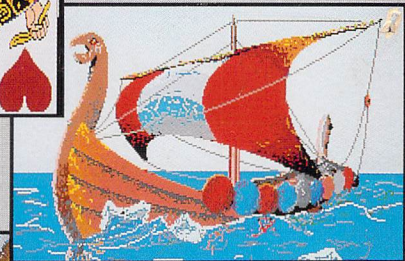
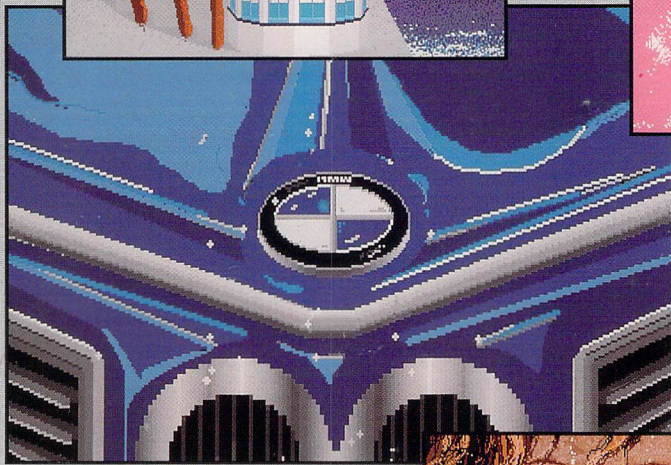
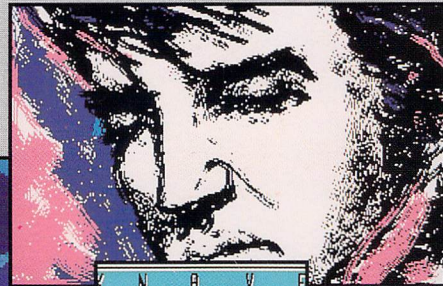
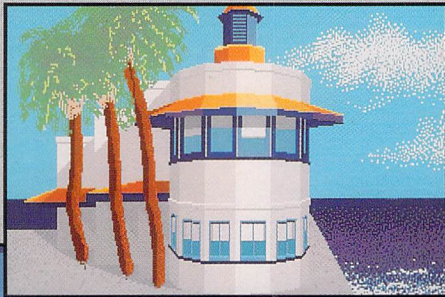


# BECOMING AN **AMIGA** ARTIST

*An illustrated guide to Amiga graphics, sound and animation.*

*Features an 8-page graphics gallery in full color.*



**Vahé Guzelimian**

**Norbert K. Kuhnert • Gia L. Rozells**

**SCOTT, FORESMAN AND COMPANY COMPUTER BOOKS**



# **Becoming an Amiga Artist**

**Vahé Guzelimian  
Norbert K. Kuhnert  
Gia L. Rozells**

**Scott, Foresman and Company**  
Glenview, Illinois • London

*To our parents.*

Color plates (with the exception of *Knave*) and Figures 1.2, 2.18, 2.19, 2.20, 2.21, and 2.22 reproduced by permission of The Savannah College of Art and Design. Figure 2.23 reproduced by permission of Jim Alley. *Knave* and figures 2.12, 2.13, 5.1, 5.2, 5.3, 5.4, 5.5, 5.6, 5.7, 5.8, 5.9, 5.13, 5.14, 5.17, and 5.18 reproduced by permission of Gene Brawn. Figures 5.12, 5.13, 5.16, 5.19, and 5.20 reproduced by permission of Electronic Arts.

Amiga, Kickstart, Workbench, AmigaBASIC, AmigaTutor, Graphicraft, and Musicraft are registered trademarks of Commodore-Amiga, Inc.  
Deluxe Paint, Deluxe Print, and Deluxe Video Construction Set are registered trademarks of Electronic Arts.  
Penmouse+ is a registered trademark of Kurta Corporation.  
Aegis Animator, Aegis Images, and Aegis Impact are registered trademarks of Aegis Development.  
FutureSound is a registered trademark of Applied Visions.  
The Music Studio is a registered trademark of Activision.  
Easyl is a registered trademark of Anakin Research, Inc.

#### **Library of Congress Cataloging-in-Publication Data**

Guzelimian, Vahé.

Becoming an Amiga artist.

Includes index.

1. Computer art—Technique. 2. Amiga (Computer)—Programming. I. Kuhnert, Norbert K. II. Rozells, Gia L. III. Title.

N7433.8.G89 1987 702'.8'566765 87-4532

1 2 3 4 5 6 RRC 92 91 90 89 88 87

ISBN 0-673-18527-3

Copyright © 1987 Vahé Guzelimian.

All Rights Reserved.

Printed in the United States of America.

#### **Notice of Liability**

The information in this book is distributed on an "As Is" basis, without warranty. Neither the author nor Scott, Foresman and Company shall have any liability to customer or any other person or entity with respect to any liability, loss, or damage caused or alleged to be caused directly or indirectly by the programs contained herein. This includes, but is not limited to, interruption of service, loss of data, loss of business or anticipatory profits, or consequential damages from the use of the programs.

Scott, Foresman Professional Publishing Group books are available for bulk sales at quantity discounts. For information, please contact the Marketing Manager, Professional Books, Professional Publishing Group, Scott, Foresman and Company, 1900 East Lake Avenue, Glenview, IL 60025.

# PREFACE

Whether you're an accomplished artist or a doodler, a business professional or a home finance manager, a kid or a grown-up, if there's an Amiga in your life, you probably can't keep your hands off it. *Becoming an Amiga Artist* will help you and your Amiga accomplish just what you want, as quickly and easily as possible.

The Amiga's exciting capabilities appeal to the creative spark in all kinds of people. Designers from four-year-old Mercedes McComb to graphic artist Andy Warhol have found the Amiga the ideal bridge from the vision of the mind's eye to the satisfied smile that accompanies the final product.

Whether you're programming your Amiga or using commercial software packages, working in an office, in a studio, or at home, *Becoming an Amiga Artist* is indispensable. In it you'll find the information you need to create outstanding graphics, sound, and animation. Its illustrated step-by-step instructions show how to use

- Deluxe Paint
- Impact!
- AegisDRAW
- Aegis Animator
- Deluxe Video

You'll learn how to begin and what software to buy or avoid. You'll learn advanced techniques, tips, and shortcuts not available anywhere else.

*Becoming an Amiga Artist* also teaches you, in simple English, how to program

- graphics
- sound
- speech
- animation

using AmigaBASIC. Sample programs—including waveform generators, flying spaceships, patterned graphics, and a speech experimenter—will develop your BASIC programming skills and help you apply them to produce your own exciting animation.

*Becoming an Amiga Artist* explains the use of digitizers, genlock devices, and MIDI interfaces. It tells how to transfer images from the screen to paper, film, and videotape. And, it describes how to create “desktop videos” using your Amiga, a videocassette recorder (VCR), and a video camera.

With detailed how-tos and a special chapter on using AmigaDOS for file management, *Becoming an Amiga Artist* is the most comprehensive, best-illustrated reference available. With it and your Amiga, you’ll be able to create artwork, advertisements, business reports, technical animations, and videos you’ve only imagined until now.

## **ACKNOWLEDGMENTS**

---

This book has profited from discussions we’ve had with enthusiastic Amiga users and developers across the country.

We offer special thanks to our friends at the San Diego Amiga Users Group; to Bill Gladstone for all of his help; and to the many software and hardware developers who responded to our numerous questions about their products. And, thanks also to Professor Jim Alley, the Savannah College of Art and Design (SCAD), and the SCAD students for their contributions.

G.R., N.K., and V.G.

---

## ILLUSTRATIONS AND PROGRAMS

---

All of the illustrations for this book were made on the Amiga computer by Gia Rozells and Norbert Kuhnert, except for the better ones which were done by Gene Brawn and the SCAD students. The programs were all written by Norbert.





# CONTENTS

<b>1</b>	<b>The Amiga—The Right Tool</b>	<b>1</b>
	The Mind Set Free	1
	The Amiga in Art	3
	The Amiga in CAD	4
	The Amiga in Music	4
	The Amiga in Business	4
	The Amiga in Education	4
	The Amiga in Science	5
	The Amiga in Entertainment	5
	The Integrated Brain	5
	The Basics	6
	Hardware	6
	Software	7
	Documentation	9
	Upgrades and Options	9
	You're on Your Way	9
<b>2</b>	<b>Painting Software</b>	<b>11</b>
	Comparing Painting and CAD programs	11
	Using Deluxe Paint	12
	Try the Examples	13
	Deluxe Paint Tutorials	13
	Creating a Flyer	13
	Creating Patterns	17
	Using Fill Patterns	18
		<b>vii</b>

---

Getting to Know Cycle Draw	22
Creating a Painting with Cycle Draw	24
Creating Three-Dimensional Objects	25
Creating an Underwater Scene	29
Advanced Examples	32
Tips and Techniques	36
Tracing Images onto the Screen	36
Graphics Tablets	36
Digitized Camera Images	38
Low, Medium, and High Resolution	39
Multitasking	39
Using Aegis Images	40
Using Graphicraft	40
Clip Art	40
Using Deluxe Print	41
Converting Deluxe Print “Group” Images to Deluxe Paint	42
<b>3</b> Business Graphics	<b>45</b>
The Alternatives	45
Software Options	45
Creating Charts and Graphs with Impact	46
Limitations	47
Producing a Sample Slide Show	47
Creating a Bar Chart	48
Creating a Pie Chart	52
Creating a Line Graph	53
Creating Icon Charts	55
Automatic Graphing with Icons	57
The No-Pest Chart	58
Creating a Slide Show	59
Presentation	60
<b>4</b> Aegisdraw—A Graphics Processor	<b>63</b>
Graphics Primitives	63
Drawing Strategies	64
Two Ways of Drawing a Cube	64
Drawing a 3D Diamond within a Box	65

---

Perspective Drawings	76
One-Point Perspectives	76
Scaled Drawings	80
Rearranging the Furniture in a Room	80

**5****Animation Software 87**

Creative Tools	87
Aegis Animator	94
Using Animator	95
Advanced Tips for Animator	106
Editing the Script	108
Deluxe Video Construction Set	114
Creating a Video Slate	115
Adding Color Bars to the Slate	127
Advanced Tips for Deluxe Video	128

**6****Programming Graphics in AmigaBASIC 131**

AmigaBASIC Graphics Skeleton Program	131
Skeleton Graphics Program	132
The AmigaBASIC Screen	132
The Default Screen	133
Using a Custom Screen	133
Defining a Window	134
Window Attributes	135
Screens	137
Windows	137
Color Resolution	138
Selecting Colors	139
Output Primitives	142
Drawing Points	142
Drawing Lines	144
Line Patterns	145
Drawing Shapes	148
Drawing Circles, Arcs, and Ellipses	150
Drawing Filled Areas	151
Filling with Patterns	155

Graphics Library Subroutines	156
Selected Kernal Graphics Routines	158
Opening a Font	158
Setting a Font	159
Closing a Font	159

## 7

## AmigaBASIC Animation 161

Blitter Objects (BOBS)	161
Sprites	162
Virtual Sprites	163
Sprites or Bobs: Which to Use?	163
Features of Bobs and Sprites	163
Creating an Object—Using the Object Editor	164
Loading the Object Data	166
Creating the Object within AmigaBASIC	166
Displaying Objects	167
Temporarily Removing Objects	167
Permanently Removing Objects	167
Selecting Object Colors	168
Bob Colors	170
Assigning Object Parameters	172
Positioning Objects	173
Determining Current x and y Object Positions	173
Moving Objects	173
Determining Current x and y Object Velocities	174
Accelerating Objects	174
Determining Other Object Functions	175
Detecting Collisions	175
Selective Collision Detection	177
Sample Program BOUNCE.BAS	178

## 8

## Sound Programming, Software, and Hardware 181

Digital Sound	181
Digitizing	182
Programming Sound with AmigaBASIC	183
The SOUND Command	185
The WAVE Command	187

---

	Sound Software and Hardware	195
	Hardware	195
	Software	196
<b>9</b>	<b>AmigaBASIC Speech</b>	<b>199</b>
	The SAY Command and the TRANSLATE\$ Function	199
	Voice Array Settings	200
<b>10</b>	<b>Video Applications</b>	<b>205</b>
	Using a VCR	205
	Video Digitizers	206
	The Technology	206
	Using Digi-View	208
	Using Genlock	209
	Using Amiga Genlock 1300	210
	Other Genlocks	211
<b>11</b>	<b>File Management</b>	<b>213</b>
	The CLI	213
	Opening the CLI	213
	Data Disks	214
	Files	214
	Creating Drawers	216
	Other AmigaDOS File Management Commands	218
	File Protection	220
	Backing up Entire Disks	220
	Creating a RAM Disk	221
<b>12</b>	<b>Screen Reproduction Techniques</b>	<b>223</b>
	Printers	223
	Shopping for a Printer	227
	Plotters	228

**xii      Becoming an Amiga Artist**

---

Screen Shots	229
Film Recorders	230
Straight-Off-The-Disk Slides and Separations	230
Slideshows	230
VCRs	231
<b>Appendix</b>	<b>233</b>
Vendor Reference	233

# **The Amiga— The Right Tool**

The Amiga computer's hardware and software design make it the right tool for graphics and sound applications. It is extremely powerful and flexible, and it preserves the freedom essential to artistic creativity. For the first time, you can combine music, animation, speech, and video into the kind of product only specialists could create before.

The Amiga is a powerhouse. It has speed, multitasking ability, and color graphics incomparable in its price range. With it a small business can compete with much larger firms, and an artist can produce intricate designs quickly and efficiently.

The Amiga is not just a graphics computer or just a number cruncher. With the Amiga you can run your spreadsheets and be an artist and designer on one machine, even all at one time. It provides new dimensions of fun and practical graphics (Figure 1.1).

---

## **THE MIND SET FREE**

Traditionally, computer operators had to know a lot about developing and using extensive lists of arcane commands. But learning, remembering, and typing the commands can be major chores that continuously

**FIG. 1.1** The Amiga monitor, central processing unit, keyboard, and mouse.



occupy an operator's mind and interfere with the concentration required for creativity. So the Amiga uses icons (cleverly designed little pictures and symbols) to present choices to the user. The user can recognize, intuitively, what the icons mean. Tasks that traditionally required entire command sequences are reduced to the single click of a mouse button. This frees the user to be creative and productive.



How will you use your Amiga? You may have a specific application in mind or several projects in the planning. Chances are you'll discover many uses that you haven't even considered yet. The following sections describe just a few of the places where the Amiga is right at home.

## The Amiga in Art

The Amiga's graphics power and 4,096 colors provide homes, schools, and businesses with a portable art studio, a portable print shop, and even a portable photographic studio and darkroom. Artists such as Andy Warhol and the students of the Savannah College of Art and Design (SCAD) are using the Amiga's power to break new ground in the visual arts (Figure 1.2).

FIG. 1.2 A sample of the graphics you can produce on the Amiga. Painting by Robynne Engel of SCAD.



## **The Amiga in CAD**

Computer-aided design (CAD) benefits from the Amiga's power. The computer eliminates the need for rulers, pencils, and compasses in making scaled drawings of floor plans, mechanical parts, and furniture, for example.

## **The Amiga in Music**

With its four-channel stereo sound the Amiga is a musical instrument all by itself. The sounds it produces compete with those made by dedicated synthesizers, and it's a powerful, programmable digital sampler.

Musicians are changing the way they produce music by incorporating the Amiga into their studios for sequencing, sampling, and writing music. With available software and hardware, you, too, can digitize sounds, write musical scores, and simulate instruments with your Amiga—even if you've never played a note before.

## **The Amiga in Business**

The Amiga is a tremendous addition to an office. Its multitasking ability, speed, and expandable memory make it a workhorse for spreadsheets, word processing, graphics creation, and database applications. And its iconic interface eliminates the need for costly, time-consuming training sessions. Specialized businesses such as fashion design, advertising, and architecture have already discovered the Amiga's potential.

## **The Amiga in Education**

The Amiga's educational potential is almost limitless. Typing programs teach beginners to use the keyboard. The software responds to the user's mistakes and successes. Foreign language tutorials and talking word processors take advantage of the Amiga's advanced speech capabilities. Programs on topics including space exploration, natural history, and geography are unlocking young minds by providing an interactive environment that's exciting and fun. High-level language support in AmigaBASIC, C, Pascal, FORTRAN, and FORTH provides an excellent opportunity for you to learn computer programming.

---

## The Amiga in Science and Engineering

Scientists and engineers are using the Amiga's animation, color, and digitizing capabilities to model real-world phenomena. A scientist at the Scripps Institution of Oceanography in La Jolla, California, uses the Amiga's digitizing capabilities to study aquatic vegetation distribution in the ocean. He also uses BASIC programs to model the thermal convection and diffusion of nuclear power plant cooling water.

## The Amiga in Entertainment

The Amiga is a natural entertainer. Its high-speed animation capabilities and four-channel sound production make it a knockout games machine. The Amiga has one processor just for animation and a chip devoted solely to sound production, so when you play one-on-one basketball on the Amiga, the players move with speed and skill and you hear the crowd, the bounce of the ball, and the buzzers. Much like the games in a video arcade, the Amiga's flight simulators put you in control of a speeding jet in aerial combat, and its tank wars pit you against obstacles and enemies on the ground. If you're interested in more cerebral pursuits, play chess against the Amiga. The challenges are endless.

## THE INTEGRATED BRAIN

---

How can one machine do so much so well? A look inside the Amiga reveals the secret. Most computers have only one processor, but the Amiga uses *three*.

The Motorola 68000 microprocessor serves as the central processing unit (CPU), which controls all other processes in the computer. It is well known for both its speed and its reliability. A 6500 microprocessor is used just to read the keyboard. And the graphics coprocessor adds support by relieving the CPU of many graphics management chores that would otherwise cost time. This gives the Amiga exceptional speed for animation.

Besides the three microprocessors, two very specialized chips provide speech and full four-channel stereophonic digital sound.

The CPU acts as the “boss,” performing many major processes on its own, but distributing the tasks that the other microprocessors and specialized chips can perform more efficiently.

## **THE BASICS**

---

You don't need to understand all the inner workings of the Amiga to appreciate or to use it. The CPU and iconic interface take care of almost everything for you. A basic familiarity with the main components of your machine will come in handy, however, and give you a working Amiga vocabulary. The following sections describe the hardware and software you have if you own or use the Amiga in its basic configuration.

### **Hardware**

**THE MONITOR.** The Amiga monitor features full color and a 13-inch screen suitable for creating highly detailed, multicolored pictures. The monitor supports three video operating modes:

- *Analog Mode.* This RGB (red, green, blue) mode is selected by pushing the switch on the back of the monitor to the far right. Analog mode is the only mode utilized by the Amiga. This mode provides 80-column  $\times$  25-line full-color capability. The maximum resolution is 640  $\times$  400 pixels (dots). A palette of 32 colors can be chosen from the 4,096 colors available.
- *Digital Modes (positive and negative).* These modes make it possible for you to use the Amiga monitor on personal computer systems, such as the IBM PC, that use the digital mode.

Adjustment controls for horizontal position, vertical hold, color, tint, brightness, contrast, and volume are hidden under the panel on the front. Adjust the various levels to suit your needs (depending on whether you're working in a bright room, with high-resolution, etc.), but remember that turning down the brightness extends the monitor's life.

**THE MAIN UNIT.** The CPU is the controller for the Amiga. It has 256k bytes of random access memory (RAM), with another 256k bytes of user-transparent memory that stores Intuition, the Amiga's operating

---

system. You can expand the RAM to 512k bytes by simply opening the cover on the front and slipping in a 256k RAM card. You'll probably want to do this since more than half of the software programs available require at least 512k.

The disk drive is on the right side of the main unit. It uses 3.5-inch 880k microdisks. The disks are inserted label side up, metal end first. To remove a disk, press the eject button just beneath the drive. When the Amiga is reading from or writing to a disk, the disk drive light goes on. Don't try to eject a disk while the light is on; you may lose data from the disk.

On the left panel of the main unit is the on/off switch. The back and the right side have 11 ports for plugging in various peripherals, such as a printer, joystick, mouse, trackball, graphics tablet, VCR, genlock device, video digitizer, modem, or second disk drive. Both serial and parallel ports are supported, as well as a full 68000 bus expansion.

**THE KEYBOARD.** The Amiga's 89-key detached keyboard uses 10 function keys and a numeric keypad. The *Amiga User Guides* explain how to use the various function and control keys.

**THE MOUSE.** The mouse makes it easy to operate the Amiga. Moving the mouse on a desktop causes a pointer to move on the screen. You can use the mouse and the screen icons to make choices quickly. And the mouse works as a versatile painting and drawing tool in graphics programs. The *Amiga User's Guide* describes the powers of this small but valuable controller.

## Software

In addition to the Amiga's hardware, the basic configuration also includes software labeled Kickstart, Workbench, and Amiga Extras.

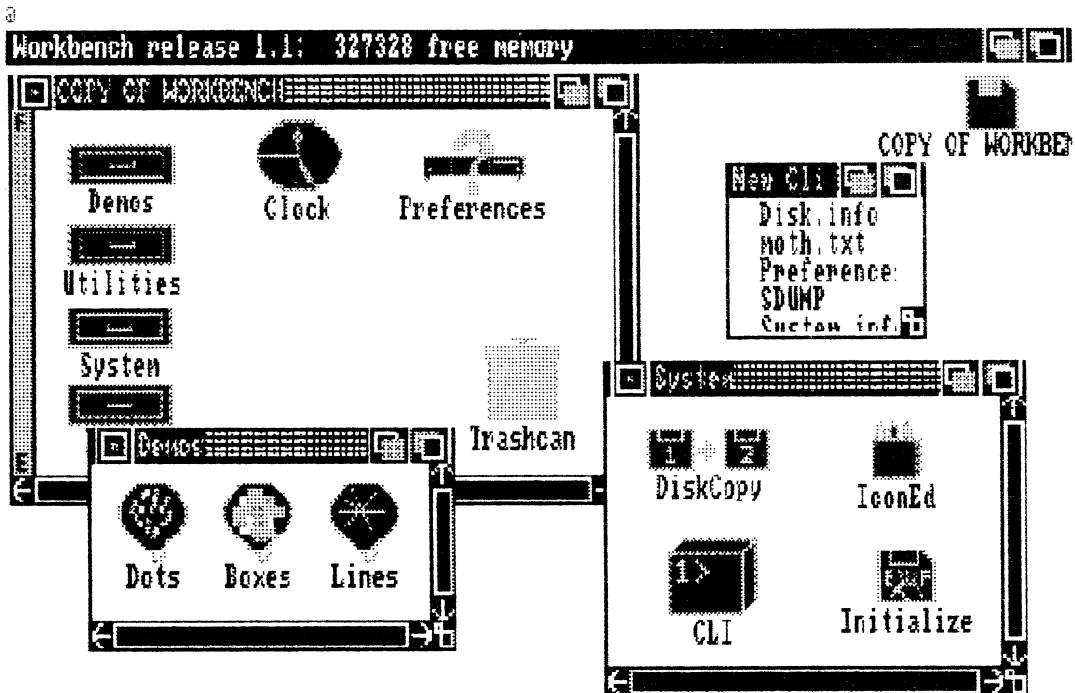
**THE KICKSTART DISK.** The Kickstart disk contains Intuition, the Amiga's operating system. Kickstart must be loaded before you can operate the computer. Rather than storing Kickstart on ROM chips, Commodore-Amiga stores it on disk so that when they upgrade the operating system you'll be able to exchange your current disk for the new version. When you turn on the Amiga CPU, you'll hear a start-up tune and the screen will show a hand holding a Kickstart disk. This is a request for you to put the Kickstart disk into the drive. Wait while the disk drive

hums (the Amiga is reading the disk), and when the red light goes off, the machine will prompt you with another hand requesting the Workbench disk.

**THE WORKBENCH DISK.** This disk provides the tools for most of the Amiga's file managing functions. Using the mouse and the icons you can create files, retrieve files, copy disks, and so on. Commercially available software has the Workbench file on the disk, so you don't always have to insert your Workbench disk to use the software. With a working environment composed of icons such as the pictures of trash cans and disks shown in Figure 1.3, the Amiga is very easy to use.

**THE AMIGA EXTRAS DISK.** The Extras disk contains AmigaBASIC, Amiga Tutor, and some interesting demo programs. To use the disk, insert it after Kickstart and Workbench.

FIG. 1.3 The Workbench screen and multiple windows.



---

## Documentation

Two manuals are included in the binder labeled *Amiga User Guides*. The first is the *Introduction to Amiga*, which contains documentation on everything from setting up the Amiga to using Workbench, adding peripherals, and using AmigaDOS. The second, *Amiga Basic*, tells how to use the AmigaBASIC language by Microsoft. It provides a full tutorial to get you started as well as a complete list of commands.

---

## UPGRADES AND OPTIONS

As you read this book, you'll learn about many peripherals you might want to attach to the Amiga for specialized purposes. Some of the possible upgrades and options are up to 8 megabytes of RAM; 300-, 1200-, and 2400-baud modems; a genlock interface; color digitizers; musical keyboards that attach via a MIDI interface; hard disk drives; multifunction cards; printers; plotters; joysticks, and graphics tablets.

The software options are equally numerous and rapidly expanding. Commodore-Amiga has many offerings, and hundreds of outstanding products have been provided by independent developers inspired by the groundbreaking capabilities of the Amiga. Commodore-Amiga has encouraged these developers to produce high-quality products by making the Amiga a product ready for expansion.

---

## YOU'RE ON YOUR WAY

Now you have an idea of what you've got: a desktop computer with huge potential for productivity and creativity in just about any area you can think of. Whether your business is commercial landscape design, research analysis, medical illustration, graphic art, technical documentation, engineering, hairstyling, fashion design, teaching, advertising, or child's play, this book will help you discover many novel ways to put your Amiga to use. You'll learn basic methods and advanced tips and shortcuts to save you time.

Almost anything you can design with pencil, brush, or camera can be designed with the Amiga. From brochures, manuals, and paintings to maps, tools, and cars. The fun starts in Chapter 2—get ready to paint with the Amiga's 4,096 colors.





# Painting Software

This chapter focuses on painting programs, the creative tools for such diverse pursuits as art, advertising, and making stationery or greeting cards. Chapter 4 discusses computer-aided design (CAD) programs used typically for architectural drawings, drafting, construction designs, and parts modeling. If you're wondering which type of program suits your needs, the following specific comparison will help.

---

### COMPARING PAINTING AND CAD PROGRAMS

---

Painting programs make it simple to draw freehand pictures on the screen. You control the color of each individual pixel (dot) that appears on the screen, and you have the freedom to create simple or detailed shapes, in any of 4,096 colors, anywhere on the screen.

You cannot draw precisely measured objects such as floor plans, however, without creating a grid and holding a ruler up to the screen. Also, the freedom of addressing each pixel means that objects created with these dots can't be picked up and moved separately from other objects or background. That is, a painting program can identify only the individual pixels, not the objects they form. CAD programs, on the other hand, identify every object in a drawing independently of other objects and the background. Each object can be manipulated separately.

- You can paint on the screen more quickly.
- The program monitors the screen contents pixel by pixel, not object by object.
- You have more flexibility in color variation and subtlety of images.

CAD programs are object-oriented graphics editors. If you draw a circle or a house, the program remembers it as a distinct object that can be moved, saved, and sized independently of the rest of the page.

With CAD programs you can do some freehand drawing, but their strong points are the tools they offer for drawing precisely to scale using grids and coordinates. In short, with CAD programs

- The components of the drawing have identities.
- The objects within a drawing can be edited.
- Coordinate systems are built into the program or specified by the user available to create scaled drawings.
- You can perform object transformations (translations, scaling, and rotations).

Note that Chapter 3 discusses a program that falls between the painting and CAD programs. It is designed especially for creating professional quality business charts and graphs.

If you flip through the pages of Chapters 2, 3, and 4, the pictures will give you a better idea of the applications for painting, CAD, and intermediate programs.

## **USING DELUXE PAINT**

---

Several powerful, exciting painting programs make it possible for artists and those who can't even draw a stick figure to create sophisticated artwork for personal, artistic, or business purposes. This chapter focuses on Deluxe Paint because it is the most flexible and widely used painting program. Other terrific programs are listed at the end of this chapter. Whichever you choose, start out by reading its manual. This will introduce you to the basic features of the program and will get your creative juices going. Then follow the step-by-step examples in this book (using Deluxe Paint) to learn useful advanced concepts and to get advice from experienced Amiga artists.

---

Deluxe Paint, by Electronic Arts, has features to suit everyone from children to accomplished graphic artists. The following pages introduce you to some advanced techniques and tips not mentioned in the manual.

As you experiment, be sure to try using some of the keyboard commands listed here and in the manual; they'll increase your speed after you've mastered the mouse. Also, before you start, be sure to format a blank disk according to instructions given in Chapter 11 or in the *Amiga User's Guide*, so you can save the pictures you create.

## Try the Examples and Take It from There

When you've experimented with all the examples in this chapter, you're sure to find hundreds of new combinations and uses for the techniques. And you'll be prepared to put them to work.

## DELUXE PAINT TUTORIALS

---

The following introductory paintings use a variety of important techniques. They're all made in low resolution mode using Deluxe Paint version 1.0. (You'll learn more about what lo-res is later in the chapter.) The techniques will apply to later versions of the program and, artistically, to other programs with similar tools.

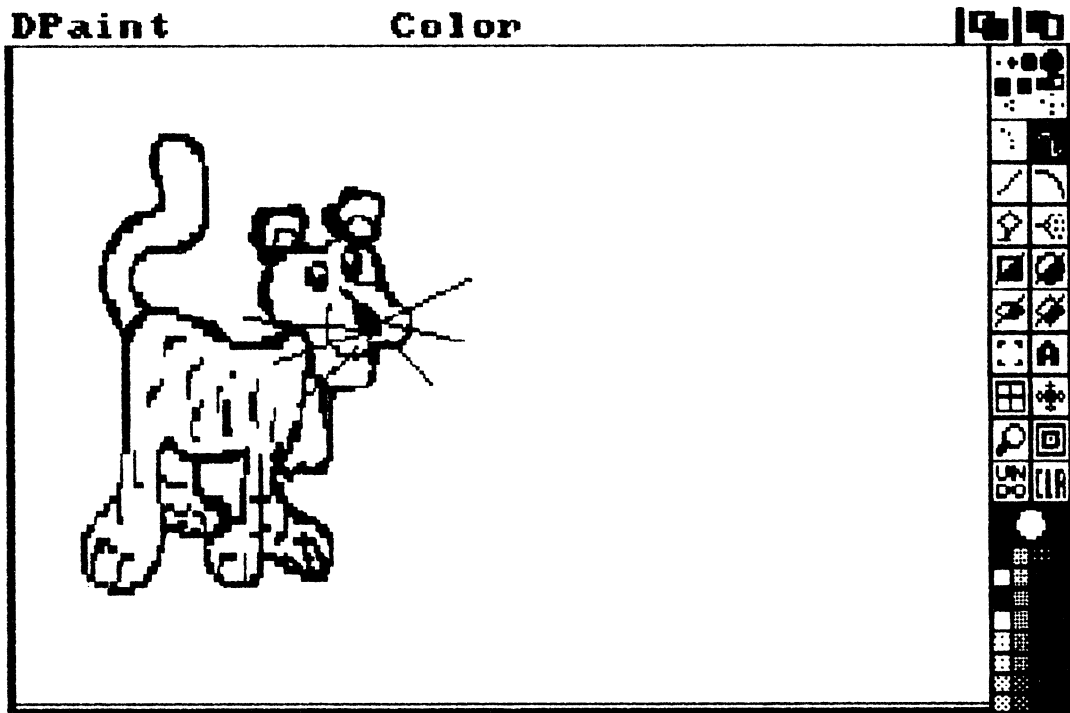
### Creating a Flyer

The "Missing Cat" flyer shows how Deluxe Paint can bring a simple everyday sign to life.

**DRAWING THE CAT.** Use a small brush (1 to 2 × 2 pixels) to sketch your cat on the screen. It could be detailed like the example in Figure 2.1, or simply made out of circles. Or, you could trace a picture from a book onto a piece of transparent acetate, tape it to your screen, and outline it with your mouse. (See "Tips and Techniques" later in this chapter for more on this method.) Give your cat some shading and features.

**USING FILL.** Select a color and use *Fill* to color your cat. If the paint leaks out of the figure into the surrounding area, use *Undo* to clean

FIG. 2.1 Drawing the cat.



it up. Then examine the outline of the cat using *Magnify* if necessary, to find the “leak” (a break in the outline), and repaint the leaky spot (see Figure 2.2).

If you have trouble pinpointing a leak when doing a fill, watch the horizontal progress (up and down the screen) of the fill and note where the color first appears outside of the fill boundary. Then search in this general area, and you’ll find the leak.

**MOVING THE CAT.** To place your cat on a different part of the page, pick it up as a brush, using the *Brush selection* tool, *Clear* the screen, and click the cat down wherever you like. If you want to resize the cat, try using the commands in the *Brush* menu to *Stretch*, *Double*, or *Halve* its size. If you stretch the figure, the outline may become a little jagged. Smooth it out by going over the outline with a small brush.

**ADDING THE TEXT.** Choose *Load Fonts* from the *Font* menu and select a typestyle. Select the *Text* tool. Use the mouse to place the box on

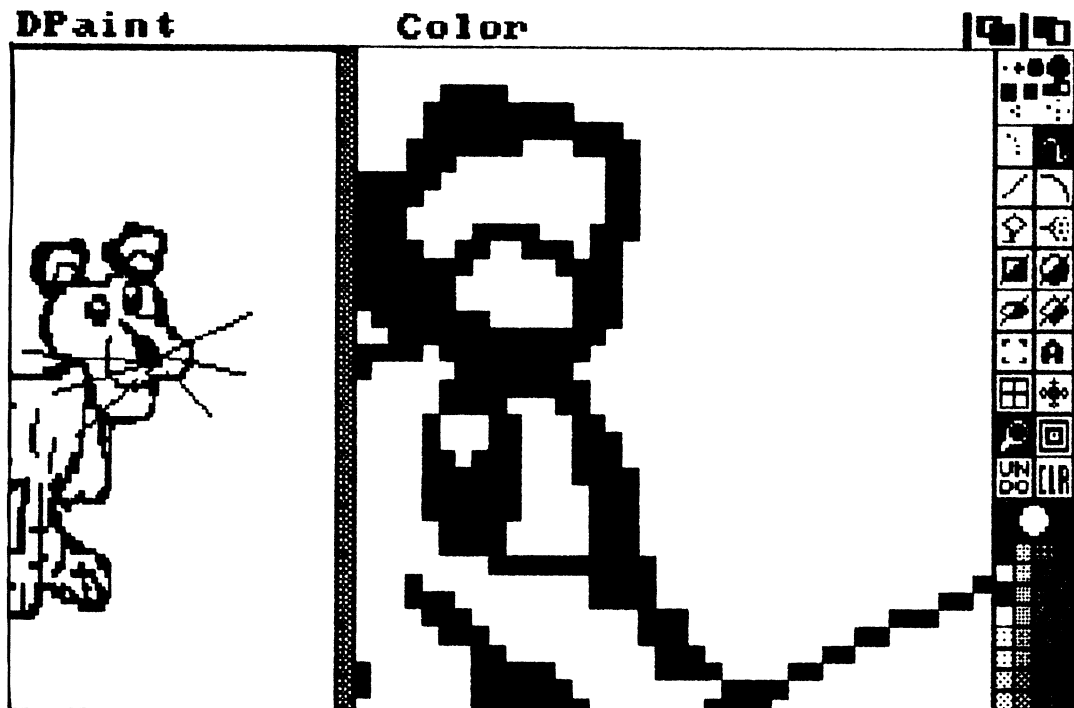
the screen where you'd like the text to start. Typing on the keyboard will enter text on the screen. Backspace to erase, or erase with a brush using the background color.

It's usually easier to type your text on an open area of the current or spare screen. (You'll learn more about using the spare screen in the next example.) Then pick the text up as a brush for exact positioning. Unless you want the box around the word, be sure to use *Pick* (click right button on the center dot of the foreground/background color indicator) to set the background color before you pick the text up.

Also, centering is easy. Any time you pick up an object as a brush, it's held at its center point. Just use the coordinate display from the *Preferences* menu to reference the placement point. For example, to center an object horizontally on the screen (that's 320 pixels/2) position the brush so that the x coordinate reads 160. Then release. Presto.

**ENLARGING THE TEXT.** To enlarge the word "Missing," pick it up as a brush and resize it using the *Stretch* item under *Size* in the *Brush*

FIG. 2.2 Using *Magnify*.



menu. Then click it down in the correct place. Remember, text is just like any other object on the screen—you can resize it, erase, or fill it with other colors (Figure 2.3).

When using the *Size* tool, experiment by using it with the *Shift/Hold Z* with the *Grid* tool turned on.

**ADDING A BORDER.** To put a solid border around your picture, choose a large brush (square or round) and select the *Outline Rectangle* tool. Press the F10 key. The menu bars will disappear. Move the cursor to the upper lefthand corner of the screen and a box to the lower righthand corner; release. If any of the edges of the screen aren't filled with the proper background color, *Fill* them in to match. Press F10 again to get the menu bars back.

FIG. 2.3 The Missing Cat poster.

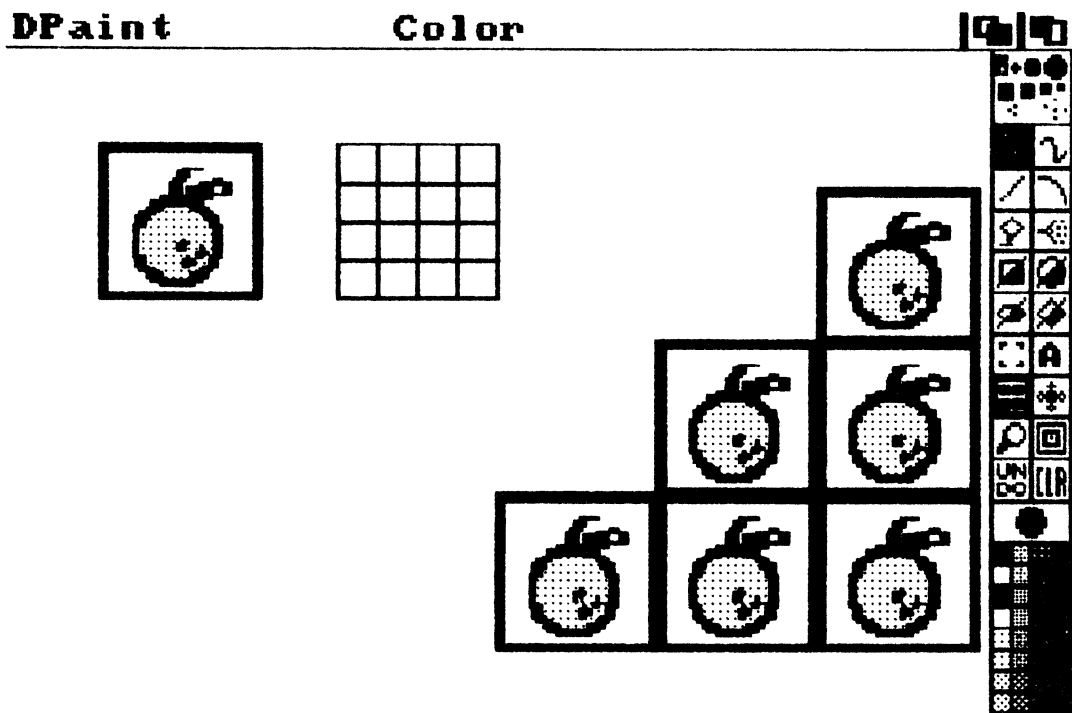


## Creating Patterns

You've used the *Fill* command to add solid color to large spaces. What if you want to fill such spaces with a pattern instead? For instance, if you drew a living room scene, you could use *Fill* to make patterned wallpaper. Here's a way to do that using the *Grid* tool, *Fill*, and the *Spare* screen. The instructions here are for systems with 512k memory. If you have 256k, you can adapt them by using the special instructions at the end of the example.

**CREATING A BORDER PATTERN.** Think of the pattern as a screen covered with repeating patterned tiles. In Figure 2.4, the pattern piece is the tile with an orange on it. You can create block patterns like floor tiles,

**FIG. 2.4** The original orange pattern piece (left), the grid (middle), the pattern painted six times with the grid turned on.



intricate filigree designs, or textures for wallpapers, borders, or tree bark. Draw a piece of the pattern you'd like to use anywhere on your screen.

**SIZING THE GRID.** Click on the *Grid* tool with the right mouse button. A small tic-tac-toe outline will appear. You can adjust its size by dragging the mouse while you hold either button down. Position the top left corner of the grid over the top left corner of your pattern piece. Click and drag until the grid covers the whole pattern. Then release.

You've made a screen custom filled to the size of your pattern. The invisible grid will remain, constraining your drawing so that tile corners will automatically fall at grid intersections. Figure 2.3 shows how the size of the grid equals the size of the pattern piece. If you have any trouble sizing the grid, the automatic x-y display of the grid size in the menu bar will help. Also, keep in mind that the x and y sizes include the right and bottom borders of one of the grid rectangles.

**REPEATING THE PATTERN.** Pick up your pattern piece as a brush. Paint it across the page. It will align itself with the grid intersections to create a patterned area.

Note that if the pattern is detailed, when you put the pattern pieces side by side the edges may not align exactly. The grid will create a one-pixel overlap on all sides. Plan for this by creating your pattern one pixel larger than you need it. The orange pattern was stamped six times with the *Grid* function turned on.

## Using Fill Patterns

**FILLING OBJECTS.** Clear the screen. Draw an outline of a woman in a kimono and a house (Figure 2.5). You'll fill in the kimono with a print and the house with a thatched pattern. Since you don't want to fill her face and hands, the umbrella, or the surrounding space with these patterns, mask these areas by filling them in with a color you won't be using anywhere else (Figure 2.6). Leave the spaces you want filled, the kimono and the house, filled with the background color. (Or switch the palette's foreground and background colors.)

**USING THE SPARE SCREEN.** Use the *Spare* option from the *Picture* menu and choose the submenu item *Swap* to go to the blank (*scratchpad*) screen. A shortcut for this is to press the *J* key.



FIG. 2.5 The outlined Kimono drawing.



