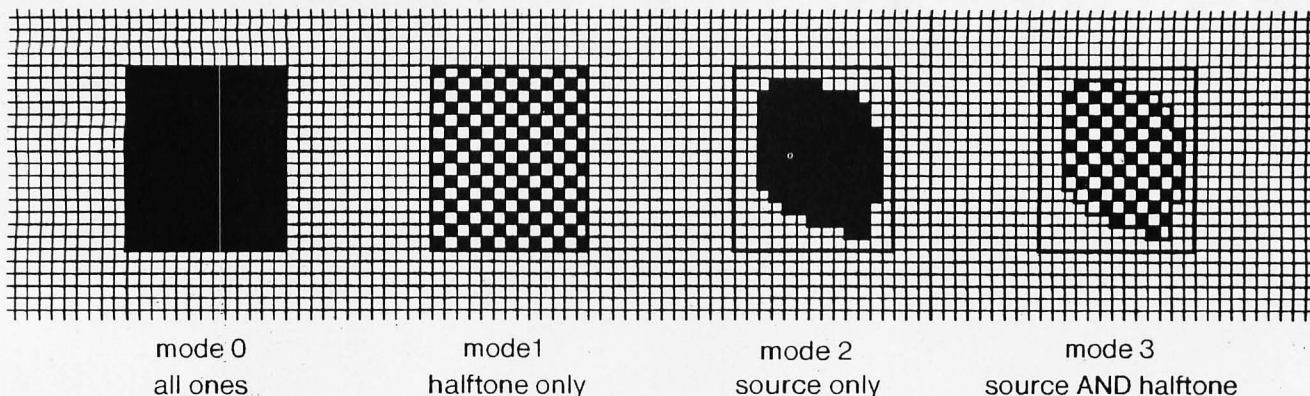


Figure 4: BitBlt's four possible source modes.

- Mode 2—Source only (supplies source pixels)
- Mode 3—Source AND halftone (supplies source bits masked by halftone pattern)

Figure 4 illustrates the effect of these four modes with the same source and

destination and a regular gray halftone.

### Combination Rule

The examples above have all stored their results directly into the destination. There are actually many possible rules for combining each source

element S with the corresponding destination element D to produce the new destination element D'. Such a rule must specify a white or black result for each of the four cases of source being white or black and destination being white or black.

Figure 5 shows a box with four cells corresponding to the four cases encountered when combining source (S) and destination (D). For instance, the cell numbered 2 corresponds to the case where the source was black and the destination was white. By appropriately filling the four cells with white or black, the box can be made to depict any combination rule (there are sixteen possible rules altogether). The numbers in the four cells relate the rule as depicted to the integer value that selects that rule. For instance, to specify that the result

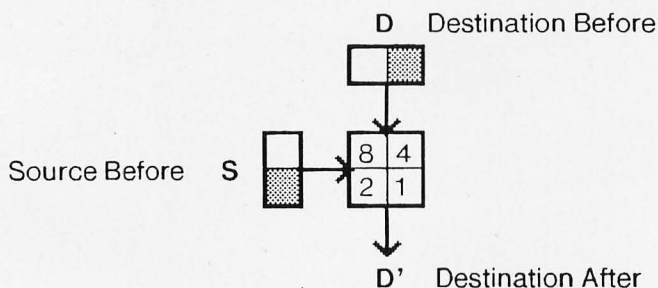


Figure 5: A BitBlt combination diagram. This diagram, when filled in, specifies the effects of a given combination (or "rule") on all combinations of dark and light source and destination cells. Each combination is given a number equal to the sum of the cells that are darkened. See figure 6 for examples.

# DISASTER INSURANCE.

## PROTECT YOUR HARDWARE FROM THE UNEXPECTED.

Not to mention the unavoidable pollutants in the air. Performance robbing dust, grime, spills and static electricity.

Cover Craft Dust Covers help extend the useful life of your computer equipment at a fraction of the cost. Perhaps that's why more people throughout the world rely on Cover Craft Dust Covers than any other brand.

Visit your local dealer or contact Cover Craft.

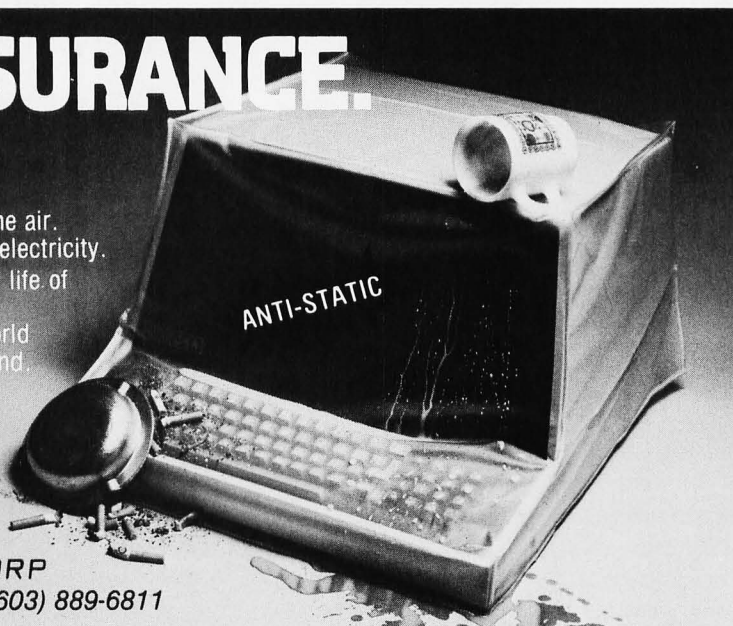
Dust Covers for most terminals, disks, printers, modems, etc.

**\$8.95-\$15.95**  
Shipping extra.



**COVER CRAFT** CORP

PO Box 555, Amherst, NH 03031 • (603) 889-6811





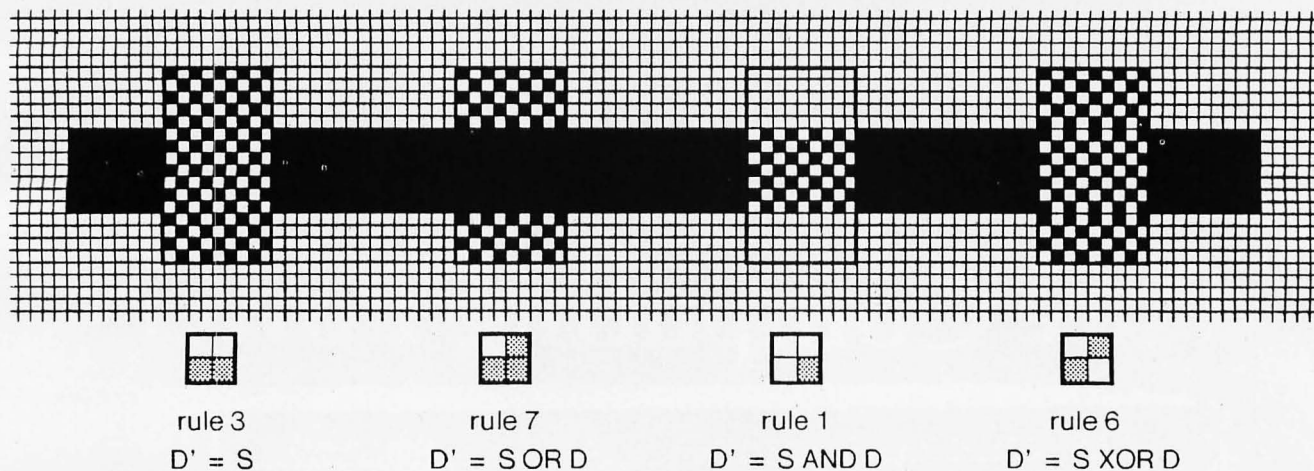


Figure 6: Four common combination rules.