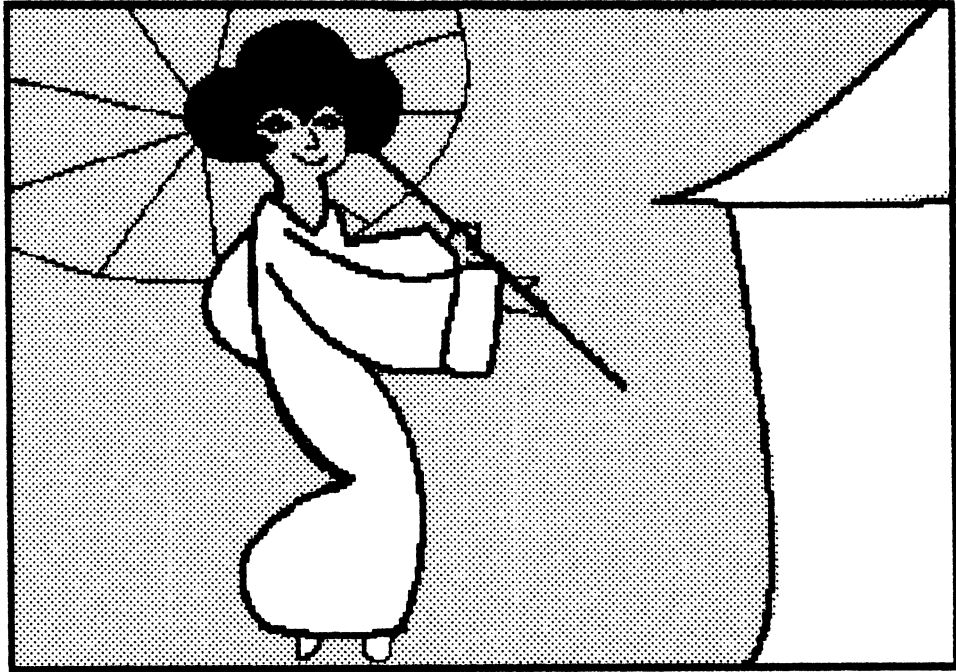
**CREATING THE PATTERNS.** Use the pattern method described in the previous example to create a print for the kimono print on half the screen and a thatched pattern on the other half (Figure 2.7).

Swap back to the house screen. Go to the *Spare* option again, and this time choose the *Merge-in-back* submenu item. Since the kimono and the house are filled with the background color, when you merge the screen they will fill with the patterns (Figure 2.8).

If the art is complicated, or a color you're using as a background color appears in another part of the image that you don't want to fill with the pattern, use the *J* key to flip back and forth between the screens as you erase the area from the pattern screen that shouldn't appear on the main screen.

Note that the brush modes (freehand dot/line, straight line, and rubber band) are independent of the screen display. That is, you can start drawing a line on the main screen, swap screens, and still be drawing the same line. Try this: assume that you're erasing some very fine detail from

FIG. 2.6 Mask the areas to remain unpatterned.



one drawing in order to fill it with something else from the spare screen (using *merge-in-back*).

You'll need to set up your two screens so the two images are positioned at the same xy coordinates. Set up the first screen normally. Use the *J* key to swap screens and pick up the image from the spare screen as a brush. Swap to the main screen with the *J* key, and position the brush exactly where you want it to merge using the main screen's image as a guide. Without moving the mouse, use the *J* key to swap screens, and stamp the brush into position.

Similarly, you can use this technique to start drawing a straight line on one screen and, still holding the mouse button, flip to the other screen to check its position. Then flip back and release the button to draw the line. This is a very useful technique.

**FINISHING THE PICTURE.** Now you can add fresh colors to the woman's hair, face, and hands, and the umbrella and background. Also, check the upper section of the screen under the menu bars by pressing

F10. The menu bars will disappear, and you can make sure the umbrella and house roof reach to the edges of the screen. If they don't, draw them in as needed. This upper section will show when the picture is printed.

**SPECIAL INSTRUCTIONS FOR 256k.** With 256k you can draw your patterns in an empty corner of your picture, cut off small pieces of them with the *Brush selection tool*, to fill in the kimono and house, and then clean up the corner. This method can be difficult with intricate scenes, however, so try the following method, too.

**FOR 256k OR INTRICATE PAINTINGS IN 512k.** If you're planning a complex picture, with small sections of fill pattern, you can follow these steps: sketch the objects, fill them in with patterns, cut them out as brushes, clean up any messy sections, and Save them on a data disk as brushes. Later, when you've completed the rest of the drawing, paste them into the painting in the appropriate places.

FIG. 2.7 Patterns made on the Spare screen.

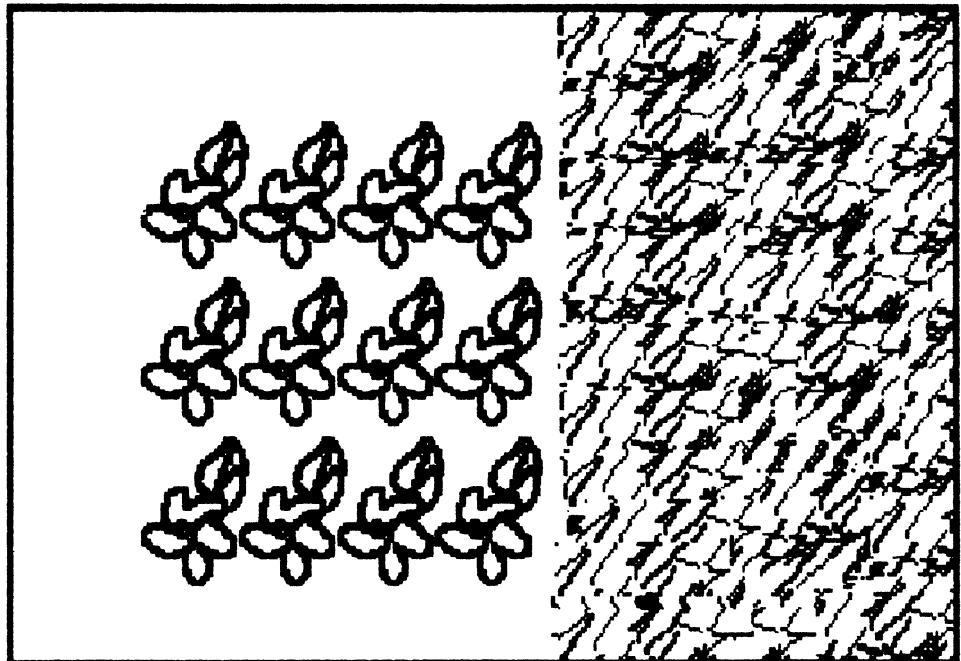
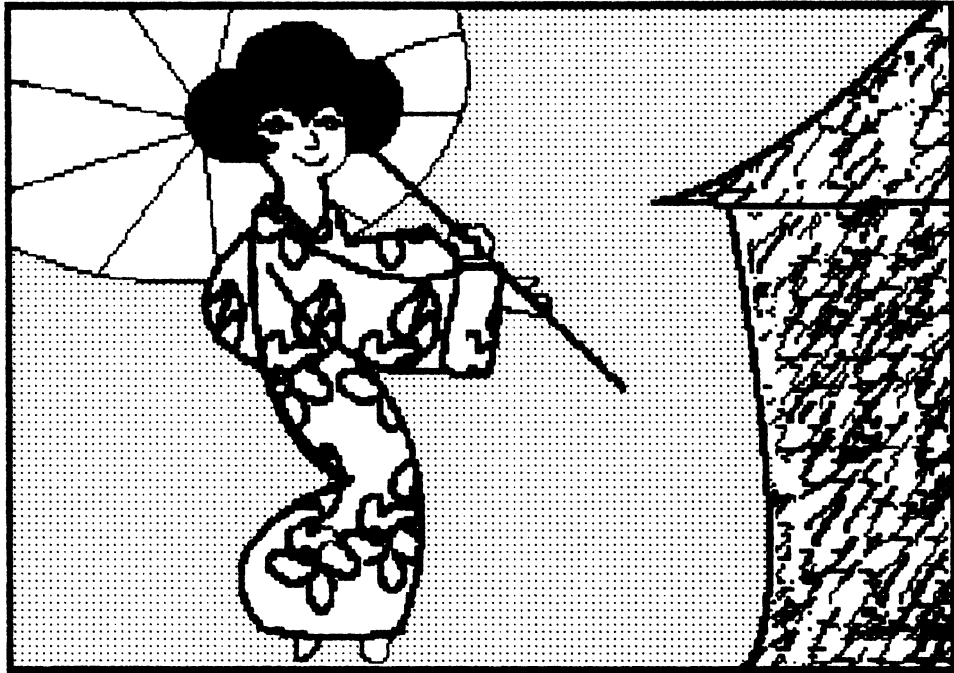


FIG. 2.8 The Merge-in-back function.



**CREATING A PATTERN BRUSH.** Another technique to create a master “pattern brush” that contains many small patterns. You can then load this brush and stamp it down, pick up the pattern you need, erase the master brush, and use the new smaller brush. This saves the time it would take to load and search your pattern files. Group your patterns according to type, and name the master brush files accordingly. Any time you want to add to a brush, just stamp it down, put the new pattern in place, pick the brush up again, and resave it.

### Getting to Know Cycle Draw

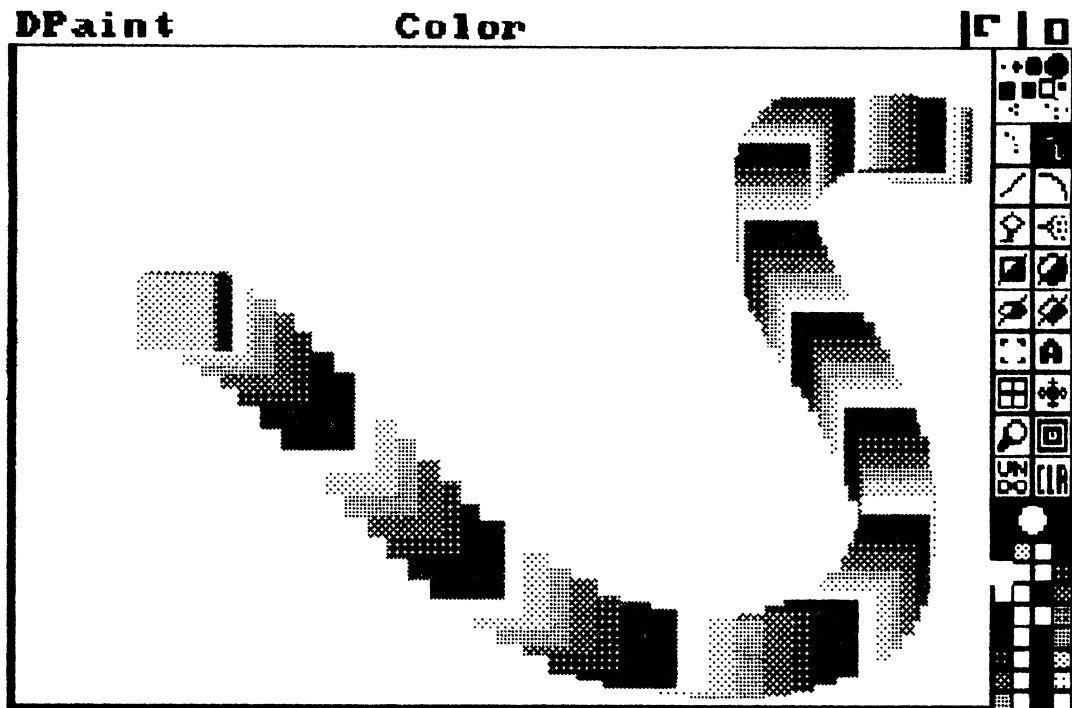
Cycle drawing can be used to create animated effects in your paintings. Experiment with the following example first and then use it in the three-dimensional painting example.

**SETTING A CYCLE RANGE.** Click with the right button on the *Foreground Color Indicator* to bring up the palette. You may designate

three ranges, C1, C2, and C3. Choose C1 first by clicking on the C1 indicator. Then, to establish a selection of colors to be animated, click on a color, then on *Range*, and then on a different color. The bracket shows the selected range. Adjust the cycle speed control bar for fast or slow cycling. At this point, you can repeat the process to establish C2, the second range of cycle colors, but for this example, it isn't necessary. Click on *OK*.

**PAINING AND CYCLING.** Select any brush and the *freehand line* tool. Pick a foreground color that's in the cycle range of C1, C2, or C3. (You'll be painting within just one range.) Turn on *Cycle* from the *Mode* menu. To create a large brush like the one used in Figure 2.9, choose a large square or circle brush with the right button, and drag to enlarge it even more. Then, begin painting. Experiment for a while, using the *Tab* key to turn the cycling on and off.

FIG. 2.9 Cycle drawing.



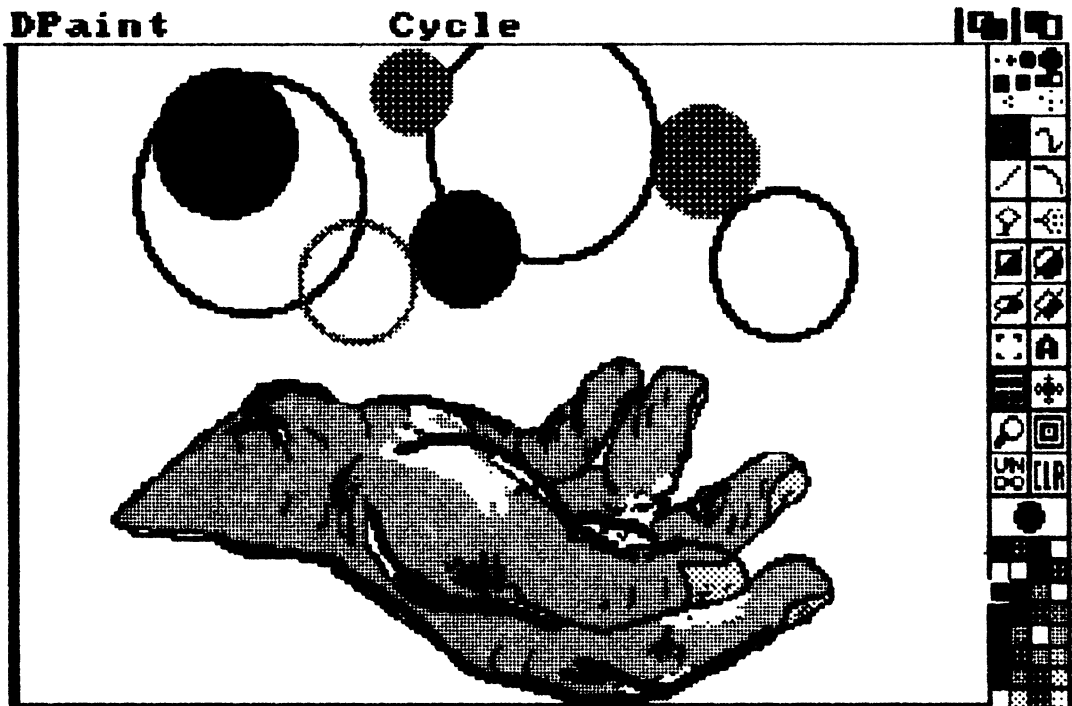
## Creating a Painting with Cycle Draw

This painting shows a juggler tossing balls. The cycling techniques you'll use to animate this picture can also be used to animate such things as flashing marquees, moving dancers, waterfalls, firework drawings, and many backgrounds.

**CREATING THE HAND.** Draw or use acetate to trace a hand onto the screen. *Fill* it with color. Add lines for creases, and create shading using darker colors with a thick brush or airbrush and *Blend*, *Smear*, or *Shade* as shown (Figure 2.10).

**DRAWING THE MOVING BALLS.** Choose a range of cycle colors in the palette. Turn on *Fast Feedback* from the *Preferences* menu. (This is so useful, you might want to leave it on all the time.) Use the *Circle* tool to draw the balls. Be sure to choose a different color for each one, all

FIG. 2.10 The Juggler.



from the cycle range you designated. Or, to make it very easy, choose *Cycle* from the *Mode* menu, click on the first color in your cycle range, and begin to draw. The colors are changed automatically in sequence. This will continue until you turn *Cycle* off.

Also, try drawing your circles with the *Oval* tool. Notice that you can make them very round, more so than those drawn with *Circle*. Use *Tab* to start the animation, and use it again to stop.

## Creating Three-Dimensional Objects

For many paintings, you'll want a simple way to create the illusion of depth. The following are several methods you can use to make a ground with a grid on it, objects, and text with the look of 3D.

**CREATING THE GROUND.** To create ground that seems to recede and move, click on the *Grid* tool with the right button and enlarge it to about 1/2- × 1-inch boxes.

Make the background color a pale blue or gray. Use a dot brush, and the *Freehand Dot* tool, and draw dots across the bottom of the page at every other grid point (about every 2 inches, which makes about five dots across the page). Midway up the page, draw a line of dots at every grid point (about 11 dots across the page).

Use the *Straight line* tool, a small brush, and a dark color to define a grid on the screen. Draw a line from the top center point to the bottom center point, then draw the other converging lines. Draw the horizontal lines (Figure 2.11).

Bring up the palette by clicking the foreground color selector with the right mouse button and create a range of browns and greens. Define these colors as the *Cycle* (C1, C2, or C3) range.

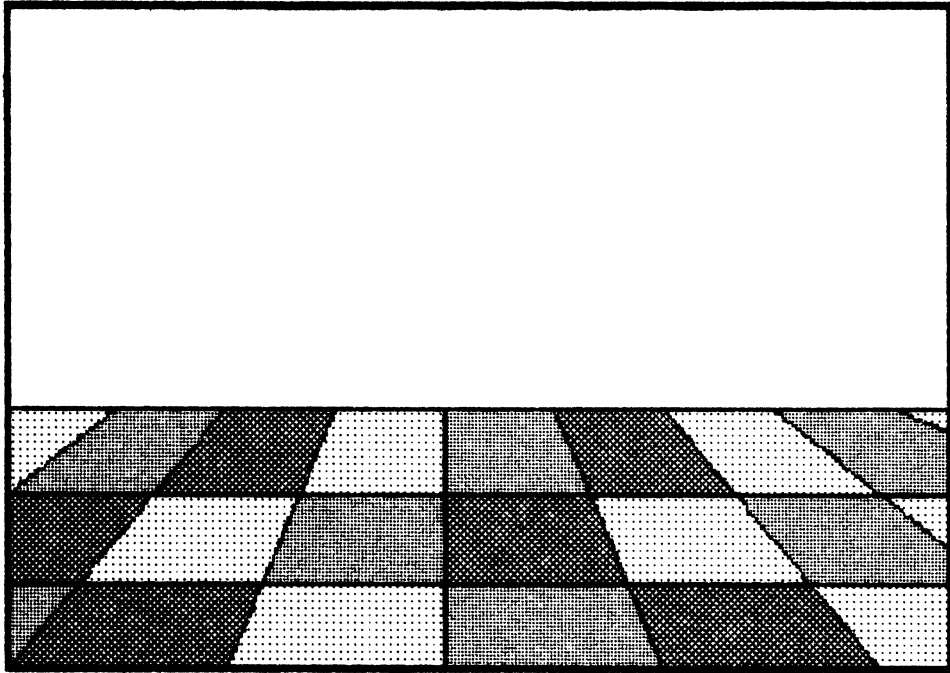
Turn off the *Grid*. *Fill* in the squares with various colors from the range you defined. You now have what appears to be a receding background, like a patchwork of field plots seen from a plane.

If you turn the *Cycle* function on and off (use *Tab*), the ground will appear to move or a bird painted over it would appear to fly.

**CREATING THE SKY.** If you wish, add a lighter symmetrical grid for the sky area and fill in the squares.

**CREATING THE SPHERE.** Select the palette requestor. Set the gray range (last 12 colors in the default palette) to a range from white to dark

FIG. 2.11 The ground.



blue. Click on the *SH* box and check the brackets. They should cover the range of colors you've just changed. If they don't, reset them (just like cycle). Turn off the palette.

Set the foreground color to the darkest blue. Choose the *Solid Oval* tool and draw out a circle (the *Circle* tool doesn't make circles as round as the *Oval*).

Pick up the circle and click a copy down on the lower right of the screen. Change the foreground color to the background color so you can see what you're doing.

Select *Shade*. Grab the circle you have just drawn and, using it as a brush, swirl it over the circle image, holding down the right mouse button. Do this until the circle is dark blue. Then, place the brush image over the circle image, just a little up and to the left, leaving a small crescent of the circle showing. Click the left mouse button. Continue this process until you have a shaded circle like that shown in Figure 2.12.

Now, select *Blend*, the 5-point brush, and the *Airbrush* tool, in that order. Holding the left mouse button down, use left to right and top to bottom motions to blend the bands of blue so that their boundaries disappear.

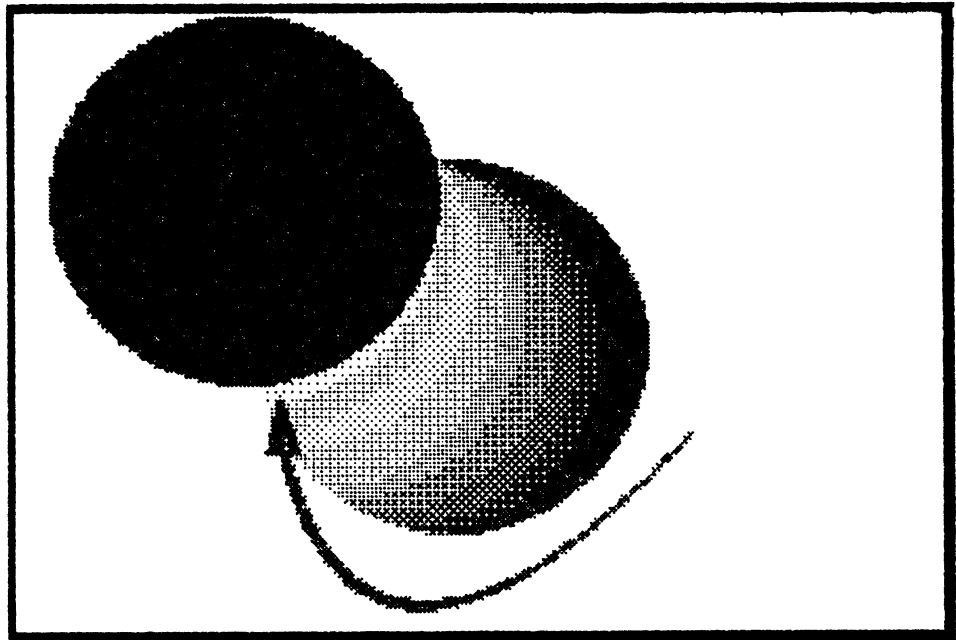


Select *Shade*, the 5-point brush, and the *Airbrush*. Use a circular motion to give the surface of the sphere some texture. Use the left mouse button to lighten and the right to darken until you get an effect you like.

**ANTIALIASING.** Circles, in low resolution, generally look pretty ragged around their edges. A way to smooth these edges is to use progressively lighter shades of color moving away from each jagged edge. This is called *antialiasing*.

A simple way to do this in Deluxe Paint is to pick up the circle as a brush. Set the background color to the lightest color in your shading range and clear the screen. Select *Color* from the *Mode* menu. Select any other color from the *Shade* range. Click down the now solid circle. Select *Shade* from the *Mode* menu and the *Circle* tool. Place the faint image exactly over the circle on the screen. Holding down the right mouse button, draw away from the center about two pixels in both the x and y directions (use the x-y display in the menu bar to see how far you've gone). As you do this, you'll see a soft edge form around the circle. Release the mouse button.

FIG. 2.12 Creating the sphere.



Select *Object* from the *Mode* menu and click the sphere design a little to the left of center over the solid circle.

Pick the new image up as a brush (be sure the background color is the same for both screens) and *Swap* screens (*J* key). Then, stamp the new image about midway up the screen.

**CREATING THE SHADOW.** Use the *Hollow Oval* tool to draw the shadow outline. *Fill* the outline with black (Figure 2.13).

**CREATING TEXT.** There is another simpler way to make shadows. Set the foreground color to red and type in the word "Surprise!" (including the exclamation point). Figure 2.14 uses a custom font, but any font can be used. Cut out the word using the *Brush Selection* tool. Choose *Size* from the *Brush* menu, *Stretch* and drag to enlarge the letters. Hold down the *Shift* key while stretching to keep the characters in proportion. (Or, instead of *Stretch* you could use the *Double* option.)

Change the *Mode* menu to *Color*. Choose black as the foreground color. Stamp down a copy of the word in black. Switch back to the

FIG. 2.13 Creating the shadow.

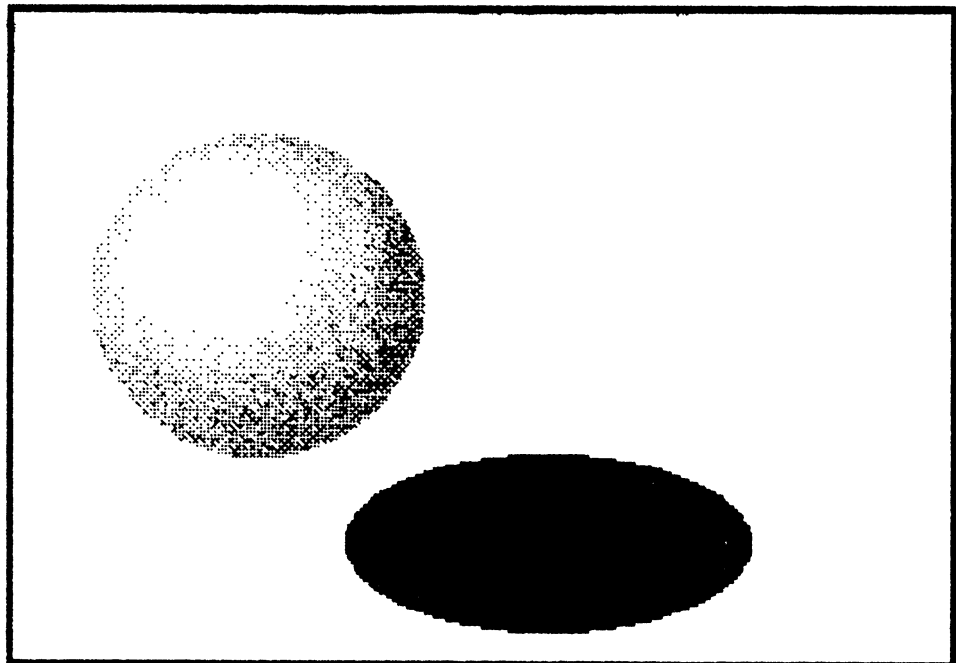
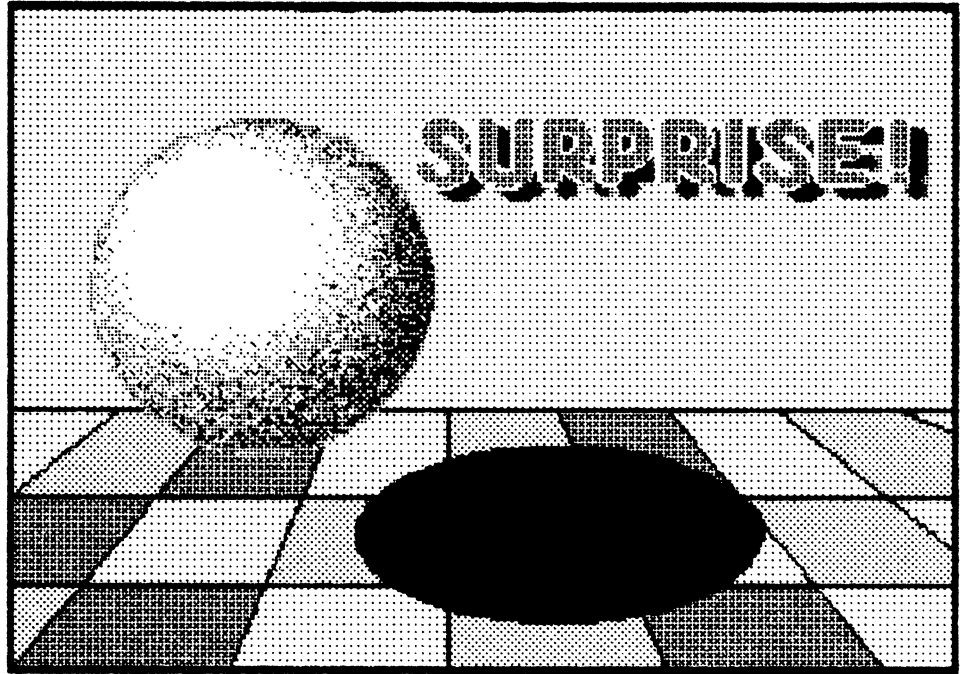


FIG. 2.14 The 3D drawing.



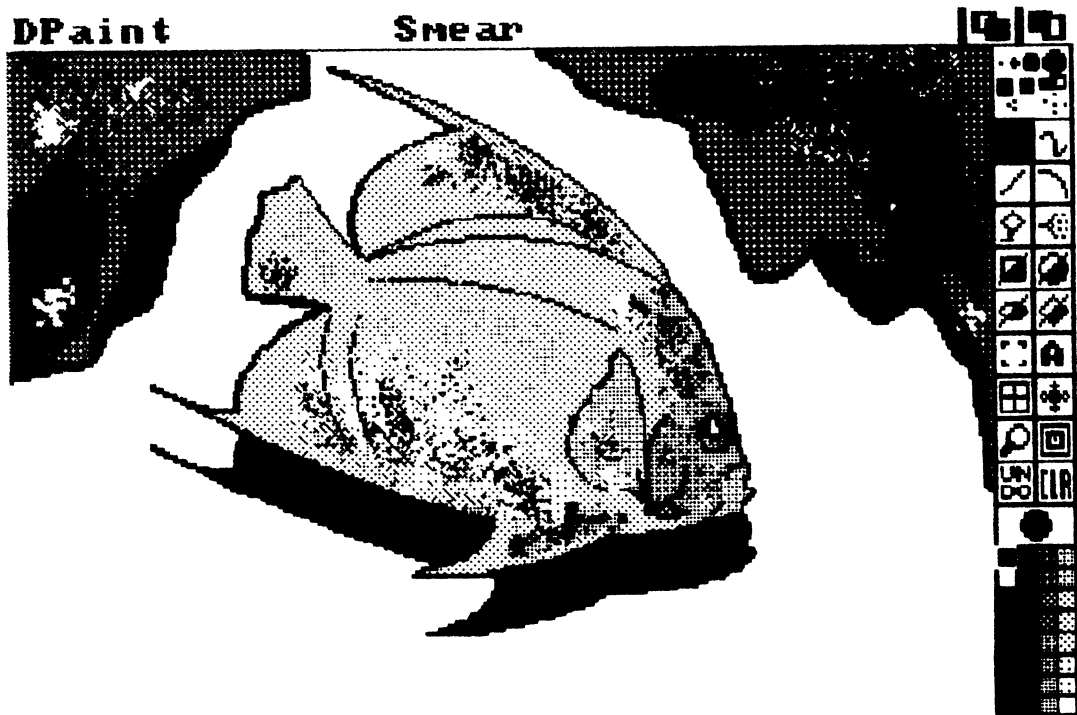
*Object mode.* Stamp a red copy of the word just above and to the left of the black one to create shadowed, 3D letters as shown in Figure 2.14. If you want to antialias the word, use the procedure described above.

## Creating an Underwater Scene

The fish scene incorporates advanced techniques of shadowing and shading (Figure 2.15). The instructions that follow explain how to create a new color palette and a special brush, and how to use the *scratchpad* feature.

**SETTING THE COLOR PALETTE.** Take a moment to create a new color palette with underwater colors such as ranges of blues and dark blue-grays, some sea greens, and oranges; and include a red, pink, pale yellow, and golden yellow. You should always set your palette before you begin to paint. This helps you plan the painting. As you work on the painting, remember, if you make a mistake, *Undo* it immediately, before doing another step.

FIG. 2.15 The underwater scene.



**DRAWING THE FISH.** Draw the outline of the fish in black with a small brush. *Fill* your fish with a base color, such as orange. Select *Airbrush* and a single- or three-pixel brush and use darker colors to create shading. Use the darkest colors on the lower stomach, the tail and the fins.

**CREATING THE SHADOW.** Pick up your fish as a brush. Choose black as the foreground color. Choose *Color* from the *Mode* menu. Stamp down a black fish shadow (Figure 2.16). Switch back to *Object* mode. Stamp your original fish on top and to the left of the shadow (Figure 2.17). Choose any new brush.

**SHADING AND HIGHLIGHTING THE FISH.** With a single-pixel brush and the *Airbrush* (adjust the nozzle to its smallest size by selecting it with the right button and sizing with the left), spray yellow highlights

along the upper edge and fin. *Fill* the eyes and small fins with solid yellow.

Use *Magnify*, the *Freehand* line tool, and a small brush to fill in areas of white and pale gray highlights in the small fins, mouth, and eye.

Use *Magnify* to perfect the shading and highlighting. For instance, you could *Magnify* successive portions of the upper fin and use *Smear*, *Blend*, and *Shade* to mix the colors. You must set your palette before you *Shade*. This is exactly like setting your cycle colors, except you click *SH* instead of *C1*, *C2*, or *C3* before you choose the *Range*.

With *Magnify* you'll be able to see each pixel moving as you *Smear* and *Shade*. Try using both small and thick brushes.

Use the same procedure to *Blend* the colors on the belly and body. Use *Zoom* to adjust the magnification: click on it with the left button for a closer look and the right button to move farther away. You'll find that the more detailed and advanced your painting is, the more you'll be working in the *Magnify* mode.

FIG. 2.16 The shadow.

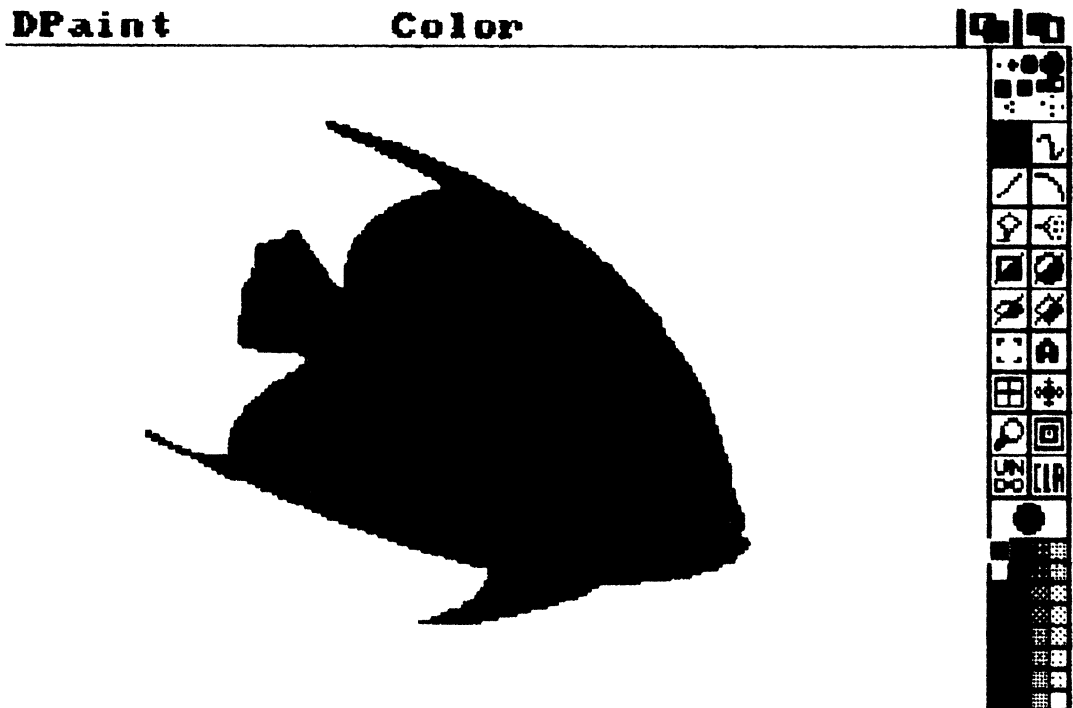
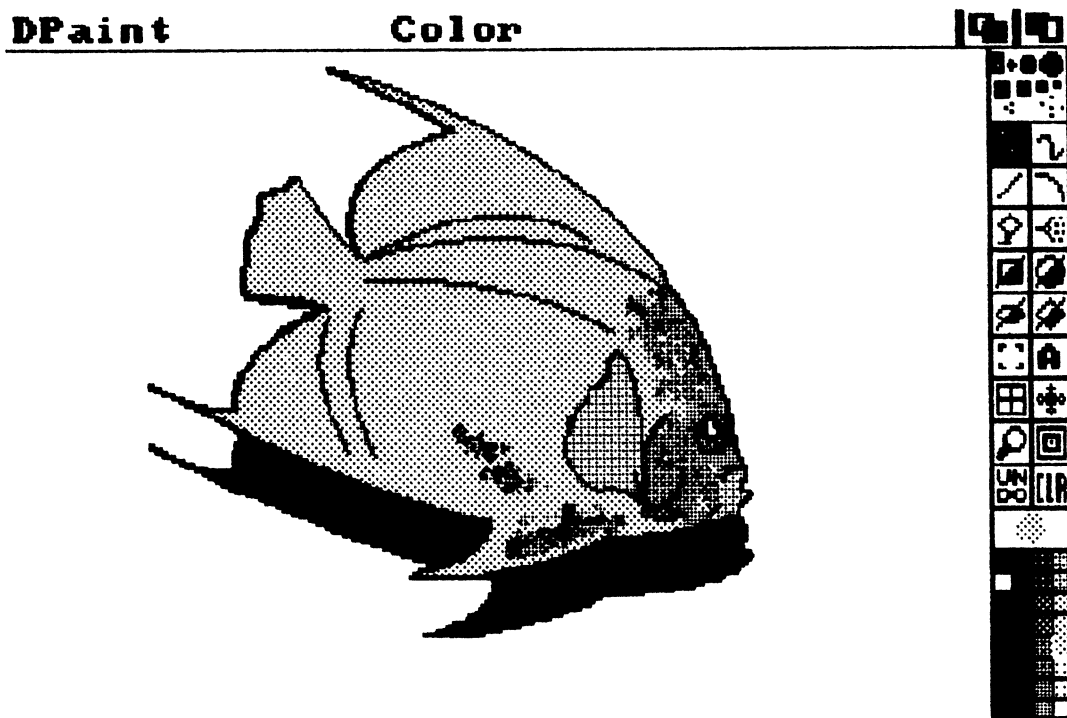


FIG. 2.17 The fish placed over the shadow.



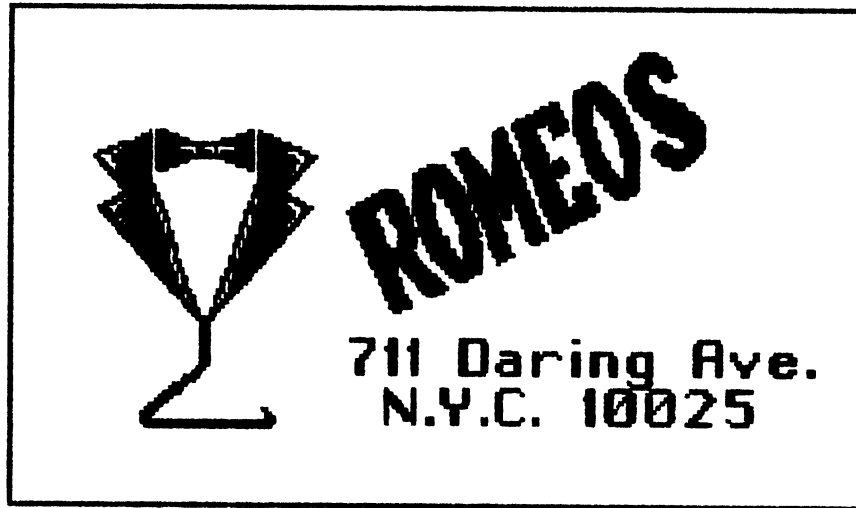
**MAKING THE BACKGROUND.** Add background structures such as rocks, shadows, and sea plants.

## ADVANCED EXAMPLES

The following advanced examples shown in Figures 2.18 to 2.23 build on the techniques practiced in the preceding section. These beautiful paintings were created by art students at the Savannah College of Art and Design (SCAD) under the tutelage of Professor Jim Alley.

This group of artists has created some of the most fascinating work to date on the Amiga. In fact, SCAD received the first 25 systems off the production line. This is still one of the largest collections of Amigas in a single setting. These paintings are only samples of the possible uses of Deluxe Paint and the techniques you've learned. You'll find many more of your own.

**FIG. 2.18** A business card designed by Janet Cook using Deluxe Paint in lo-res.



**FIG. 2.19** An intricate painting created by Ian Christman using Deluxe Paint in lo-res. This painting features the use of borders and patterns.

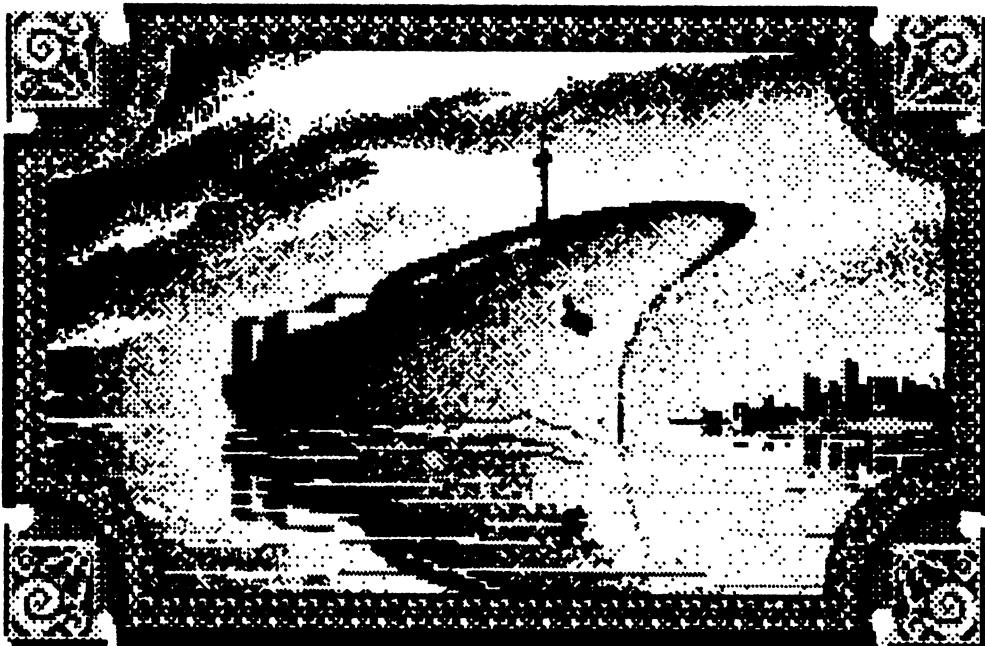
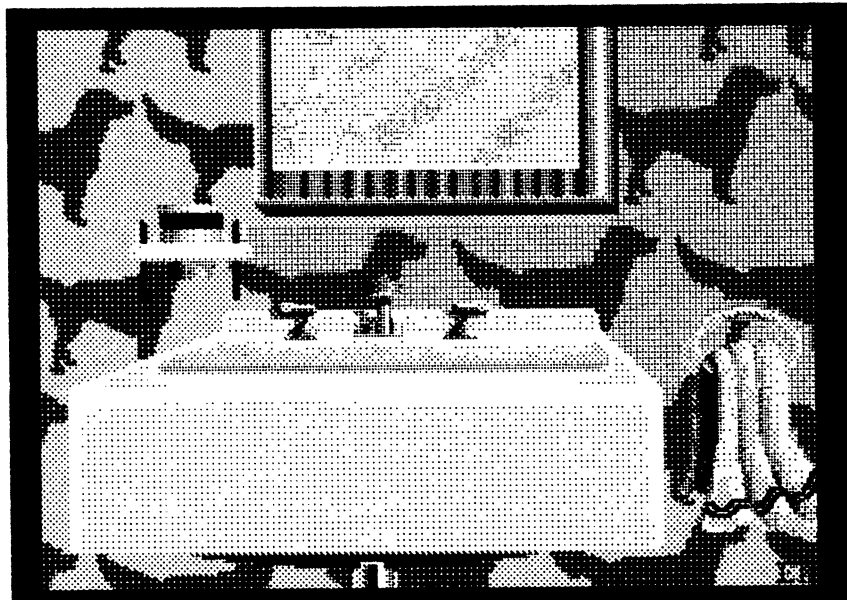


FIG. 2.20 Cary Grant and a takeoff on pop art by Nancy Pettibone using Deluxe Paint in lo-res and text as part of the graphics.