should be black wherever the source or destination (or both) was black, we would blacken the cells numbered 4, 2, and 1. The associated integer for specifying that rule is the sum of the blackened cell numbers, or  $4 + 2 + 1 = 7$ .

Figure 6 illustrates four common combination rules graphically. Each is described by a combination diagram, its integer rule number, and the actual logical function being applied. The earlier case of ORing can be seen in left center of the figure. This case is often described as painting "under" the destination because existing black areas remain black.

### Smalltalk Access to BitBlt

In this section, we present the Smalltalk interface to BitBlt and take a detailed look at the application of BitBlt to text display and line drawing. In preparation, you will need some additional context, which we present here before describing class BitBlt.

Besides class *Form*, two additional classes are used extensively in working with stored images, *Point* and *Rectangle*. *Points* contain *x* and *y* coordinate values and are used for referring to pixel locations relative to the top left corner of a *Form* (or other point of reference). By convention, *x* increases to the right and *y* down, consistent with the layout of text on a page and the direction of TV scanning. A *Rectangle* contains two *Points*: *origin*, which specifies the top left corner, and *corner*, which in-

dicates the bottom right corner of the region described. Class *Point* provides protocol for access to the coordinates and for various useful operations such as translation and scaling. Class *Rectangle* provides protocol for access to all the coordinates involved and other operations such as intersection with other rectangles. It may be useful to note the parallel between classes *Point*, *Rectangle*, *Form* and classes *Number*, *Interval*, *IndexedCollection*. *Numbers* index *Collections* and *Points* index *Forms*. *Intervals* select *subCollections*, and *Rectangles* select *subForms*.

Figure 7 shows the complete representation of the *Form* shown in figure 1. The width and height are stored as *Integers*. The actual pixels are stored in a separate instance of class *Bitmap*. *Bitmaps* have almost no protocol, since their sole purpose is to provide storage for *Forms*. They also have no intrinsic dimensionality, apart from that projected by their own *Form*, although the figure retains this structure for clarity. It can be seen that space has been provided in

the *Bitmap* for a width of 16; this is a manifestation of the hardware organization of storage and processing into 16-bit *words*. *Bitmaps* are allocated with an integral number of words for each row of pixels. The integral constraint on row size facilitates movement from one row to the next during the operation of *BitBlt* and during scanning of the display screen by the hardware. While this division of memory into words is significant at the primitive level, it is encapsulated in such a way that none of the higher-level graphical components in the system need consider word size.

### Class BitBlt

The most basic interface to *BitBlt* is through a class of the same name. Each instance of *BitBlt* contains the parameters necessary to specify a *BitBlt* operation. The *BitBlt* protocol includes messages for initializing the parameters and one message, *copyBits*, that causes the primitive operation to take place. The class template for *BitBlt* is given in table 1.

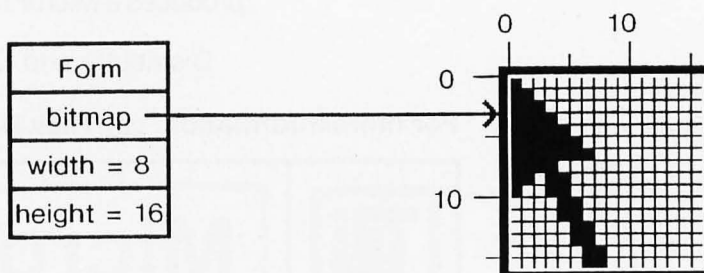


Figure 7: The complete representation of figure 1.



class name	BitBlt
superclass	Object
instance variable names	destForm sourceForm halftoneForm combinationRule destX destY width height clipX clipY clipWidth clipHeight sourceX sourceY
instance messages and methods	
<i>setup</i>  <b>destForm: form1 sourceForm: form2 halftoneForm: form3 rule: rule destRectangle: destRectangle clipRectangle: clipRectangle sourceOrigin: sourceOrigin    </b>  destForm ← form1. sourceForm ← form2. halftoneForm ← form3. combinationRule ← rule.  destX ← destRectangle minX. destY ← destRectangle minY. width ← destRectangle width. height ← destRectangle height.  clipX ← clipRectangle minX. clipY ← clipRectangle minY. clipWidth ← clipRectangle width. clipHeight ← clipRectangle height.  sourceForm == nil iffFalse: [sourceX ← sourceOrigin x. sourceY ← sourceOrigin y].  self copyBits  <i>operations</i>  <b>copyBits     &lt; primitive &gt;</b>	

Table 1: Class template for class BitBlt.

The state held in an instance of BitBlt allows multiple operations in a related context to be performed without the need to repeat all the

setup. For example, when displaying a scene in a display window, the destination Form and clipping rectangle will not change from one

operation to the next. This situation occurs frequently in the graphics kernel, as demonstrated in the following section.

### Image Synthesis of Text

Much of the graphics in the Smalltalk system consists of text and lines. These high-level entities are synthesized by repeated invocation of BitBlt. In this section and the next, we examine these two important applications more closely.

One of the advantages derived from BitBlt is the ability to store fonts compactly and to display them using various combination rules. The compact storage arises from the possibility of packing characters horizontally one next to another (as shown in figure 2), since BitBlt can extract the relevant bits if supplied with a table of left *x* coordinates of all the characters. This is called a *strike* format, from the typographical term meaning a contiguous display of all the characters in a font.

The scanning and display of text is performed in the Smalltalk-80 system by a subclass of BitBlt. This subclass inherits all the normal state, with destForm indicating the Form in which text is to be displayed and sourceForm indicating a Form containing all the character glyphs side by side (as in figure 2). In addition, this subclass defines further state information, including:

- text—a String of Characters to be displayed
- textPos—an Integer giving the current position in text

## Bower-Stewart & Associates SOFTWARE AND HARDWARE DESIGN

### \$GOLD DISK\$ CP/M® Compatible Z-80 Software

Available for all 8-5" SS-SD IBM format systems including TRS-80®, Northstar, SD Systems. Also available on 5" double density Superbrain®.