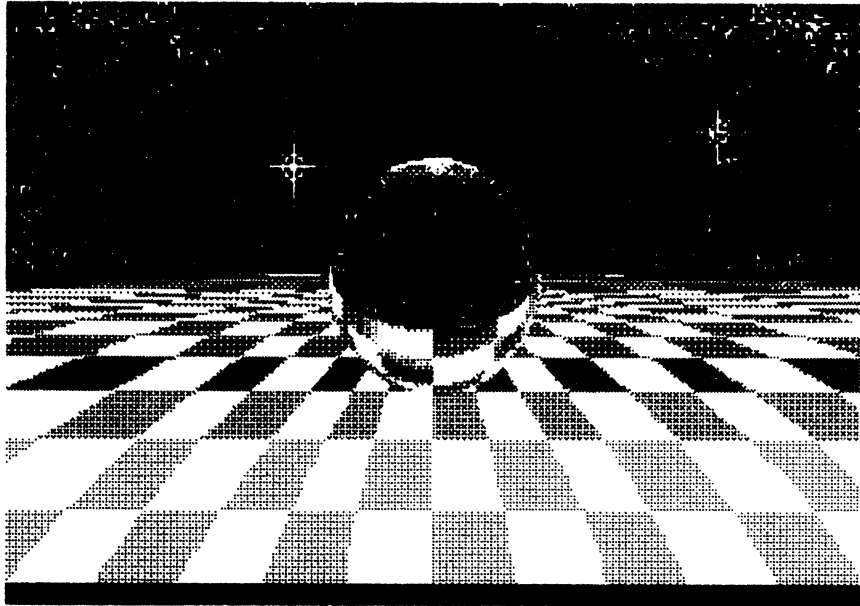


FIG. 2.21 Any subject can be used in Amiga art, even the bathroom sink as drawn by Ellen Reiser in Deluxe Paint using lo-res.

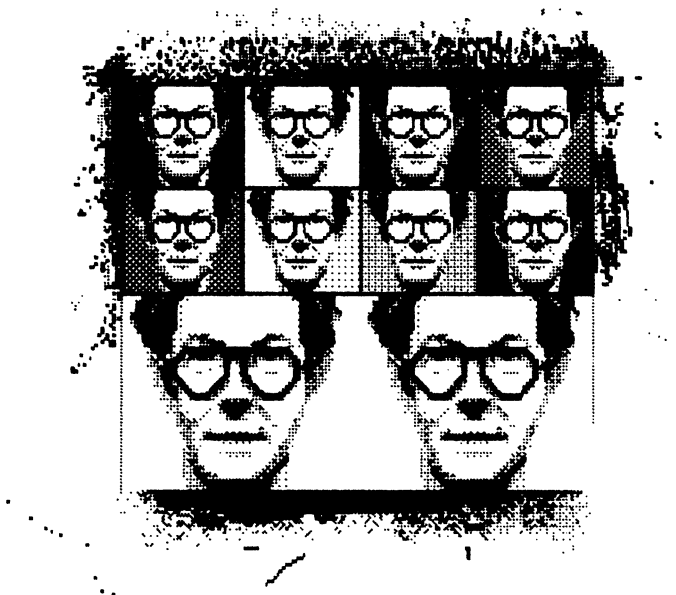




**FIG. 2.22** The advanced version of the sphere and grid example, by Jerry Donnelly using lo-res Deluxe Paint.



**FIG. 2.23** The Amiga gives you the ability to repeat an object and use it in various combinations as done by Professor Jim Alley using Deluxe Paint in lo-res.



## TIPS AND TECHNIQUES

---

### Tracing Images Onto the Screen

**ACETATE TRACING BY HAND.** For those who aren't gifted artists, working from outlines traced onto acetate is an easy and inexpensive way to transfer drawings onto the screen. It enables you to use an original sketch, a photograph, or a picture from a magazine.

The method simply requires taping a piece of acetate over the original drawing and tracing it with a fine-tipped marking pen. Then use masking tape to hold the acetate onto the screen, and trace the outlines using the *Freehand* tool (or, as in the fish scene, use the *Curved line* tool) and the single-pixel brush. If the image you want to use is too large or small for the screen, use a photocopy machine to enlarge or reduce it before tracing.

**CAMERA IMAGES ON ACETATE.** For images that are too complex to trace by hand, have pictures, photographic slides, or negatives made into black-and-white transparencies. The enlarged transparency can then be taped to the screen and traced. Many copy centers and photo shops can make transparencies for you.

### Graphics Tablets

A graphics tablet is a touch-sensitive graphic input device that you write on with an electronic stylus. You can draw over an original with the stylus, and what you draw will be reproduced on the screen.

You may have tried tracing over a drawing with the mouse. This doesn't work because there's no direct relationship between how far you move the mouse over the paper and the relative movement produced on the screen. Digitizing pens, however, offer *absolute positioning*. That is, what you draw is what you get, very accurately.

Several digitizer tablets are available for the Amiga. Unfortunately, they are more expensive than a mouse, ranging from \$295 to more than \$1000. However, a digitizer tablet may be a good investment for you. The ease with which you can trace elaborate drawings and the precision of up to 1000 points per inch (PPI) available for technical drawings could dramatically change the quality of the work you produce.

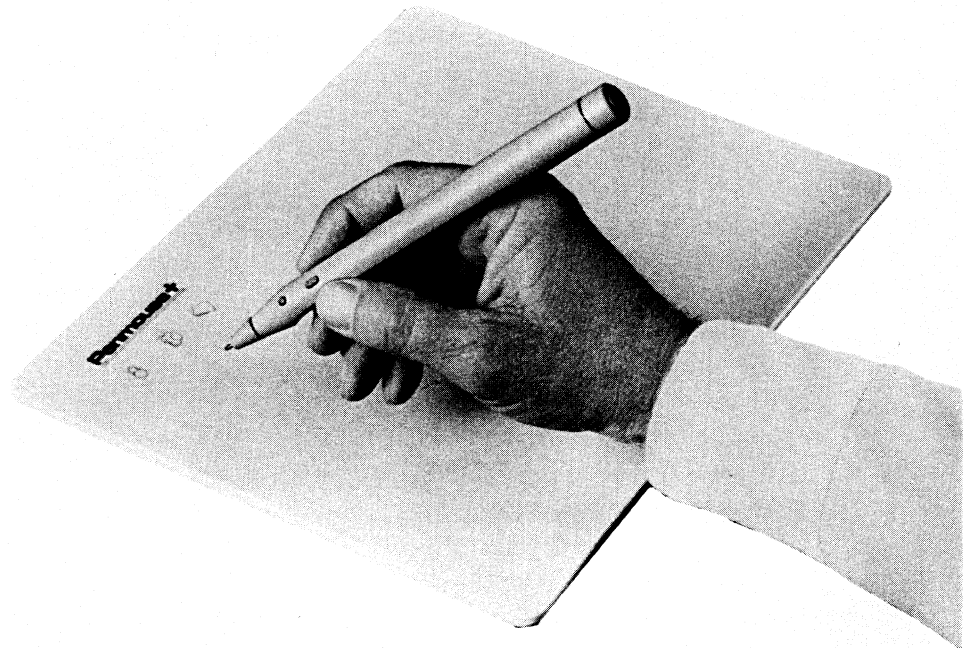
Kurta Corporation offers the Kurta Penmouse +, which functions both as a graphics tablet with absolute positioning and as a mouse using

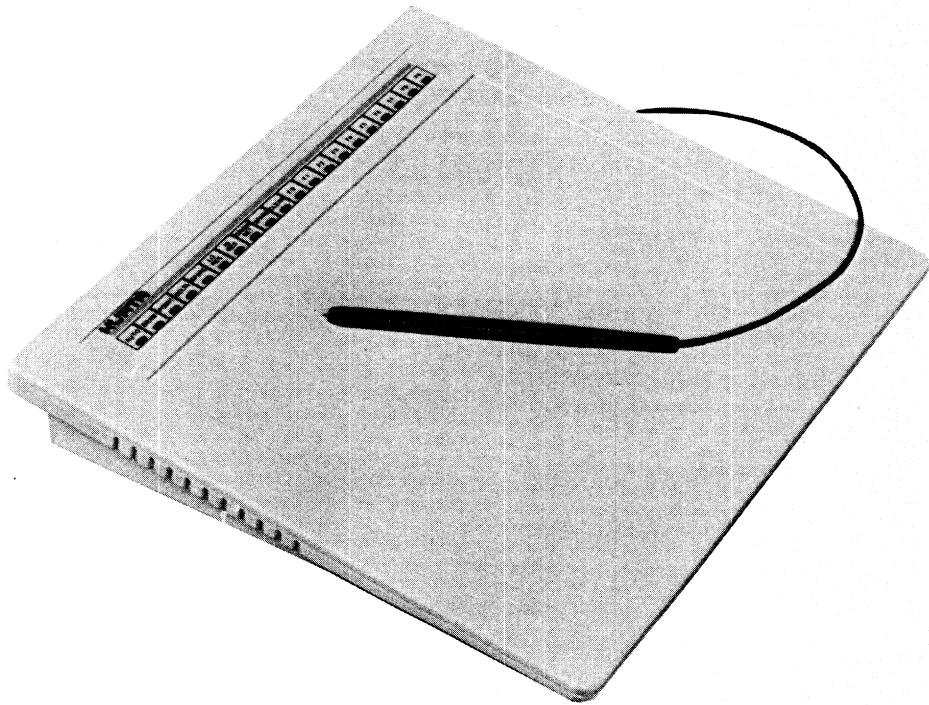
relative positioning. It includes two components: a cordless pen and a 1/4-inch thick tablet (Figure 2.24). It is soft-key selectable between a graphics tablet mode and a mouse mode. It uses an all-electronic movement detection method, so it can be rotated and operated in any direction and is not dependent on friction, quality of the surface, or optical, or mechanical characteristics. The Penmouse + has up to 200 PPI.

Kurta also offers several larger professional tablets including the Series ONE shown in Figure 2.25 in three sizes (8.5 × 11 inches, 12 × 12 inches, and 12 × 17 inches). The 8.5 × 11 inch tablet has a 8-degree slope and a ratio of active area to total area of 70%. The tablet is microprocessor-based with multiple operating modes, selectable output formats, downloading from the host, and a programmable menu for ease of use. The resolution is up to 200 PPI.

Anakin Research Inc., of Rexdale, Ontario, Canada, has created Easyl, a pressure-sensitive graphics tablet, for use with the Amiga. Easyl is 8.5 ×

**FIG. 2.24** The Kurta Penmouse +.



**FIG. 2.25** The Kurta Series One.

13 inches. It has a resolution of  $1,024 \times 1,024$  pixels. Anakin also offers compatible registration pegs for producing frame-by-frame character animation with EasyL. The user makes a series of drawings on paper in the traditional animation method. The sheets of paper are held in place on the graphics tablet by register pegs, and the user traces the paper drawings with the EasyL pen to input them to the Amiga. (See the Vendor Reference section of this book for more details.)

### **Digitized Camera Images**

Wouldn't the best possible solution to putting images onto the screen allow you to point a camera at an object and immediately transfer it to the Amiga? Absolutely. And it's possible. Chapter 10 describes how you

can use a video camera and software to capture images on your screen and then alter them by changing colors, resizing them, and adding text and graphics using any popular painting program.

These images are tremendous creative tools for designers and artists. And, for those who aren't artists but can at least point a camera, they form the basis for high-quality visual products that they couldn't execute on their own.

## Low, Medium, and High Resolution

When you type *dpaint* to open Deluxe Paint, you get the low resolution (lo-res)  $320 \times 200$ , 32-color version used in the example paintings. With 512k you can get the medium resolution (med-res)  $640 \times 200$ , 16-color version by typing *dpaint m*. And to get the  $640 \times 400$ , 16-color full interlace mode version, type *dpaint h*. With 256k, med-res gives you only four colors, and you can't get high resolution at all.

If you have two disk drives, med-res will not normally allow you to use a Spare screen, and in high-res the disk must be accessed often just to use the program. To use your spare screen more easily and save about 22.5k, unplug the second drive before turning on your Amiga. Or, you can get two screens in med-res by loading Deluxe Paint from the Command Line Interface (CLI) with the command *dpaint med 3*. This limits you to the first 8 colors in the normal palette of 16, but sometimes this is all you need. You may be able to start a project with *dpaint med 3* and then switch to the 16-color mode to finish. This technique used in the high-res mode won't give you an extra screen, but it will allow you to use much larger brushes.

## Multitasking

If while you're running Deluxe Paint, you'd like to be able to change Preferences or use AmigaDOS, you can use two drives by putting Deluxe Paint in drive 1 and booting up the Workbench disk in drive 0. Open CLI and type *cd df1:*. Then type *run dpaint* to start the program. Slide down the Deluxe Paint screen by dragging down the menu bar, and click on the CLI icon to activate it.

You can also copy the *run* command from Workbench onto your Deluxe Paint disk. For single drives, use the *copy c/run to ram:* command. Then insert your Deluxe Paint disk and type *df0:c/copy ram:run to df0:c*. Boot Deluxe Paint at the CLI 1> prompt by typing *run dpaint*.

For two drives, put Workbench in drive 0 and Deluxe Paint in drive 1. Copy `c/run` to Deluxe Paint using the `copy c/run to df1:c` command.

Note: if this above use of the CLI is new to you, see Chapter 12 for an introduction.

## Using Aegis Images

Aegis Images, from Aegis Development, is another powerful painting program. It comes packaged with Animator, discussed in Chapter 5, to make a potent combination.

Images has 10 different shape tools, 20 brushes and patterns, and several standout special effects, including an Under painting function, Pantograph, Antialias, and Wash. It also makes use of a Fast Menu and brush and pattern editors, which are extremely helpful.

It comes with a comprehensive, easy-to-follow user's guide to get you going. You could certainly create all of our examples and more with Images. And you would be able to use the extra special effects it includes. Some professional artists have found that Images is not as flexible as Deluxe Paint. Despite the fact that it offers extra tools, the user interface is a bit more cumbersome for heavy use. This won't be noticeable to most users however, and getting Images and Animator in one package makes it a great value.

## Using Graphicraft

Graphicraft is Commodore's own painting software for the Amiga. It was the earliest painting program and was bought by many users. It's easy to use but may not have all the tools a professional artist would prefer. Still, it has plenty of choices for most uses and can be used to create any of the example paintings.

## Clip Art

Both Electronic Arts and Aegis Development have created clip art, disks filled with objects and scenes drawn by accomplished artists, that can be used in any of your pictures. Decide what you need or like—a picture of a car, for example—look it up on the disk, and drop it into whatever you're working on. Clip art is particularly useful for business purposes. When time is of the essence, you can simply import a U.S. map or a city scene for your graph or report rather than spending the time it would take to design that piece yourself.

Need a completed background for a prehistoric scene, a beach view, an elephant, a bar chart, an alien being? You can find one ready to use somewhere on these disks.

## Using Deluxe Print

Deluxe Print, from Electronic Arts, is a printer graphics program that comes with its own clip art. It is designed to help you create quickly customized stationery, calendars, greeting cards, labels, and banners. You could create these items using a painting program, but not as quickly and easily as this (Figures 2.26 and 2.27).

For instance, Deluxe Print presents you with a ready-to-use greeting card format, a choice of borders, various text styles, and a choice of logos and colors. And, you can add freehand objects. Presto—in 5 minutes you have a personalized greeting card. Also, the disk is a great resource because you can export the clip art pictures to any other paint or animation program for use in your own drawings.

FIG. 2.26 A sample Deluxe Print screen.

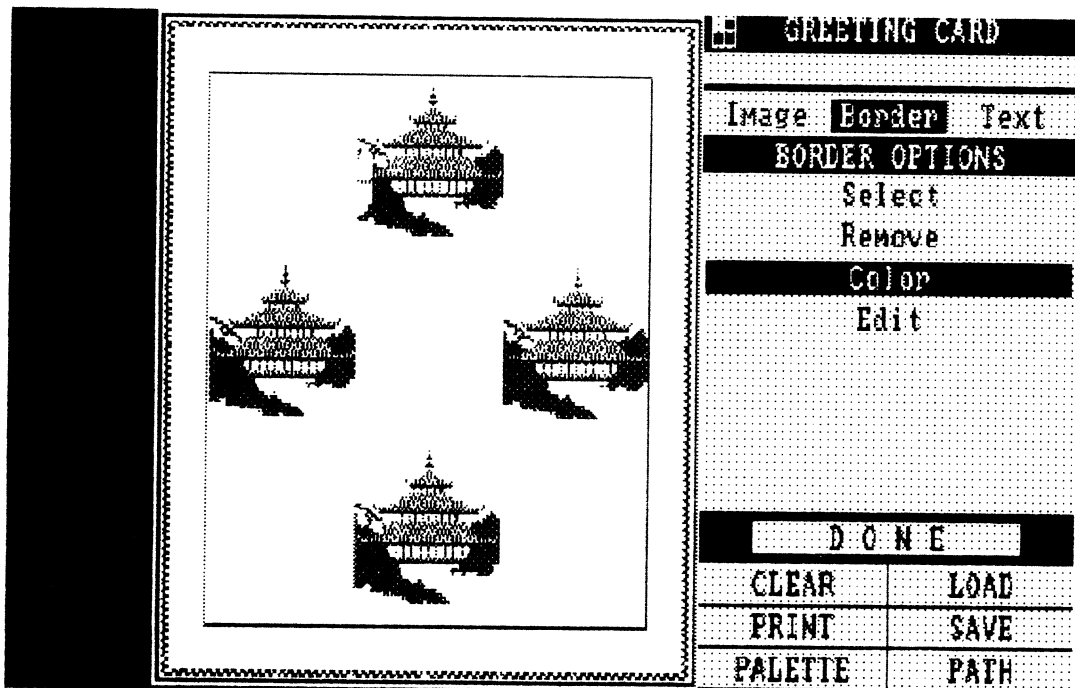
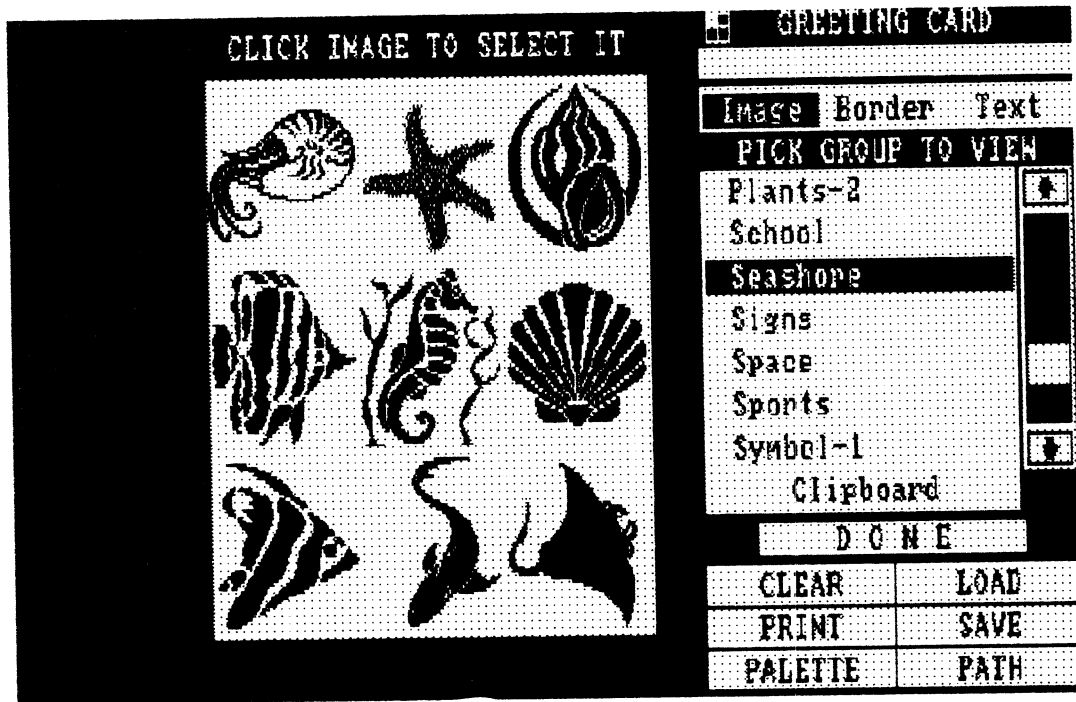


FIG. 2.27 Just a few of the choices you have in Deluxe Print.



## Converting Deluxe Print "Group" Images to Deluxe Paint

Besides using art from Deluxe Print with Deluxe Paint, you can do the reverse. You can import files created and saved with Deluxe Paint for use with Deluxe Print. If you've seen Deluxe Print, you may have noticed that some of the files on the Art Disk supplied with the program were created in just this way.

Also supplied with the program are what Deluxe Print refers to as "group" files. These are clip art collections of images you can use to create some spiffy cards, letterhead, and so on. They certainly can make your life a lot easier and make your work look very professional.

Unfortunately, it's not easy to use Deluxe Print's group files with Deluxe Paint, because they're in the wrong (not IFF) format. But there is a way.

Open Deluxe Print and select the *Label* format. Use *Select* to store the group file in memory. *Select* an image you want to export to Deluxe



Paint. Place the image on the label background. You might have to do this in steps, with the larger two- and three-piece pictures like the “downhill racer.” Select the *Save* option. Enter a file name. Click on *Export*. Repeat this for all the images you want to export. Exit Deluxe Paint.

Load Deluxe Paint in any mode. Select the *Load Brush* option. Be sure your data disk with the Deluxe Print images is in a disk drive, and then tell Deluxe Paint you want to look there for the brush (see the manual for details). Select the med-res drawer (don’t worry if you’re not in med-res—it works anyway).

Load the brush. When the brush appears, it will probably be all jumbled up. Select *Remap* from the *Brush* menu. If you’re in other than the med-res mode you will have to resize the brush so that its original proportions are preserved. From either lo-res or high-res modes select the *Size* option from the *Brush* menu and choose *Halve*, then reselect the *Size* option and choose *Double Vert*.

Voila! A new Deluxe Paint image for your clip art files.



# **Business Graphics**

Presentation quality graphics for use in reports, brochures, training manuals, and advertisements are everyday needs for most businesses. Corporate America spends up to \$100 for the outside production of one 35 mm slide for use in a sales presentation. Now, for under \$2000, the largest or smallest business can have the facilities for producing hard-copy graphs, charts, slideshows, and even animated graphics presentations using the Amiga. This is a breakthrough for producers and audiences—it will increase both demand and use of graphics. Above all, it increases your choices.

---

## **THE ALTERNATIVES**

Your first choice was buying the Amiga, which is the first personal computer with the hardware capabilities to make color graphics production and low-cost hook-up to a VCR for showing graphics presentations possible.

Next, you'll need to determine what software will be appropriate for your business needs and how the final product will be presented.

### **Software Options**

The painting programs listed in Chapter 2 and the CAD program in Chapter 4 can all be used to create colorful and detailed charts, graphs,

and business illustrations with text. If you already own such a package and only need an occasional graph or chart, you won't need another program. If your business requires a large output of high quality graphics and the use of data from spreadsheets however, you will benefit by choosing a product specifically designed to make such graphics quickly and professionally.

Two of the most dynamic programs available are Impact! by Aegis Development Corp., and Deluxe Video Construction Set by Electronic Arts. Impact! produces top-quality slide shows. Deluxe Video creates animated presentations including animated titles, charts, and graphs. For more on Deluxe Video, see Chapter 5 on animation programs. For more on Impact!, keep reading.

## CREATING CHARTS AND GRAPHS WITH IMPACT!

---

With Impact! you can enter tables of numeric data and then select the type of chart you'd like—bar, line, or pie. Impact! does all the calculations and gives you a chart to which you can add text, colors, or freehand drawing objects. Impact! will also accept ASCII text files, so you can input data straight from your current spreadsheet programs (see C-1 of the Impact! manual).

The program runs in the  $640 \times 200$  mode. It is an object-oriented program (like CAD), but also offers paint program features such as, freehand, line, frame, circle, and arc.

Text is offered in five fonts with point sizes from 4 to 18. But you can easily reduce or enlarge the text beyond these ready-to-use sizes. You can also choose to have the letters underlined, with drop shadows, outlined, bolded, and italicized.

The text is entered in a *Table* or text screen. This is particularly powerful because it gives you the ability to use a built-in word processor to take any amount of text you input and rearrange it to fit into any area of the page that you define.

You can also create *Icons*, which are symbols up to  $64 \times 32$  pixels that can be placed anywhere on the page and can be stacked as symbols on charts. This feature can be used to design unique and striking bar charts quickly and easily.

*Slides* are collections of graphic objects (charts, graphs, text, etc.) placed on the screen of Impact! to create a drawing or image. The slides

may be saved and played back in a show. You cannot print any object that you create in Impact! until you plot it onto a slide. For instance, after you create several graphs, you can resize and combine them, say, shrink four graphs and put them on one slide. Then, you can design a presentation of the slides in a selected sequence, thus creating an entire show detailing the growth of your company, next year's sales projections or this year's profits.

In this show, you choose the transitions, called *Wipes*, from one slide to the next: fade in/out; curtain up/down, left/right; spiral in/out; random; or trickle. You also determine the wipe delay and how long each slide will be shown.

## Limitations

**SPEED.** Impact! keeps everything you draw on the screen. If you make an alteration on the image, the new image is drawn over the old ones. This is useful because you can perform sequential Undo functions to get back to any previous level. However, this feature also makes you wait frequently and for long periods while the screen redraws itself.

**MEMORY.** On a 512k machine, one complicated slide that includes a lot of data can use up all available memory. If this happens, the system will crash and you may lose your work. If a data disk is available, the program will try to save the current work to it. Recognize that with 512k you may occasionally lose work. It is crucial that you save your work frequently, in various stages, to avoid losing everything.

This program benefits tremendously from the addition of extra memory. For businesses, a 1-megabyte expansion card can be considered a must. It will give you more working memory for creating slides and will speed up the change of slides when running a slide show. A hard disk also improves slide shows by providing more space for slides and speeding up the presentation.

## PRODUCING A SAMPLE SLIDE SHOW

---

The following section is a detailed explanation of how to create a variety of unusual charts and graphs. It also describes how to combine them into a professional slide show. "Presentation Graphics with Im-

pact!" is a detailed manual provided with the software. It introduces the features and tools of the Impact! program and provides one tutorial for creating a basic graph. The examples in this chapter can be used to follow that example or own their own. They include advanced uses that you can incorporate in your own reports and presentations.

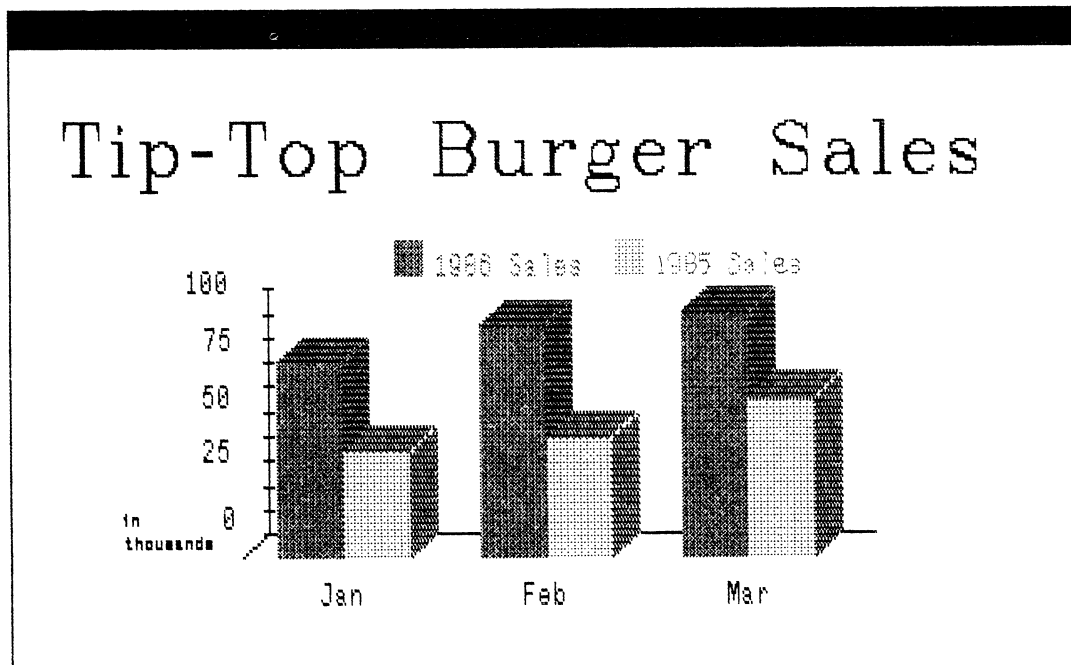
## Creating a Bar Chart

**ENTERING DATA.** To enter data for the chart shown in Figure 3.1, choose *New* from the *Project* menu, using the right mouse button. To enter the Graph Builder, highlight *Graph* and then release the button.

A requestor box will appear. Click inside the edit box until the prompt shows that it is active. Type in the name of the Series (data for a chart) that you'll create. Call it "1986 Sales." When the Series Requestor appears, type in the following data:

Jan	80
Feb	95
Mar	100

FIG. 3.1 A 3D bar chart.



---

Close the series by clicking in the box in the upper left corner (the close gadget). Choose Open Series from the Graph menu and type in the following data with the title "1985 Sales":

Jan	80
Feb	50
Mar	65

Close this Series Requestor.

You will be creating a graph that compares the data of the two years. Note that in order for the computer to match them accurately, the titles must match and the labels must match. Jan and Jan match; but don't use Jan and jan. Also, don't use any commas. Only numbers are accepted.

It's useful to type in the year with the larger data values first. Then, if you create a chart with bars in front of each other, the taller bars (values) will automatically appear behind the shorter. That's why you typed the 1986 data before the 1985 data.

**SELECTING THE TYPE OF GRAPH.** Choose *Bar* from the *Graph* menu. When the bar *Graph Selector* appears, click on *3D* to create a graph that compares the sample years with side-by-side bars. When the chart is drawn, choose *Okay*.

**CHANGING THE BACKGROUND COLOR.** There are several ways to change the black background to another color. One is to go into the *Colors* menu and choose *Edit*. In the *Edit* menu set the zero color (black, at the left) to whatever color you want the background to be. Do this by clicking on black, then on *Copy*, then on a color you won't be using. That color will now be black. Go back to the zero box (black), click on it, and then use the slider controls to change its color. When printing onto paper, you'll usually want to set this to white; otherwise, the black screen will print out as black on paper.

Another method is to go into the *Colors* menu and click on the color you want for the background menu to make it the active color and choose *Frame/Filled* from the *Shapes* menu. Move to the upper left corner of the workscreen and pull a frame with the left button to the lower right-hand corner (covers the entire screen). The graph will redraw over the new background color. If you have 512k this method will use up too much memory, but is a useful technique on other occasions.

**CHANGING THE COLOR OF THE LEGENDS AND TIC MARKS.**

Choose *Special* from the *Graph* menu. This section gives you the opportunity to change the colors of the tic marks, axis lines, and legends by clicking on a color and then clicking in the box right-hand boxes until they turn the selected color. You can also turn the legends on or off. Pages 5-20 of the manual describe the other functions of this selector.

**DELETING THE NUMBERS.** The numbers on the left hand axis have decimal points attached to them. Round numbers look much better. To change them, click *Select* in the *Edit* menu to *On*. The pointer will have the work "edit" attached to it. Click it over the left hand numbers. They will highlight to show that they are selected for editing. Choose *Delete* under *Region* in the *Edit* menu. The column of numbers will disappear.

**TYPING IN TEXT.** To type in new numbers (100, 75, 50, etc.), select *Font* from the *Text* menu and set it to *System*. Set the point size to 8.

Go to the *Color* menu and slide down to *Text Colors*. Highlight *Foreground* and release. Go back and click on the color you want the numbers to be (they should match the month labels).

Choose *Text* from the *Shapes* menu. Click the cursor on the screen where you want the text to appear. A dagger appears there. Type in the numbers.

Set the point size to 6 and type in the label "in thousands."

You now know how to type in text, but there is also a shortcut you can use in this case. You can leave the numbers with the decimal places until you plot this graph onto a slide. In the *Slide Builder* you can use the background color, a small brush, and the line tool to draw over the decimal places. (You can't paint over them in the *Graph Builder*.)

**MOVING THE LEGENDS.** To move the legends so they don't overlap the bars, choose *Select* from the *Edit* menu. Slide the highlight over the *On* selection.

Use the pointer with "Edit" attached to click on both legends to be moved. They will highlight. Go back to the *Edit* menu and select *Region* and *Move*. Move the box attached to the cursor, to the spot where you'd like the legends to be and click the left button. The legends will appear there. You can choose *Move* again if the new position isn't quite what you wanted. Go back to *Edit* and turn *Select* to *Off*.



---

**CHANGING THE SIZE OF THE GRAPH.** Click on the sizing gadget in the lower right corner of the graph builder and move the cursor in to reduce the size of the graph by about one third.

**PLOTTING THE GRAPH.** Select *Plot* from the lower left corner of the graph builder. When the requestor appears, select *Yes*.

Move the reference frame that appears by pushing the upper left corner with the cursor. Place it at the bottom of the screen and then click the left button. When the frame reappears, click the right button and it will disappear.

**BUILDING THE SLIDE.** You are now in the Slide Builder. Here you can add to the slide but cannot change any of the elements you created in the Graph Builder. In order to change such items, select the entire graph with *Edit* and then from *Region* choose *Edit*. This will put the chart back into the Graph Builder.

In the Slide Builder you can give the graph a title. Choose *Text* from the *Shapes* menu. Then choose *Font* from the *Text* menu. Choose *Regular* and adjust the point size to 8. Select *On* next to the *Shadow* option.

Go to the *Color* menu. Highlight the *Text Colors* option and highlight *Foreground*, then release. Go back and click on blue. The *Foreground* color indicator will change to blue, indicating that any text you type will be blue. You can repeat the process to select the colors for the background and shadow colors you'd like to use.

Select *Text* from the *Shapes* menu. Move into the slide and click the left button on the spot where you'd like to place the title of the chart, "Tip-Top Sales," and type it in.

**CREATING A BORDER FRAME.** For a final touch, choose a large brush from the *Brush* menu, select a color, and choose *Frame/Normal* from *Shapes* to pull a frame around the graph. You won't always want to use this for slide shows, but it's useful for printed graphs.

**SAVING THE SLIDE.** Save the slide to your prepared data disk.

- If you have one disk drive, remove the Impact! disk and insert the data disk. Select *Drawer* from the *Project* menu. If it says *df0*,

select *Okay*. If not, type in *df0:*. Select *Save* from *Project* and highlight slide.

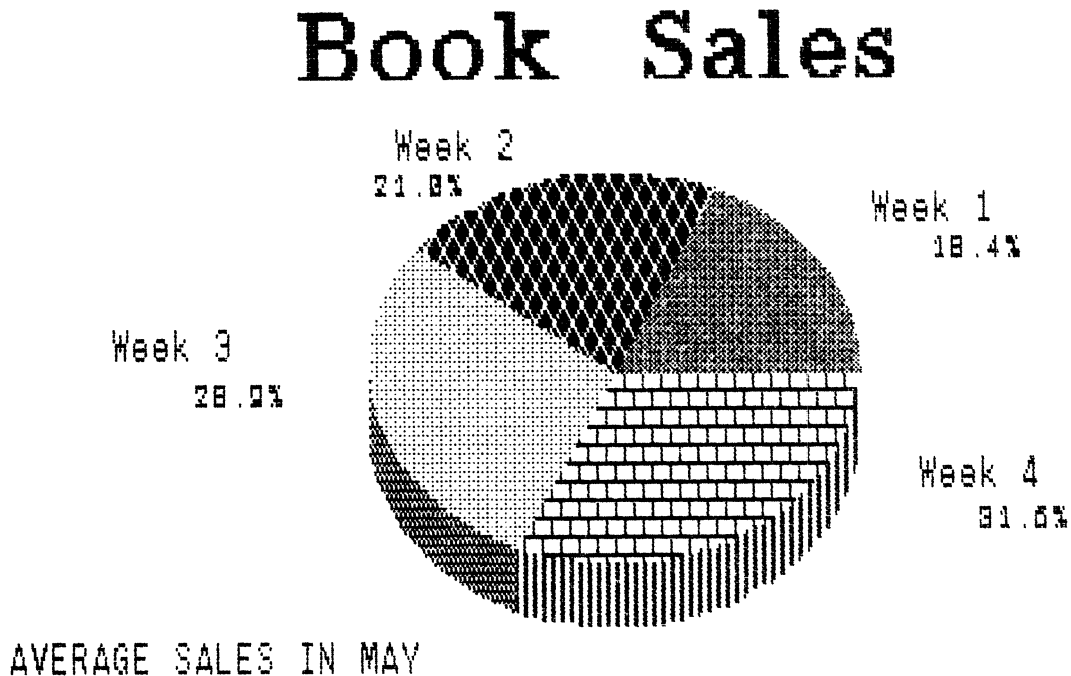
- If you have two disk drives, place the data disk in the external drive. Select *Drawer* from the *Project* menu. Type in *DF1:* and *Okay*. Select *Save* from *Project* and type in the title. (In the future you might put the data disk in the external drive as soon as you start the program.)

You've created the first slide for your sample slide show.

## Creating a Pie Chart

You've already been through all of the basic steps for creating a slide using a bar chart. Now set up the pie chart shown in Figure 3.2 to try out a few other features.

FIG. 3.2 A 3D pie chart



---

**ENTERING THE DATA.** Select *New* and *Graph* from the *Project* menu. Type in the title, "Book Sales," and fill in the following data for the series:

Week 1	3500
Week 2	4000
Week 3	5500
Week 4	6000

Close the series.

**CREATING THE PIE.** Select *Pie Chart* from the *Graph* menu. And select the left pie. A pie will be drawn on the screen. Then choose *3D*. A new pie will draw over the old one. Click on *Okay*.

**EXPLODING THE PIE.** Make the largest slice stand out by exploding it. Do this by choosing *Select/On* from the *Edit* menu. Click on the slice to choose it for editing. Then choose *Explode* from the *Region* menu. You can replace the slice by choosing *Explode* again.

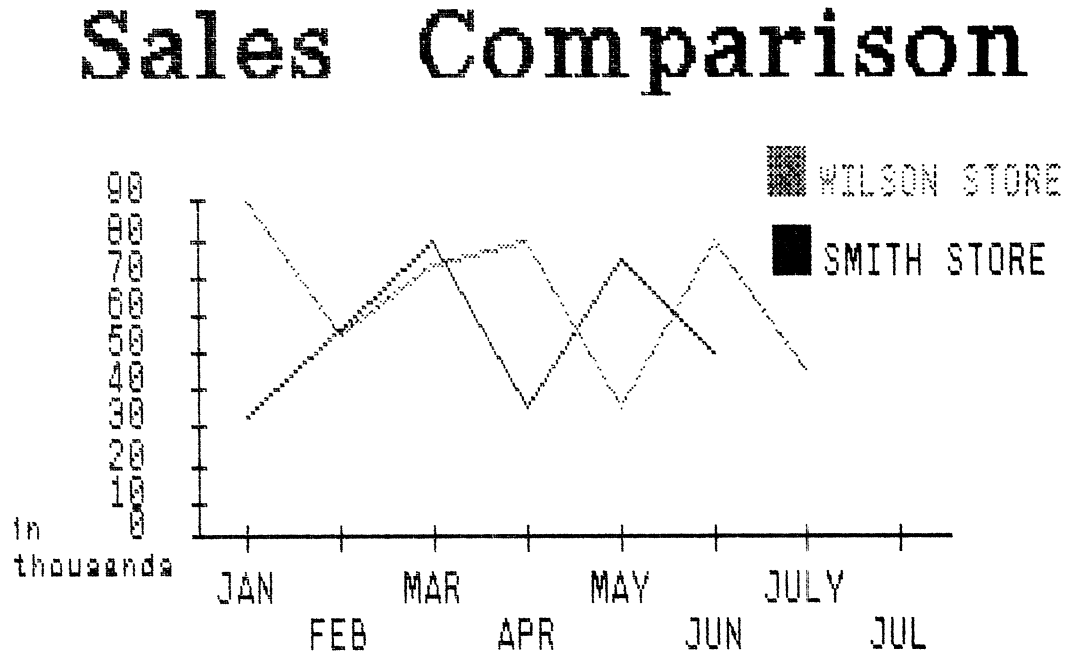
**CHANGE THE COLOR OF A SLICE.** To change the color of a slice (or to use patterns), use *Select/On*, click on the piece to *Edit*, and click on the color you want in the *Color* menu.

**FINISHING THE GRAPH.** To finish the graph, type in the legend "Average Sales in May," plot the graph onto a slide, add a title, and save the slide. (If necessary, review the "Creating A Bar Chart" example for descriptions of how to do these last steps.)

## Creating a Line Graph

Line charts are the best method of displaying data to show trends in many businesses. Figure 3.3 shows just one of the line graphs it is possible to make using Impact!. To create it, use the methods you've learned to open a new graph and type in the following data.

FIG. 3.3 A line chart for showing trends.

**ENTERING THE DATA.****Wilson Store**

Jan	35
Feb	50
Mar	70
Apr	90

Open a new series with the following data.

**Smith Store**

Jan	22
Feb	50
Mar	80
Apr	100

Close the last series.

**CHOOSING THE GRAPH.** Choose *Line* graph from the *Graph* menu and choose the upper right hand graph.

**ADJUSTMENTS.** Use techniques from the preceding examples to move the legends to appropriate position, add the legend “in thousands,” plot the graph to a slide, and add a title.

**CHANGING THE NUMBERS.** To delete the .0 decimal places from the left hand numbers column, set the background color as the active color, choose a small brush, and use the line tool from *Shapes* to draw over the .0 column.

**SAVING THE SLIDE.** Use the techniques under “Saving the slide” in the “Creating a Bar Chart” example.

## Creating Icon Charts

**CREATING A HAMBURGER CHART.** What if you could use burgers rather than bars for the Tip-Top Burger graph? It would be a great visual statement, such as the graph in Figure 3.4 which you can make using the Icon Builder. Start by selecting *New* and *Graph* from the *Project* menu. Use “Burger Chart” as the title and enter the following data.

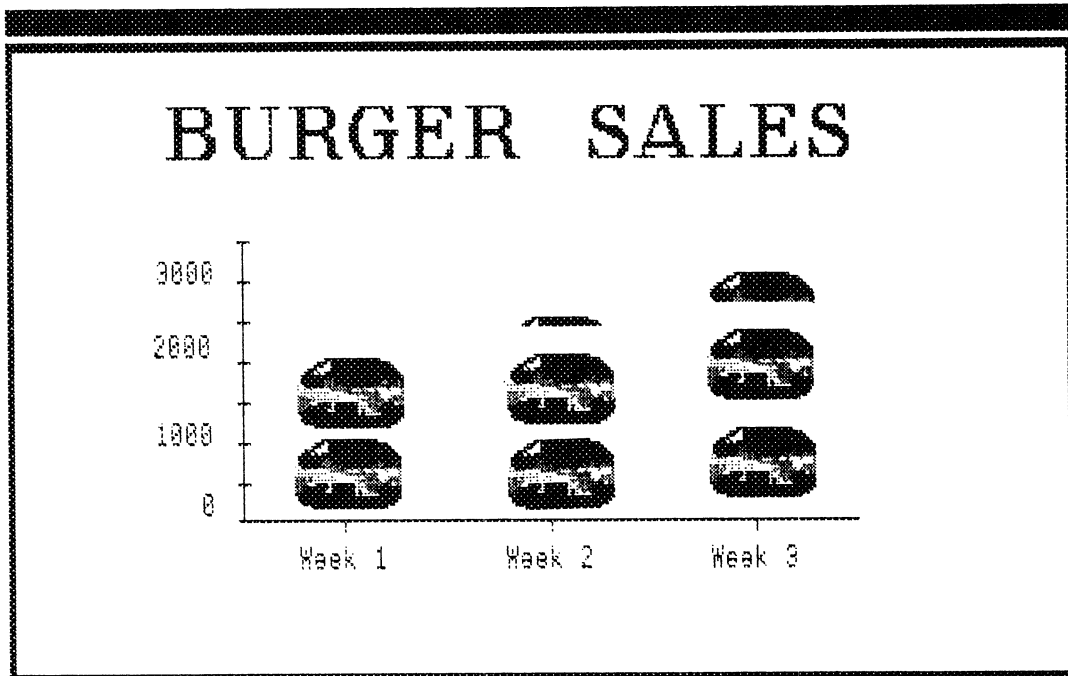
Week 1	1500
Week 2	2000
Week 3	3500

Close the series box.

**SELECTING THE CHART.** From *Graph*, select *Bar* chart. Choose the upper left basic bar chart. Select *Okay*.

**CLEARING THE GRAPH.** Select *Draw* from the *Shapes* menu. Set the active color in the *Colors* menu to red. Select the upper right brush from the *Brushes* menu. Place a box at the top of each of the bars to denote how tall they are. Then, use *Delete* under *Region* in the *Edit* menu to *Delete* all of the bars and the legend “Burger Chart.” Now there are no bars, just marks to show where they were.

FIG. 3.4 An icon chart.



**SIZING THE GRAPH.** Use the sizing gadget to reduce the size of the graph to about one third of the page. You should have a blank graph, except for the red marks.

**PLOT THE GRAPH.** Choose *Plot* and place the reference square in the middle of the page. Click on the right button to remove the reference frame.

**ADDING TEXT.** Use the “Creating the text” procedure described in the “Creating a Bar Chart” example to place the title and the legend “Number of burgers sold in sample weeks.”

**CREATING THE ICON.** Next, use the Icon Builder to create the burgers for the chart. Under *Project*, choose *New/Icon*. The Icon Builder will appear.

**ADJUST THE COLOR PALETTE.** The default color palette doesn’t contain the brown you’ll need for the bun, so go to the *Color* menu and choose *Edit*. Within the edit box, click on a color you won’t be using and move the sliders until you make a brown you like. Select *Okay*.

**DRAW THE BURGER.** In the drawing area of the Icon Builder, start by matching the background color of the icon to the background of the graph by clicking on whatever color your background is in the color bar, clicking on *Fill*, and clicking within the drawing area. The area will fill with the chosen color.

Use *Draw* and click on brown to begin drawing the hamburger bun. Use green to add some lettuce, red for tomatoes, white for onions, and yellow for mustard. Make the burger as large as you'd like within the drawing area, it will actually be only as large as the smaller version to the upper right corner.

**SAVING THE ICON.** When you've finished, use the "Saving the slide" procedure described under "Creating a Bar Chart" to save the icon to a data disk (*Save/Icon*) with the file name "Burger icon." Now the icon is saved for future use.

**PLOTTING THE ICON.** To use the icon now, click on *Plot* in the Icon Builder. A reference frame will appear. Stamp copies of the burger icon onto the blank graph up to the height of the red marks. If any of the marks still show after adding the burger icons, paint over them with the background color and any brush.

To get partial icons, as shown in Figure 3.4, use the trick of reversing the order. Stamp the first burger on the mark. Then, stamp the icon box over most of the first burger and work down the column, adjusting the position as needed. Do any clean-ups with a small brush and the current background color.

**ADJUST THE NUMBERS.** Delete the .0 column from the left hand numbers column by painting over them with the background color, a small brush, and the line tool from *Shapes*.

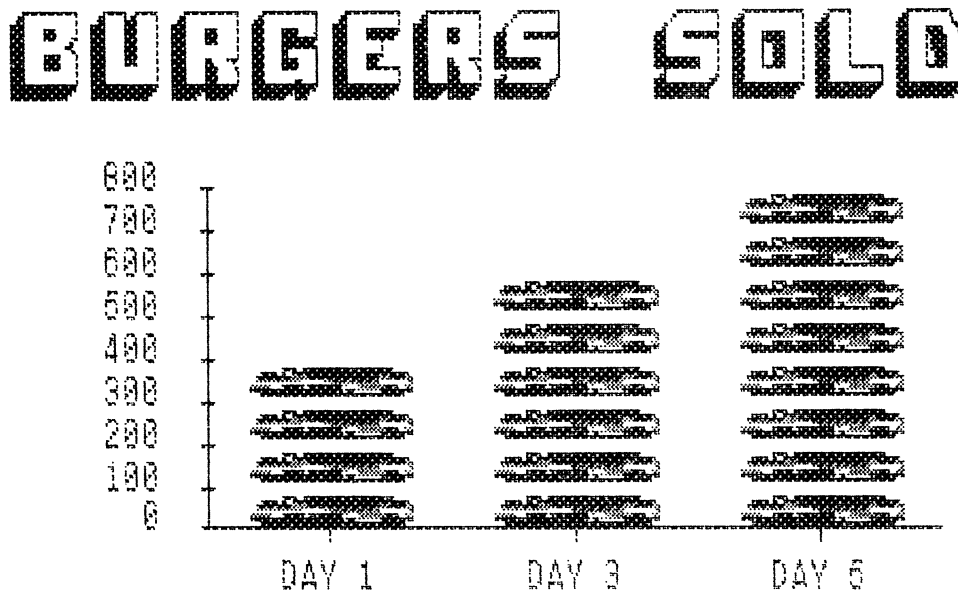
**ADDING A TITLE.** Use the procedure for "Creating the text" in the "Creating a Bar Chart" example to add a title with the Meteor font.

**SAVE THE SLIDE.** Save the slide with the title "Hamburgers."

## Automatic Graphing with Icons

**GRAPHING WITH ICONS.** Now that you have an icon saved to disk, you can use the automatic graphing function rather than stamping to make a graph with icons. An icon graph is shown in Figure 3.5.

FIG. 3.5 An automatically graphed icon chart.



**ENTERING THE DATA.** Select *New* and *Graph* from the *Project* menu. Enter the following data:

Day 1	400
Day 2	600
Day 3	800

Close the series.

**SELECTING THE ICON.** Select *Icon* from the *Graph* menu and type in the file name of the icon "Burger icon." Select *Okay*. The program will use the icons for the graph. You'll see, however, that they are not exactly like the burgers you originally drew; they'll be thinner.

**FINALIZE THE GRAPH.** Using the techniques you've learned, finalize the legends, plot to a slide, and add a title using the *Shadow* font.

## The No-Pest Chart

The "No-Pest Bug Spray" chart shows another way to use icons to express a message in personalized graphs.



**CREATING THE CHART.** Use the same method as in the Burger Chart to prepare a blank chart but this time make it horizontal by clicking on the “horizontal” box.

To make the chart (shown in Figure 3.6) easily readable, reverse the numbers on the lower axis by deleting the figures that automatically plot and retyping them.

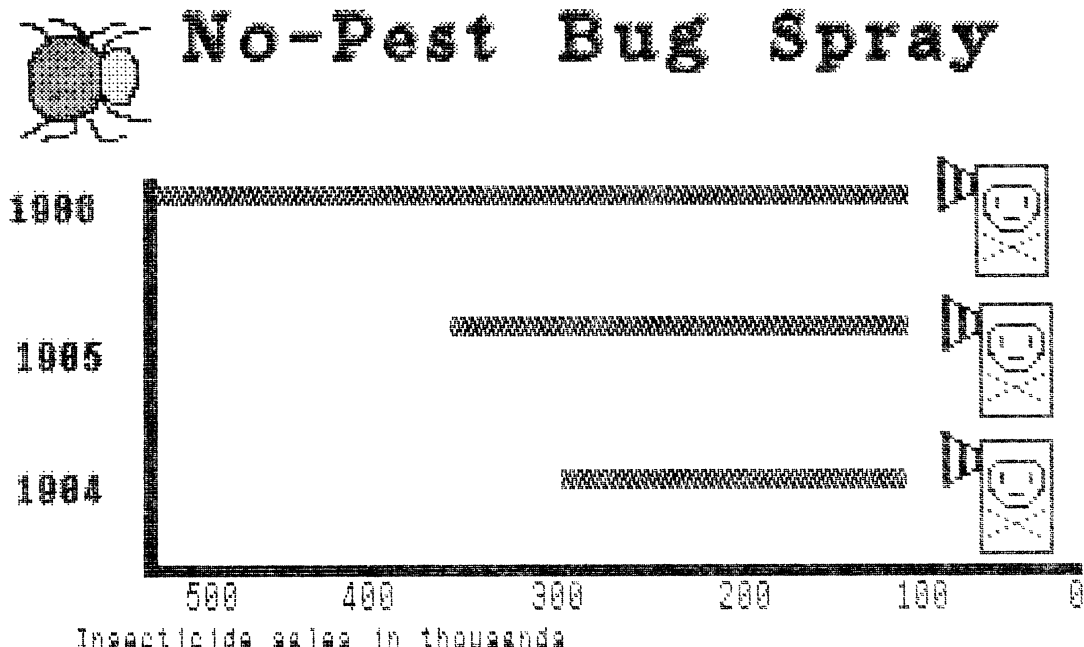
Create the can icon and stamp it in the appropriate places. Use *Line* from *Shapes*, a thick brush, and the yellow/black pattern from *Colors* to draw the spray lines.

Use a new Icon Builder to create the spider icon and add it to the graph.

### Creating a Slide Show

Practice making a slide show by using the example slides you’ve created. Put them together using the Show Builder. If you are using Impact!, Quit the program. Then from the Impact! Workbench screen, choose the SlideShow icon. (If you are using the CLI, type in SlideShow at the 1> prompt.)

FIG. 3.6 Using icons for unusual charts.



**USING SLIDESHOW.** Select *New* and *Show* from the *Project* menu. Set the data drawer with *Drawer* from the *Project* menu. When the directory of slides appears, select them in the order you'd like them to appear and adjust the settings for each one.

The settings specify the type of wipe you want to use, the delay of how long the complete wipe takes, and the pause time the image will be displayed.

For instance, put the first slide you created into the show by clicking to activate *Add*, clicking on the title "Tip-Top Sales," highlight *Down* for the type of *Wipe*, activate the *Pause Time* box and type in 6, and set the *Wipe Delay* to 1. Follow that procedure for each of the other slides.

**PLAYING THE SHOW.** Play the finished slide show by choosing *Auto* ( . . . ) from the *Show* menu. Press the space bar to stop the show. Close the show window by clicking in the close gadget.

**SAVING THE SHOW.** Use *Save* and *Show* from the project menu to save the show to a data disk.

**MAKING A SHOW FOR DISTRIBUTION.** To make a slide show to distribute to non-Impact! owners, copy *SlideShow* onto a disk containing the disks and show file for the slide show. Do not copy *Impact!* onto this disk, copying only the *SlideShow* program allows you to distribute the disk without giving away the program.

## PRESENTATION

---

After you've created charts, graphs, and slide shows, you'll need to output the product in a form suited to your business, budget, and audience.

Your options include using:

- Printers or plotters to produce paper hard copies of graphs or charts
- A 35 mm camera to shoot images from the screen
- A film recorder that captures images directly from the screen onto film
- Color separations created directly from disk files by an outside service

- Slide shows presented on the Amiga or a larger group monitor
- Slide shows taped onto VCR cassettes
- Graphics and animation sequences overlayed onto preexisting videotape

See Chapter 12, "Screen Reproduction Techniques," for further information on the above techniques.



# **AegisDRAW— A Graphics Processor**

Just as a word processor inserts, deletes, and modifies words, sentences, and paragraphs, AegisDRAW inserts, deletes, and modifies graphics elements. Unlike painting programs, which keep track of nothing but colored pixels, DRAW retains the identity of every element within every object, whether the elements are visible or hidden. DRAW also stores information on the positioning, attributes, layering, and visibility of each element.

## **GRAPHICS PRIMITIVES**

---

The simplest elements used to build objects are called *graphics primitives*. “Primitive” describes their role as the basic building blocks of more complex objects. Lines, rectangles, polygons, points, arcs, circles, and text, for example, are primitives.

Each graphics primitive acquires a specific set of attributes as you draw it. For example, a line has a specified weight (thickness), a color, and a pattern. As each primitive is drawn, it acquires the complete set of current attributes defined by AegisDRAW as defaults, or selected by you from pulldown menus.

## DRAWING STRATEGIES

Unless you're creating only the simplest of drawings, you'll need to plan your drawing strategy before you start. In many cases, a little forethought will help you to avoid unnecessarily limiting your drawings. Formulate your strategy to use the powerful features of AegisDRAW, but recognize its limitations.

As you learned in Chapter 2, AegisDRAW is an object-oriented graphics editor designed to manipulate distinct shapes. The way your shapes can be manipulated depends entirely on how you define them. When you create an object as a collection of shapes, keep in mind how the shapes will be assembled and used to make the object. A simple example will illustrate this point.

### Two Ways of Drawing a Cube

Visually, the two cubes shown in Figure 4.1 are identical. To see how they were constructed, their shapes can be dragged apart using the *Drag*

FIG. 4.1 Visually identical cubes.

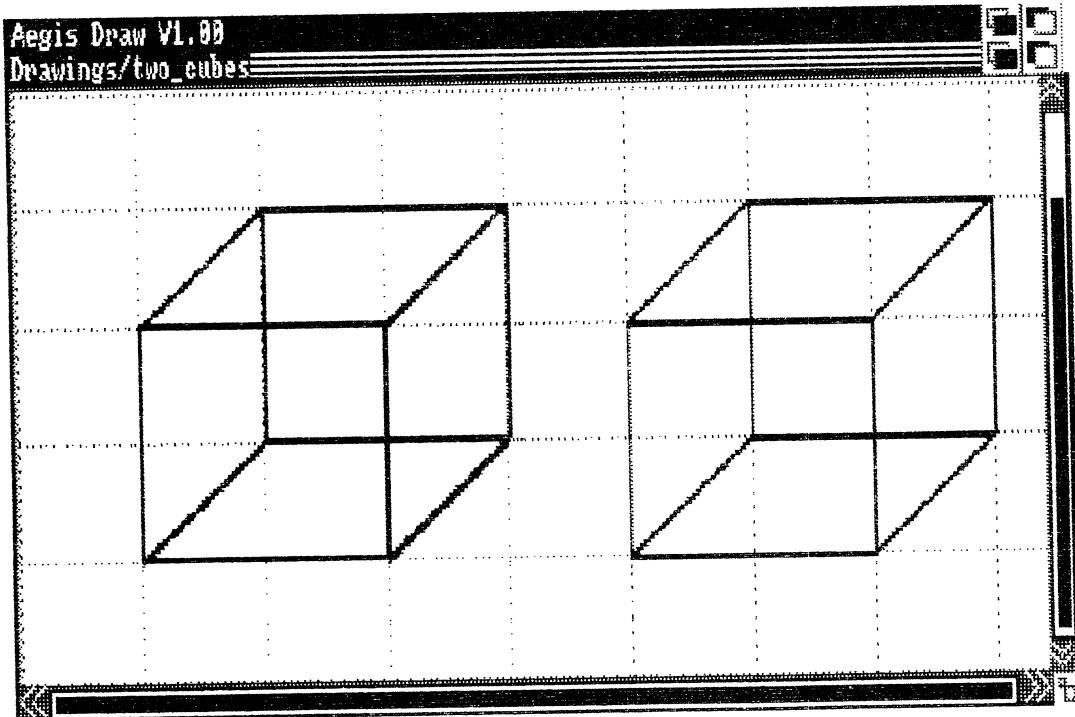
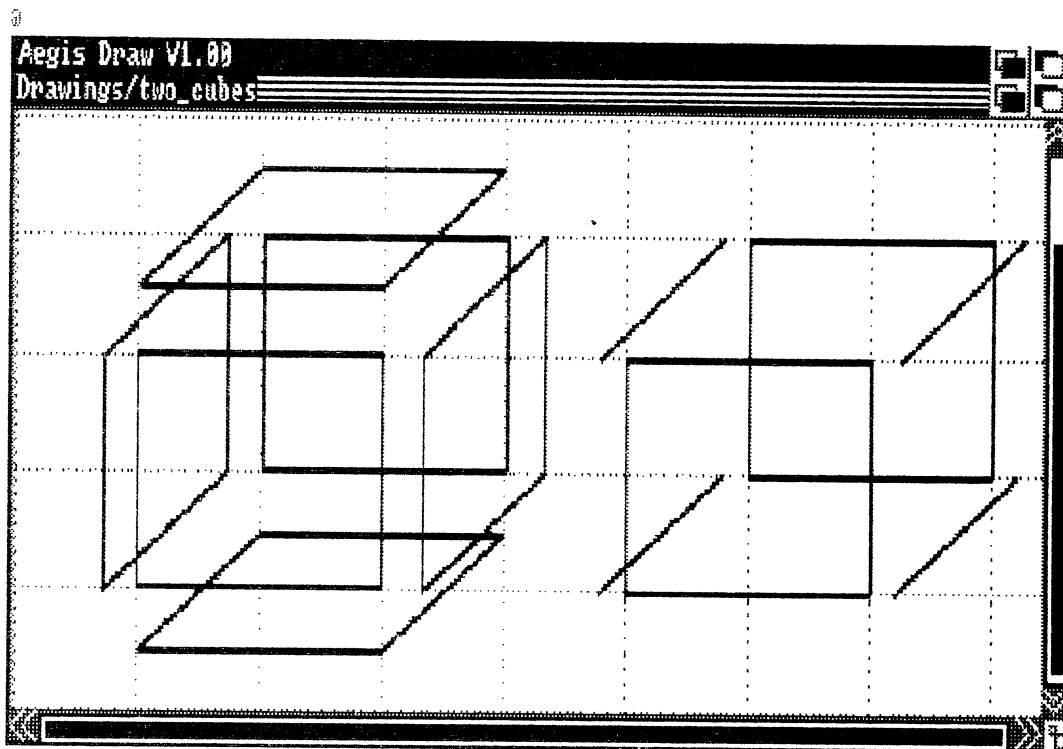


FIG. 4.2 Structurally different cubes.



it option from the *Tools* menu. As you can see in Figure 4.2 the cube on the left is composed of six distinct faces, while the cube on the right is composed of two faces and four line segments. (It's also possible to build a cube using twelve line segments.) The proper way to build the cube depends on how you want to manipulate its components.

Suppose you want to solid-fill the sides to create a box. Clearly, the cube on the left allows you the freedom to fill each of the six faces independently. The cube on the right has only two complete faces, so only the front and back can be solid-filled.

### **Drawing a 3D Diamond Within a Box**

If you haven't already done so, open AegisDRAW. Use the sizing gadget in the lower right corner to enlarge the drawing surface to its maximum size. Select *Ruler On/Off* from the *Display* menu to toggle the rulers off. Select *Grid Size* from the *Options* menu and modify the x and y values to 20.000. If you don't like the current grid pattern, choose a new

one from the *Options* menu, but remember that the grid is there for your own reference. Don't make it any bolder than you need to, or it may draw your attention away from your drawings. Check the *Grid Snap* option under the *Preferences* menu to make sure it's on. The *Grid Snap* feature will help you to position the sides of the box precisely by forcing each vertex to lie exactly at the intersection of the grid lines closest to where you position the cursor.

To finish setting up the graphics environment, you can select many options at once by holding down the right mouse button (the menu button) while clicking on the desired features with the left mouse button (the selection button). When you release the menu button, all the selected options will be implemented. You can practice this by clicking on a *Line Weight* and *Line Pattern* from the *Options* menu and a color in the *Color* menu. Don't release the menu button until all three options have been selected.

**DRAWING THE BOX.** Now that the graphics environment is right, draw the first (left) face of the cube as a closed polyline. To do this, click the left mouse button at the first vertex of the face, move to the next vertex, and press the left mouse button again. Continue this procedure, clicking only once at each vertex, until you position the final line to complete the face. At the last point, double click the left mouse button. The first face will have the line weight, pattern, and color you've chosen. Figure 4.3 shows the first face of the cube.

Note: if you make a positioning mistake but you haven't yet double-clicked the left mouse button, then you can press the right mouse button to cancel what you've done. You can use the *Undo* option under the *Edit* menu to remove the last object drawn. The *Eraser* option under the *Tools* menu lets you remove any object from the display.

After you've drawn the first face, draw the bottom face, as shown in Figure 4.4, using the same procedure. Make sure you define the bottom face as a closed, four-sided polyline. If you use only three lines, you won't be able to fill the face later on.

Select *Rectangle* from the *Tools* menu and draw the back face, as shown in Figure 4.5 by clicking at any of its corners and double-clicking at its diagonally opposite corner.

Select *Clone* from the *Tools* menu. Make a copy of the left face by clicking on one of its lines. Choose a line whose position is not shared by any other lines. Then position the copy as the right face and click again. (If you select the wrong face, remember that you can press the right mouse button to cancel the selection.) Figure 4.6 shows the right face in position.



FIG. 4.3 Draw the left face using the line tool.

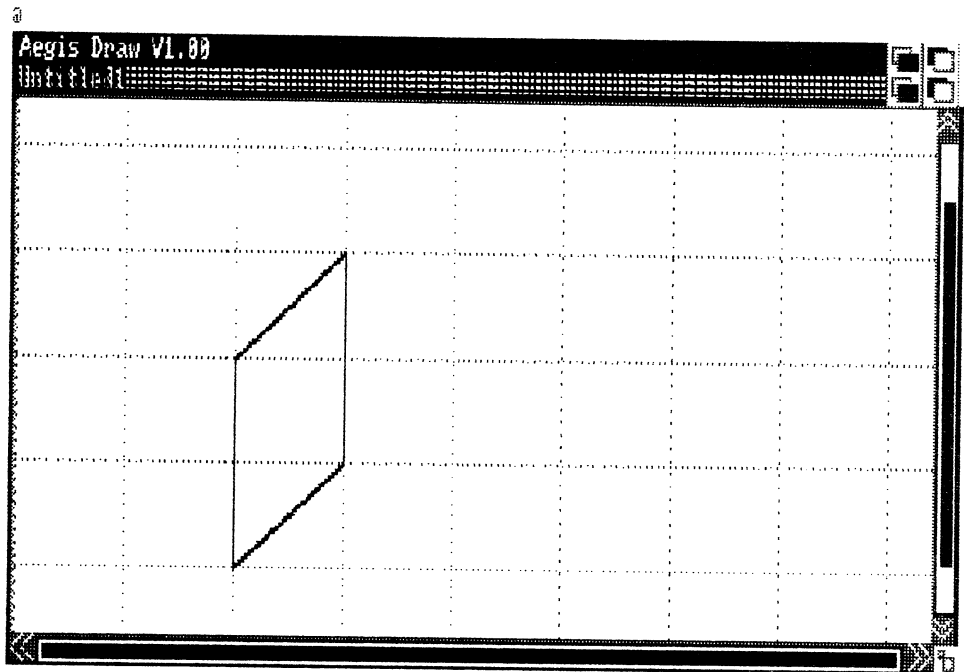


FIG. 4.4 Draw the bottom face using the line tool.

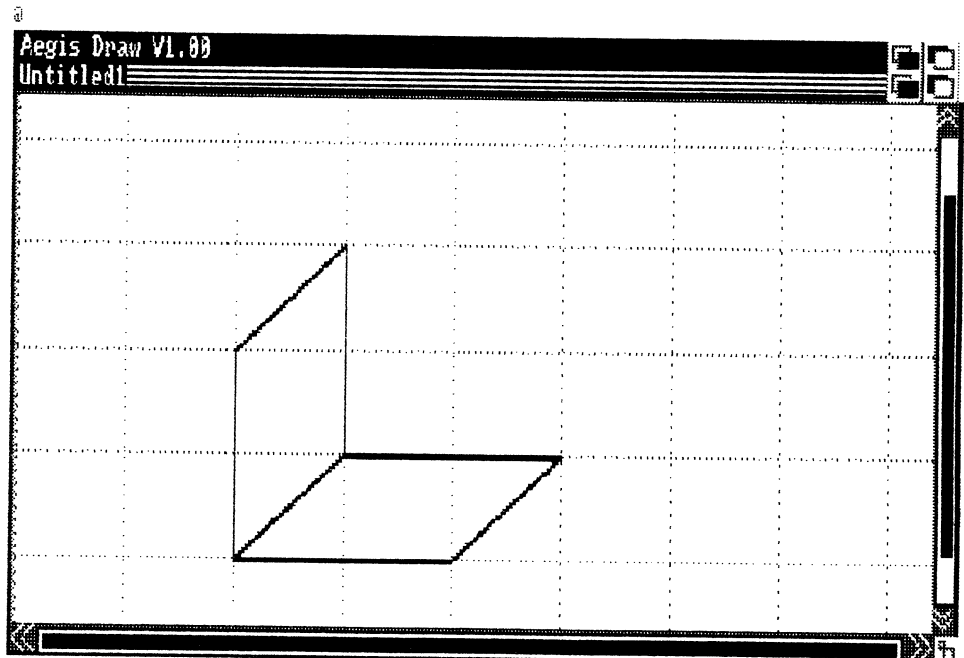


FIG. 4.5 Draw the back face using the rectangle tool.

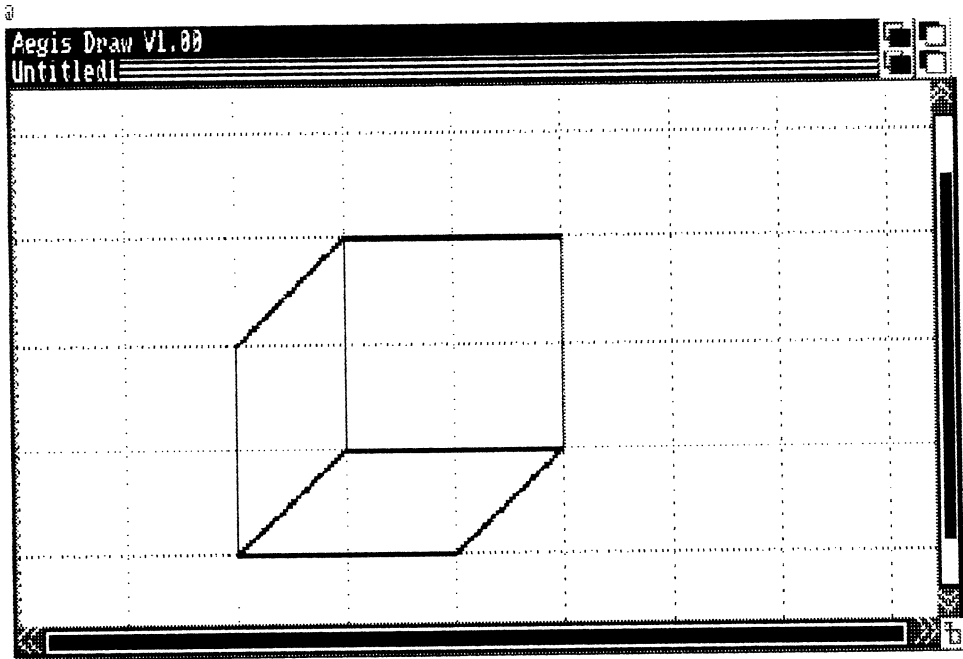
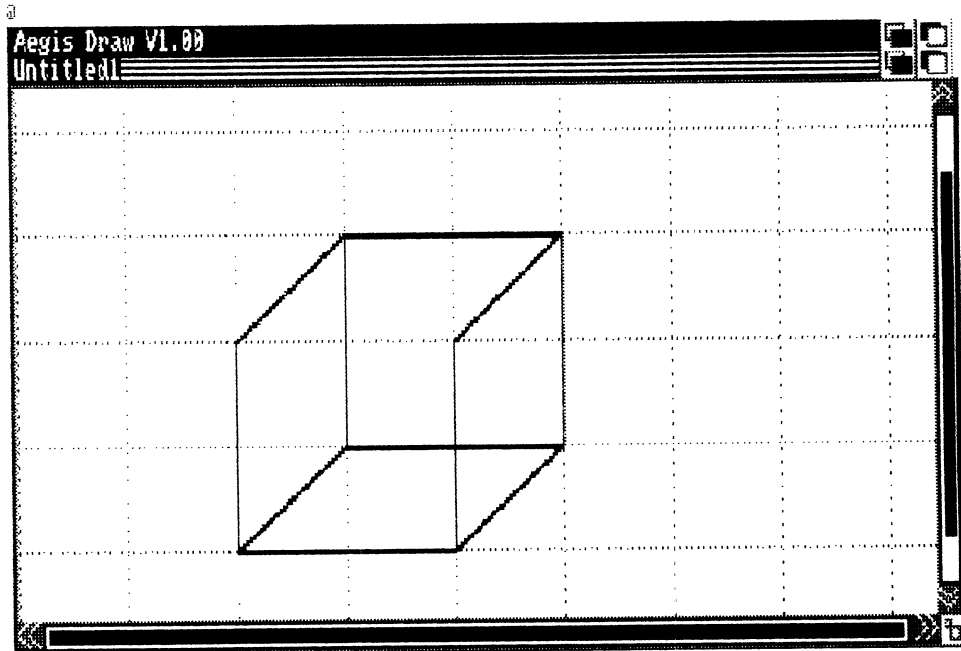


FIG. 4.6 Create the right face as a clone of the left face.



Create the front face as a clone of the back face to get the result shown in Figure 4.7.

At this point, the cube appears complete. But don't be fooled! Only five faces have been drawn. To prove it, turn off *Grid Snap* and select *Drag it* from the *Tools* menu. Move the sides away from each other as shown in Figure 4.8 by clicking on them and dragging.

Create the top face as a clone of the bottom face. Figure 4.9 shows the top face in place.

Now that the cube has all its faces, it can be transformed into a box with the addition of solid fill colors. To solid-fill a face, select a color, turn the *Filled* option under the *Preferences* menu to *On* and select the solid fill pattern from the *Options* menu. Pick the *Color* option from the *Tools* menu. This option looks at the current color, fill option, and fill pattern to see how an enclosed shape is to be modified. Select a face and fill it with a specified color (see Figure 4.10).

Repeat the procedure until all the faces are solid-filled with different fill colors, as shown in Figure 4.11. Use the *Drag it* tool with *Grid Snap* turned on to move the faces together (Figure 4.12).

FIG. 4.7 Create the front face as a clone of the back face.

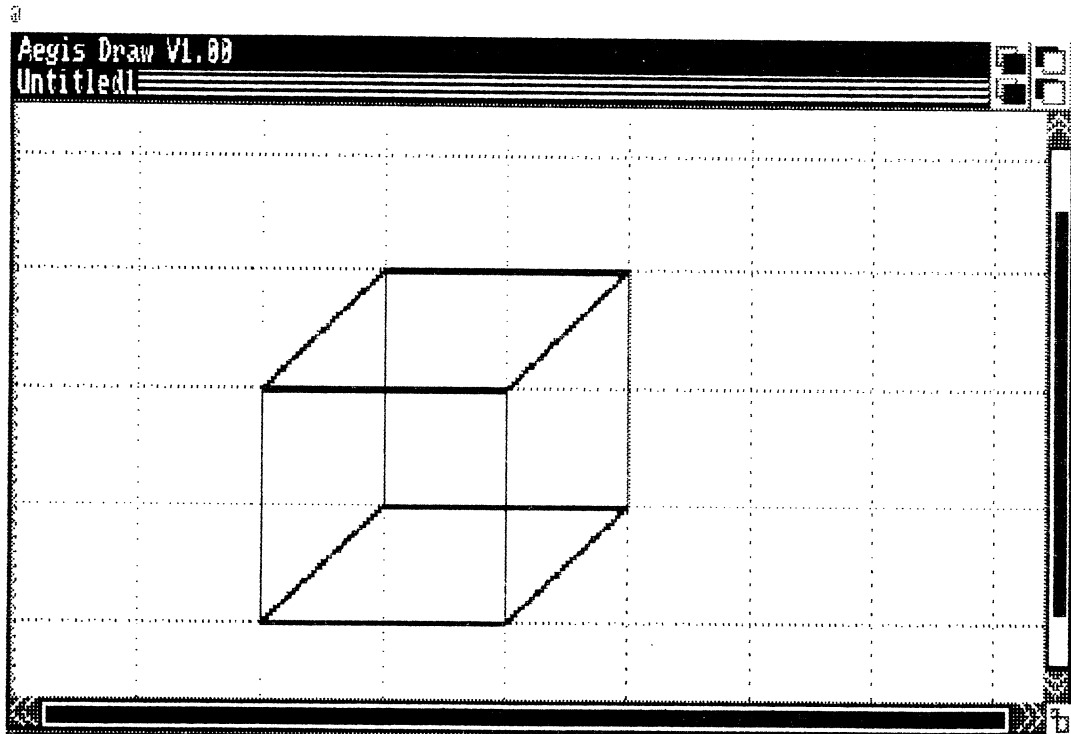


FIG. 4.8 Drag the faces apart

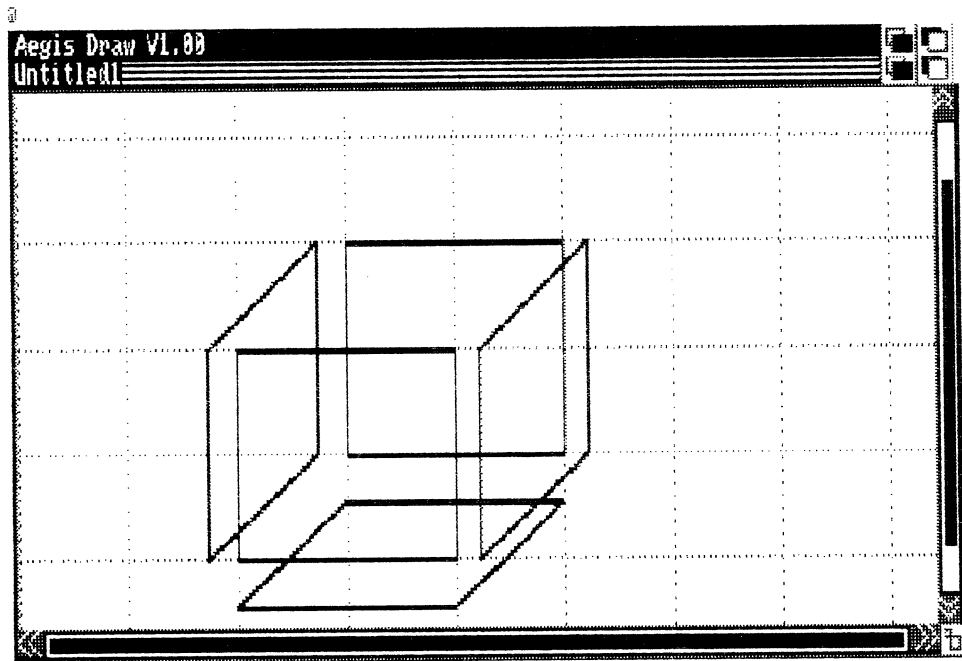


FIG. 4.9 Create the top face as a clone of the bottom face.

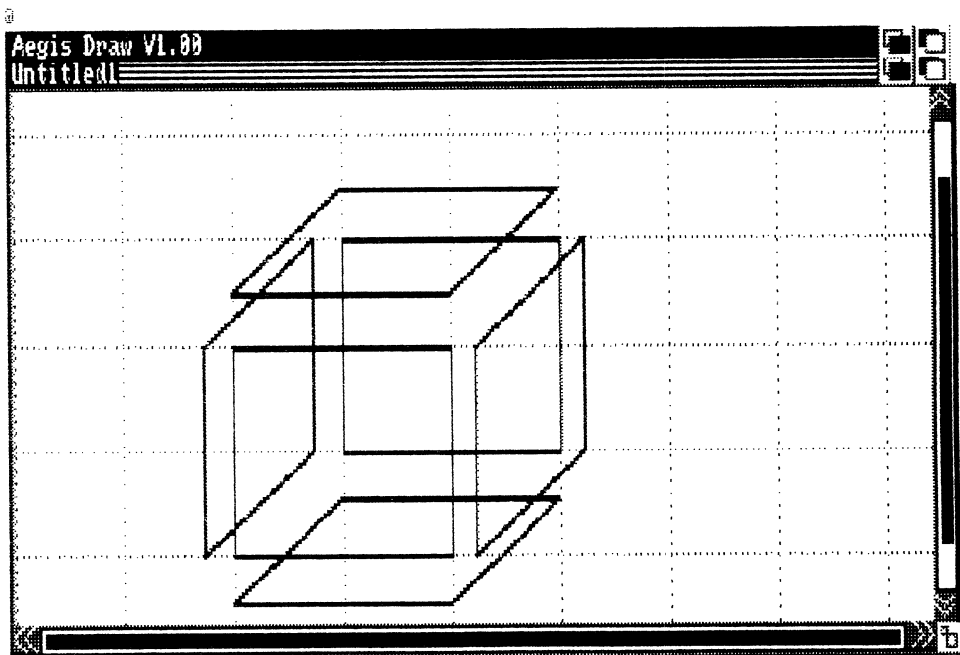


FIG. 4.10 Fill in the back face.

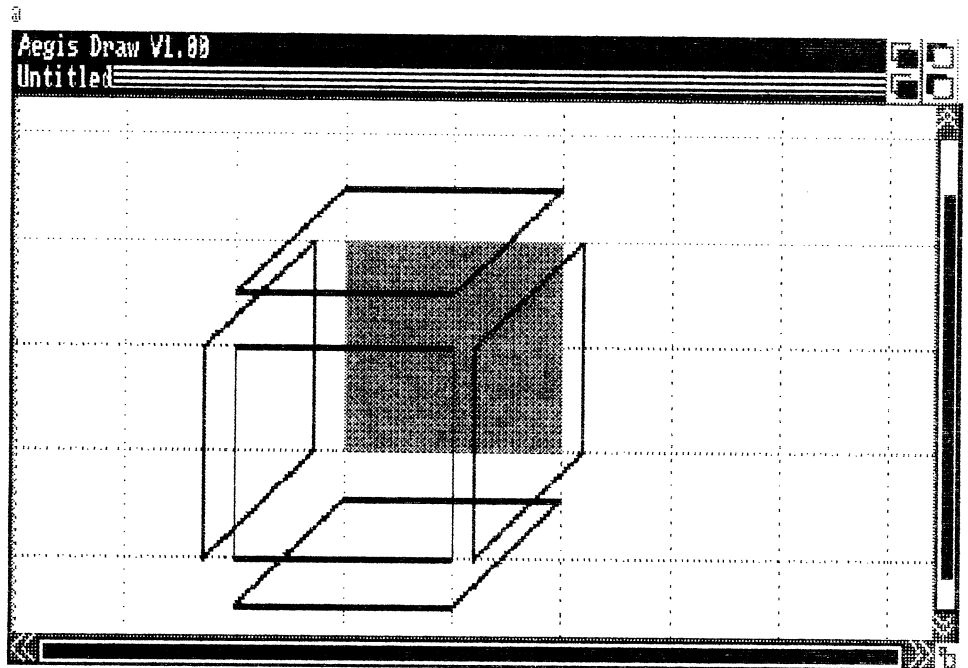


FIG. 4.11 Fill the other faces.

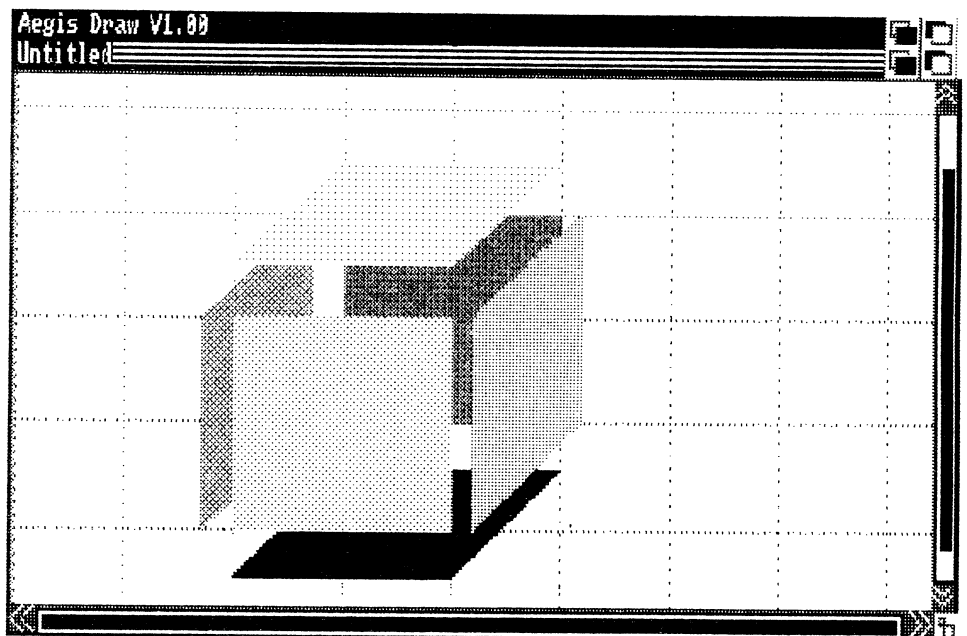
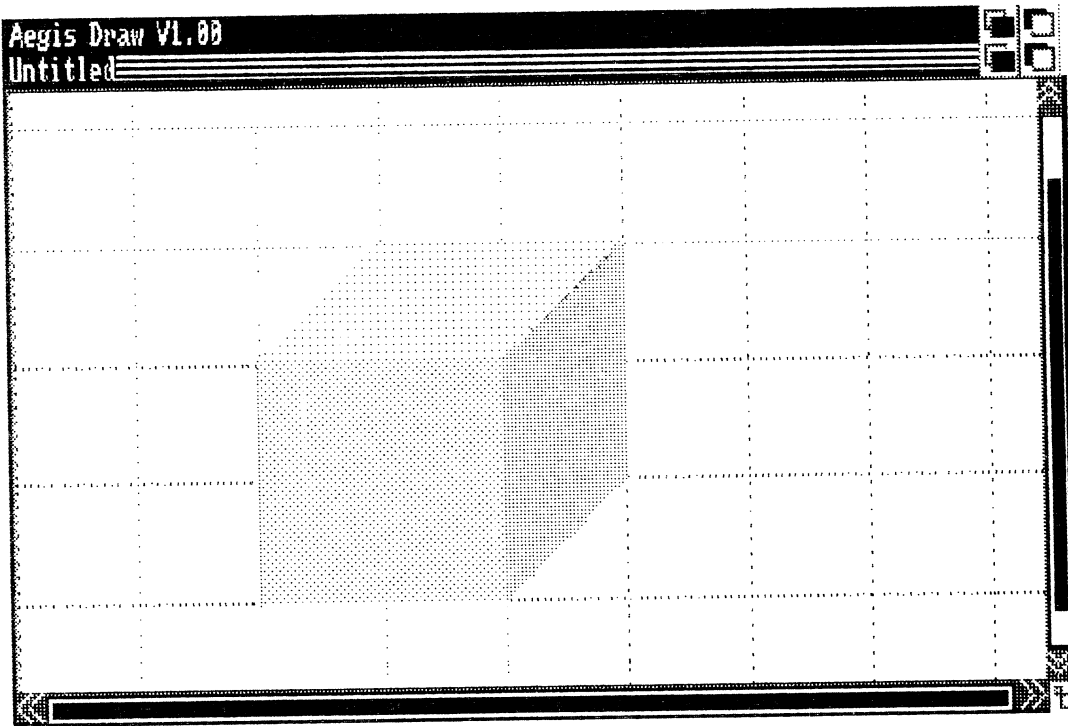


FIG. 4.12 Drag the faces together.