**\$175.**  
ppd

#### Un-can your canned software!

**Z-80 Disassembler** Feel coupled up with your canned software? Our Z-80 Disassembler recreates assembly language source files from absolute code enabling users to easily tailor programs to meet their specific needs. The Preconditioner works with the Disassembler to decode ASCII.

Credit cards: Immediate service, free 24 hr. phone - we will credit invoice. Checks, M.O.'s: Ten workday hold. CA. res: Add tax.



**\$50.**  
ppd

#### Great looking letters & reports!

**E-Z Text** A unique word processor organized around user-created text files, embellished with simple control commands, which supports such 'BIG GUYS' features as Automatic Footnoting, Table Spacing, Heading, Paging, Left & Right Margins, Proportional Spacing and MORE, at a 'LITTLE GUYS' price tag.



State system & controller. Allow time for surface mail. Trademarks: Digital Research, Radio Shack, Intertec.

POST OFFICE BOX 1389 HAWTHORNE, CALIFORNIA 90250 213 / 676-5055



Listing 1: The scanWord: method scans or prints text.

**scanWord: endRun**

```

| charIndex |
< primitive > "May be implemented internally for speed"
[charIndex > endRun] whileTrue:
  [charIndex ← text at: textPos.
   (exceptions at: charIndex) > 0
   ifTrue: [! exceptions at: charIndex].
   sourceX ← xTable at: charIndex.
   width ← (xTable at: charIndex + 1) - sourceX.
   printing ifTrue: [self copyBits].
   destX ← destX + width.
   destX > stopX ifTrue: [! stopXCode].
   textPos ← textPos + 1].
textPos ← textPos - 1.
↑ endRunCode

```

•xTable—an Array of Integers giving the left x location of each character in sourceForm

•stopX—an Integer that sets a right boundary past which the inner loop should stop scanning

•exceptions—an Array of Integers that, if non-zero, indicate that the corresponding characters must be specially handled

Once an instance has been initialized with a given font and text location, the scanWord: loop given in listing 1 will scan or print text until some

horizontal position (stopX) is passed, a special character (determined from exceptions) is found, or the end of this range of text (endRun) is reached.

The check on exceptions handles many possibilities in one operation. The space character may have to be handled exceptionally in the case of text that is padded to achieve a flush right margin. Tabs usually require a computation or table check to determine their width. Carriage return is also identified in the check for exceptions. Character codes beyond the

range given in the font are detected similarly and are usually handled by showing an exceptional character, such as a little lightning bolt, so that they can be seen and corrected. The printing flag can be set false to allow the same code to *measure* a line (break at a word boundary) or to find where the cursor points. While this provision may seem over-general, two benefits (besides compactness) are derived from that generality. First, if one makes a change to the basic scanning algorithm, the parallel functions of measuring, printing, and cursor tracking are sure to be synchronized. Second, if a primitive implementation is provided for the loop, it exerts a threefold leverage on the system performance. The scanWord: loop is designed to be amenable to such primitive implementation; that is, the interpreter may intercept it and execute primitive code instead of the Smalltalk code shown. In this way, much of the setup overhead for copyBits can be avoided at each character, and an entire word or more can be displayed

**HOW TO GET THE SOFTWARE CATALOG WITH APPLE'S STAMP OF APPROVAL.**

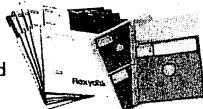
We're picky at Apple. So you can imagine how picky we are about which programs we included in our catalog called Special Delivery Software. No, we don't develop these programs here at Apple. But they're so well done, so applicable and so easy to run that we give them our Special Delivery stamp of approval. They're useful and exciting programs - special programs you've never heard about before. But the only way you can get these programs is through our Special Delivery Software catalog. To get this catalog and all the catalog products, just visit your local authorized Apple dealer. Or grab your nearest telephone and dial. The phone call won't cost you a cent. And neither will the catalog: (800) 538-8400. In California (800) 672-1424. Apple computer inc.

**FIRST CLASS**



## SPECTACULAR OFFERS

BASF "FLEXYDISK"...  
Superior Quality data  
storage medium.  
Certified and guaranteed  
100% error free.



SINGLE SIDED-SINGLE DENSITY

5 1/4" or 8" Diskettes ..... 10/\$24  
5 1/4" or 8" Vinyl Storage Pages ..... 10/\$5

### MAXELL-DISKETTES

The best quality  
diskette money can buy.  
Approved by Shugart  
and IBM.



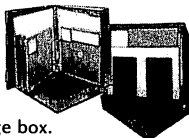
Sold only in boxes of 10

5", 1 side ..... \$3.30  
8", 1-side ..... \$3.90.  
5", 2-side ..... \$4.25  
8", 2-side ..... \$5.60

ALL MAXELL DISKETTES ARE DOUBLE DENSITY

### LIBRARY CASE...

3-ring binder album.  
Protects your valuable  
programs on disks  
Fully enclosed and  
protected on all sides.  
Similar to Kas-sette storage box.



Library 3-Ring Binder ..... \$6.50  
5 1/4" Mini Kas - sette/10 ..... \$2.49  
8" Kas-sette/10 ..... \$2.99

### DISKETTE DRIVE HEAD CLEANING KITS

Prevent head crashes and  
insure efficient, error-  
free operation.



5 1/4" or 8" ..... \$19.50

### SFD CASSETTES

C-10 Cassettes ..... 10/\$7  
(All cassettes include box & labels)

Get 8 cassettes, C-10 sonic and  
Cassette/8 library album for  
only ..... \$8.00  
(As illustrated)



### HARDHOLE

Reinforcing ring of  
tough mylar protects  
disk from damage



5 1/4" Applicator \$3    5 1/4" Hardholes \$6  
8" Applicator \$4    50/8" Hardholes \$8

VISA • MASTERCARGE • MONEY ORDERS  
CERTIFIED CHECK • FOR PERSONAL CHECKS  
ALLOW TWO WEEKS • C.O.D. REQUIRES A 10%  
DEPOSIT • CAL. RES. ADD 6% SALES TAX  
MIN \$2 SHIPPING & HANDLING • MINIMUM  
ORDER \$10 • SATISFACTION GUARANTEED  
OR FULL REFUND

Write for our free catalog

# ABM PRODUCTS

8868 Clairemont Mesa Blvd.  
San Diego, CA 92123

Toll Free

1-800-854-1555

For Orders Only

For information or California orders

(714) 268-3537

Listing 2: The drawLoopX:Y: method draws lines.