With AegisDRAW, the order of creation of elements becomes the default order of display. If you created the faces in the order specified in the example, then the back, left, and bottom faces will automatically appear behind the front, right, and top faces. If one of your cube faces is being drawn in front when it should be in the back, use the *Back* option under the *Tools* menu to change its relative position.

**DRAWING THE DIAMOND AND PUTTING IT INTO THE BOX.** To further illustrate the grouping of shapes and front/back positioning, you'll create a reference object. The next example shows a diamond composed of six solid filled triangular shapes. Filled shapes can be created immediately (without building a wire frame model first) by turning the *Filled* selection under the *Preferences* menu to *On* before creating the shape. Don't forget to select the desired color and fill pattern.

Draw the diamond in Figure 4.13 to the convenient scale provided by the grid marks.

To place the diamond in the box, you must first resize it to an appropriate scale. The simultaneous scaling of the six triangular shapes requires that they first be “grouped” together. This is accomplished with the *Group* selection under the *Edit* menu. Click and drag to surround the diamond with the selection rectangle. All of the objects completely within the rectangular area you specify will be grouped together as a whole. Use the *Sizer* tool under the *Tools* menu to resize the diamond (Figure 4.14).

The next step is to open the box by removing its lid. Do this by using the *Drag it* tool in the *Tools* menu. Your result should look like Figure 4.15.

Because the triangular shapes that make up the diamond have been grouped together with the *Group* operation, the entire diamond can now be dragged into the box (see Figure 4.16). Notice, however, that the diamond still appears to be in front of the box. This can be corrected

FIG. 4.13 Draw the diamond using a convenient scale.

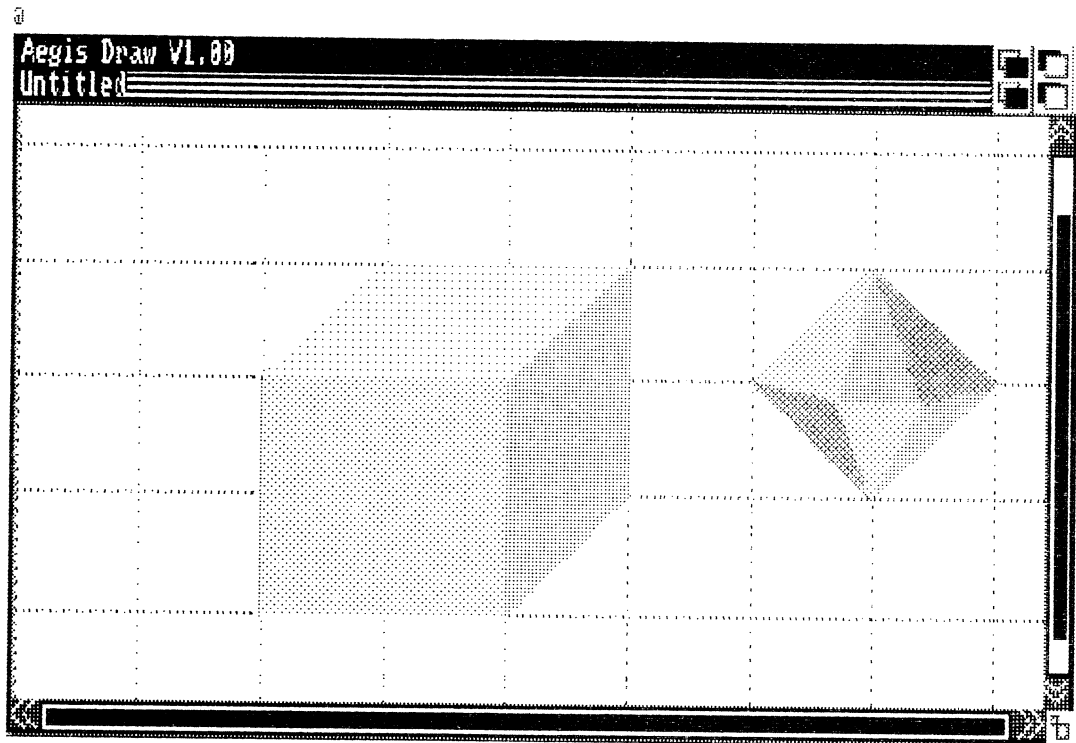


FIG. 4.14 Scale the diamond after grouping its faces.

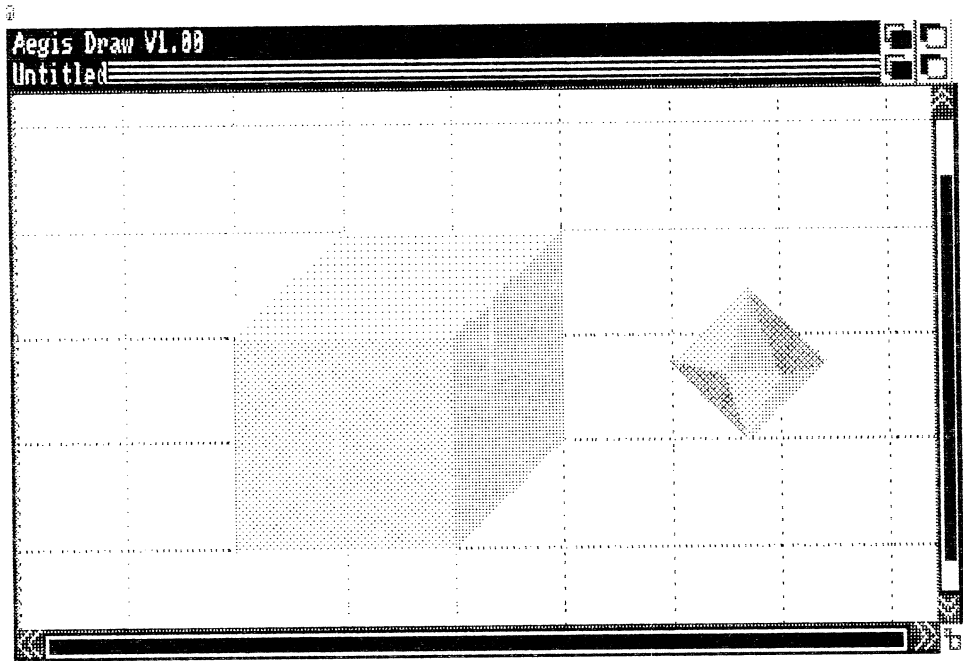


FIG. 4.15 Open the box.

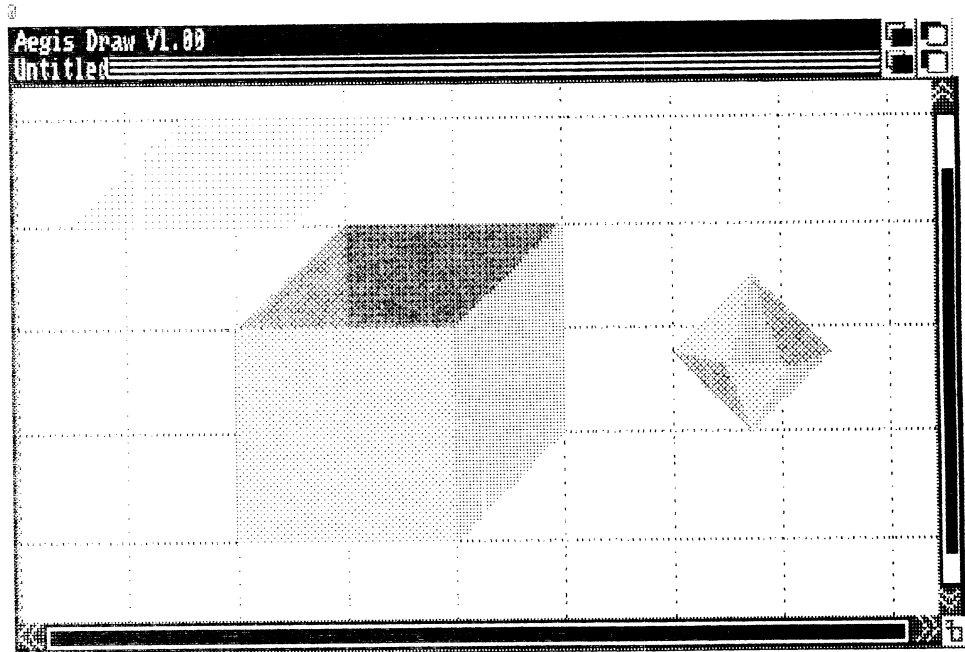




FIG. 4.16 Drag the diamond over the box.

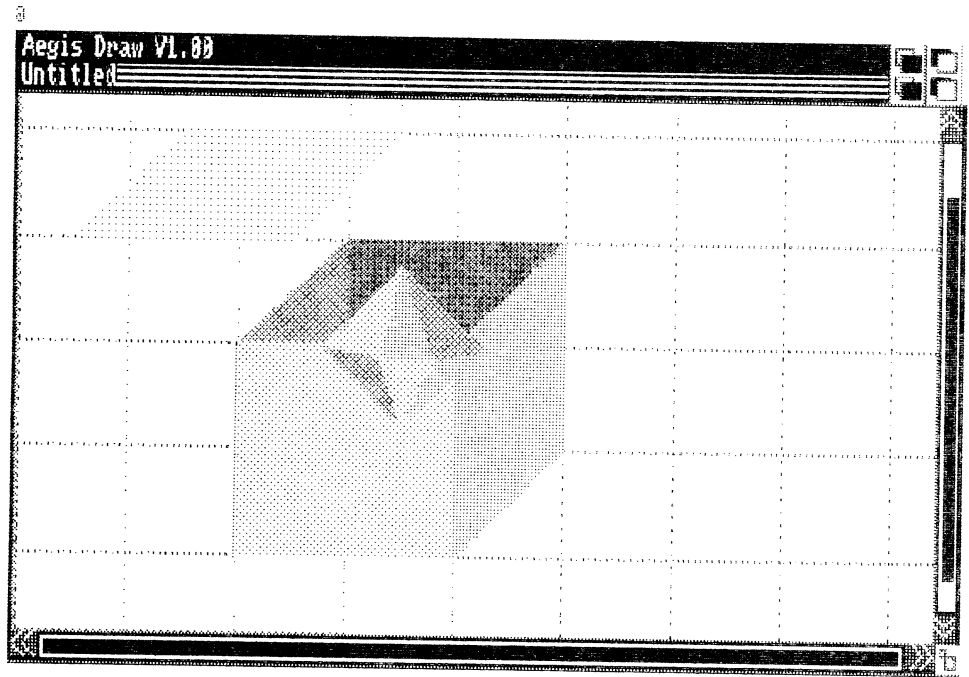
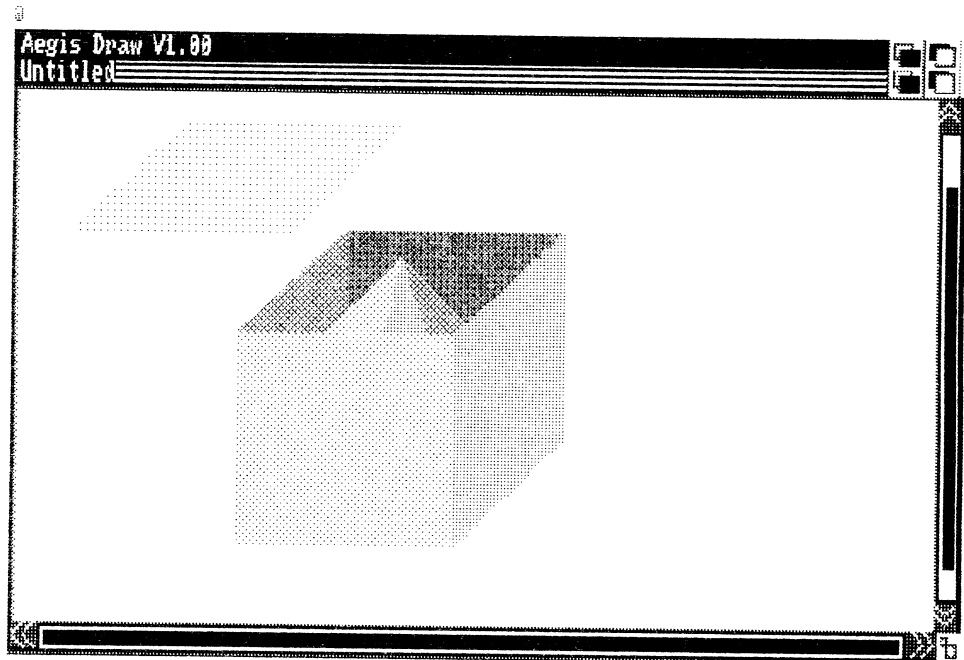


FIG. 4.17 Order the object drawing sequence.



using the *Backer* tool in the *Tools* menu. To draw the shapes in the correct order, click on the shapes in the following sequence: front cube face, right face, diamond, left face, back face, bottom face. (An easy way to derive a list like this is to create it in reverse order. Reading the list in reverse order indicates the back to front order of positioning.) Now turn off the grid. You should see the result shown in Figure 4.17.

You can save the entire drawing for future use as either a part or a drawing by selecting the *Save As* option in the *Project* menu.

The boxed diamond example illustrates many of the powerful features of AegisDRAW. By far the most important feature is the identity the program gives to every shape or grouping of shapes within the drawing. Being able to isolate and modify the shapes gives you a great deal of flexibility.

## PERSPECTIVE DRAWINGS

---

Adding perspective to three-dimensional images enhances the perception of depth in your drawings. As an object in the real world appears smaller the farther away it lies, objects created by AegisDRAW can also be scaled and positioned in relation to an imaginary viewer's eye. This section teaches you to draw three-dimensional objects using one-point perspective.

### One-Point Perspectives

Choose a convenient grid size and grid pattern, and then draw a line to represent the horizon. Place a reference mark somewhere along the line to represent the vanishing point. Select *Rectangle* from the *Tools* menu and draw two unfilled rectangles as shown in Figure 4.18 somewhere below the horizon.

Select a new color to draw lines connecting the four vertices of each rectangle with the vanishing point as shown in Figure 4.19. Turn the grid off and use the *Back* tool in the *Tools* menu to force the projection lines to be drawn behind the rectangles. If you have trouble picking the line you want, turn the *Grid Snap* setting in the *Preferences* menu to *Off* and try again.

Draw the top face on each of the cubes as a four-sided element. You can tell that the face is properly positioned when the projection lines disappear under the shape, as in Figure 4.20.

FIG. 4.18 Establish a horizon and a vanishing point

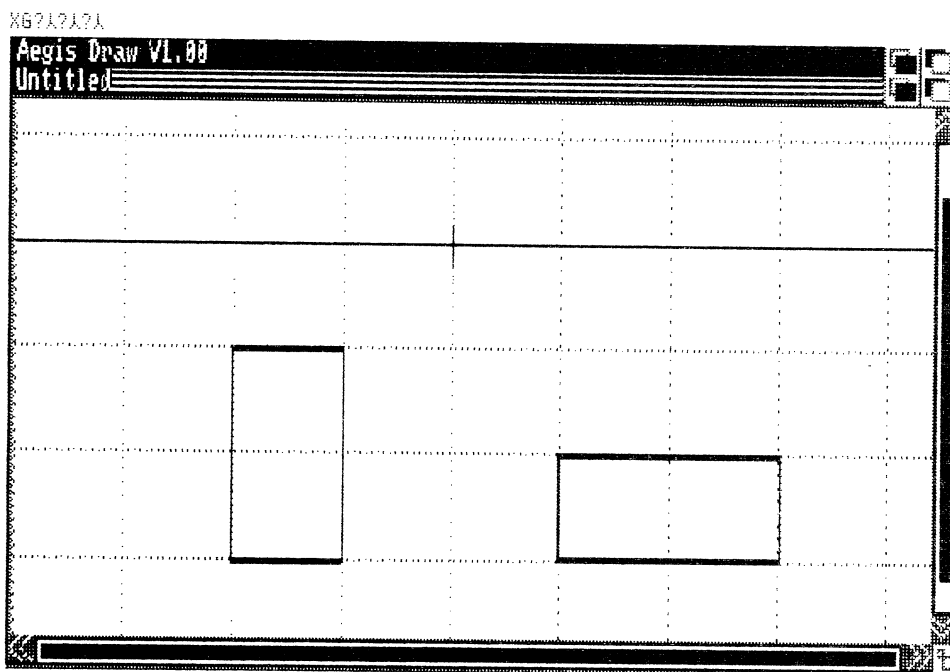


FIG. 4.19 Connect the four vertices of each rectangle with the vanishing point

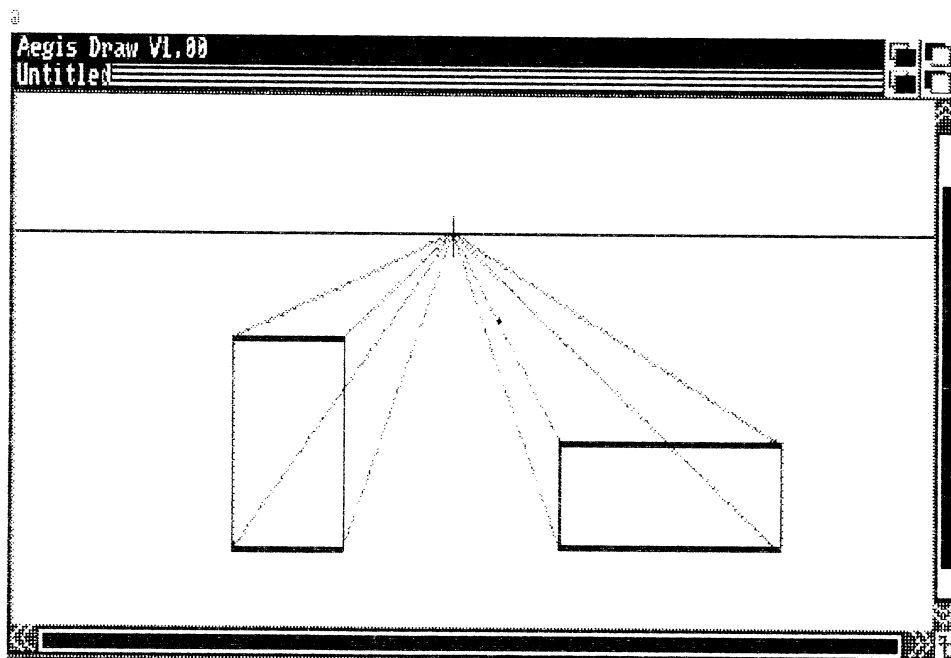


FIG. 4.20 Use the projection lines for reference to draw the top face.

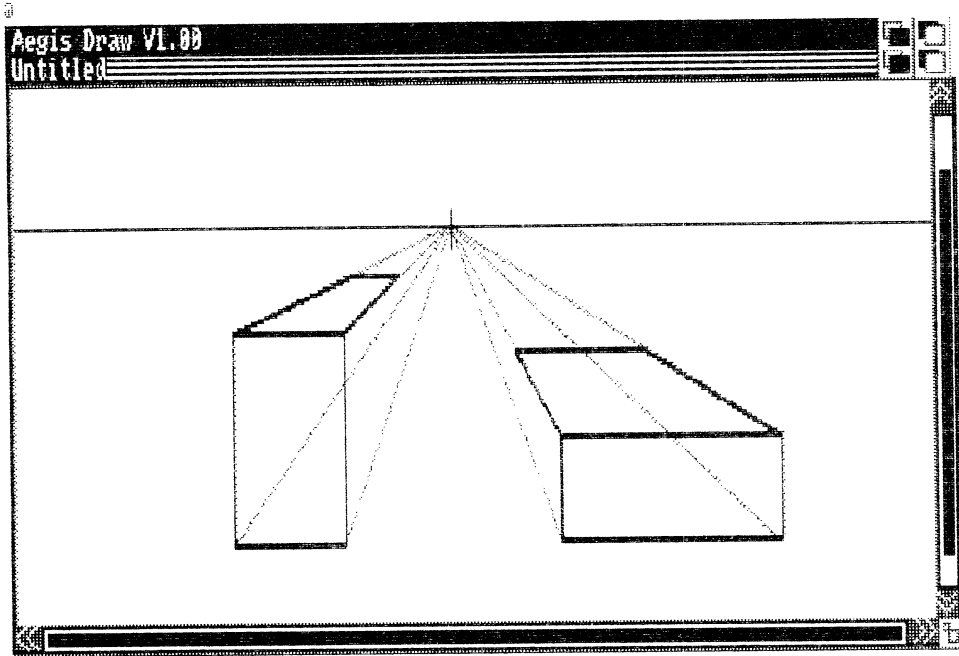


FIG. 4.21 Define the faces as enclosed polygons.

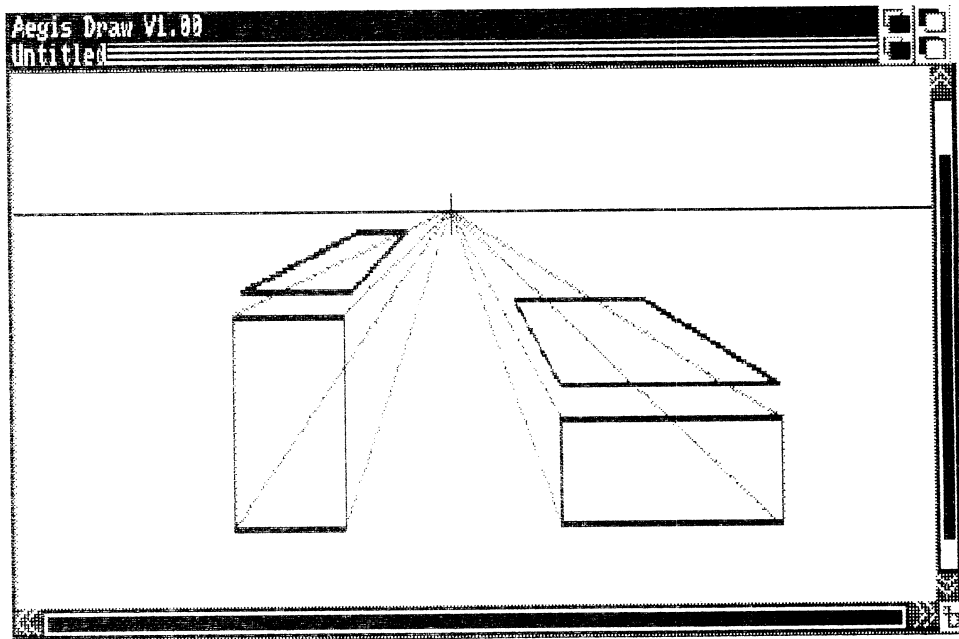


FIG. 4.22 The completed cubes.

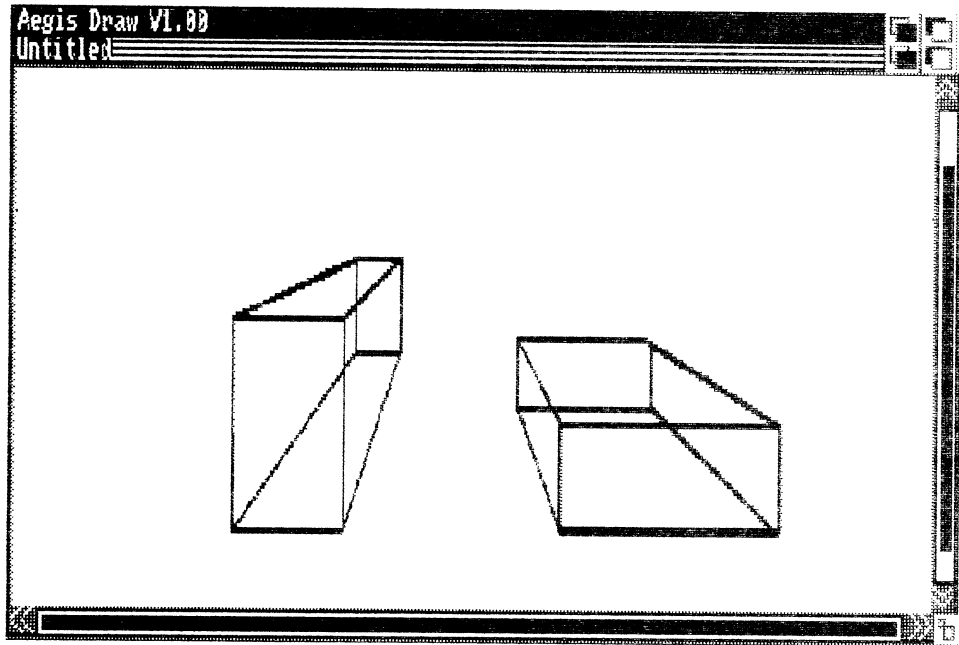
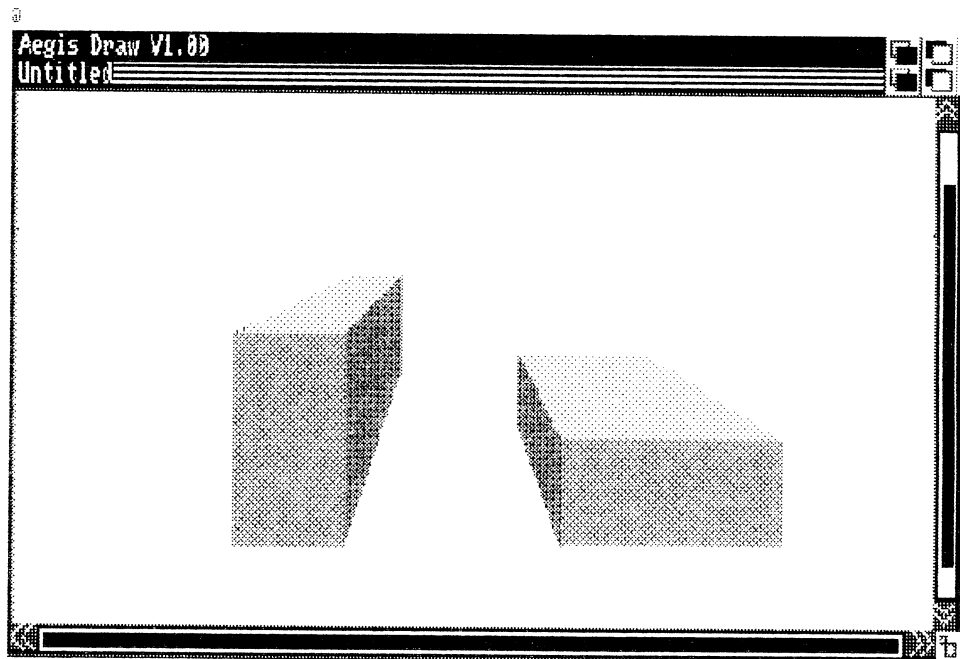


FIG. 4.23 The completed boxes.



The top faces are dragged away in Figure 4.21 to emphasize that four sides are used to define each one as an enclosed figure. This will be important when you fill the faces with solid colors.

Draw the other faces, and delete the horizon and projection lines using the *Eraser* tool from the *Tools* menu. Your drawing should look like Figure 4.22.

Use the *Color* tool from the *Tools* menu with the *Filled* option in the *Preferences* menu turned to *On*. Fill each of the faces with the desired color, as shown in Figure 4.23.

## SCALED DRAWINGS

---

AegisDRAW provides many convenient tools to help you create precisely scaled drawings for the design of real-world objects such as furniture and houses. The range of numeric values displayed by the rulers is under user control. Also, you can easily display the coordinates of any point and the distance between any two points.

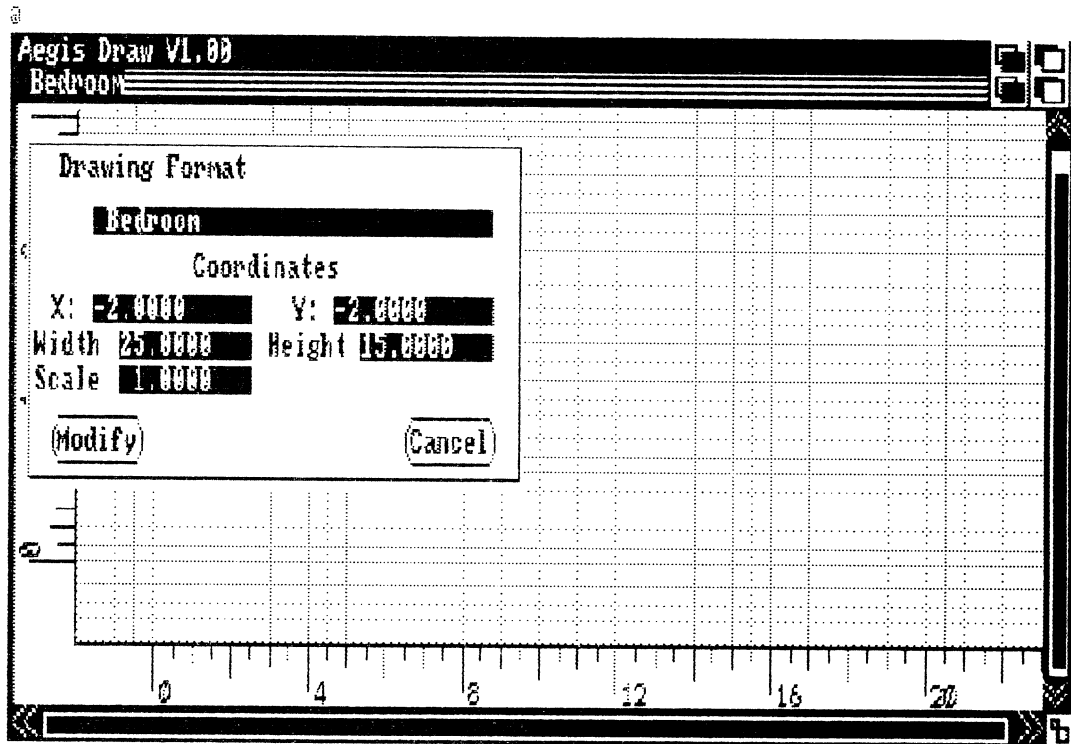
### Rearranging the Furniture in a Room

Rearranging the furniture in a room provides an excellent example for creating a scaled drawing. Trial-and-error furniture moving often results in more work than necessary, because you're likely to move each of the pieces more than once. Some arrangements might never even be attempted because of the effort involved. The tools provided by AegisDRAW give you a practical and fun way to try every arrangement you can think of, with a minimum of physical exertion.

**DRAWING THE SCALED ROOM.** Use a tape measure to measure the room you'd like to draw. After starting AegisDRAW, select *Format* under the *Options* menu and enter a drawing name and the x-y coordinates in the lower left hand corner. Choose a width and height large enough to enclose the room, but not so large that the room will occupy only a small fraction of the drawing surface. Also, leave some working room around the edges. The values shown in Figure 4.24 are based on a bedroom measuring 18.5 ft by 12.0 ft.

Click on *Grid Size* in the *Options* menu, and select a grid size that uses the required accuracy. The sample bedroom needs an accuracy only to the nearest half foot, so the default grid size of 0.5 is fine. If you want

FIG. 4.24 Formatting a drawing.



accuracy to the nearest inch, enter the decimal equivalent of 1/12 (0.0833333). Click on *Modify* to initiate the change (see Figure 4.25).

Select *Line* from the *Tools* menu, the thickest line weight from the *Options* menu, and black from the *Colors* menu. Click on *Numeric Display* in the *Preferences* menu. Using the coordinate (0,0) as your starting point, draw a line of the right length to represent one of the bedroom walls. Use the numeric display in the upper left corner to precisely select the endpoints. When the *Grid Snap* feature is on, the numeric display will automatically show the coordinate of the grid point closest to the cursor (see Figure 4.26).

Draw the rest of the room, remembering to include important features—windows, doors, closets, and so on. When the drawing is complete, you should have a precisely scaled floor plan, as shown in Figure 4.27.

**DRAWING THE FURNITURE.** Change the grid size as needed before constructing each piece of furniture. Don't worry about furniture

FIG. 4.25 Setting the grid sizes.

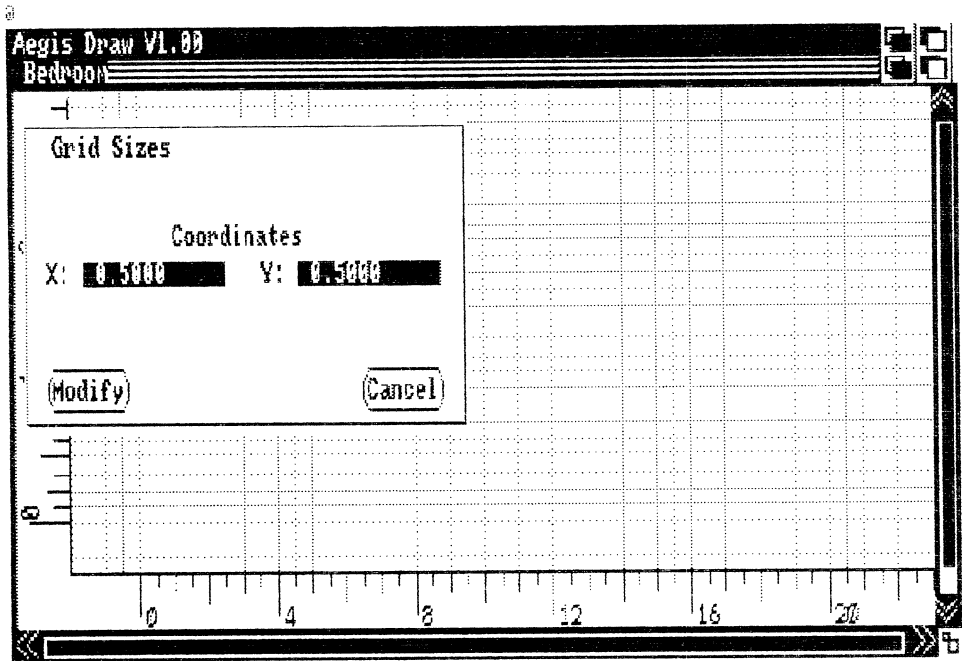


FIG. 4.26 The Grid Snap feature.

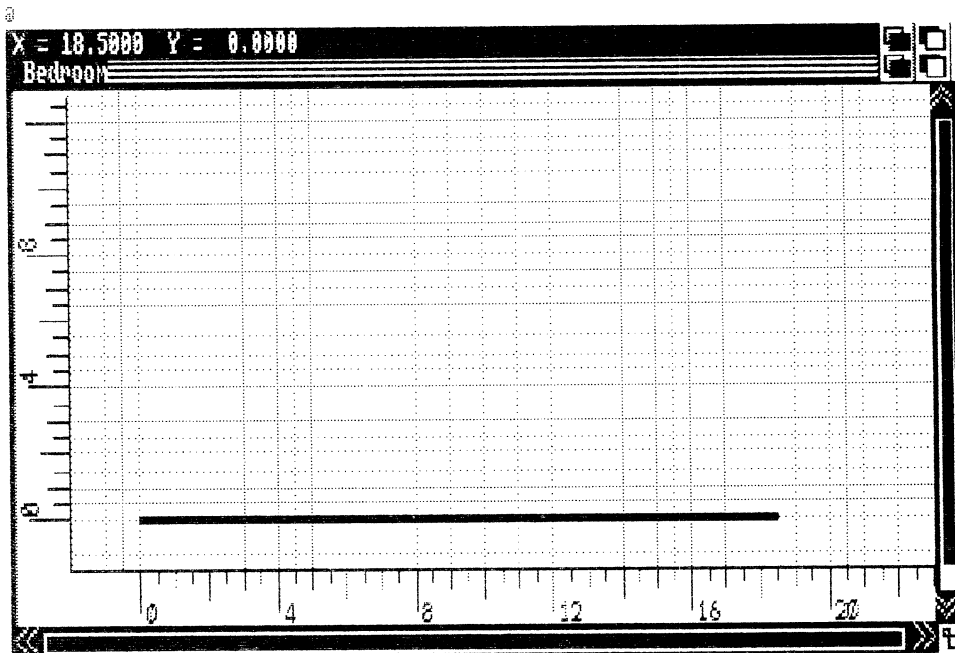
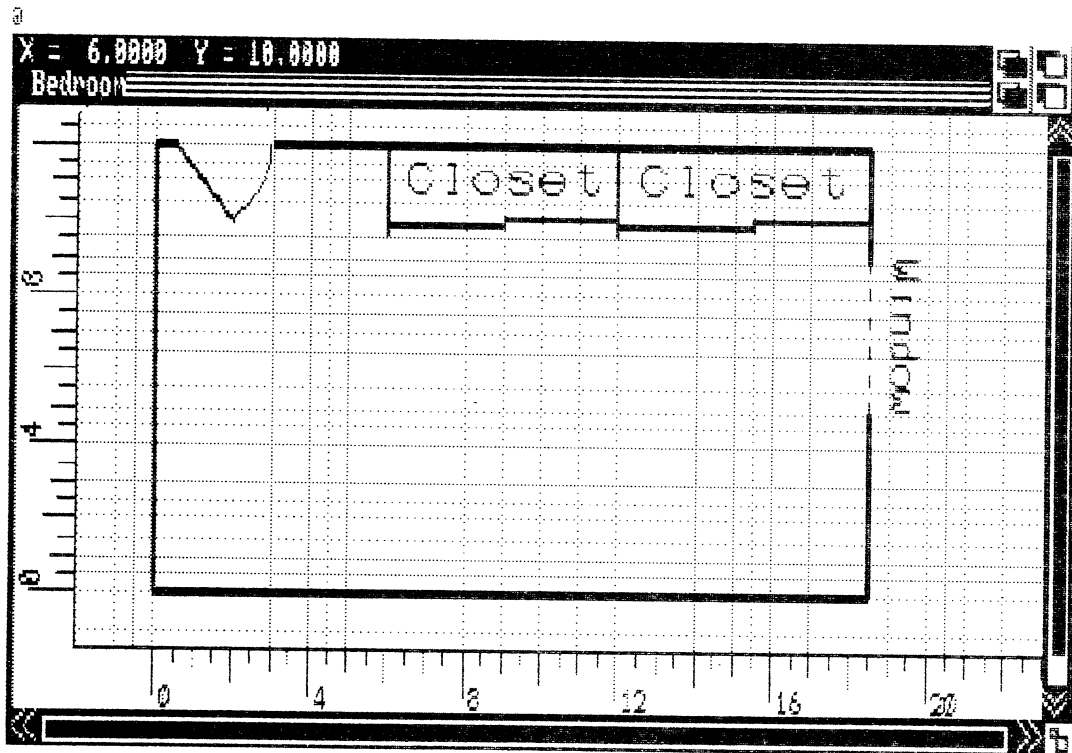




FIG. 4.27 The precisely scaled floor plan.



positioning or orientation at this point, just create the piece using the *Dimension* tool or zoom in on the piece and check its size against the rulers. For very small objects, it's often useful to zoom in before creating the object (see Figures 4.28 and 4.29).

Use the *Eraser* tool from the *Tools* menu to remove the dimension lines when each piece is complete.

Figure 4.30 shows the result after all the furniture has been carefully drawn to scale.

**GROUPING THE FURNITURE COMPONENTS.** Before you start moving furniture, remember to group the objects composed of more than one piece into a whole. For example, the bed is composed of four separate objects: the bed cover, a triangle representing the turned down corner, and two pillows. To collect all the objects comprising the bed into a unit that can be moved as a whole, use the *Group* command. All objects completely enclosed in the rectangular region you select will be grouped together as one piece.

FIG. 4.28 Using Dimension and Zoom.

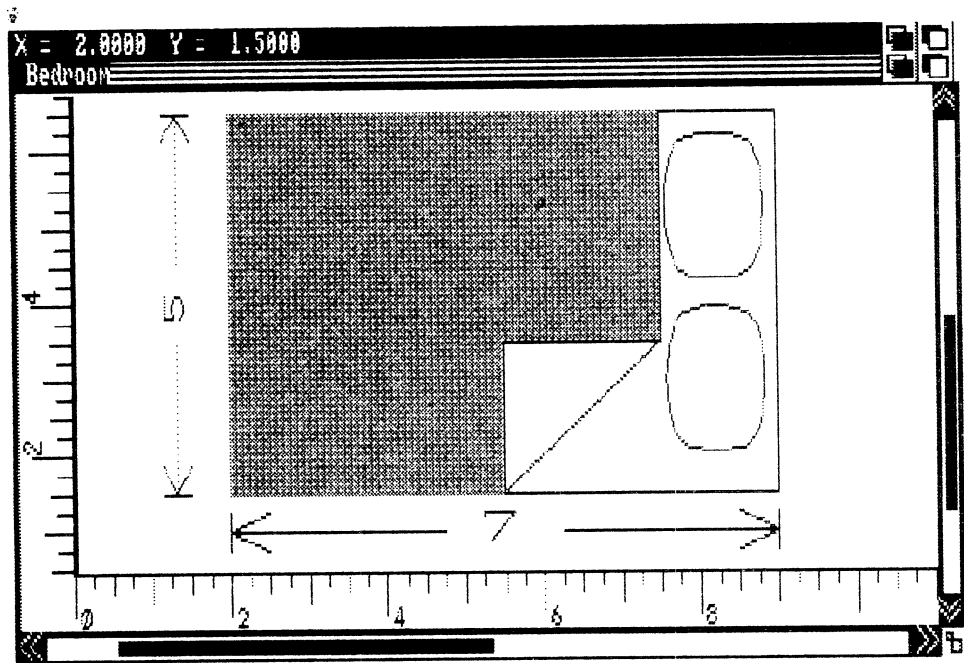


FIG. 4.29 For small objects, zoom in before creating the object.

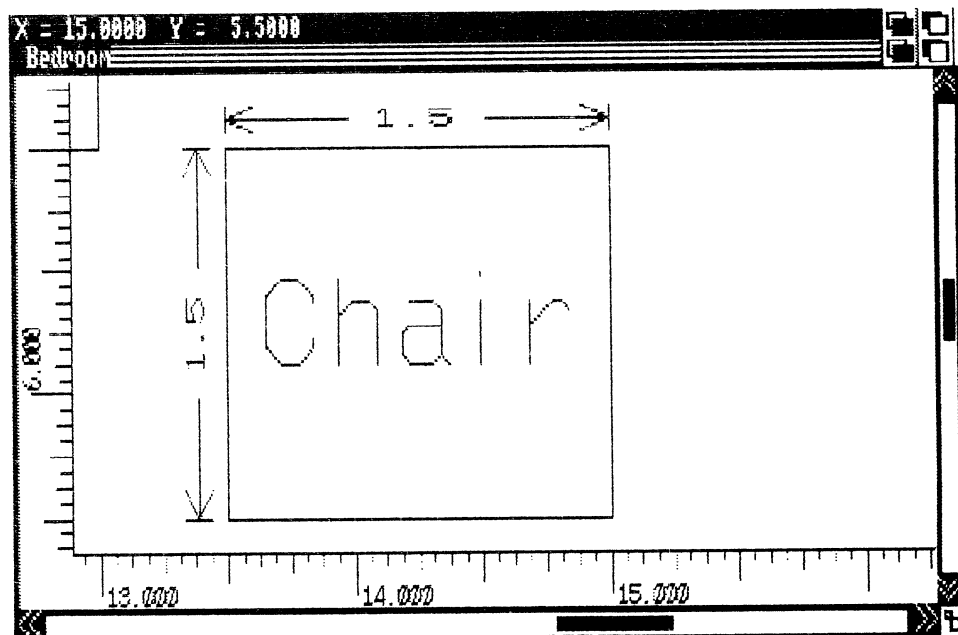


FIG. 4.30 The result after the furniture is drawn to scale.

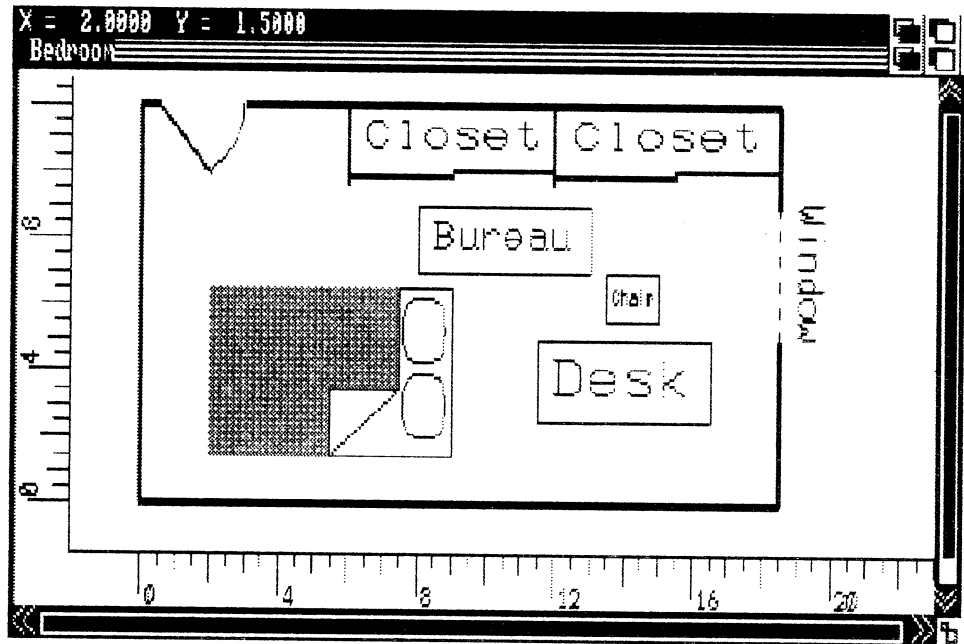


FIG. 4.31 Moving the furniture—arrangement 1.

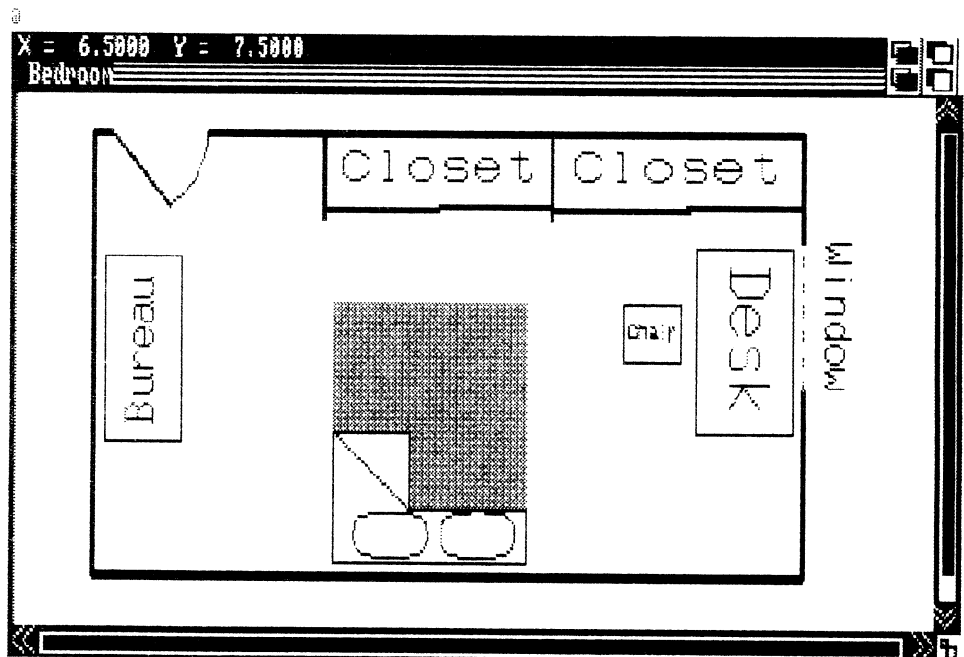
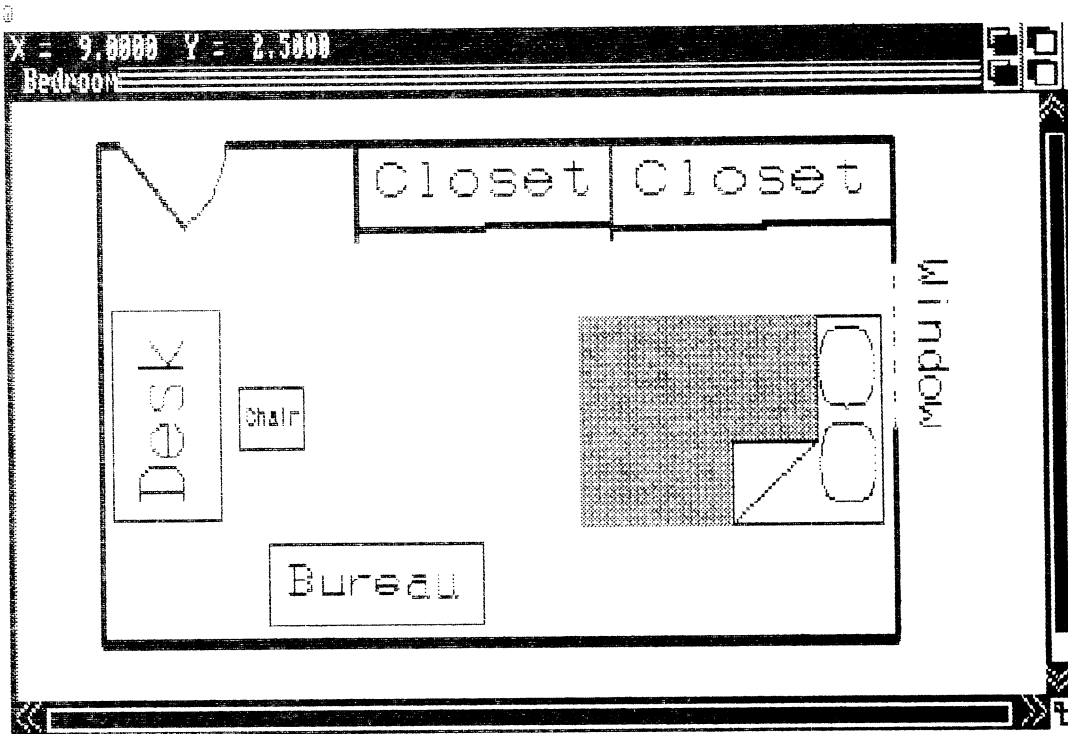


FIG. 4.32 Moving the furniture—arrangement 2.



**MOVING THE FURNITURE.** Once the furniture components have been grouped, each piece can be independently translated and rotated to new positions within the room using the *Drag it* and *Rotator* tools from the *Tools* menu. Figures 4.31 and 4.32 show two possible bedroom arrangements.

# **Animation Software**

Your first encounter with animation was probably the Saturday morning cartoons. Today you see animation daily in television commercials, training and educational films, public relations videos, sales reports, point-of-purchase displays, and feature films.

The Amiga is an important innovation in the history of animation: the advent of low-cost, computer animation accessible to any small business or individual. Whether you want to make animated “movies” for art’s sake, teach a technique, explain this year’s sales figures, or promote a product or service, animation on the Amiga is an eye-opening communication tool (Figure 5.1).

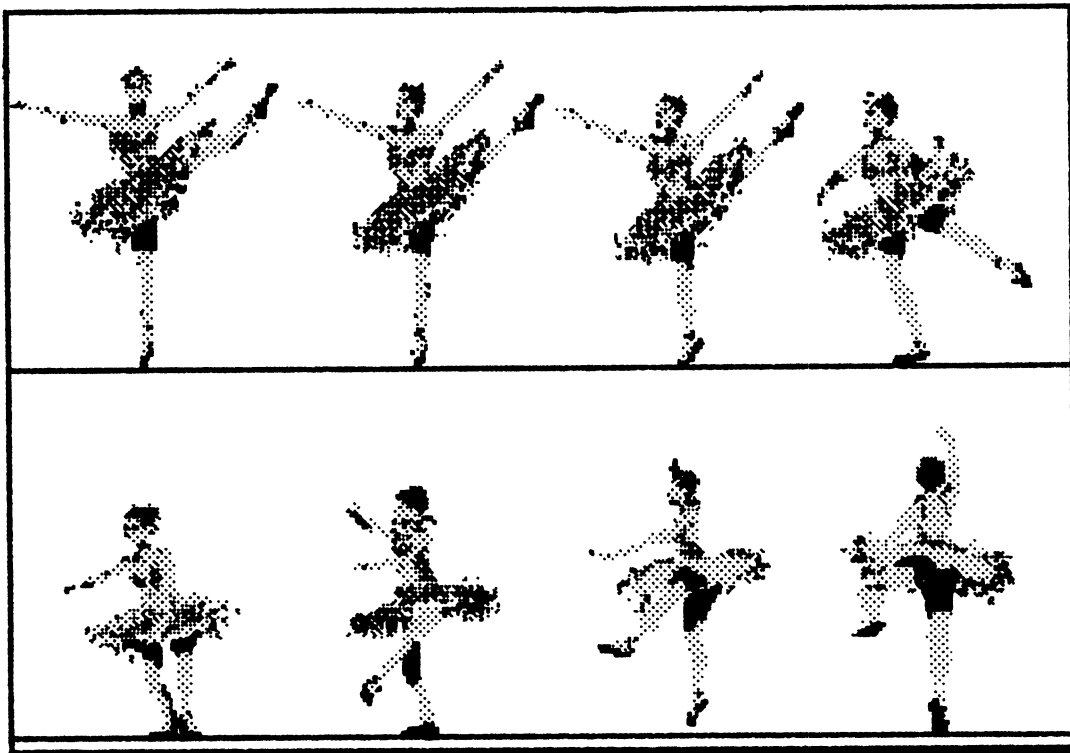
## **CREATIVE TOOLS**

---

With practice, you can program animated graphics yourself using AmigaBASIC, as introduced in Chapter 7. This software is particularly suited for creating titles, games, or relatively simple sequences.

For more complicated pursuits, two leading-edge software packages are available: Animator by Aegis Development, and Deluxe Video Construction Set by Electronic Arts. These exciting programs use entirely different methods of animation. Although they’re both flexible, each is best suited to distinct purposes.

FIG. 5.1 The ballerina dances across the screen using an animation program. She was created by Aegis Development and adapted by Gene Brawn.



Animator is a *metamorphic animation editor*: a program that lets you create an image and then change its shape as you animate without redrawing the image. It gives you 32 colors and allows you to create objects and to manipulate them on the screen. You can watch as you move them and adjust the look of the animation to your liking. It is an intuitive program that lets you change things quickly and test different moves. When finished, you can tape your animated sequences on a VCR, and use conventional production techniques to add music or other audio. Because of the range of colors available, speed of creating, smooth movement, and complete flexibility of what you can create, Animator is useful for creating very high-quality business sequences and is best for any artistic uses.

Deluxe Video is based on a more traditional approach to animation using scripts, timing, and cells. With Deluxe Video's unique "visual

script," you organize tracks for background and foreground visuals, interactive controls (what part of the animation occurs when and how) and even music and sounds. Eight colors are available in the foreground and eight in the background. The user interface is extremely well organized and sensitive to the animator's needs. It provides icon programming and easy organization of many elements. Deluxe Video is, in fact, a "visual programming language."

Electronic Arts has highlighted Deluxe Video's use in advertising, business presentations, commercials, and sales displays. They have even included automatic script generators that can prepare animated titles and charts for presentations using raw data that you enter from the keyboard as the only input. Deluxe Video is well suited to these purposes and is also easy to understand and practical for home users who don't need a full 32-color palette and aren't facing the time pressures of professional video production.

Table 5.1 compares the extensive capabilities of the two animation programs. Your choice will depend entirely on your potential business or personal uses, your artistic goals, and your preference for a certain type of interface. Both programs are similarly priced and in both cases 512k is required and two disk drives are recommended. The performance of both programs, Deluxe Video in particular, is vastly improved by the addition of extra memory. Keep in mind that this table is based on the first released versions of the programs. The data may change as updated versions are released.

**TABLE 5.1 AEGIS ANIMATOR AND DELUXE VIDEO: A QUICK COMPARISON**

<b>Feature</b>	<b>Animator</b>	<b>DVideo</b>	<b>Comments</b>
<i>General</i>			
* Manufacturer	Aegis	EA	
* Version	1.1	1.0	
* Graphic Mode Used	lo-res	lo-res	
* Color Palette	32		Foreground & background share palette
		16	Foreground—8 colors Background—8 colors
* Import Graphics?	Y		Any IFF lo-res
		Y	Any IFF lo-res
* HAM Mode?	N	N	Auto convert to 8 color

TABLE 5.1 (continued)

Feature	Animator	DVideo	Comments
<i>General</i>			
* Dual Playfields?	N	Y	Amiga dual playfield mode allows separate control of foreground and background memory. Screens are limited to palette of 8 colors
* Memory Requirements	at least 512k		2 drives recommended
a. Program Size (in bytes)			
1. Editor	180k	512k	Maximum program size. Minimum. Small amount of room left for workbench. Extensive use of overlays. Configuration depends on available memory.
2. Player	52k	110k	Stand-alone programs
3. Max script size	Dynamic	Dynamic	Varies with memory size
b. Object Memory			
1. Standard (chip)	120k	Dynamic	Depends on memory and script size, includes sounds
2. Extended (fast)	Dynamic	72k/200k	Fixed increments. Can be modified
c. Music Files	NA	32k/24k	Tracks//max p/instrument
d. Sound Files	NA	24k	Not sep memory
* Use Ram Disk?	Y	Y	If available
* Use Fast Memory?	Y	Y	
* Use Workbench?	Y	Y	
* Use Clipboard?	N	N	Simulated
* Copy Protected?	N	Y/N	Key disk protection or unprotected optional
* Cell Animation?	Y	Y	
* Tweening?	Y	N	DVideo automates frame generation between effects
<i>User Interface</i>			
* Type	by example		Animator records user movements of objects & tools



TABLE 5.1 (continued)

Feature	Animator	DVideo	Comments
<i>User Interface</i>			
		hybrid	DVideo is programmed primarily through an iconic interface. Objects, including text can also be placed visually when they appear or are moved
* Memory Display?	Y		Total remaining, in numeric characters
		Y	Graphic display of memory used
* Use Menus?	Y		All commands available from menu bar
		Y	Files, edit & playback from menu bar. All others generated in context
* Alternate Command Input	Y		Iconic "Fast Menu." No keyboard equivalents
		Y	Some keyboard equivalents of menu items
* Help?	Y		On-screen from "fast menu"
		N	User's manual
* Editing	Y	Y	Cut, splice, insert, kill
* Display X-Y Position?	N	Y	
* Preview?	Y		Single tween or full animation
		Y	Loads playback module
* Script Format	ASCII	Binary	
* Edit Script?	Y		Any ASCII text editor
		N	Within programmer only
* Merge Scripts?	Y		Both must fit in memory, otherwise use text editor
		N	
* Merge Scenes?	Y	Y	
* Chain Scripts?	Y	Y	Player utility only
* Program Summary?	Y		From script
		Y	Printed summary of completed animation
* Debug Tools	Y		Control of playback speed only
		Y	Speed control, hide objects, memory use

TABLE 5.1 (continued)

Feature	Animator	DVideo	Comments
<i>User Interface</i>			
* Error Trapping	Y		Saves current work when out-of-memory. Not reliable. Easily confused if too many objects deleted. Memory management generally poor
		Y	One of the best. Will let you lose work if save-to-disk fails
<i>Effects</i>			
* Color Control	Full		Cycle, range, spectrum, fade, adjusting gadgets
		Full	Same as above. Also predefined palettes and color locking
* Fade In/Out	Y		Fade out is automatic
		Y	Foreground manual
			Background automatic
* Wipes	Y		Foreground manual
		Y	Manual
* Background Cuts?	N		Automatic, 5 styles
		Y	No buffering, must load from disk
			Backgrounds are double buffered
* Text Modes	None		Imported as object
		All	Uses any available system font
			All text modes: JAM invert and complement
* Text Rotate	N		Text is bitmapped object
		Y	Special vectored font only
* Polygons?	Y(2-D)		Line, star, regular hex, circle, and user drawn
			Filled and outline
			Single color only
		Y(2-D)	23 predefined shapes
a. Rotate polygons?	Y	Y	X, Y and Z axis rotates.
b. Maintain perspective?	Y		Can toggle off
		Y	

TABLE 5.1 (continued)

Feature	Animator	DVideo	Comments
<i>Effects</i>			
c. Change Shape?	Y		Shrink, expand, add, & delete points. Change Z-plane position
d. Stamp to background?	N	Y	Horizontal & vertical only
e. Group-to-move?	Y	N	
f. Clone?	Y	Y	
g. Save polygons to disk?	Y	—	
* Built-in-Patterns?	N	Y	Background & text
* Strobe?	N	Y	Use clone tool
		Y	Automatic
* Bitmap Objects?	Y	Y	
a. Type	IFF	IFF	
b. Import w/Palette	N	Y	
c. Change Size?	N	Y	Jerky
d. Maximum Size	one full screen		
e. Number of colors	32	8	Dvideo will auto-convert up to 32 colors to 8
* Audio?	N	Y	
a. Music		SMUS	IFF
b. Sounds		8SVX	IFF
<i>Other Features</i>			

*Deluxe Video*

In addition to the extensive list of features above, Deluxe Video also includes (in its first release) these utility programs:

- A. A "scene generator" that is designed to make simple animations painless for the light or inexperienced user. Simply enter the desired information and the program takes over, producing a finished script
- B. An animator that allows you to incorporate short, predesigned animations into your script. That is, you may combine several sequential images (IFF format bitmaps from DPaint, Images, etc.) into a single file that can then be imported into your animation and incorporated into your animation using the *AniSeq* command.
- C. A file unpacker. This program is necessary because DVideo scripts, bitmaps, music, and sounds are stored as a single file in order to make transfer of animations as easy as possible. You may select to unpack all or single objects & sounds

TABLE 5.1 (continued)

Feature	Animator	DVideo	Comments
<i>Other Features</i>			
			D. A printed report and file compaction utility. The reporter provides an extensive list of program details. The file compactor scans the script and removes any unused files Future products include predesigned templates that you will be able to use with the scene generators.
	<i>Animator</i>		A player utility is provided for Animator. This utility addresses the “auto playback” shortcomings of the original release. Turns glow off and allows entry of a time command into the script file. Outstanding menu selector design. Cannot run from the workbench. Future products for the Animator include animated clip art files.

Separate tutorials for Animator and Deluxe Video are included in this chapter. The instructions assume that you're using two disk drives and that you're familiar with the screen and tools of the program. The examples show step-by-step advanced techniques that you'll be able to apply later in your own animations. Each example highlights the unique program features and gives tips on how to create high-quality animated sequences.

## AEGIS ANIMATOR

Aegis Animator uses *tweening* to control the movement of objects on the screen. Begin by thinking of tweens as frames. In traditional animation, various images created on separate frames are strung together in quick succession to give the illusion of movement. Movies work the same way: a series of static pictures linked together and projected very fast create the semblance of motion.

When the major movie studios such as Warner Bros. created large animation departments, “key frame” animation was initiated. The key animator would draw the character at key points of movement. The apprentice animators would then draw all the individual stages of movement of the character between the established key points. You

guessed it. These talented apprentices were the in-betweeners or “tweeners.”

When using Animator, you are the master artist. You create the object and designate the key beginning and ending positions. The computer is the tweener and draws all the in-between frames (tweens) for you.

Unlike traditional animation, however, Animator’s frames are not just single pictures; they’re frames over time. That is, each frame is a unit of time in which many things can happen. In traditional animation, frame 1 might contain an airplane. In Animator, you can draw the airplane and then have it start flying, all in frame 1. It doesn’t take numerous frames to create motion. Each frame can contain motion and then be strung together with other frames containing more motion.

Above all, Animator is economical in its requirements of the user’s time and effort.

Though the process sounds a bit complicated when you read it, it takes only moments to get up and running using the tutorial in the “Inside Aegis Animator” manual. Reading the tutorial will make it much easier to follow the Animator example presented here. The manual’s tutorial is a simple animation; the one in this chapter is not difficult, but ends up looking very polished and uses advanced techniques. If you’re familiar with the Animator tools, however, you can do this tutorial on its own.

The example takes you step by step through an animation that looks great, integrates the use of many of the major Animator tools, and is easier than you’d imagine. You’ll be able to incorporate the techniques in other advanced animations you create.

## **Using Animator**

The following sample animation uses objects created previously in a paint program. Animator gives you some simple tools for creating shapes, but they’re not designed for making complex backgrounds or objects. To make high quality animations, and to enjoy playing with the program, take time to plan your story and paint your scenes in advance. Save them on a data disk, and import them into your animation. (See Chapter 9 of the manual for information on using images created outside of Animator.) Also, if you need to save time or if you’d rather animate than paint, try

using one of the many clip art packages available (clip art is described in Chapter 2). With these you can simply choose the scenes you want and drop them into your animation.

For this example (called “The Smoker”), prepare three pictures:

1. The background of a Bette Davis-like woman holding a cigarette.
2. A window or frame made by redrawing her right eye as a closed eye and then saving a rectangle of the closed eye as a window, (i.e., as a brush in Deluxe Paint [be sure to add .win to your filename] or as a frame in Images).
3. Text that says “Amiga Art and Animation,” also saved as a window.

You could draw a similar picture using a woman’s face and hand (as simple or elaborate as you want), or use drawings from a clip art package. (For the sake of just getting some animation practice, even a stick figure woman will do.) In the example you’ll animate her right eye so that it blinks, make several streams of smoke that float up to the top of the screen, have a text title slide in, and then fade the entire screen to black.

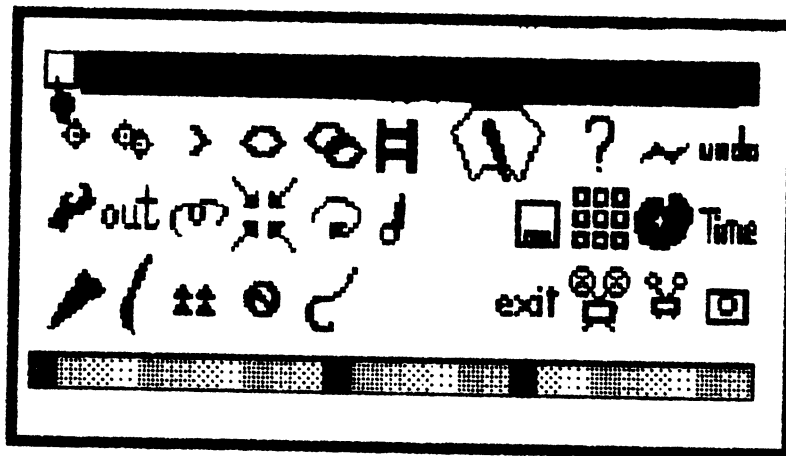
**STARTING THE ANIMATION.** Boot Animator. When the screen appears, click in the close gadget to remove the *Fast Menu* since you’ll be working from pulldown menus for now (Figure 5.2).

If this isn’t your first Animator project today, from the *Project* menu, choose *Storyboard*. When the storyboard screen comes up, choose *Delete* from the *Edit* menu and click the round cursor over the first box. This assures that you’ve got a clear storyboard with maximum available memory when you start. Simply choosing *New Script* from the *Project* menu does not reclaim all of the memory. So, don’t use *New Script*. Use the *Storyboard* to kill any old animations.

Next, choose *Into* from the *Project* menu and click the cursor, which is now the word “Into,” over the first box. This will bring up a fresh screen on which to begin your animation. Some colors on your screen may be glowing. To stop this feature, click on the palette icon in the *Fast Menu*. Click *Glow On* to turn it off. Close that requestor and close the *Fast Menu*.

**IMPORTING THE BACKGROUND SCENE.** Import the background for your animation from the data disk on which your prepared objects are stored. To do this, select *Storage* from the *Project* menu. When the

FIG. 5.2 The Fast menu.



storage screen appears you must tell the program to change from the internal drive DF0: (where it's reading the Animator disk) to the external drive or DF1:, which is the data disk. Choose *cd* for change directory. Insert the data disk into the external drive (or, with a single-drive system, remove the Animator disk and insert the data disk). When the requestor asks "New Directory?" type in Df1: and hit return.

You will see the names of the files loading in the upper bar. Once the names of the files are loaded they're kept handy in memory. This takes up a little of the memory you might use for your animation, but it saves time and makes Animator a pleasure to work with. If you'll be creating a large animation and wish to salvage some of the memory that a large directory would use up, transfer just the files that you'll use to a new disk, so that the directory will be as small as possible.

After the directory loads, choose *pic*. Click on or type in the name of the background painting you want to use and select *Load* (see Figure 5.3). When the background scene is loaded, close the *Storage* menu.

**ADJUSTING THE TWEENS.** From the *Time* menu, choose *Next Tween* four times in a row. From the *Project* menu, choose the submenu *Time*. You'll see in the triple boxes that you are now in tween 5 (Figure 5.4). Close this box. You move to the fifth tween because in preplanning this script it was decided that that would be an appropriate point at which to make her eye blink.

FIG. 5.3 Loading the background.



Remember that it is a good practice to plan your scripts in advance. That way you will be able to load all of your “objects” first, before you animate the moves. Then you will know how much memory you have available for the rest of your script. If you don’t have enough memory, you can adjust your script by deleting or redrawing an object in order to release additional memory before you’ve invested hours only to find out too late that you’ll have to begin again.

**ANIMATING HER EYE.** To animate the woman’s eye, choose *Storage* from the *Project* menu. Choose *win*, enter the name of the closed-eye file, and select *Load*.

Depending on how you saved the closed eye window, it will either appear directly over her open eye, where it belongs, or in the upper right



corner of the screen (see Figure 5.5). If it appears in the upper corner, drag it with the hand into position over the open eye and click it down. If it appears in the proper place, you still need to move the cursor over the closed eye and click it once to officially set the window (also called a raster) in place.

Before doing anything else, move to the *Next Tween* (this will be tween 6). In this tween she opens her eye again. Choose *Destroy* from the *Create* menu. Click the round cursor over the closed eye area to select that object and then click the left button again to kill it. Now you're back to the open eye, which was, after all, just hidden behind the closed eye.

**VIEWING THE EYE MOVEMENT.** The fun part: select *Replay All* from the *Time* menu. Allow a few moments for the first four tweens to go

FIG. 5.4 Checking the tween number.



FIG. 5.5 The closed-eye raster.



by, and then watch for her eye to blink. Note that it takes just two tweens to make this move. Traditional animation would probably add a few more frames, the eye open, half closed, and then closed, for instance. But this isn't necessary here because persistence of vision, the way a television picture is scanned, and your own imagination take care of the extra steps for you.

In fact, to save memory and to achieve smoothness in your animations, always start by using just a few tweens. Later, if needed, you can add more tweens. But often, simpler is better. Two steps often produce a smoother movement than five.

**ADDING THE TEXT.** Before you start to enter the text into your animation, go to *Time* in the *Project* menu and make sure you're at the last frame by selecting *End* and then selecting *Next Tween* to create a

new frame. Then, again in the *Project* menu, choose *Storage*. Click on *win* and enter the title of the text file you've saved as a window. Select *Load*.

When the text appears, grab it with the hand cursor and move it below the lower left corner of the screen until it almost disappears below the screen. Leave just a little of the box so you can grab it again. Usually Animator remembers every move you make, but this first one is ignored. However, as soon as you let go of the cursor, all following moves are remembered, so be sure the title is where you want it before releasing the button. Close the *Fast Menu*.

You want the text to rise from its out-of-sight position below the bottom of the screen to about the middle of the screen. If you just used a move tool (the hand) to drag it up to the middle there's no guarantee that you would keep to a straight line. The text would probably wobble into place. Instead, use the *Path* tool under the *Move* menu. It allows you to plot a straight line that the text can follow.

Choose the *Path* tool from *Move*. To use the *Path* tool, go to the bottom of the screen and click in the area above where you put the text, the box will begin to glow. Then, pull a line up to a midpoint on the page (see Figure 5.6). Make sure that there are no jagged points on the line before you click the right button to officially set the path. The text will then appear suddenly at the end of the *Path*. Now, is the position okay? If it's too high or too low, use the path tool to adjust the position up or down. These moves won't be replayed—only the move from the beginning of the tween to the end.

**VIEWING THE TEXT MOVE.** Use *Replay Tween* to see how the text moves into the scene or *Replay All* to see the entire sequence.

**CREATING THE SMOKE.** Use the submenu *Time* in the *Project* menu and select *Begin* to rewind the animation to tween 1. Now you will add the rising smoke to the beginning of the animation. The script is planned so that the smoke will rise and hit the top of the screen by the end of tween 4, that way it doesn't distract the viewer from the eye movement.

First, select white from the *Color* menu. From the *Create* menu choose *Polygon/Filled*. The cursor is now a triangular shape. Click the tool at the base of the ash on the cigarette. Drag the first line about 1/2 inch straight up and click the left button. Then make a 1/4 inch line to the right and click the left button to complete the polygon (a triangular

FIG. 5.6 Using the *Path* tool.

move raster to desired position



shape much like the shape of the cursor) by making the final line back to the base of the ash and clicking the right mouse button. You should now have a small, long, white triangle rising from the cigarette (Figure 5.7). Note: if dark colors in your painting make it hard to see the lines the cursor draws, you might want to go back and turn on *Glow On* again. The lines you're working with will then be highlighted when active.

From the *Move* menu, choose *Morph* and *Hook*. Click the hook over the triangle to highlight it (Figure 5.8). Then click on the points and pull them up about an inch. Also hook the midpoints of the sides of the triangle and pull them in so that the bottom of the triangle is skinnier than the top.

Move to the *Next Tween*. Use the *Hook* again to pull the smoke up more. Also, try using the *Morph/Loop* tool to manipulate the smoke a different way. You'll notice that *Hook* can create a point anywhere on the

line segments, *Loop* can't. Therefore, if you find you need a point to change a shape, from a square to a circle, for example, use the *Hook*. *Loop* can, however, move points and push lines outward.

Choose *Next Tween* and repeat the process two more times until the smoke just hits the top of the screen. If you use the *Player* program created by Aegis to play your animation, there will be no menu bars on the screen. So, in order to fill the top of the screen, drag the smoke into the menu bar. You'll see it when it's replayed on the *Player*. But, be careful. Don't just push points under the bar because they'll jam up and you'll have a mess. Use the *Hook* so that the points are consolidated.

**VIEWING THE ANIMATION.** To see what you've done so far, choose *Replay All* from the *Time* menu. If you like it, choose *Play Loop*

FIG. 5.7 Making smoke from polygons.



FIG. 5.8 Using the *Hook* tool.

to run the entire animation repeatedly and get a feel for the timing. To stop it, click the right mouse button.

If you'd like to refine your morphing skills, try adding a second stream of smoke. Try to make it snake up the screen like real smoke does. Remember that exaggeration is crucial to animation. If real smoke moves in many small streams, animate it using just two thicker streams. Start the second stream slightly after the first (thus, tween 2 or later).

**ADDING A FADE TO BLACK.** Finish the sequence with a transition that fades the screen to black. Make sure that you are on the last tween (tween 7) by selecting *End* from the *Time* menu under *Project*.

From *Project* choose *Fast Menu*. Click on the palette-shaped object and when the color requestor appears look at the right-most color in the palette at the bottom of the requestor. If it's glowing from green to purple, you're okay. Otherwise, select *Glow On*. Do this to assure that when you

use the *Replay Tween* option, you'll be able to find your way to the cursor and menu bars after the screen fades to black. If you don't have glow on, you could lose all your work by not being able to locate anything on the black screen (and you haven't saved any part of the animation to this point!).

Still working in the color requestor, adjust the range selector (the arrow line above the color bar), so that it stretches the length of the bar from left to right. Click on black to select it as the color to fade to. A cross will appear to show that it's selected. Click on *Fade*. Now you will see why it's important to turn the glow on.

**VIEWING THE COMPLETE ANIMATION.** To see your entire animation, choose *Replay All* (Figure 5.9). Stop the animation by clicking the right button during a lighted portion of the sequence.

**FIG. 5.9** The completed animation.



**SAVING THE SCRIPT.** To save the entire animation go to the *Storage* menu, choose *Script*, type in a title, hit return, and choose *Save*. This example was prepared, so you didn't need to save it sooner, but it's usually best to *Save* your work more often than just at the end just in case you run out of memory and lose the script.

## ADVANCED TIPS FOR ANIMATOR

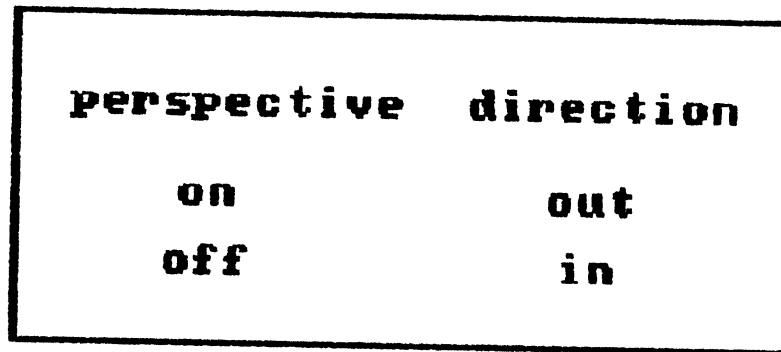
---

These tips are based on version 1.1; your copy may be a newer, improved version.

- Learn to use the *Fast Menu*, it will save you hours!
- Don't use the *New Script* selection, instead go into the *Storyboard*, *Delete* old scripts, and use *Into* to enter the first frame of a new script. This reclaims all possible memory.
- A newly loaded object, such as the closed eye window, will automatically use the current palette. To use the object's original palette, load the object into an empty *Animator* script or painting program screen and save it as a *col* (color) file. The *col* file can then be loaded before loading the object into the animation to preserve its palette. (But be aware that this will also change the background colors, you can't have it both ways, so plan ahead.)
- Note that the *Global Speed*, *Glow*, and *Perspective* settings are not preserved when you save your animation. You must manually reset them before replaying the sequence. For best results, use the default settings for scripts that you will want to auto play for audiences using the *Player*. The global speed can be entered into an *Animator* script with a text editor if you want to set it for playback with the *Player* (see "Editing the Script").
- The maximum size that you can allocate when loading *Animator* with the "ani xxxxx command" is 180000 on a 512k system. Be careful: allocating this much memory can cause a system crash in which you would lose your work. Use the memory meter frequently.
- Also, the *Player* program in the original version always loads scripts with the default memory size, so you can't auto play an animation created with expanded memory. Later versions of *Player* have solved this problem, so try to get the most recent one.



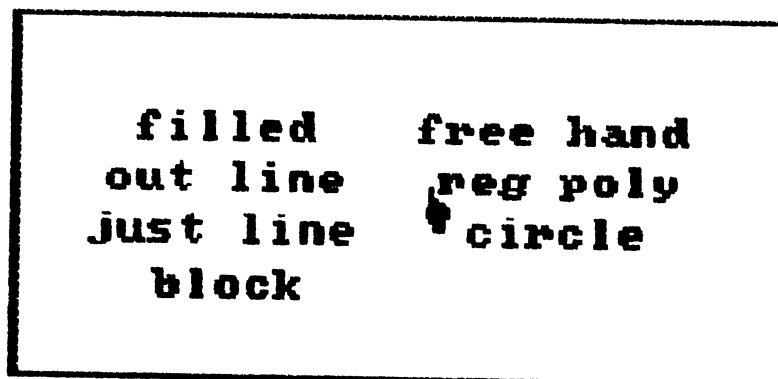
FIG. 5.10 The perspective requestor.



- The Fast Menu contains two options not available on the menu bar. When you click on the function icon (a bent line), and then click on *Out*, you get a requestor for controlling perspective (whether you want to use a vanishing point or not). Usually you would, but if the lines become too jagged, you might want to turn it off (Figure 5.10).

When you click the long red triangle icon with the function icon selected, a polygon selector appears. This contains a selection called “regular poly,” which actually allows you to make hexagons not available elsewhere. The upper bar will say “Circles” but you’ll be drawing hexagons (Figure 5.11).

FIG. 5.11 The polygon requestor.



- The *Time* submenu under *Project* is the heart of Animator, since timing is the soul of animation. This menu has three gadgets and two requestors. You've used *Begin*, *End*, and the triple box counter, which move you to different points in the animation. There are also two gadgets for setting the global speed (the pace of the entire animation) and the local speed (the amount of time it takes for an individual tween to pass.)
- When polygons are rotated on the x or y axis, they tend to creep along their respective perpendicular axis (depending on the direction of rotation) and the shape may distort. For best results, adjust a polygon's shape with the *Hook* or *Loop* tools and its position with the *Grab* tool after each rotation.
- Save your work often, just in case the system crashes! Because, unfortunately, this otherwise excellent program doesn't have the best memory management. The *Ani-Panic* function will work under normal conditions, but it is undependable with memory expanded.

## EDITING THE SCRIPT

---

Aegis Animator is programmed by example, that is, you show the program the actions you want to take and they are then recorded and used when the animation is replayed. This is probably the fastest and easiest way to program complicated animations, but it has the drawback of being memory hungry, particularly when you are experimenting with different moves.

Every animation created with Aegis Animator has associated with it a file named `<yourfilename>.script`. This file contains the list of commands that you executed when programming the animation. Many of these commands are no longer valid, particularly the *Moves*. Animator is smart enough to change some commands when you do, but others, like the *Move* commands, are retained because the program must assume that you really do want to move your character to these screen points. What should be a 5000–6000 byte animation file, after several iterations, could easily become a 30,000 byte monster. Fortunately, there is a solution.

Every Animator script file is stored in ASCII text format and can be edited with any text editor (ED, Textcraft, EMACS, Scribble, etc.). All you

have to do is load your word processor and the script file and begin editing.

To demonstrate the editing process, use the sample output from the "Smoker" script that you created earlier in this chapter. If you tried the example you'll realize that you probably made many more moves than you had to, particularly when creating the smoke polygons. This is the section of the script that will best lend itself to editing. For the purposes of this explanation it is assumed that you have a 512k system. (If your system is configured with more memory, it is suggested that you run the animator and the text editor concurrently. That way you will be able to test your edit decisions as you make them. But be careful! Save your edits before you run them; the animator may crash the system and destroy your edits.)

When you have the file loaded and it appears on the screen, take a moment to look at the structure of the script. Compare it with the example provided at the end of this chapter. You'll notice that it is divided into small sections much like paragraphs. The first section is the basic description of the animation. It includes the file name followed by a series of numbers and a list of graphics files used in the animation (except for the background file, which is usually listed in the first tween).

Here is the first "paragraph" of the animation script:

```
*script goodbye.script 15 320 200
*version 14
*ground_z 512
*speed 55
*define AMIGA_BITMAP CigText1.win
*define AMIGA_BITMAP Smoke1.win
*define AMIGA_BITMAP WinkingCigarette.win
```

First, check the list of files. Does the final animation use all of these files? If not, just delete the unused files. This will reclaim a lot of memory when the animation is loaded, because this list is used by the program to load all of the graphics into memory before the animation is run.

The next thing you'll want to note is the first number following the name of the script (15 in this case). This is the total number of tweens in this animation. If you delete any tweens, you'll have to change this number to correspond to the new count. For now, you can ignore the 320 and 200 digits, which you may have guessed is the screen resolution (lo-res).

The item named *\*speed* is the speed at which this animation will be played back. Normally, this item will NOT appear in a script produced with the animator. But, if you wish to use the stand-alone Player software to show off your animations, and your animation will play back at other than the default speed setting, you *must add this line manually*. Play around with different values to get a feel for the range of settings.

```
*tween 0 200 200 10
*act 12 LOAD_BACKGROUND -2 greycigarette.pic
*act 101 INIT_COLORS -1 0 32
    (0 0 0 )(240 240 240 )(240 0 0 )(224 0 0 )
    (208 0 0 )(192 0 0 )(176 0 0 )(128 0 0 )
    (160 160 176 )(144 144 192 )(112 112 208 )(80 80 208 )
    (48 48 224 )(0 16 240 )(192 0 224 )(64 16 0 )
    (96 32 16 )(128 64 32 )(160 96 64 )(192 128 96 )
    (192 112 112 )(192 112 112 )(208 128 128 )(208 144 144 )
    (224 160 160 )(240 176 176 )(240 192 192 )(176 176 176 )
    (144 144 144 )(112 112 112 )(80 80 80 )(48 48 48 )
*act 37 INSERT_POLY 0 0 1 7 0 0 0
    (144 95 512 0 )(149 88 512 0 )(149 82 512 0 )
    (146 78 512 0 )(144 80 512 0 )(147 84 512 0 )
    (146 88 512 0 )
*act 7 MOVE_POINT 0 4 2 -2 0
*act 4 KILL_POINT 0 3
*act 7 MOVE_POINT 0 6 0 0 -32
*act 6 MOVE_POLY 0 0 0 0
*act 6 MOVE_POLY 0 0 0 -99
*act 7 MOVE_POINT 0 0 5 5 0
*act 37 INSERT_POLY 1 0 1 7 0 0 0
    (149 100 413 0 )(149 88 413 0 )(149 82 413 0 )
    (146 78 413 0 )(146 78 413 0 )(147 84 413 0 )
    (146 88 381 0 )
```

The next “paragraph” is the first tween in this animation. The numbers following the word *\*tween* are: the beginning time (0 here), the length or running time of this tween (200) and the end time (200). Note carefully the final number (10). This is the total number of commands contained in this tween. You’ll have to change this number as you make changes to the script. You’ll also notice that the first tween is a little different than the following ones. Usually (but not always) this is where the background picture is loaded and the color palette is set.