### drawLoopX: xDelta Y: yDelta

```
| dx dy px py p i |
< primitive >
dx ← xDelta sign.
dy ← yDelta sign.
px ← yDelta abs.
py ← xDelta abs.
self copyBits.           "first point"

py > px
ifTrue:                   "more horizontal"
  [p ← py/ 2.
   1 to: py do:
     [:i | destx ← destx + dx.
      (p ← p - px) < 0 ifTrue: [desty ← desty + dy. p ← p + py].
      self copyBits]]
ifFalse:                   "more vertical"
  [p ← px/ 2.
   1 to: px do:
     [:i | desty ← desty + dy.
      (p ← p - py) < 0 ifTrue: [destx ← destx + dx. p ← p + px].
      self copyBits]]
```

Listing 3: Methods for image magnification. @ is a shorthand message that returns a new Point whose x-value is the receiver (on the left) and whose y-value is the argument (on the right). Points respond to the + and \* messages by distributing them over each of the coordinates.

### magnify: rect by: scale spacing: spacing

```
| wideForm bigForm |
wideForm ← Form extent: (rect width * scale x) @ rect height.
wideForm spread: rect from: self by: scale x
  spacing: spacing x direction: 1 @ 0.
bigForm ← Form extent: rect extent * scale.
bigForm spread: wideForm asRectangle from: wideForm by: scale y
  spacing: spacing y direction: 0 @ 1.
↑ bigForm
```

"First expand horizontally"

"Then expand vertically"

### spread: rect from: sourceForm by: scale spacing: spacing direction: dir

```
| slice sourcePt |
slice ← Rectangle origin: 0 @ 0 extent: dir transpose * self extent + dir.
sourcePt ← rect origin.
1 to: (rect extent dot: dir) do:
  [:i |
    "slice up the original image"
    self copy: slice from: sourcePt in: sourceForm rule: STORing.
    sourcePt ← sourcePt + dir. slice moveBy: dir * scale].
1 to: scale - spacing - 1 do:
  [:i |
    "smear out the slices, leave some space"
    self copyAllTo: 1 @ 0 in: self rule: ORing]
```

"transpose returns a Point with swapped coordinates"

"dot product selects direction of stretch"

directly. Conversely, the Smalltalk text and graphics system requires implementation of only the one primitive operation to provide full functionality.

### Line Drawings, Image Synthesis

The same design principle applies in the support for drawing lines. By using BitBlt, one algorithm can draw lines of varying widths, different halftone "color," and any combina-

tion rule. To draw a line, an instance of BitBlt is initialized with the appropriate destination Form and clipping window, and with a source that can be any Form to be applied as a pen shape along the line. Starting from the stored destX and destY, the line-drawing loop, drawLoopX:Y: (listing 2), accepts x and y delta values and x and y step values as necessary, calling copyBits at each point along the line. The method used

**FREE**  
with software purchase —  
One year subscription to **InfoWorld**

**Ad#16**

**ULTIMATE SOFTWARE PLAN**  
✓ (New items or new prices)

# DISCOUNT SOFTWARE

We'll match any advertised price on any item that we carry. And if you find a lower price on what you bought within 30 days of buying it, just show us the ad and we'll refund the difference.  
It's that simple.

Combine our price protection with the availability of full professional support and our automatic update service and you have the Ultimate Software Plan.

It's a convenient, uncomplicated, logical way to get your software.

CP/M users: specify disk systems and formats. Most formats available.

## CP/M®

**ARTIFICIAL INTELLIGENCE**  
Medical (PAS-3) .....\$849/\$40  
Dental (PAS-3) .....\$849/\$40

**ASYST DESIGN**  
Prof Time Billing .....\$549/\$40  
General Subroutine .....\$269/\$30  
Application Utilities .....\$439/\$30

**COMPLETE BUS. SYSTEMS**  
Creator .....\$269/\$25  
Reporter .....\$169/\$20  
Both .....\$399/\$45

**COMPUTER CONTROL**  
Fabs (B-tree) .....\$159/\$20  
UltraSort II .....\$159/\$25

**COMPUTER PATHWAYS**  
Pearl (level 1) .....\$ 99/\$25  
Pearl (level 2) .....\$299/\$40  
✓ Pearl (level 3) .....\$549/\$50

**DIGITAL RESEARCH**  
CP/M 2.2  
NorthStar .....\$149/\$25  
TRS-80 Model II (P+T) .....\$159/\$35  
Micropolis .....\$169/\$25  
Cromemco .....\$189/\$25  
PL/I-80 .....\$459/\$35  
BT-80 .....\$179/\$25  
Mac .....\$ 85/\$15  
Sid .....\$ 65/\$15  
✓ Z-Sid .....\$ 90/\$15  
✓ Tex .....\$ 90/\$15  
DeSpool .....\$ 50/\$10

**D.M.A.**  
Ascom .....\$149/\$15  
DMA-DOS .....\$179/\$35  
CBS .....\$369/\$45  
Formula .....\$539/\$45

**GRAHAM-DORIAN**  
General Ledger .....\$729/\$40  
Acct Receivable .....\$729/\$40  
Acct Payable .....\$729/\$40  
Job Costing .....\$729/\$40  
Payroll II .....\$729/\$40  
Inventory II .....\$729/\$40  
Payroll .....\$493/\$40  
Inventory .....\$493/\$40  
Cash Register .....\$493/\$40  
Apartment Mgt. ....\$493/\$40  
Surveying .....\$729/\$40  
Medical .....\$729/\$40  
Dental .....\$729/\$40

**MICRO-AP**  
S-Basic .....\$269/\$25  
Selector IV .....\$469/\$35

**MICRO DATA BASE SYSTEMS**  
HDBS .....\$269/\$35  
MDBS .....\$795/\$40  
DRS or QRS or RTL .....\$269/\$35  
MDBS PKG .....\$1295/\$60

**MICROPRO**  
WordStar .....\$319/\$60  
Customization Notes .....\$ 89/\$25  
Mail-Merge .....\$109/\$35  
WordStar/Mail-Merge .....\$419/\$85  
DataStar .....\$249/\$60  
WordMaster .....\$119/\$40  
SuperSort I .....\$199/\$40

## MICROSOFT

Basic-80 .....\$289/\$30  
Basic Compiler .....\$329/\$30  
Fortran-80 .....\$349/\$30  
Cobol-80 .....\$574/\$30  
M-Sort .....\$124/\$30  
Macro-80 .....\$144/\$20  
Edit-80 .....\$ 84/\$20  
MuSimp/MuMath .....\$224/\$25  
MuLisp-80 .....\$174/\$20

## ORGANIC SOFTWARE

TextWriter III .....\$111/\$20  
DateBook II .....\$269/\$25  
Milestone .....\$269/\$25

## OSBORNE

General Ledger .....\$ 59/\$20  
Acct Rec/Acct Pay .....\$ 59/\$20  
Payroll w/Cost .....\$ 59/\$20  
All 3 .....\$129/\$60  
All 3 + CBASIC-2 .....\$199/\$75