Next is the command *INSERT\_POLY 0 . . .* which is the command to create the polygon that represents the smoke coming from the cigarette. From now on you will be looking for the commands associated with poly #0. For the purposes of this explanation, ignore poly #1, which is also smoke but essentially duplicates the first. Normally, you would also be editing that too.

The next command you see is *MOVE\_POINT*. The first number following is, as always, the polygon identification which is followed by the point number (every polygon is made up of a given number of points.) You won't be changing these, although you could. Many times when you move a point you're adjusting the shape of an object and might make several moves to get it just right. In that case you would probably want to edit out the unused moves. This time, however, keep the moves because you are trying to give the impression of moving smoke.

The next command is *MOVE\_POLY 0* followed by three '0's. The first 0 is the number of pixels you moved the polygon in the x direction (horizontally), the next the vertical or y move and the last, the z or forward and backward move. Since the values here are all 0, it means that you grabbed the polygon but didn't move it anywhere! This is a perfect candidate for editing a useless instruction.

Now, the tween looks like this:

```
*tween 0 200 200 9
*act 12 LOAD_BACKGROUND -2 greycigarette.pic
*act 101 INIT_COLORS -1 0 32
    (0 0 0 )(240 240 240 )(240 0 0 )(224 0 0 )
    (208 0 0 )(192 0 0 )(176 0 0 )(128 0 0 )
    (160 160 176 )(144 144 192 )(112 112 208 )(80 80 208 )
    (48 48 224 )(0 16 240 )(192 0 224 )(64 16 0 )
    (96 32 16 )(128 64 32 )(160 96 64 )(192 128 96 )
    (192 112 112 )(192 112 112 )(208 128 128 )(208 144 144 )
    (224 160 160 )(240 176 176 )(240 192 192 )(176 176 176 )
    (144 144 144 )(112 112 112 )(80 80 80 )(48 48 48 )
*act 37 INSERT_POLY 0 0 1 7 0 0 0
    (144 95 512 0 )(149 88 512 0 )(149 82 512 0 )
    (146 78 512 0 )(144 80 512 0 )(147 84 512 0 )
    (146 88 512 0 )
*act 7 MOVE_POINT 0 4 2 -2 0
*act 4 KILL_POINT 0 3
*act 7 MOVE_POINT 0 6 0 0 -32
```

```
*act 6 MOVE_POLY 0 0 0 -99
*act 7 MOVE_POINT 0 0 5 5 0
*act 37 INSERT_POLY 1 0 1 7 0 0 0
      (149 100 413 0 )(149 88 413 0 )(149 82 413 0 )
      (146 78 413 0 )(146 78 413 0 )(147 84 413 0 )
      (146 88 381 0 )
```

Note that in addition to removing the unused instruction, we have also changed the count of instructions on the *\*tween* line. **THIS IS VERY IMPORTANT.** If you fail to keep the count accurate, Animator may become confused and crash!

There is one other move of poly #0 in this tween. In this move there are 0's in the first three places but a -99 value in the last. This means we moved the poly toward us in the z plane. In other words you want to have the smoke pass in front of something else in this animation. You planned ahead for this and executed this instruction here. But you may need to do this and won't discover it until it is too late (in the animation.) You could remedy this in the editing by inserting this instruction (with the appropriate value) while you edit the script. Don't forget to update the number of instructions in the tween though!

The final example in this tutorial is simple but tedious. You may want to keep a calculator handy for this one (or better yet, use the workbench calculator tool!) Here you will consolidate some moves on points. If the same point is moved several times within the same tween, animator carries out every move as it is recorded (unlike polygon moves which are averaged.) This gives you the ability to do sophisticated morphing (changing of the shape of a polygon over the life of the tween). In this case, however, you want these points to move smoothly from point A to point B. Smoothness is more important than complexity.

The procedure is to group all like points and their values, which is why you need the calculator. Remember that these values are not absolute screen locations, but rather the distance the point moved. So, all you have to do is find every occurrence where point x was moved during this tween and total the x moves, y moves, and z moves. The resulting values are then represented by a single command `MOVE_POINT # <total x> <total y> <total z>`.

Start with this:

```
tween 200 200 400 21
*act 7 MOVE_POINT 0 3 -6 -20 0
*act 7 MOVE_POINT 0 3 -3 -2 0
```

```
*act 7 MOVE_POINT 0 3 -4 0 0
*act 7 MOVE_POINT 0 3 -6 2 0
*act 7 MOVE_POINT 0 6 -1 4 0
*act 7 MOVE_POINT 0 2 -11 1 0
*act 7 MOVE_POINT 0 0 -10 -1 0
*act 7 MOVE_POINT 0 2 -10 1 0
*act 7 MOVE_POINT 1 3 -17 -19 0
*act 7 MOVE_POINT 1 4 -14 4 0
*act 7 MOVE_POINT 1 2 -14 4 0
*act 7 MOVE_POINT 1 5 -3 6 0
*act 7 MOVE_POINT 1 1 -3 2 0
*act 7 MOVE_POINT 1 6 2 5 0
*act 7 MOVE_POINT 1 3 2 23 0
*act 7 MOVE_POINT 1 2 -2 1 0
*act 7 MOVE_POINT 1 2 -1 1 0
*act 7 MOVE_POINT 0 3 -3 1 0
*act 7 MOVE_POINT 1 3 1 -3 0
*act 7 MOVE_POINT 1 4 2 -3 0
*act 7 MOVE_POINT 1 2 5 -7 0
```

Add everything up and get this:

```
tween 200 200 400 10
*act 7 MOVE_POINT 0 3 -22 -19 0
*act 7 MOVE_POINT 0 6 -1 4 0
*act 7 MOVE_POINT 0 2 -21 0 0
*act 7 MOVE_POINT 0 0 -10 -1 0
*act 7 MOVE_POINT 1 3 -14 1 0
*act 7 MOVE_POINT 1 4 -12 1 0
*act 7 MOVE_POINT 1 2 -12 -1 0
*act 7 MOVE_POINT 1 5 -3 6 0
*act 7 MOVE_POINT 1 1 -3 2 0
*act 7 MOVE_POINT 1 6 2 5 0
```

You've reduced the number of elements from 21 to 10, a substantial savings when realized in many tweens over the course of an entire animation.

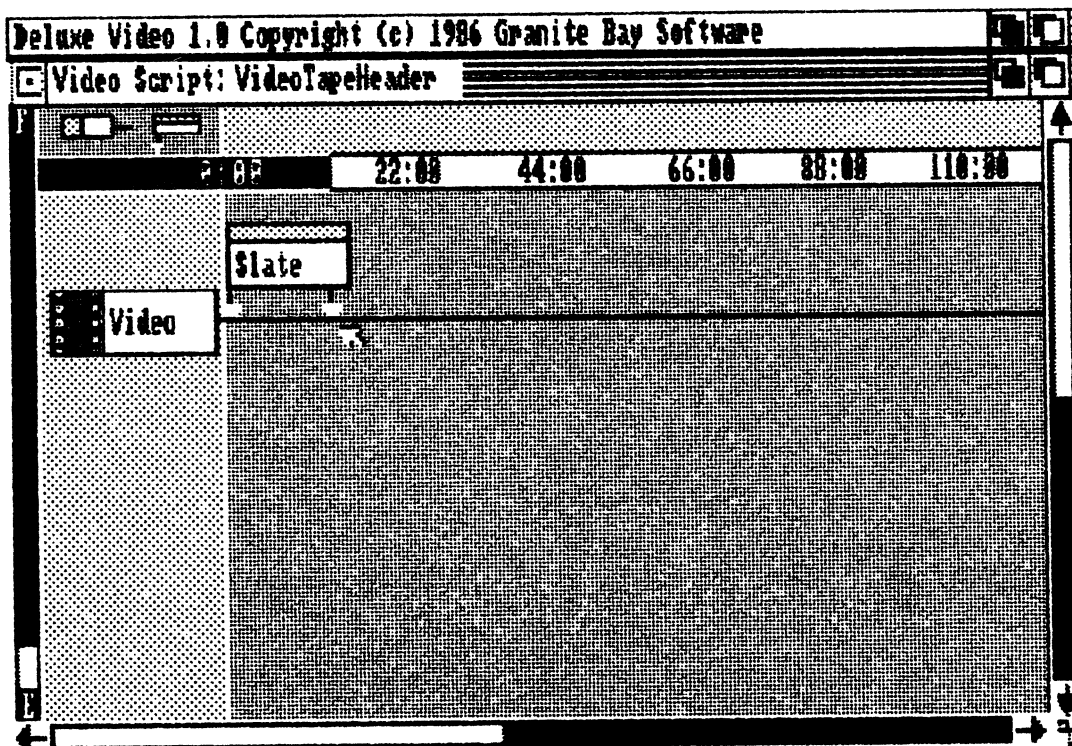
This same idea can be applied to many other commands produced by animator. Try this example first, then examine your own scripts for redundant or unused instructions. You'll find that once you are more comfortable with this procedure your productivity will increase tremen-

dously because you'll have more memory to work with and you'll probably be more likely to experiment with new ideas when you can fix them later. For a real challenge, try creating an animation entirely with a text editor!

## DELUXE VIDEO CONSTRUCTION SET

Deluxe Video Construction Set, by Electronic Arts, is a package developed with significant research and input from traditional animators and animation studios. This influence is evident in the traditional linear organization of the program. Every element is laid out clearly and efficiently using a type of storyboard or chart that shows tracks holding each element of your animation. A sample storyboard is shown in Figure 5.12.

FIG. 5.12 Sample storyboard.





The user interface is extraordinarily thoughtful and well planned. Unlike Animator, you don't build your animation on the screen. Instead you program your script with textual icons that show what objects are being used and exactly where, when, and how they appear. In this respect, when you use Deluxe Video, you are utilizing one of the most advanced examples of the next wave of "visual" programming languages.

All animation takes advance planning. With Deluxe Video, you generally can't see your moves and adjust with them as you build your animation, so it takes even more planning. The "Deluxe Video Advanced Users Guide" available from Electronic Arts contains a planning sheet that helps you to keep track of your animation. You might also get such a sheet in any basic animation textbook. The usefulness of planning in this way cannot be overemphasized. It will save you many hours and will make it easier to give your work a professional look.

The following sample animation takes you step by step through the creation of an animated sequence using Deluxe Video. Although you could probably build the necessary background in Deluxe Video, the best procedure is first to prepare the background in a paint program and import it to your animation. This saves animating time and provides the high quality look of a carefully prepared scene. The example can be done on its own, but is even easier if you have read the Deluxe Video manual or are at least familiar with the tools. It also assumes that you are using two disk drives.

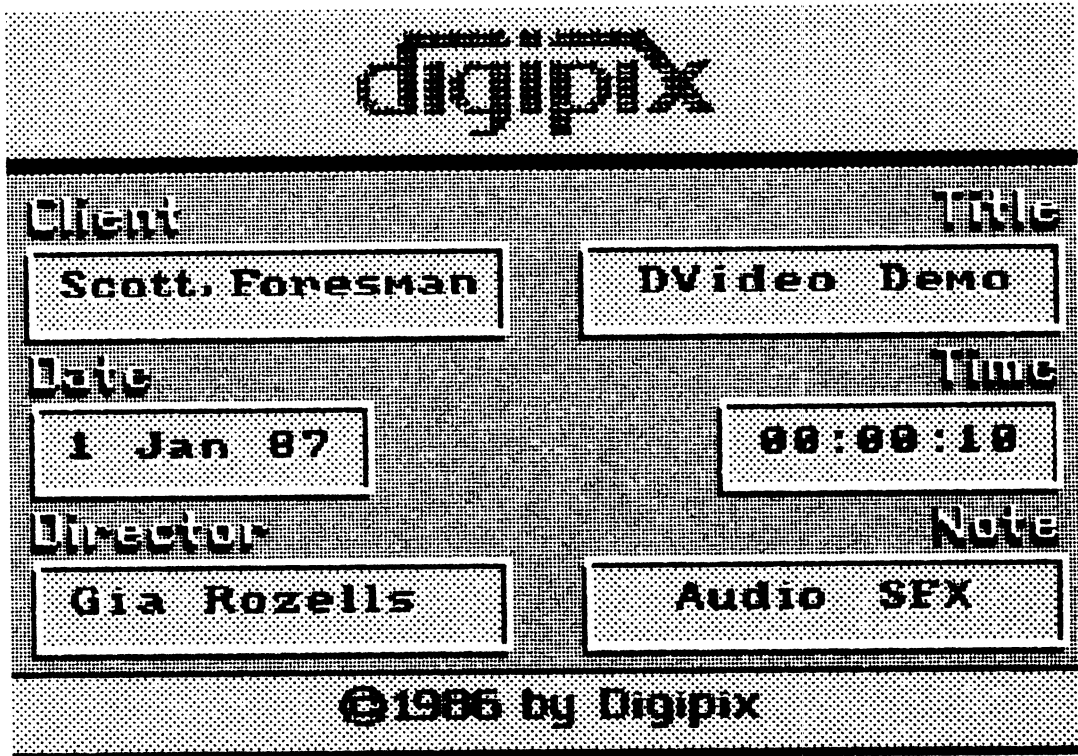
## **Creating a Video Slate**

Since Deluxe Video is ideally suited for "desktop video" production, this example provides you with a tool to use in all of your features that you may save on video tape. It is a video slate that you record at the beginning of a videotape to tell you what program (your animation, business presentation, or home movie) follows. This is a professional device to identify the several elements of the tape: the client, the title, the date, the length, the director, and additional notes.

You can make your own slate to your personal specifications. The sample, shown in Figure 5.13, looks and reads like a professional studio's slate; using it will give your work a studio look. In addition to identifying the tape, it uses an animated countdown to lead into the beginning of the show.

**CREATING THE SLATE.** Start by painting the background as shown in Figure 5.14, in a paint program. The rest will be completed in Deluxe Video.

FIG. 5.13 The Video Slate.



For the company name put your business or perhaps your family name ("Smith Family Video Library"). You can make the rectangles as fancy as your artistic talent allows. Simple boxes with drop shadows are attractive. The standard topaz and ruby fonts will work for the type. There's a lot of information on a slate, so you should probably use topaz 8/9 or ruby 8/9.

This example is meant to be fun as well as useful, so embellish the idea as much as you like. When finished, store your prepared background on disk in a subdirectory named "Pictures."

**STARTING THE PROGRAM.** Load Deluxe Video. Drag the camera icon out onto the Workbench from the window. Close the window. This frees up some memory. Click on the camera to activate it.

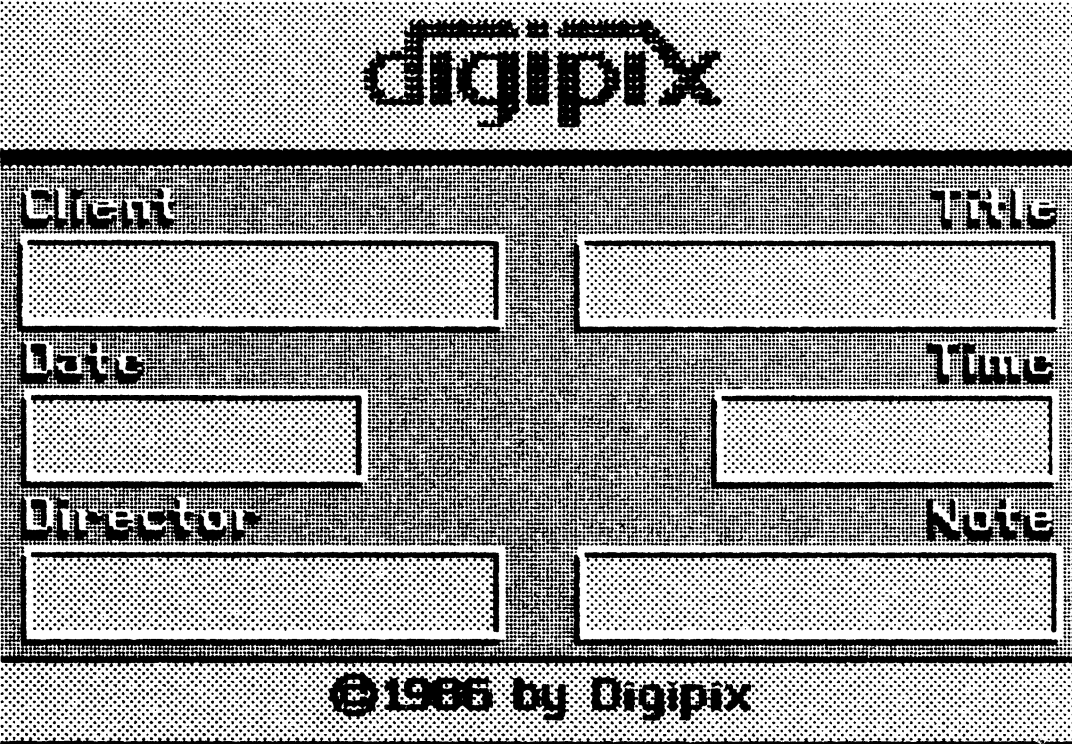
(Depending on whether you've used the program today, the following may apply. Since loading this program requires a key disk method, you'll

probably forget to take that disk out. While the program is loading a requestor will say “Deluxe Video is write protected.” Click *Cancel*. The edit screen will appear and another requestor will appear saying that a workfile can’t be opened. Again, *Cancel*.)

At this point the program can’t do anything until you load your work disk (a clean or fairly empty disk) into the external drive or tell Deluxe Video to look at the program disk for the file. Whenever Deluxe Video works on a file it stores a copy of it onto the selected data disk, so until it has such a disk it won’t begin.

Note: if you ever run out of room at the end of creating something and find that you can’t save it to disk, the requestor will say that it couldn’t save the file and ask you to “Retry or Cancel.” Don’t ever Cancel. You’ll lose all your work. Simply remove the full disk and insert a new one, then select *Retry*. So be sure that you have enough room on your disk for two copies of your finished animation before you start.

FIG. 5.14 The background drawing.



The image shows a screenshot of a software interface for creating a background drawing. At the top center, the word "digipix" is displayed in a stylized, lowercase font. Below this, the interface is organized into a grid of input fields. The first row contains two fields: "Client" on the left and "Title" on the right. The second row contains "Date" on the left and "Time" on the right. The third row contains "Director" on the left and "Note" on the right. Each field is represented by a rectangular box with a thin border. At the bottom center of the interface, the text "©1986 by Digipix" is displayed.

Client	Title
Date	Time
Director	Note

©1986 by Digipix

**SETTING THE DATA DRAWERS.** After you've inserted the data disk select *Data Drawers* from the *Options* menu. The requestor that appears asks you to tell the program where to look for certain items. (The data disks that you are using should all be prepared with these subdirectories on them. See Chapter 11 for how to do this.)

Generally it's advisable to select DF1: for the videos, pictures, and objects you will use in this animation. DF0:, the program disk, contains a default set of sounds, music, and instruments. You should use these files in order to save space on your data disk. Click *OK* or *Cancel* to close the menu. Now, select *Snapshot*, which will save the settings permanently, i.e., *Deluxe Video* will remember these settings and use them every time you load the program.

**STARTING THE ANIMATION.** Select *New* from the *Project* menu. If you grab the right arrow of the *New Scene* icon you'll see that the default time setting is 20:00 seconds. Drag the arrow to the left until it reads 12:00 seconds. This will be the length of your animation.

Double click on the *Scene* icon to bring up the *Scene Script* window. Drag an *Empty Track* onto the work area. Select *Background*. Drag an *Empty Effects* icon onto the track. Select *Lock Colors*. Drag the left arrow to the far left (02:00). Drag the right arrow to 12:00 seconds.

Grab another *Empty Track* and select *Foreground* and repeat the process.

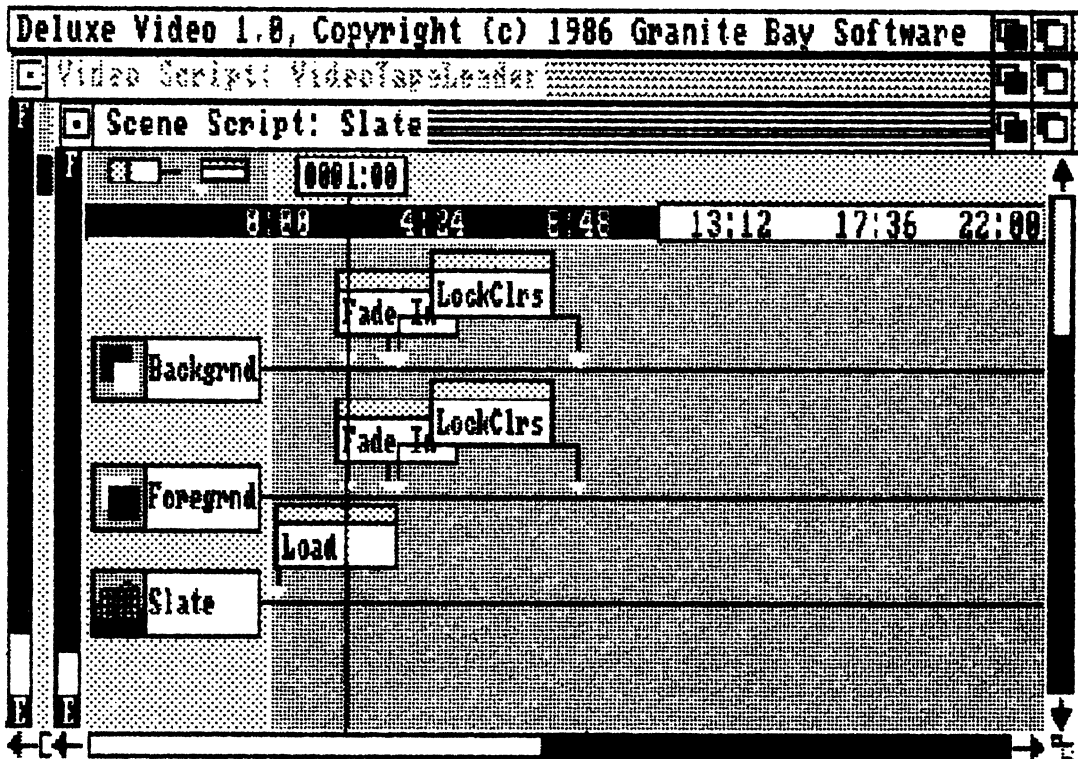
**IMPORT THE BACKGROUND.** To import the background slate scene you drew from your painting program, grab another *Empty Track* and pull it down into the area on the left side of the *Script Window* (the working area). Imported backgrounds are always *Pictures*, so click on *Picture* in the requestor.

Select your file (from the painting program) from the *Picture Requestor* "On Disk" list. Now click *Select*. The track icon should now be labeled with the name of your file, as shown in Figure 5.15, where the file is called "Slate."

**CREATING A FADE IN.** Grab an *Empty Effect* icon (the small one on the right) and place it on the track a little to the right of the beginning of the track.

When you see the *Load Effect Requestor*, click on "Get Ready." Now drag another *Empty Effect* icon on the track and choose *Fade In* from the requestor. This will begin your animation with a slow (1 second) fade

FIG. 5.15 Importing the background.



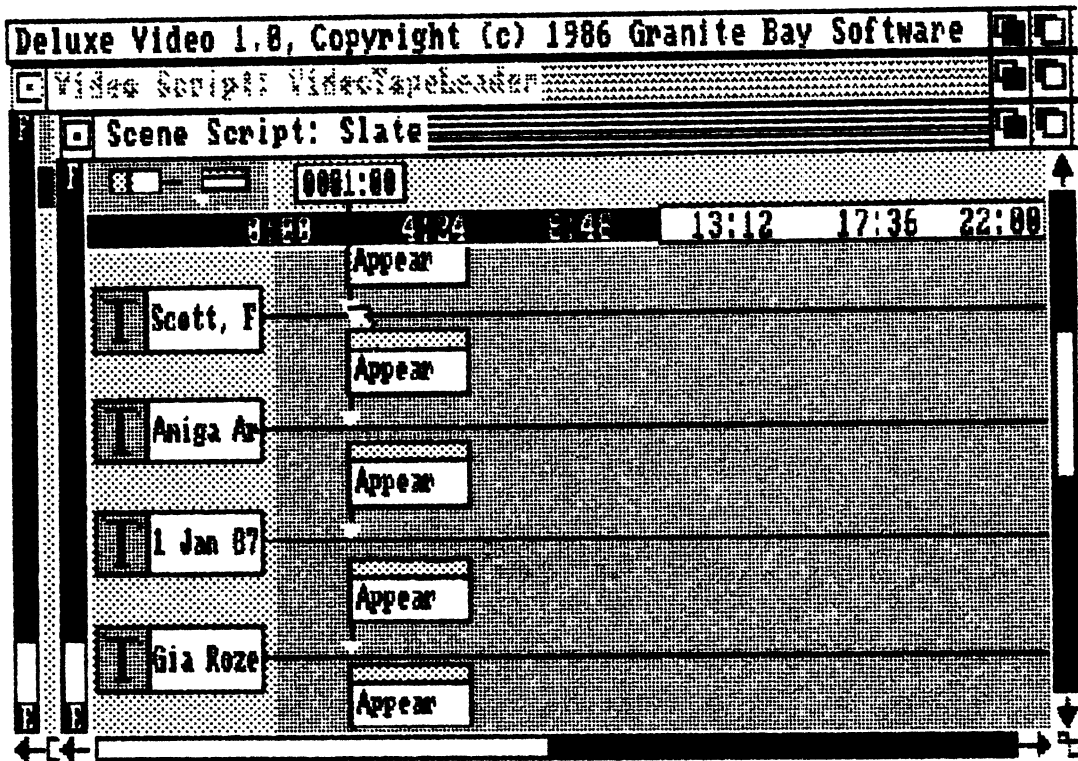
in. Adjust the right arrow on this icon to speed up or down the fade-in time. The default fade color is from black which happens to be what you want.

**CREATING THE TEXT.** Grab a new *Empty Track* and place it below the first one. Choose *Text Line*. This track is a text track on which you'll enter the information titles (see Figure 5.16).

When the text requestor appears, click in the empty upper box labeled *Text* to activate it. Type in the client's name. Choose *Topaz 9*. This is normally a little too small for television text, but okay for one viewer (you).

Click on the *TV* icon. The text, font, and colors you've chosen will appear at the bottom of the screen. The default colors are yellow on black, an effective combination. You may use these colors or play with

FIG. 5.16 Creating the text.



the numbers that select palettes to see other color combinations. Choose *Okay* when you're satisfied. (Good colors from the default palette are 5, 6, and 7 for foreground and 0 or 1 for background. Other combinations are possible by first changing the palette from the color menu. If at all possible use a composite monitor or color television to preview your work. Some colors look good on an RGB monitor but just don't work on televisions.)

**PLACING THE TEXT.** Now to place this text on the background, put an *Empty Effect* on the text track prior to the *Fade In*. When the requestor asks "Add what kind of effect?" Choose *Appear*.

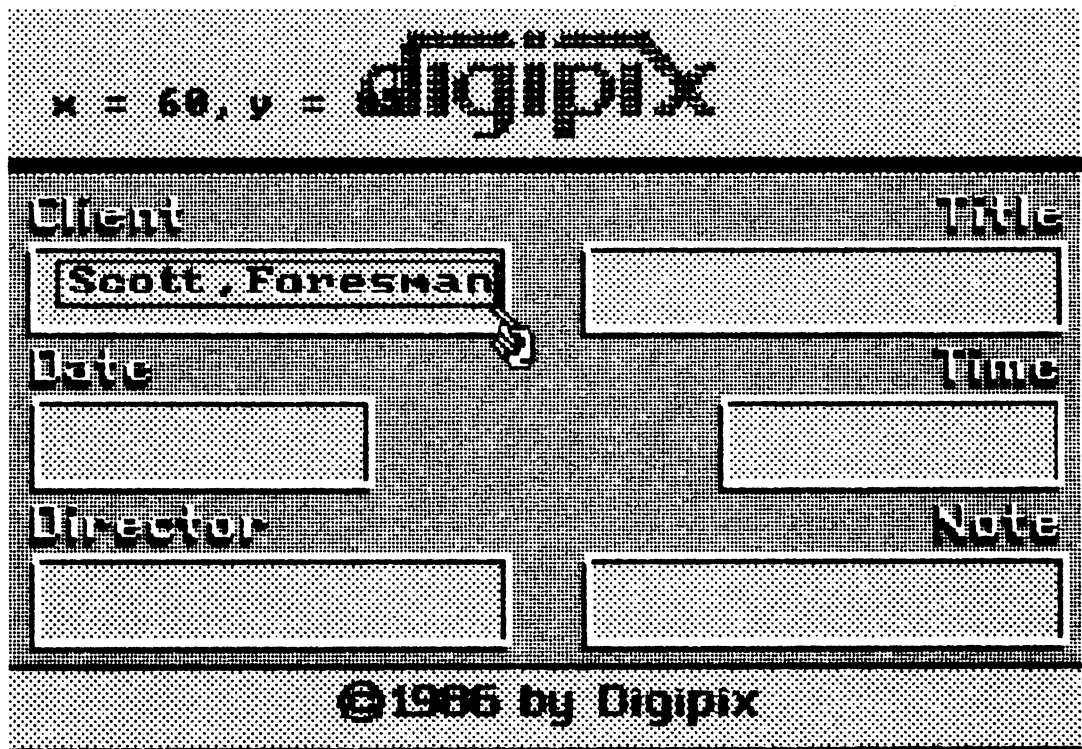
The requestor will ask "Appear Where?" Click on the TV icon. This will allow you to see the background slate. (If for some reason it isn't showing, close this requestor and go back to the *Project* menu, click on *Picture* and *Show*, and then try again.)

The text you created is probably sitting at the bottom of the screen. You can click and hold on it and grab it with the mouse. Drag it and place it in the "Client" box (see Figure 5.17). Click on the Behind gadget (left) in the upper right corner of the screen.

Note that you can see a white rectangle in the gray area of the requestor box. This box is positioned correctly in relation to the actual screen. The area represented is actually larger than the real screen. In the future you might be able to use this to your benefit. For example, if you moved the text to the far left of the gray requestor, it wouldn't appear on the background scene. Until it appears, it takes up no memory. Thus, you could save a piece of animation without tying up memory from the parts pool until you wanted to pull it into the animation. For now, the text is perfect where it is, so select *Okay*.

Repeat the above process for each text box, inserting whatever text is appropriate for you. You want all of the text in the animation to appear

FIG. 5.17 Placing the text.



at the same moment. Therefore, make sure the icons all have the same start time as the end of the fade-in. Check this by clicking the left arrow of each one and reading the time box at the left side of the time bar. (As you become a more advanced user of this program, you might want this slate to be a little more exciting. Try staggering the time at which each title appears so that they “pop” on one at a time before the countdown begins.)

**LOADING THE SOUNDS.** It is advisable to load sounds and music early in your work, even if they won’t be used right away. To make room on your work screen scroll up or down using the scroll gadget on the right. Add a new *Empty Track*. Click on *Sound*.

When the *Sound Requestor* appears choose *Bloop*. This is a digitized sound available on the program disk. You’ll see that when it loads it will appear under the *In Video* column. This indicates that it’s in the parts pool for your animation. (Be sure to delete anything that isn’t being used.) Any part listed here uses your valuable memory.

Click on the *Speaker* icon to preview the bloop sound. Choose *Select* to add it to your script.

**PLACING THE SOUND.** Put an *Empty Effect* on the sound track (at :21). Select *Fetch*. Now adjust the position so that the time reads 00:00. Get another *Empty Effect* and place it on the slate track to the right of that one. Select *Play*.

The first effect shows when the sound will be fetched into the program and the second one indicates when it will be played. Generally, a sound only needs to be fetched once, even if you play it many times. (As long as the parts pool isn’t too full.)

The *Play Effect Requestor* appears. Now you can adjust the sound. Move the rate slider until you get a crisp sound (about 3/4 inch from the left).

Adjust the volume to maximum. You can usually leave it there unless you want to lower the volume to create a background sound beneath others.

Adjust the stereo slider to the left if you only have one speaker. Otherwise, leave it in the center for stereo.

Click on the speaker icon to hear the sound. When you like it, select *Okay*. (If you don’t like the *Bloop* sound, simply *Cancel*. Then double click on the *Bloop* track icon. Now you can select another sound. Don’t forget to delete *Bloop* from the parts pool.)



**SETTING THE DURATION OF THE SOUND.** The two arrows on the play effect icon indicate the duration of the bloop sound effect. Pull arrows apart to play it more than once, or push them together to clip it shorter. Note that you could create new sounds, add an echo, etc., by overlapping several sounds, starting (or repeating) a new sound before the first is finished.

**CREATING POLYGON TEXT COUNTDOWN NUMBERS.** The 10-second countdown will actually be from 10 to 2 (a count of 8). The last two seconds are a fade to black to allow time for a smooth transition into the beginning of the tape. (Actually, this is a technical element that allows your VCR's picture to stabilize before you cut to it.)

You'll be creating polygon text for these countdown numbers. Creating polygon text is much like typing the information text you did previously.

Add a new *Empty Track*. Select *Polygon Text*. When the *Polygon Text Requestor* appears type in "10." Click on the TV icon to see the numbers and colors. Choose *Shadow*. Click on the various numbers to check various color combinations to see which you prefer. Try reversing the same combination you used for the text. If you want to use other color palettes, check in the manual for instructions on how to set other foreground colors.

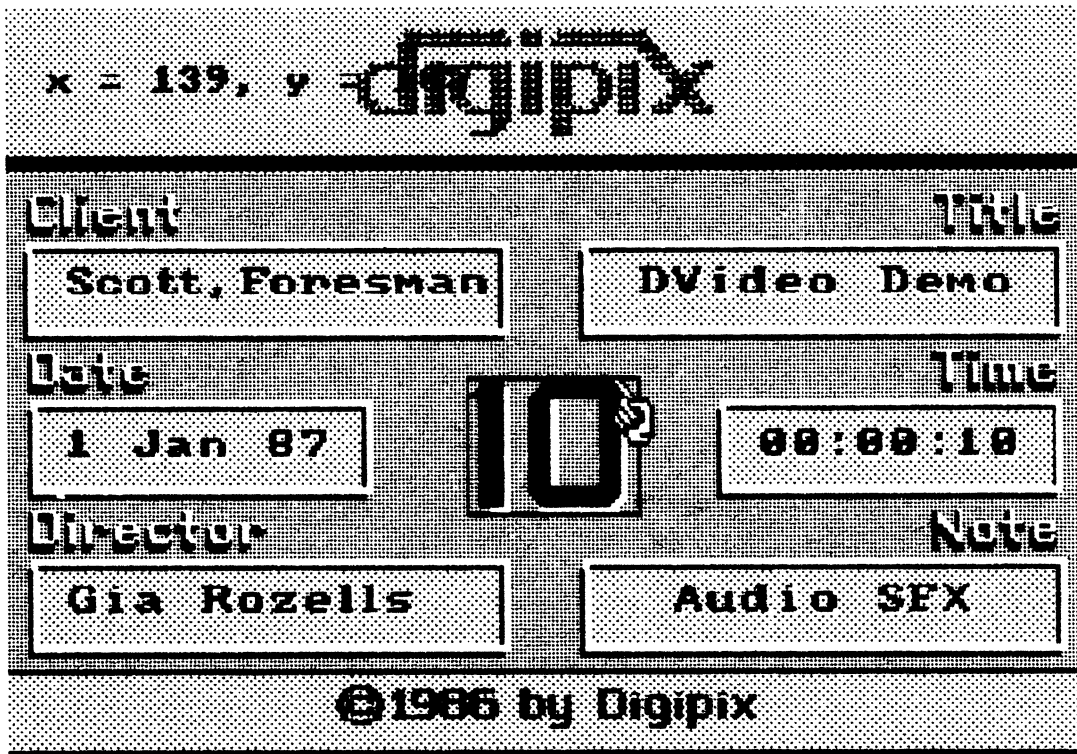
Note that for videotaping, large areas of the color red are not recommended. It tends to smear on home televisions.

After you've selected your colors (color 5 from the default palette looks good on screen), select *OK*.

**PLACING THE NUMBERS ON THE SLATE.** Pull down a new *Empty Track*. Select *Appear*. Click on the TV icon to see the background. The 10 should be near the center at the top of the screen (Figure 5.18). Drag it to the blank area in the center of the screen. Remember the "y" location; you'll need it later. Click on *Center*. This centers the number horizontally on the vertical line on which you placed the text (not in the vertical center of the screen). Click on the *Behind* gadget and choose *OK*.

**SETTING THE TIME THE NUMBERS APPEAR.** Drag the left arrow for the 10 effect icon to 1:00 second. This sets the 10 to appear at 1 second into the video. You can allow a little more time before the count starts, but don't forget to adjust the scene icon on the video track too.

FIG. 5.18 Placing the polygon numbers.



**MAKING THE NUMBERS DISAPPEAR.** Select a new *Empty Effect*. Place it behind the 10 effect icon on the 10 track. Choose *Disappear*. Slide the time arrow to 1:50. This sets the 10 to disappear at 1:50.

**CREATING AND TIMING THE OTHER NUMBERS.** Select a new *Empty Track* and choose *Polygon Text*. Repeat the above process for the number 9. When you preview the background to see where the 9 is located on the screen, drag it to the same "y" coordinate as you did with the 10. Click OK, then click center.

Adjust time so that the 9 appears at 2:00 seconds and add a new *Disappear* effect to make it vanish at 2:50.

Repeat the above process for numbers 8 through 3. Each number should appear at the next full second and leave on the x:50.

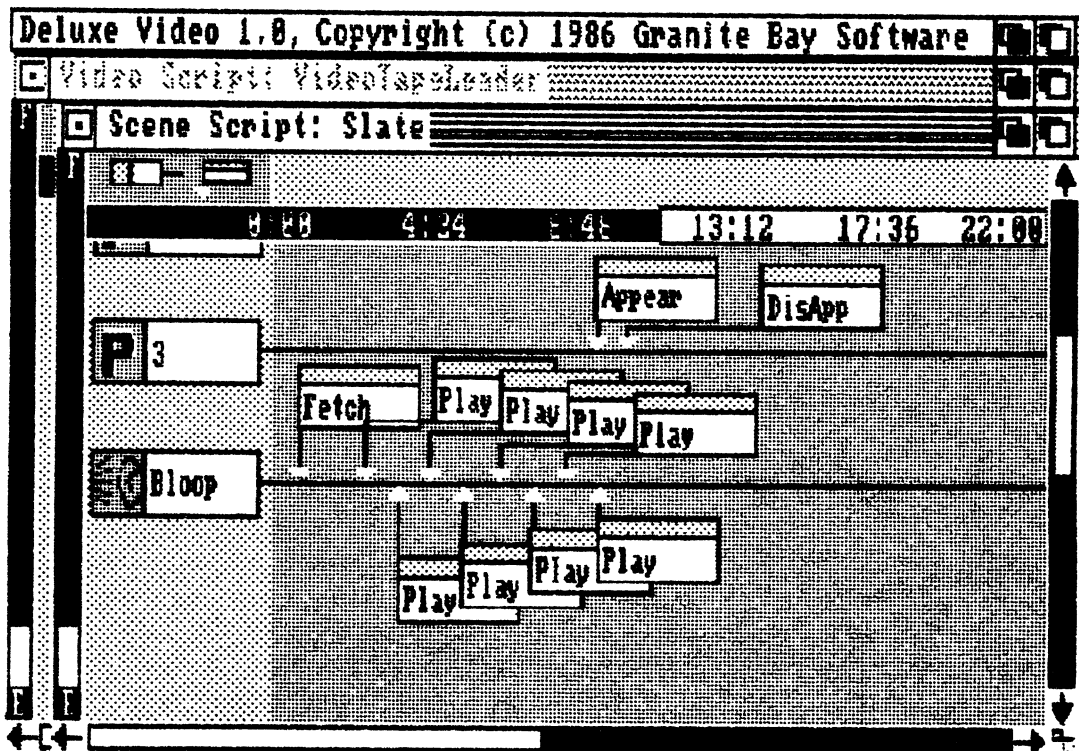
**SYNCHRONIZE THE SOUNDS.** Next you'll synchronize the sounds so that the bleeps occur when the numbers flash onto the screen.

Go to the bleep track and make sure that *Fetch* is located to the far left of the track at time 00:21.

Grab *Play* and move it until it is aligned with the 10 icon. That is, the bleep must occur at 1:00 second into the video, synchronized with the appearance of the 10.

Create a new bleep effect for each of the other numbers and synchronize the times to the appearance of each of them. That is, on the same sound track, set up eight bleep icons at 1:00, 2:00, 3:00, etc. (see Figure 5.19). To let you know when the countdown is almost finished, you might want to make the bleep at 3 a little higher or lower in pitch than the others by adjusting the playback speed when you get its *Play Requestor*.

FIG. 5.19 Synchronizing the sounds.



**VIEWING THE ANIMATION.** To see what you've created so far, go to the *Project* menu and select *Play Video*.

Go back to the working area by clicking on the upper left hand button on the Remote Control.

**ADDING THE FADE TO BLACK.** To finish the video use a fade to black that begins at 9:00 (when the 2 would appear if you were using 2 and 1). A fade to black is useful because fading in and out with black makes videotape editing easy (see the "Deluxe Video Advanced Users Guide" for tips on this).

Go back to the *Background* track. Put an *Empty Effects* icon just at the end of the *Lock Colors* effect. Now select *Fade Out*. Do the same with the *Foreground* track. Set the fade times to about 30:00. Be sure that both fades are exactly the same length (see Figure 5.20).

FIG. 5.20 Setting the fade outs.