## PEACHTREE\*

General Ledger .....\$399/\$40  
Acct Receivable .....\$399/\$40  
Acct Payable .....\$399/\$40  
Payroll .....\$399/\$40  
Inventory .....\$399/\$40  
Surveyor .....\$399/\$40  
Property Mgt. ....\$799/\$40  
CPA Client Write-up .....\$799/\$40  
Mailing Address .....\$349/\$40

## SOFTWARE WORKS

Adapt (CDOS to CP/M) .....\$ 69/\$na  
Ratfor .....\$ 86/\$na

## SOHO GROUP

MatchMaker .....\$ 97/\$20  
WorkSheet .....\$177/\$20

## STRUCTURED SYSTEMS

GL or AR or AP or Pay .....\$599/\$40  
Inventory Control .....\$599/\$40  
Magic Worksheet .....\$219/\$40  
Analyst .....\$199/\$25  
Letterlight .....\$179/\$25  
QSort .....\$ 89/\$20

## SUPERSOFT

Diagnostic I .....\$ 49/\$20  
Diagnostic II .....\$ 84/\$20  
Disk Doctor .....\$ 84/\$20  
Forth (8080 or Z80) .....\$149/\$25  
Fortran .....\$219/\$30  
Fortran w/Ratfor .....\$289/\$35  
Other .....less 10%

## TCS

GL or AR or AP or Pay .....\$ 79/\$25  
All 4 .....\$269/\$99

## UNICORN

Mince .....\$ 99/\$25  
Scribble .....\$ 99/\$25  
Both .....\$189/\$50  
Amethyst .....\$299/\$75

## WHITESMITHS

"C" Compiler .....\$600/\$30  
Pascal (incl "C") .....\$850/\$45

## "DATA BASE"

FMS-80 .....\$649/\$45  
dBASE II .....\$629/\$50  
Condor .....\$599/\$30  
Condor II .....\$899/\$50  
Access/80 .....\$749/\$50

## "PASCAL"

Pascal/MT+ .....\$429/\$30  
Pascal/Z .....\$349/\$30  
✓ Pascal/UCSD .....\$399/\$50  
Pascal/M .....\$149/\$20

## "WORD PROCESSING"

✓ WordSearch .....\$179/\$50  
SpellGuard .....\$229/\$25  
VTS/80 .....\$259/\$65  
Magic Wand .....\$289/\$45  
Spell Binder .....\$349/\$45

## "OTHER GOODIES"

The Last One .....\$549/\$95  
SuperCalc .....\$269/\$50  
Target .....\$189/\$30  
BSTAM .....\$149/\$15  
Tiny "C" .....\$ 89/\$50  
Tiny "C" Compiler .....\$229/\$50  
CBASIC-2 .....\$ 98/\$20  
Nevada Cobol .....\$129/\$25  
MicroStat .....\$224/\$20  
Vedit .....\$105/\$15  
ESQ-1 .....\$1349/\$50  
MiniModel .....\$449/\$50  
StatPak .....\$449/\$40  
Micro B+ .....\$229/\$20  
Raid .....\$224/\$35  
String/80 .....\$ 84/\$20  
String/80 (source) .....\$279/\$na

## APPLE II®

## INFO UNLIMITED

EasyWriter .....\$224  
Datedex .....\$349  
Other .....less 15%

## MICROSOFT

Softcard (Z-80 CP/M) .....\$259  
Fortran .....\$179  
Cobol .....\$499

## MICROPRO

Wordstar .....\$269  
MailMerge .....\$ 99  
Wordstar/MailMerge .....\$349  
SuperSort I .....\$159

## PERSONAL SOFTWARE

Visicalc .....\$ 99  
Visicalc II .....\$159  
CCA Data Mgr .....\$ 84  
Desktop/Plan II .....\$159  
Visiterm .....\$129  
Visidex .....\$159  
Visiplot .....\$149  
Visitrend/Visiplot .....\$229  
Zork .....\$ 34

## PEACHTREE\*

General Ledger .....\$224/\$40  
Acct Receivable .....\$224/\$40  
Acct Payable .....\$224/\$40  
Payroll .....\$224/\$40  
Inventory .....\$224/\$40

## "OTHER GOODIES"

✓ dBASE II .....\$329/\$50  
VU #3 (use w/Visicalc) .....\$ 49  
Super-Text II .....\$127  
Data Factory .....\$129  
DB Master .....\$184  
OEM (complete  
acting) .....\$399  
Charles Mann .....less 15%  
STC .....less 15%

is the Bresenham plotting algorithm (*IBM Systems Journal*, Volume 4, Number 1, 1965). It chooses a principal direction and maintains a variable,  $p$ . When  $p$ 's sign changes, it is time to move in the minor direction as well. This procedure is another natural unit to be implemented as a primitive, since the computation is trivial and the setup in *copyBits* is almost all constant from one invocation to the next.

## Image Processing

We have seen how BitBlt can copy shapes and, in the foregoing examples, how repeated invocation can synthesize more complex images such as text and lines. BitBlt is also useful in the manipulation of existing images. For example, text can be made to look bold by ORing over itself, shifted right by one pixel. Just as complex images can be built from simple ones, complex processing can be achieved by repeated application of simple operations. Here, we present three examples of such structural manipulation: magnification, rotation, and the game of Life. These examples were devised by the author in collaboration with Ted Kaehler.

As we shall see in the next two sections, many applications of BitBlt are very simple, such as filling a *Form* with white, or copying all of one *Form* to some location in another. Smalltalk provides for such casual use of BitBlt through a wide range of simple messages to class *Form*, such as:

```
someForm fillAll: white.
someForm copyAllTo:
    destLocation in: destForm.
```

We will not list all such messages here. In the examples that follow, the reader should be able to infer the meaning from the message names and the accompanying explanations.

## Magnification

It is often useful to magnify an image for closer scrutiny and especially to allow convenient alteration of stored *Forms*. Photo 1 shows this function providing user control over the font used for display of text.

ORDERS ONLY—CALL TOLL FREE VISA • MASTERCARD

1-800-854-2003 ext. 823 • Calif. 1-800-522-1500 ext. 823

Overseas—add \$10 plus additional postage • Add \$2.50 postage and handling per each item • California residents add 6% sales tax • Allow 2 weeks on checks. C.O.D. ok • Prices subject to change without notice. All items subject to availability • ® — Mfgs. Trademark.

## THE DISCOUNT SOFTWARE GROUP

6520 Selma Ave. Suite 309 • Los Angeles, Ca. 90028 • (213) 666-7677  
Int'l TELEX 499-0032 BVHL Attn: DiscSoft • USA TELEX 194-634 BVHL Attn: DiscSoft •  
TWX 910-321-3597 BVHL Attn: DiscSoft



**Listing 4:** The rotate method. This method rotates an image of size  $2^n$  by  $2^n$  one quarter-turn clockwise.

```
rotate | mask temp quad |
temp ← Form extent: self extent.
mask ← Form extent: self extent.      "set up the first mask"
mask copy: mask asRectangle halftone: white rule: STORing.
mask copy: mask asRectangle/2 halftone: black rule: STORing.
quad ← self width/2. "the size of a quadrant"
[quad >= 1] whileTrueDo:
[ "First exchange left and right halves"
mask copyAllTo: 0 @ 0 in: temp rule: STORing.
mask copyAllTo: 0 @ quad in: temp rule: ORing.
self copyAllTo: 0 @ 0 in: temp rule: ANDing.
temp copyAllTo: 0 @ 0 in: self rule: XORing.
temp copyAllFrom: quad @ 0 in: self rule: XORing.
self copyAllTo: (0 - quad) @ 0 in: self rule: ORing.
temp copyAllTo: quad @ 0 in: self rule: XORing.
"Then flip the diagonals"
self copyAllTo: 0 @ 0 in: temp rule: STORing.
temp copyAllFrom: quad @ quad in: self rule: XORing.
mask copyAllTo: 0 @ 0 in: temp rule: ANDing.
temp copyAllTo: 0 @ 0 in: self rule: XORing.
temp copyAllTo: quad @ quad in: self rule: XORing.
"Compute the next fine mask"
mask copyAllFrom: (quad/2) @ (quad/2) in: mask rule: ANDing.
mask copyAllTo: quad @ 0 in: mask rule: ORing.
mask copyAllTo: 0 @ quad in: mask rule: ORing.
quad ← quad/2]
```

The character for "7" has been presented magnified nine times. Using a pointing device, the user has blackened some cells to provide a European style "7," and the result can be seen in both the upper-left and lower-right windows on the screen.

A simple way to magnify a stored Form would be to copy it to a larger Form, making a big dot for every little dot in the original. For a height  $h$  and width  $w$ , this would take  $h \times w$

operations. The algorithm presented in listing 3 (as two messages to class Form) uses only a few more than  $h + w$  operations.

The magnification proceeds in two steps. First, it slices up the image into vertical strips in wideForm separated by a space equal to the magnification factor. These are then smeared, using the ORing function, over the intervening area to achieve the horizontal magnification. The process is then

repeated from wideForm into bigForm, with horizontal slices separated and smeared in the vertical direction, achieving the desired magnification. Figure 8 illustrates the progress of the above algorithm in producing the magnified "7" shown in photo 1.

## Rotation

Another useful operation on images is rotation by a multiple of 90 degrees. Rotation is often thought to be a fundamentally different operation from translation, and this point of view would dismiss the possibility of using BitBlt to rotate an image. However, the reader must consent that the first transformation shown in figure 9 is a step toward rotating the image shown: all that remains is to rotate the insides of the four cells that have been permuted. The remainder of the figure shows each of these cells being further subdivided, its cells being similarly permuted, and so on. Eventually each cell being considered contains only a single pixel. At this point, no further subdivision is required, and the image has been faithfully rotated!

Each transformation shown in figure 9 would appear to require successively greater amounts of computation, with the last one requiring several times more than  $h \times w$  operations. The tricky aspect of the algorithm below is to permute the subparts of every subdivided cell at once, thus performing the entire rotation in a constant times  $\log_2(h)$  operations. The parallel permutation of many cells is accomplished with the aid of two auxiliary Forms. The first, mask, carries a mask that selects the upper left quadrant of every cell; the second, temp, is used for temporary storage. A series of BitBlt operations exchanges the right and left halves of every cell, and then another series ex-

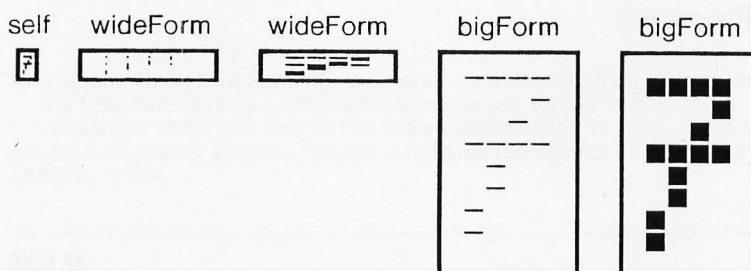


Figure 8: Magnification with BitBlt. See the text for more details.



Figure 9: Image rotation with BitBlt. See the text for more details.

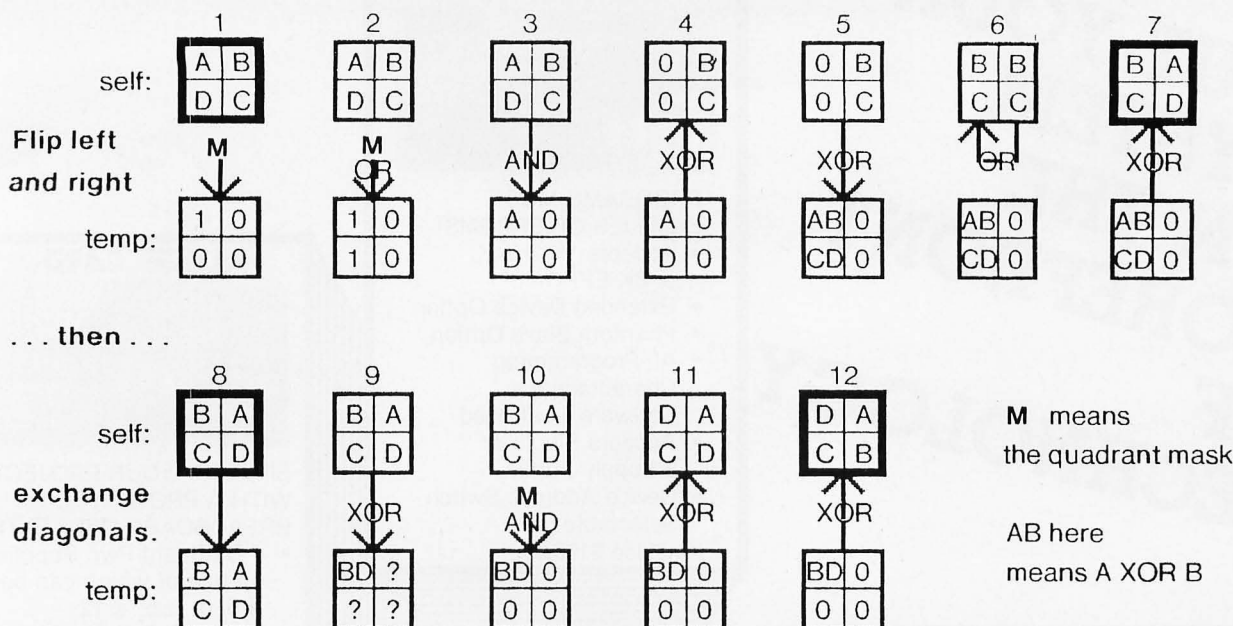


Figure 10: Permuting four quadrants of a cell.

changes the diagonal quadrants, achieving the desired permutation. The complete method for rotation is given in listing 4.

Figure 10 traces the state of temp and self after successive operations. The offsets of each operation are not shown, though they are given in the program listing. After twelve operations, the desired permutation has been achieved. At this point, the mask evolves to a finer grain, and the process is repeated for more, smaller cells. Figure 11 shows the evolution of the mask from the first to the second stage of refinement. The reader will note that the algorithm presented here for rotation is applicable only to square forms whose size is a power of two. The extension of this technique to arbitrary rectangles is more involved and is left as an exercise for the reader. A somewhat simpler exercise is to apply the above technique to horizontal and vertical reflections about the center of a rectangle.

## The Game of Life

John Conway's game of Life is

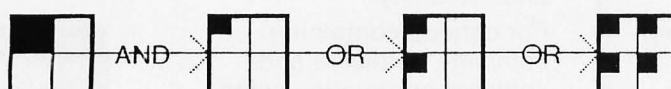


Figure 11: Refinement of the quadrant mask.

probably well known to readers of BYTE. It is a fairly simple rule for successive populations of a bitmap. The rule involves the neighbor count for each cell—how many of the eight adjacent cells are occupied? Each cell will be occupied in the next generation if it has exactly three neighbors, or if it was occupied and has exactly two neighbors. This is explained as follows: three neighboring organisms can give birth in an empty cell, and an existing organism will die of exposure with less than two neighbors or from overpopulation with more than three neighbors. Since BitBlt cannot add, it would seem to be of no use in this application. However, BitBlt's combination rules do include the rules for partial sum (XOR) and carry (AND). With some ingenuity and a fair amount of extra storage, the next generation of any size of bitmap can be computed using a constant number of BitBlt operations.

Listing 5 gives the method for nextLifeGeneration. As shown in figure 12, the number of neighbors is represented using three image planes for

the 1s bit, 2s bit, and 4s bit of the neighbor count in binary. The 8s bit can be ignored, since there are no survivors in that case, which is equivalent to zero (the result of ignoring the 8s bit). This Smalltalk method is somewhat wasteful, as it performs the full carry propagation for each new neighbor, even though nothing will propagate into the 4-plane until at least the fourth neighbor. Some readers may enjoy improving upon this algorithm.

Many other image-processing tasks can be performed with BitBlt. The author has built a complete optical character-recognition system for Sanskrit text using the various combination rules and an operation that counts the number of black bits in any rectangle (how would you do it?).

Bitmap processing is ideally suited to VLSI (very large scale integration) implementation. Readers who are interested in this direction should check the proceedings of the Design Automation Conference, June 1981, for "Parallel Bitmap Processor," by Tom Blank, Mark Stefik, and Willem vanCleeemput.

## Efficiency Considerations

Our original specification for BitBlt has been published elsewhere



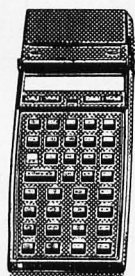
## SUPERBRAIN

By INTERTEC



64K Double or Quad Density units available. Uses two Z-80 CPU's. Commercial-type terminal with 12" monitor. Dual double density minifloppies. Over 350 kilobytes of storage (twice that with quad density drives). Two serial RS232 ports, I/O ports standard. Expandable with optional S-100 interface. Comes with CP/M™ 2.2 operating system. MiniMicroMart can supply a wide range of CP/M development and application software.

w/64K Double Density, List \$3495 ... **\$2869**  
w/64K Quad Density, List \$3995 ... **\$3395**



**HEWLETT  
PACKARD**  
**HP-41CV**  
**\$259.**



**data  
systems**  
Z19 Video Terminal



**Limited  
Time  
\$799**

List \$995



**HEWLETT  
PACKARD**  
**HP-85A**

Desk-Top  
Computer  
List  
\$3250  
**\$2749**



**HP-83**

List \$2250 **Special \$1749**

F.O.B. shipping point. All prices subject to change and all offers subject to withdrawal without notice. Advertised prices are for prepaid orders. Credit card and C.O.D. 2% higher. C.O.D. may require deposit.

— WRITE FOR FREE CATALOG —

**MiniMicroMart**

1618 James Street

Syracuse, NY 13203 (315) 422-4467

192 August 1981 © BYTE Publications Inc

**Listing 5:** The nextLifeGeneration method. This method calculates the next Life generation given the BitBlit bitmap of the current generation. See figure 12.

```
nextLifeGeneration | nbr1 nbr2 nbr4 carry2 carry4 |
nbr1 ← Form new extent: self extent + (2 @ 2).      "temp areas larger by 1"
nbr2 ← Form new extent: self extent + (2 @ 2).      "bit all around"
nbr4 ← Form new extent: self extent + (2 @ 2).
carry2 ← Form new extent: self extent + (2 @ 2).
carry4 ← Form new extent: self extent + (2 @ 2).
(1 @ 1) eightNeighbors do:
[:delta | "delta equals a different neighbor-offset each time through this loop"
carry2 copyAllFrom: 0 @ 0 in: nbr1 rule: STORing.
carry2 copyAllFrom: delta in: self rule: ANDing.      "carry into 2"
nbr1 copyAllFrom: delta in: self rule: XORing.      "sum 1"
nbr2 copyAllTo: 0 @ 0 in: carry4 rule: STORing.
carry2 copyAllTo: 0 @ 0 in: carry4 rule: ANDing.      "carry into 4"
carry2 copyAllTo: 0 @ 0 in: nbr2 rule: XORing.      "sum 2"
carry4 copyAllTo: 0 @ 0 in: nbr4 rule: XORing].      "sum 4"
nbr2 copyAllTo: 1 @ 1 in: self rule: ANDing. "perform logic to determine the survivors"
nbr2 copyAllTo: 0 @ 0 in: nbr1 rule: ANDing. "(2s AND self) OR (2s AND 1s)"
nbr1 copyAllTo: 1 @ 1 in: self rule: ORing. "all AND (NOT 4s)"
nbr4 copyAllTo: 0 @ 0 in: self rule: NOTANDing      "store next generation"
"over self"
```

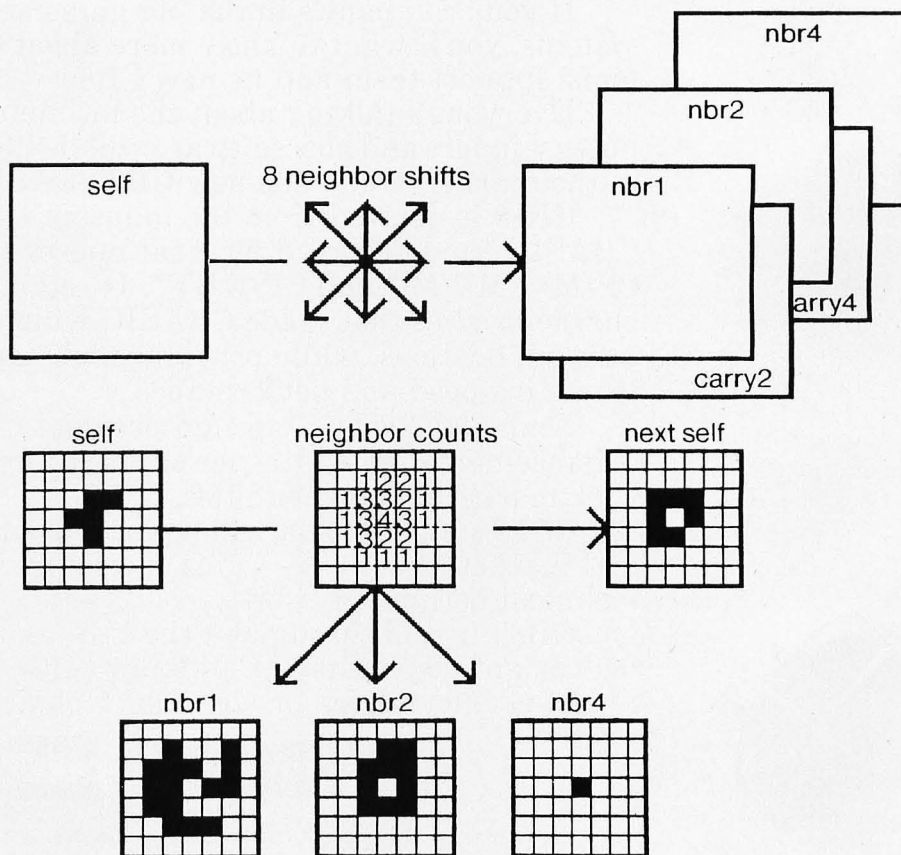


Figure 12: Counting neighbors in the game of Life.

(Newman and Sproull, *Principles of Interactive Computer Graphics*, 2nd edition, McGraw-Hill, 1979) under the name *RasterOp*. The implementation described in that reference can easily be extended to include the full set of combinations, and the addition

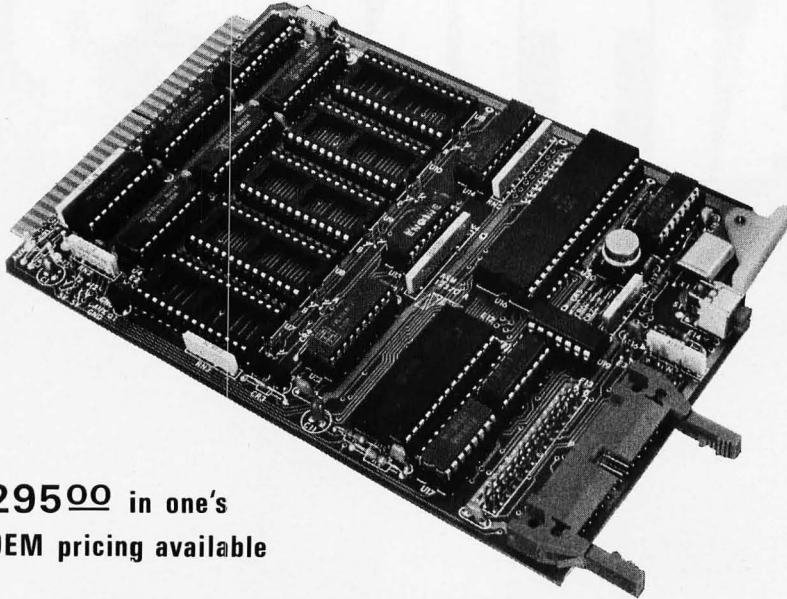
of clipping is also straightforward. Here, we add a few notes on efficiency gathered from experience.

BitBlit is so central to the user interface that any improvement in its performance has considerable effect on the interactive quality of the system

# STD BUS '6809'

The newest in Datricon's family of low-cost 4th engines.

**MEMORY** capacity to 40k bytes, **RS232/RS422 SERIAL PORT** and the powerful **6809 CPU**, all on one 4.5 x 6.5 inch **STD Bus Card**



**295<sup>00</sup>** in one's  
**OEM pricing available**

**These Specifications Tell You More!**

Every way you look at it, this powerful STD Bus Processor uses Industry-wide Standards.

**STD Bus** interface (both STD-Z80 and STD-6800 compatible) offers unprecedented user support with Analog, Power Input/Output, Disk and advanced communications protocols.

**SERIAL PORT** supports RS232C or RS422 with full modem controls including software baud rate, from 50 to 19.2Kbaud. User selectable standard since RCVR/DRVR's are factory installed.

**BYTE-WIDE MEMORY** concept permits the use of 20 currently available memory devices from 2k x 8, 4k x 8, and 8k x 8 RAM, ROM and EPROM.

## **QUALITY AND RELIABILITY**

Backed by Datricon's standard one year parts and labor warranty, 200 hour burn-in and extensive factory testing, our customers are assured of receiving high quality product.

## **D-FORTH SOFTWARE**

Datricon's popular D-FORTH software available on the Series 12 and 14 is also available on the Series 09. Optimized for control systems, D-FORTH is high-level and interactive, it is especially useful in interactive control applications such as testing and process monitor/control. Efficient memory utilization and rapid execution provide exceptional Return On Investment.

Contact Datricon's nationwide staff of highly qualified sales representatives or the factory for information.



**QUALITY WITHOUT COMPROMISE**

**503 - 284-8277      7911 NE 33RD Drive      Portland, Or 97211**

as a whole. In normal use of the Smalltalk-80 system, most calls on BitBlt are either in the extreme microscopic or macroscopic range. Let us examine these more closely.

In the macroscopic range, the width of transfer spans many words. The inner loop across a horizontal scan line gets executed many times, and the operations requested tend to be simple moves or constant stores. Examples of these are:

- Clearing a line of text to white
- Clearing an entire window to white
- Scrolling a block of text up or down

It is fortuitous that most processors provide a fast means for block moves and stores, and these can be made to serve the applications above. Suppose we structure the horizontal loop of BitBlt as the following sequence:

1. Move left partial word
2. Move many whole words (or none)
3. Move right partial word (or none)

Special cases can be provided for item 2 if the operation is a simple store or if it is a simple copy with no skew (horizontal bit offset) from source to destination. In this way, most macroscopic applications of BitBlt can be made fast, even on processors of modest power.

The microscopic range of BitBlt is characterized by a zero count for the inner loop in item 2, so that the work on each scanline involves, at most, two words. Both overall setup and vertical loop overhead can be considerably reduced for this case. Because characters tend to be less than a word wide and lines tend to be less than a word thick, nearly all text and line drawing fall into this category. A convenient way to provide such efficiency is to write a special case of BitBlt that assumes the microscopic parameters, but goes to the general BitBlt whenever these are not met. Because of the statistics (many small operations and a few very large ones), it does not hurt to pay the penalty of a false assumption on infrequent calls. ■