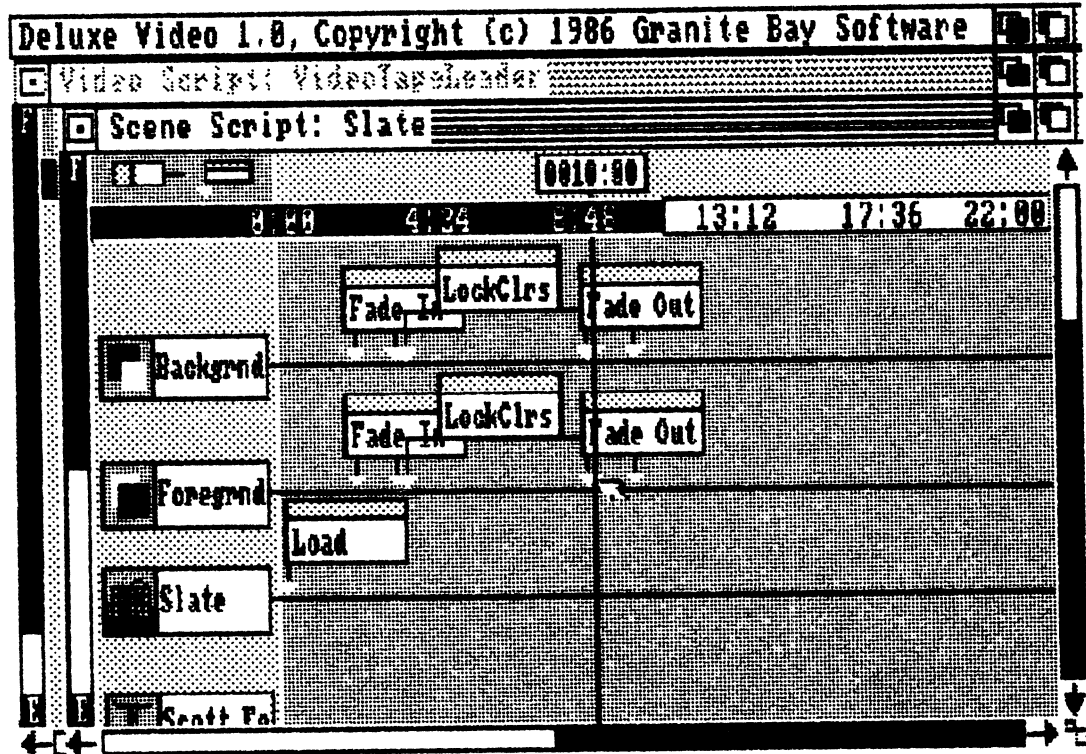
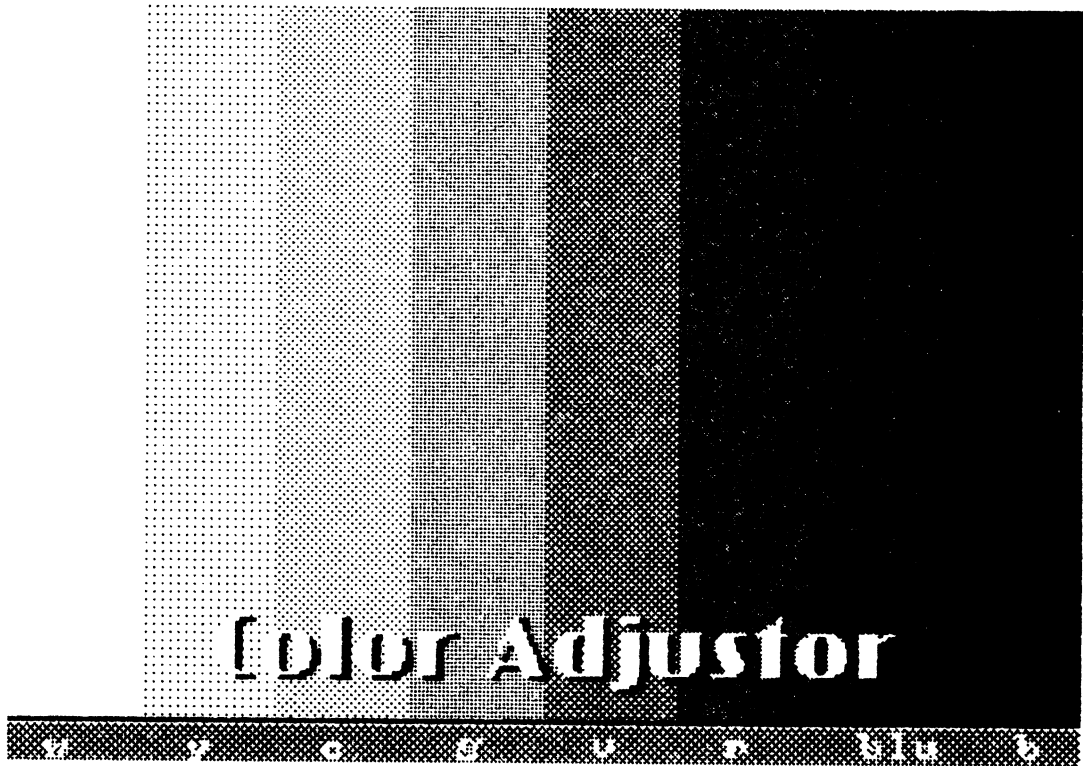


FIG. 5.21 The Color Adjustor.



**PLAY THE FINISHED VIDEO.** Now try your new video by selecting *Play Video* from the *Project* menu.

### **Adding Color Bars to the Slate**

Now that you have a finished slate, you can fill it in and use it as a “header” for any of your videotapes. This is a professional start for your programs. To complete the header you might want to include a color bar and sound test also. The color bar is a test pattern, several bars of primary colors such as you may have seen on television after a station has gone off the air (Figure 5.21).

Put the color bar on the header before your slate appears. To make one, try using the far right STD palette, which closely approximates the primary colors used in professional color bars. Use rectangle polygons

and pull them out to the vertical length of the screen. Or, for fun, make the colors pop onto the screen and rotate, or whatever you like.

The purpose is this: You create a video using animations of certain palettes and colors to create certain effects. When you play your tape on various television sets, if the colors are not the same as on your own, your video won't have the right colors. If you make a color bar display, you'll know how the colors should look. When you play your tape on a strange television set, if the colors are not correct, adjust the color controls on the set until they look as you created them. This will only be close, since you may not remember exactly and you don't have an engineer's tools to test them, but you'll remember closely enough to make your video colors look right.

Put your color bars in the slate animation by adding a new scene above the slate using the techniques you learned in the example.

## ADVANCED TIPS FOR DELUXE VIDEO

---

- You can free up memory in your parts pool by making an object appear and then immediately using the *Stamp* tool to make it part of the background and then using *Disappear* to remove the object from the pool. In this manner, the object is still visible but exists as part of the background rather than as an animated object.
- Deluxe Video will compile all the elements of the animation including sound and music as one file under one title. This makes it very easy to transfer your work from one disk to another or to transmit animations over phone lines to your friends or clients. However, if you want to get or see individual elements of a file, you'll need to use the unpacking utility provided.
- Deluxe Video has 23 polygons available in the program. You can get them by using the *Polygon Text* requestor and certain control/letter key combinations. See the manual for a list of the polygons and their key codes.
- If for some reason you see tracks in the working area but none of your effects on them, check the time setting. It's possible that you accidentally moved yourself forward in time by clicking the time gadget at the bottom of the screen.
- To save time in creating your animation, create similar effects all at the same time. For instance, create all of the text at one time.

This saves the time it takes to load the text requestor from the program disk over and over. Later, you can use *Cut and Paste* to move them to their appropriate place in the animation.

- Use *Cut and Paste* to clone objects and whole tracks as often as possible. Remember that you can modify the performance of almost all effects in Deluxe Video. This will save you considerable time. For example, you only needed to create the *Bloop* effect once. Then you just copied it for the rest of its occurrences. Similarly, if you had another sound track to add, you could just copy the entire *Bloop* track, move it to an open area, change the sound file, and adjust the already existing *Fetch* and *Play* effects to their new times on the track, and delete those you no longer need.

**CONTRIBUTOR TO THE CHAPTER.** The drawings and animation tips in this chapter were provided by Gene Brawn of San Diego, CA. Gene has had several careers spanning the range of the media business. At one time or another he has been a Director/Producer for television, a graphic designer, magazine publisher, and circulation director. His computer experience includes designing video arcade games, interactive video systems, and software. Currently, he is the owner of DigiPix, a pioneering low-cost animation company serving the broadcast, industrial, and educational markets.





# Programming Graphics in AmigaBASIC

Programming graphics in AmigaBASIC offers opportunities for programmers at all levels. Default display parameters and program control via pulldown menus provide the novice with a low user maintenance, easy-to-learn programming environment. At the same time, advanced programmers can access the lower-level operating System Kernal Routines.

The intent of this chapter is to provide you with a solid conceptual understanding of integrated AmigaBASIC programming. That may sound like a mouthful, but all it really means is that you will learn what the pieces of an AmigaBASIC program are and how they fit together. The AmigaBASIC reference manual is a great source of information for the commands, but it doesn't tell you how those commands should be integrated into a program; this chapter does.

---

## **AMIGABASIC GRAPHICS SKELETON PROGRAM**

One of the nice things about programming graphics in AmigaBASIC is the number of optional program components. In fact, almost everything

is optional, including input and output of any kind. The following is a valid AmigaBASIC program:

```
trashcan$ = "garbage"  
END
```

The only purpose for a program of this sort is to make the programmer happy in throwing out the garbage. Fortunately, very few programmers believe that calculating for calculation's sake will improve the Amiga's speed and stamina.

Calling program components optional is not to say that you can randomly omit portions of a program. For the most part, a program is a whole. Some "parts" depend on other "parts," and the key is that you understand the requirements of all the components.

Following is a schematic of a completely general AmigaBASIC graphics program. Each of the pieces is addressed in the remainder of this chapter.

## Skeleton Graphics Program

```
Include graphics libraries  
Define Screen(s)  
  Define Window(s)  
    Define Color(s)  
      Set Foreground/Background colors  
      Clear Screen  
      Initialize and Dimension variables and arrays  
      Input/Output commands  
    Close Window(s)  
  Close Screen  
End
```

## The AmigaBASIC Screen

To define the interactive and display limitations of your graphics display, AmigaBASIC provides the concept of a screen. A screen defines:

- Horizontal and vertical resolution
- Interlacing or noninterlacing
- The maximum number of colors

## The Default Screen

AmigaBASIC's default screen is known as the Workbench screen. It is a high-resolution, noninterlaced display using two bitplanes for a maximum of four simultaneous colors. The default color combinations are those selected with the Preferences program.

There are several advantages to using the Workbench screen for your BASIC programs.

- No additional memory needs to be allocated for a new screen
- You have easy access to Workbench or CLI by using the front/back gadgets
- It already exists so you don't have to do anything to set it up

## Using a Custom Screen

Despite the advantages of the default screen, often you'll want a custom-designed display. If you want any of the following, then you need to define a custom screen:

- Low resolution
- Interlacing
- More than four colors

To set up a customized screen, use the Screen command:

- Screen\_number: 1,2,3, or 4. (identifier)
- Width: not > 320 for low resolution  
not > 640 for high resolution
- Height: not > 220 for noninterlaced  
not > 400 for interlaced
- Depth: 1,2,3,4, or 5 for low resolution (# of bitplanes)  
1,2,3, or 4 for high resolution  
(# of colors =  $2^{\text{depth}}$ , e.g.  $2^5 = 32$ )
- Mode: 1 = low resolution, noninterlaced  
2 = high resolution, noninterlaced  
3 = low resolution, interlaced  
4 = high resolution, interlaced

To close a screen and free its allocated display memory, use the command

```
SCREEN CLOSE screen_number
```

The screen command defines the type of display that is possible, but graphics are not normally drawn directly on the screen. Once a screen has been defined, AmigaBASIC requires that you open a window on it before generating graphics.

## Defining a Window

A new window is opened by using the WINDOW command.

```
WINDOW window_num[, [title], [size] [, attributes] [, screen_
num]]]
```

- \* window\_num: 1,2,3,... (Number 1 is reserved for the default output window used by AmigaBASIC. This is the only window in which users can enter "immediate mode" commands.)
- \* title: window title
- \* size: (left,top)-(right,bottom).
- \* attributes: 0-31

To close a window, use:

```
WINDOW CLOSE window_num
```

To make a window the current output window without bringing it to the front, use:

```
WINDOW OUTPUT window_num
[, [title] [, size] [, attributes] [, screen_num]]]
Defines and opens the specified window.
* WINDOW CLOSE window_number
Opens a previously closed and defined window.
* WINDOW OUTPUT window_num
Selects the specified, open window as the output
window without bringing that window in front of any
overlapping windows.
```

TABLE 6.1. MAXIMUM SIZE (WITHOUT SIZING GADGET)

	Lo-res	Hi-res
Interlaced	(0,0)-(311,386)	(0,0)-(631,186)
Noninterlaced	(0,0)-(311,186)	(0,0)-(631,186)

TABLE 6.2. MAXIMUM SIZE (WITHOUT SIZING GADGET)

	Lo-res	Hi-res
Interlaced	(0,0)-(297,386)	(0,0)-(617,386)
Noninterlaced	(0,0)-(297,186)	(0,0)-(617,186)

- \* The maximum size for a window depends on:
- the screen size,
  - the interlacing mode used, and
  - whether or not a sizing gadget is used.

Tables 6.1 and 6.2 show some important limitations for maximum screen size with various options.

## Window Attributes

Windows opened by AmigaBASIC can include a number of attributes. The last parameter in the WINDOW command selects the desired attributes. Table 6.3 shows a list of attributes along with their associated values.

TABLE 6.3.

Attribute Value	Attribute
0	Plain window. Title bar if specified
1	Window contains a sizing gadget
2	Window contains a drag bar gadget
4	Window contains a depth arrangement gadget
8	Window contains a close gadget
16	Window contents are saved and redrawn when window is resized or covered

Multiple attributes are selected by adding their attribute values together. For example, to choose a sizing gadget, a depth arrangement gadget, and a window close gadget use  $1 + 4 + 8 = 13$  as the attribute value.

Example VIEW.BAS, sets up a viewing environment (a screen and two windows) and uses the PRINT command as graphics output to illustrate redirection of graphics output. Type in and run the program in VIEW.BAS.

```
REM
REM   VIEW.BAS
REM
REM   Open screen 1 in high resolution, non-interlaced
REM   mode using three bit planes.
REM
SCREEN 1,640,200,3,2
REM
REM   Open a Window on Screen 1 with a drag bar gadget
REM   and a depth arrangement gadget (2+4=6).
REM
WINDOW 2,"Window 2",(0,0)-(300,100),6,1
PRINT "Output to Window 2."
PRINT "Click mouse to continue"
Checkmouse1: IF NOT MOUSE(0) THEN Checkmouse1
REM
REM   Open a second window titled "Window 3" with all
REM   possible attributes. Show that the new window
REM   becomes the current output window by printing.
REM
WINDOW 3,"Window 3",(200,100)-(617,186),31,1
PRINT "Output to window 3"
PRINT "Click mouse to continue"
Checkmouse2: IF NOT MOUSE(0) THEN Checkmouse2
REM
REM   Make window 2 the output window without bringing
REM   it to the front.
REM
WINDOW OUTPUT 2
PRINT "Output window 2 specified"
PRINT "using the WINDOW OUTPUT command"
PRINT "Click mouse to continue"
Checkmouse3: IF NOT MOUSE(0) THEN Checkmouse3
```

```
REM
REM   Use the WINDOW command to bring window 2 to the
REM   front and make the current output window.
REM
    WINDOW 2
    PRINT "Using the window command to bring"
    PRINT "window 2 to the front and make it"
    PRINT "the current output window"
    PRINT "Click mouse to continue"
    Checkmouse4: IF NOT MOUSE(0) THEN Checkmouse4
REM
REM   Close both windows and screen 1.
REM
    WINDOW CLOSE 2
    WINDOW CLOSE 3
    SCREEN CLOSE 1
    END
```

The following is a summary of the most important features and limitations of Screens and Windows.

## Screens

- An AmigaBASIC graphics Screen is the backdrop that defines the type of display possible.
- Up to four screens may be defined, but only one at a time may be opened.
- SCREEN screen\_num,width,height,mode  
This command defines a screen. This involves allocating display memory, setting the dimensions, and setting the interlacing mode.
- SCREEN CLOSED screen\_number  
Closes the screen and deallocates its display memory.
- SCREEN OPEN screen\_number  
Reopens a previously defined but closed screen.
- A screen need not fill up the entire display surface, but it must always be bottom, left positioned.

## Windows

- AmigaBASIC does not let you use a screen until a window has been opened on it.

TABLE 6.4. Memory Requirements (kilobytes of RAM)

# Bitplanes	Low resolution		High resolution	
	No interlacing (# of kilobytes)	Interlacing (# of kilobytes)	No interlacing (# of kilobytes)	Interlacing (# of kilobytes)
1	8	24	16	32
2	16	48	32	72
3	24	72	48	112
4	32	96	64	128
5	40	120	--	---

- More than one window may be opened on a screen simultaneously.
- Graphics output always goes to the window that is currently active.
- WINDOW window\_num
- lo-res mode: 8k RAM per bitplane
- hi-res mode: 16k RAM per bitplane
- interlacing mode: 16k RAM per bitplane

Table 6.4 gives some statistics on the amount of memory required for various displays.

Notice that high resolution mode using interlacing uses 128k of RAM. This is half of the default memory dedicated to the display only. Be careful in your use of a large amount of memory, it can slow down the processor appreciatively.

## Color Resolution

The maximum number of simultaneously displayable colors is determined by the depth parameter in the SCREEN command. Due to the huge memory requirement for color displays, a finite limit exists for the maximum number of different colors you can display at one time.

- High resolution: 4 bitplanes max. =  $2^4 = 16$  colors
- Low resolution: 5 bitplanes max. =  $2^5 = 32$  colors

Deciding on the number of bitplanes (colors) depends on your application. It is always wise to dedicate as little memory as necessary to



---

your graphics display. If you must have 32 separate colors, then you need 5 bitplanes of memory and must use a low-resolution display.

## Selecting Colors

Colors are named numerically, starting with 0 and continuing to 31 for a total of 32 color names. Each color index is referred to as a *pen*, and really is just an identifier pointing to a memory location describing the color of that pen. Think of each color index as a means of identifying which “inkwell” to draw the current color from. Color selection is accomplished using the `COLOR` command.

```
COLOR [ foreground_pen_number ] [ , background_pen_number ]
```

Each of the 32 inkwells can be filled with any of the 4,096 possible shades on the Amiga, but the maximum number of simultaneously displayable colors is limited by the depth (number of bitplanes) allocated to the screen. In low resolution, a maximum of 5 bitplanes may be utilized to give a total of  $2^5 = 32$  simultaneously displayable colors. High resolution screens are limited to 4 bitplanes of memory ( $2^4 = 16$  colors).

The `CLS` “clear screen” command is used to fill the current output window with the specified background color. Default color definitions for pens 0 to 31 are listed in Table 6.5.

To redefine the color transmitted by a particular pen, use the `PALETTE` command:

```
PALETTE pen_number, red_value, green_value, blue_value.
```

The red, green, and blue values are levels of each of these primary colors defining the new color. Valid values for each color component range from 0 to 1. The RGB color model in Figure 6.1 shows that any color may be thought of as a 3D coordinate of (red,green,blue) values. For instance, white: (red = 1, green = 1, blue = 1) and black: (0,0,0).

So far, you’ve learned about the components of the AmigaBASIC graphics viewing environment. Now that you understand screens, windows, and colors, try drawing some graphics using the program `COLORBAS`.

TABLE 6.5.

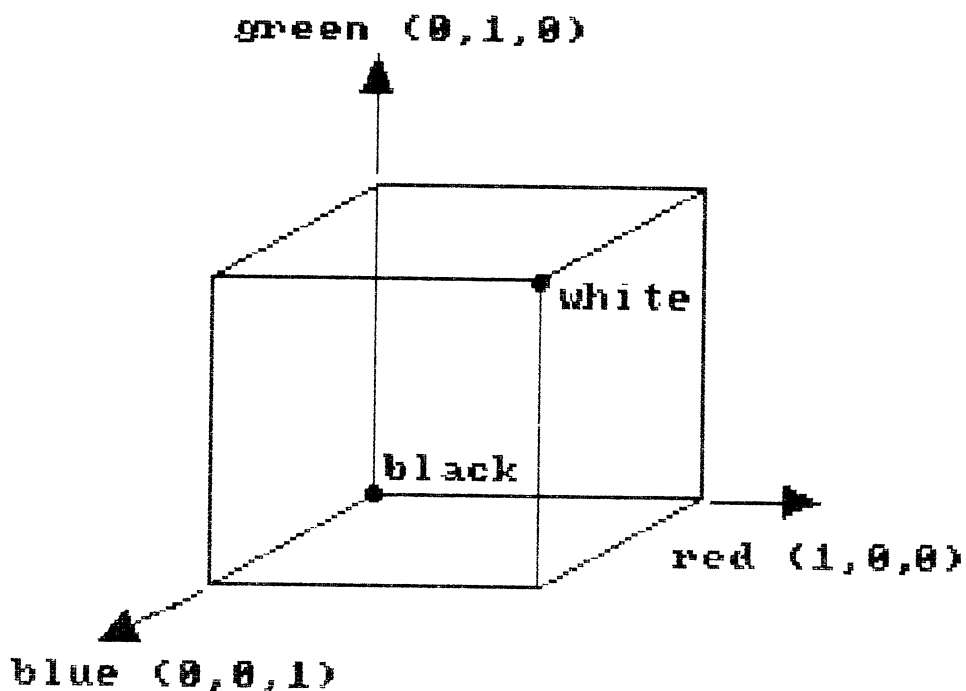
Pen	Color	Red	Green	Blue
0	Dark Blue	0	.3	.6
1	White	1	1	1
2	Black	0	0	.1
3	Orange	1	.5	0
4	Blue	0	0	1
5	Magenta	1	0	1
6	Cyan	0	1	1
7	White	1	1	1
8	Brown	.4	.1	0
9	Red-Orange	.9	.3	0
10	Lime Green	.55	1	.05
11	Gold	.9	.7	0
12	Blue	.3	.3	1
13	Violet	.55	.1	1
14	Blue-green	0	1	.5
15	Gray	.75	.75	.75
16	Black	0	0	0
17	Red	.8	.1	.1
18	Black	0	0	0
19	Tan	1	.75	.6
20	Gray (dark)	.25	.25	.25
21	Gray	.3	.3	.3
22	Gray	.4	.4	.4
23	Gray	.45	.45	.45
24	Gray	.5	.5	.5
25	Gray (medium)	.55	.55	.55
26	Gray	.6	.6	.6
27	Gray	.7	.7	.7
28	Gray	.75	.75	.75
29	Gray	.8	.8	.8
30	Gray (light)	.9	.9	.9
31	White	1	1	1

```

REM
REM  COLOR.BAS
REM
REM  This program illustrates how color indices can
REM  be modified to transmit a new color to the display.

```

FIG. 6.1 The RGB Color Model.



```

REM
  SCREEN 1,640,200,3,2
REM
REM   Open window 2 using the default window size (full
REM   display) and the default attributes (no gadgets).
REM
  WINDOW 2,"COLOR.BAS",,0,1
REM
REM   Set the background color index to 2 (default = black)
REM
  COLOR 1,2
  CLS
  PRINT "Click left mouse button to start"
  pause: IF NOT MOUSE(0) THEN pause
REM
REM   Continuously change color index 2 in a loop to a random
REM   color. The background changes color because it is
REM   drawn using color index 2.

```

```
REM
PRINT
PRINT "Click left mouse button to exit"
Checkmouse:
  red = RND
  green = RND
  blue = RND
  PALETTE 2,red,green,blue
  IF NOT MOUSE(0) THEN Checkmouse
REM
REM   Close window and screen, exit program.
REM
  WINDOW CLOSE 2
  SCREEN CLOSE 1
END
```

## OUTPUT PRIMITIVES

---

Output primitives can be roughly divided into:

- Points
- Lines
- Polylines
- Shapes
- Text
- Polygons

The word “primitive” implies that these graphics elements are generally the building blocks for much more complicated objects.

Each of the output primitives can have various attributes applied to it. For example, points can use various colors, lines can use both colors and styles and so on.

### Drawing Points

Drawing points is the process of coloring individual pixels (dots) on the screen. Two commands are useful for this task:

```
PSET [STEP] (x,y) [,pen]
PRESET [STEP] (x,y) [,pen]
```

These commands are identical except that PSE uses the foreground pen and PRESET uses the background pen when the pen specification is omitted. (x,y) is the absolute coordinate of the pixel to be colored, unless the *Step* parameter is included. Including *Step* causes the interpretation of (x,y) as a relative coordinate. See the program POINTS.BAS.

---

PSET (50,65)	Colors the pixel 50 points to the right and 65 points down from the current window's upper-left corner the current foreground color
PSET (50,65),3	Uses pen 3
PSET STEP (25,-10),3	Colors the pixel 25 points to the right and 10 points up from the <i>current pixel position</i> using pen 3

---

```

REM
REM   POINTS.BAS
REM
REM   This program draws points in the specified
REM   region using the PSET and PRESET commands.
REM
      SCREEN 1,640,200,3,2
      WINDOW 2,"Drawing Points", (0,0)-(631,186),6,1
REM
REM   Set up the color palette.
REM
      PALETTE 0,0,0,0 'black
      PALETTE 1,1,0,0 'red
      PALETTE 2,0,1,0 'green
      PALETTE 3,0,0,1 'blue
REM
REM   Draw individual points with the specified color.
REM
      PRINT "Both PSET and PRESET draw points.
      PRINT "They behave identically when the pen is specified."
      PRINT "When the pen is not specified, PSET defaults to"
      PRINT "the current foreground color, and PRESET defaults"
      PRINT "to the current background color."
      pen = 1
      FOR x = 200 TO 350
        IF x > 250 THEN pen = 2
        IF x > 300 THEN pen = 3

```

```

        FOR y = 100 TO 150
            PSET (x,y),pen
        NEXT y
    NEXT x
    PRINT
    PRINT "Click the left button mouse to continue"
    pause1: IF NOT MOUSE(0) THEN pause1
REM
REM     Use the PRESET command to draw in the background color.
REM
    PRINT
    PRINT "Draw points using the current background color"
    FOR y = 115 TO 135
        FOR x = 225 TO 325
            PRESET (x,y)
        NEXT x
    NEXT y
REM
REM     Prompt for program exit.
REM
    PRINT
    PRINT "Click left button mouse to exit"
    pause2: IF NOT MOUSE(0) THEN pause2
    WINDOW CLOSE 2
    SCREEN CLOSE 1
    END

```

## Drawing Lines

Line drawing is accomplished using the LINE command.

```
LINE [[STEP] (x1,y1)-[STEP](x2,y2),[pen_number][,b[f]]
```

The starting coordinate (x1,y1), and the end coordinate (x2,y2) must be specified. The STEP qualifier will signal the use of relative coordinates. b is used to signal that a box is to be drawn, and f instructs Amiga BASIC to fill the box. (See "Drawing Filled Areas.")

---

LINE (100,50) - (300,50)	Draws a line between (100,50) and (300,50) using the current foreground color
LINE (100,50) - STEP (200,0)	Draws exactly the same as above line using a second relative coordinate
LINE STEP (0,0) - STEP (100,50),3	Uses pen 3 to draw a line from the current pixel position to a point 100 pixels right and 50 pixels down

---

## Line Patterns

Lines are drawn using the current line pattern. The default pattern is a solid line, but can be changed using the PATTERN command.

```
PATTERN [line_pattern][,area_pattern]
```

As the second parameter suggests, you also use the PATTERN command to specify area fill patterns. Discussion of this feature is continued under "Fill Areas."

Line patterns are specified as binary (base two) numbers that are 16 digits wide:

```
1111111111111111    (solid line)
1010101010101010    (dotted line)
```

The digit "1" directs AmigaBASIC to color a pixel using the current foreground color. 0 leaves a pixel uncolored. Although basic line patterns are only 16 pixels wide, they are repeated over the entire length of the specified line.

When setting a line pattern, you may use only two representations: decimal or hexadecimal values. For example,

```
PATTERN 65535
```

gives a solid line because 65535 (base 10) = 1111111111111111 (base 2). An easier way to set line patterns is to use hexadecimal (base 16) values. Table 6.6 shows the correspondence between hexadecimal digits and dot patterns.

By dividing the 16-digit wide binary representation into four adjacent groups, you can specify line patterns quite easily (see Table 6.7).

TABLE 6.6.

Hexidecimal Digit (Base 16)	Pixel Pattern
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

To tell AmigaBASIC that you are using a hexidecimal number, add “&H” before the hex digits.

```
PATTERN &HF0F0
```

The use of hexidecimal numbers is shown in the program LINES.BAS.

TABLE 6.7.

Hexidecimal	Binary
FOFO	1111000011110000 .....
8181	1000000110000001 .....
FFFF	1111111111111111 .....



```
REM
REM   LINES.BAS
REM
REM   This program demonstrates line drawing using various
REM   colors and patterns.
REM
SCREEN 1,640,200,3,2
WINDOW 2,"Drawing Points",,6,1
PALETTE 4,0,1,0 'make color 4 green
REM
REM   Draw the first three lines using absolute coordinates.
REM
   y = 0
   FOR i = 1 TO 3
     IF i = 1 THEN PATTERN &HFFFF '1111111111111111
     IF i = 2 THEN PATTERN &HFF00 '1111111100000000
     IF i = 3 THEN PATTERN &HF0F0 '1111000011110000
     y = y + 25
     LINE (20,y) - (600,y),i
   NEXT i
REM
REM   Draw the last three lines using a combination of absolute
REM   and relative coordinates.
REM
   FOR i = 4 TO 6
     IF i = 4 THEN PATTERN &H6666 '0110011001100110
     IF i = 5 THEN PATTERN &HE4E4 '1110001111100100
     IF i = 6 THEN PATTERN &H3C3C '0011110000111100
     y = y + 25
     LINE (20,y)- STEP (580,0),i
   NEXT i
REM
REM   Pause before exiting program.
REM
PRINT "Click the left mouse button to exit"
pause2: IF NOT MOUSE(0) THEN pause2
WINDOW CLOSE 2
SCREEN CLOSE 1
END
```

## Drawing Shapes

A number of useful shape drawing commands exist in AmigaBASIC. The first uses the LINE command.

```
LINE [[STEP] (x1,y1) - [STEP] (x2,y2),[pen_number][,b[f]]
```

To draw a rectangle, use the b parameter.

---

LINE (x1,y1) - (x2,y2),,b	Draws a box with its upper left corner at (x1,y1) and lower left corner at (x2,y2). Without the b, a straight line will be drawn between the points. Use current foreground pen.
LINE (50,100) - STEP (100,100),2,b	Draws a box with its upper left corner at (50,100) and its lower right corner 100 pixels down from that coordinate. Uses pen 2.

---

These commands are used in the program FIGURES.BAS.

```
REM   Figures.bas
REM
REM   This program uses the LINE command to draw a
REM   series of unfilled figures.
REM
REM   SCREEN 1,640,200,3,2
REM   WINDOW 2,"Figures.bas",,6,1
REM   PALETTE 4,0,1,0
REM
REM   Draw a red, unfilled triangle.
REM
REM   LINE (100,60) - (160,10),3
REM   LINE STEP (0,0) - STEP(60,50),3
REM   LINE STEP (0,0) - STEP (-120,0),3
REM
REM   Draw an unfilled, white rectangle with its upper
REM   left corner at (x,y). Height = h, Width = w.
```

```
REM
  x = 100: y = 100
  h = 50: w = 120
  LINE (x,y) - STEP (w,0),1
  LINE STEP (0,0) - STEP (0,h),1
  LINE STEP (0,0) - STEP (-w,0),1
  LINE STEP (0,0) - STEP (0,-h),1
REM
REM      Call subprogram Poly to draw a figure with the
REM      specified position, radius, color, and number of sides.
REM
  col = 2: radius = 100: nsides = 5
  xcent = 400: ycent = 100
  CALL poly (xcent,ycent,radius,nsides,col)
REM
REM      Prompt user to end program.
REM
  PRINT "Click left mouse button to end"
  Checkmouse: IF NOT MOUSE(0) THEN Checkmouse
  WINDOW CLOSE 2
  SCREEN CLOSE 1
  END

REM
SUB poly (xc,yc,rad,n,col) STATIC
REM
  DIM x(20),y(20)
  twopi = 2*3.1415
  inc = twopi/n
  IF n MOD 2 > 0 THEN angle = -inc - twopi/4 ELSE angle = -inc
  IF n > 20 THEN GOTO ToomanySides
  FOR vertex = 1 TO n
    angle = angle + inc
    x(vertex) = xc + CINT(rad*COS(angle))
    y(vertex) = yc + CINT(.44*rad*SIN(angle))
  NEXT vertex
  last = n + 1
  x(last) = x(1)
  y(last) = y(1)
```

```

FOR side = 1 TO n
  LINE (x(side),y(side))-(x(side+1),y(side+1)),col
NEXT side
GOTO jump
Toomany_sides: PRINT "Too many sides"
jump:
END SUB

```

## Drawing Circles, Arcs, and Ellipses

AmigaBASIC provides the *Circle* command to draw circles, arcs, and ellipses.

```

CIRCLE [STEP](x,y),radius[,pen_number]
[,start_angle,end_angle][,aspect_ratio]]

```

The only necessary values are the coordinate of the center point and the radius. All other parameters apply to either arcs or ellipses.

The *start\_angle* and *end\_angle* apply to both arcs and “partially ellipses.” All angles are specified in radians, so it is often useful to use the conversion factor ( $\pi/180$ ), i.e., radians = dig ( $\pi/180$ ). Try creating the figures in the program CURVES.BAS.

```

REM
REM   CURVES.BAS
REM
REM   This program shows the use of the CIRCLE command to
REM   draw various figures having curved surfaces. The
REM   CIRCLE command is capable of drawing circles,
REM   arcs, pie-shaped wedges, and ellipses.
REM
SCREEN 1,640,200,3,2
WINDOW 2,"CURVES.BAS",,6,1
PALETTE 4,0,1,0 'Change to green (blue is default).
REM
REM   Draw a circle centered at (320,90) with radius 90.
REM
CIRCLE (320,100),90
REM
REM   Draw a circular arc from 8 to 270 degrees. The

```

```
REM      start and stop angles must be converted to radians.
REM
      dtr = 3.1415/180!
      sangle = 0
      eangle = 270*dtr
      CIRCLE (320,100),80,2,sangle,eangle
REM
REM      Draw a pie shaped wedge by passing negative values.
REM
      sangle = 45*dtr
      eangle = 90*dtr
      CIRCLE (320,100),70,3,-sangle,-eangle
REM
REM      Draw an ellipse by setting a non-default aspect ratio.
REM
      CIRCLE (320,100),60,4,,4!
REM
REM      Prompt the user to end program.
REM
      PRINT "Click left button mouse to end"
      pause: IF NOT MOUSE(0) THEN pause
      WINDOW CLOSE 2
      SCREEN CLOSE 1
      END
```

## Drawing Filled Areas

Filled areas are two-dimensional closed geometric figures that can be filled with:

- A solid fill color
- A user-specifiable fill pattern

There are three sets of commands to create filled areas in AmigaBASIC:

- LINE (using the "bf" parameter)
- AREA/AREAFILL
- PAINT

### USING LINE TO CREATE FILLED RECTANGLES:

LINE [STEP] (x1,y1) - [STEP] (x2,y2), pen\_number, bf

- The "bf" qualifier instructs AmigaBASIC to create a filled box.
- LINE (100,50) - STEP (50,50),bf Draws a rectangle 50 pixels on a side and fills it with the current foreground color.
- LINE (100,50) - STEP (50,50),3,bf Same as above, except pen 3 is used for the fill.

Try the program BOXFILL.BAS to see these commands at work.

```

REM
REM   BOXFILL.BAS
REM
REM   This program demonstrates the use of the LINE command
REM   to fill rectangular polygons with solid fill colors.
REM
    SCREEN 1,640,200,4,2
    WINDOW 2,"BOXFILL.BAS",,6,1
REM
REM   Draw a series of filled rectangles using default
REM   color indices.
REM
    col = -1
    FOR x = 75 TO 475 STEP 100
        FOR y = 20 TO 120 STEP 50
            col = col + 1
            LINE (x,y) - STEP (90,40),col,bf
        NEXT y
    NEXT x
REM
    PRINT "Click left mouse button to end"
    Checkmouse: IF NOT MOUSE(0) THEN Checkmouse
    WINDOW CLOSE 2
    SCREEN CLOSE 1
    END

```

### USING AREA/AREAFILL:

AREA [STEP] (x,y)

- Specify up to 20 points (one at a time) to define the vertices of the desired area. The order they are defined is the order they will be drawn.
- You need not specify the starting point twice since the first and last points will automatically be connected.
- The area will be filled with the current foreground color when the AREAFILL command is issued as shown in the program AREAFILL.BAS.

```
6
REM
REM   AREAFILL.BAS
REM
REM   This is an interactive program that uses the AREA and
REM   AREAFILL commands to fill an area with the specified
REM   fill color.
REM
SCREEN 1,640,200,4,2
WINDOW 2,"AREA.BAS",,6,1
REM
PRINT "Click the left button mouse once at each vertex"
PRINT "of the desired area. Click in box to stop collection"
LINE (0,180) - STEP (20,20),3,bf
collect:
  Checkmouse: IF NOT MOUSE(0) THEN Checkmouse
  x = MOUSE(1): y = MOUSE(2)
  IF x < 20 AND y > 180 THEN colrprompt
  PSET (x,y),1
  AREA (x,y)
  GOTO collect
colrprompt:
  PRINT "Input desired color (0-15)"
  INPUT col
  IF col < 0 OR col > 15 THEN colrprompt
  COLOR col,0
  AREAFILL 0
REM
PRINT "Click left mouse button to end"
pause: IF NOT MOUSE(0) THEN pause
```

```
WINDOW CLOSE 2
SCREEN CLOSE 1
CLS
END
```

### USING PAINT:

```
PAINT [STEP] (x,y) [,fill_pen[,border_pen]]
```

- The PAINT command is used to flood-fill an existing enclosed area.
- (x,y) is the coordinate of any point within the enclosed area.
- fill\_pen is the number of the pen used to fill the area.
- border\_pen is the pen used where the filling ends.
- If the shape is not entirely closed, the fill color will escape through the gap and spread out to cover the entire window.

### Warning:

- You can't use PAINT in a window that uses an attribute value greater than 15. Be careful! The default output window has an attribute value of 31, so it is not safe to use PAINT in that window.
- Be careful not to specify an (x,y) coordinate pair outside the current output window. Doing this can write to areas of memory that do not belong to the display, and this can crash the system.

Try the program AREAPAIN.TBAS to practice the PAINT command.

```
REM
REM   AREAPAIN.TBAS
REM
REM   This program draws a circle with an intersecting
REM   line, then fills in regions using paint.
REM
REM   WINDOW 2,"AREAFILL.TBAS",(0,0)-(320,100),24
CIRCLE (160,50),60,3
LINE (0,50) - (320,50),3
PAINT (160,45),3
PAINT (160,55),2,3
END
```



TABLE 6.8.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
0																	&H0000
1		1	1		1	1				1	1		1	1			&H6C6C
2	1	1	1	1	1	1	1		1	1	1	1	1	1	1		&HFEFE
3	1	1	1	1	1	1	1		1	1	1	1	1	1	1	1	&HFEFE
4		1	1	1	1	1				1	1	1	1	1			&H7C7C
5			1	1	1						1	1	1				&H3838
6				1								1					&H1010
7																	&H0000

### Filling with Patterns

PATTERN [line\_pattern][,area\_pattern]

- Area patterns are the 2D analog of line patterns.
- Area patterns are 16 pixels wide, but are also several lines high. Valid heights are any value of 2 up to 64 (0,2,4,8,16,32,64).
- The area fill pattern is an array of 16 bits (short integers). Dimension the array to the appropriate size for the height (e.g., DIM Pattern% (3) dimensions an array of short integers for a maximum height of 4). (The array starts and ends at index 0.)
- To define your own pattern, follow the procedure below and read the program PATTERN.BAS and Table 6.8.

1. Copy the regular grid shown in Table 6.8 (photocopying easiest).
2. Place the digit "1" in every box you wish to color.
3. Break each horizontal line into four adjacent digit breaks.
4. Write the four-digit hexadecimal representation for each line (use Table 6.8).

```

REM
REM   PATTERN.BAS
REM
REM   This program uses the PATTERN command to fill in
REM   a rectangular region.
    
```

```

REM
WINDOW 2,"PATTERN.BAS",,24
DIM pat%(7)
pat%(0) = &H0      '0000000000000000
pat%(1) = &H6C6C   '0110110001101100
pat%(2) = &HFEFE   '1111111011111110
pat%(3) = &HFEFE   '1111111011111110
pat%(4) = &H7C7C   '0111110001111100
pat%(5) = &H3B3B   '0011100000111000
pat%(6) = &H1010   '0001000000010000
pat%(7) = &H0      '0000000000000000

REM
REM   Set the line pattern to &hff00 and the area pattern
REM   to "hearts".
REM
PATTERN &HFF00,pat%
LINE (40,40) - (600,160),3,bf
END

```

## Graphics Library Subroutines

Many useful operating system graphics routines can be accessed by using the `LIBRARY` statement. This will let AmigaBASIC know how to find the routine and also how to interact with it. To access the graphics library, the appropriate statement is:

```
LIBRARY "graphics.library"
```

When you use this statement, AmigaBASIC looks in the current directory for the file "Graphics bmap." This file is included on the AmigaBASIC diskette in the BasicDemos directory (drawer). It must be present in the current disk directory when you use the `LIBRARY` statement in a program.

Operating system formalities also require that you declare certain functions. The `DECLARE FUNCTION` statement lets AmigaBASIC know that it must search the graphics library for the specified function. For example, `ReadPixel` is a graphics library function that returns the number of the pen used to color a particular pixel. Therefore, you must use the `DECLARE FUNCTION` statement:

```
DECLARE FUNCTION ReadPixel&() LIBRARY
```

The ampersand following the name indicates that *ReadPixel* is a long integer. It is necessary, so don't forget to include it as shown in the program READPIXEL.BAS.

```

REM
REM   READPIXEL.BAS
REM
REM   This program demonstrates the use of the Kernel
REM   Graphics Routine ReadPixel& and Move&.
REM
REM   LIBRARY "graphics.library"
REM   DECLARE FUNCTION ReadPixel&() LIBRARY
REM
REM   Use a high resolution, interlaced screen.
REM
REM   SCREEN 1,640,200,4,2
REM   WINDOW 2,"READPIXEL.BAS" ,,12,1
REM
REM   Display colored boxes using the default colors.
REM
REM   col = -1
REM   FOR y = 0 TO 150 STEP 10
REM     col = col + 1
REM     LINE (0,y) - STEP (40,10),col,bf
REM   NEXT y
REM
REM   rp& = WINDOW(8)
REM   ytext& = 0
REM   Checkmouse: IF NOT MOUSE(0) THEN Checkmouse
REM     x& = MOUSE(1): y& = MOUSE(2)
REM     Pen& = ReadPixel&(rp&,x&,y&)
REM     ytext& = ytext& + 10&
REM     CALL Move& (rp&,100&,ytext&)
REM     PRINT "Color Index Chosen = ";Pen&
REM     ytext& = ytext& + 10&
REM     CALL Move& (rp&,100&,ytext&)
REM     INPUT "Exit Program? (Y/N)",answer$
REM     IF answer$ = "n" OR answer$ = "N" THEN Checkmouse
REM
REM   WINDOW CLOSE 2
REM   SCREEN CLOSE 1
REM   END

```

## Selected Kernal Graphics Routines

**TEXT.** Utilizing text in your AmigaBASIC programs is extremely useful and often mandatory for quality displays. A summary of Amiga text features follows.

- Text on the Amiga is graphics. Therefore, you can mix lines, circles, filled areas, and text freely.
- Drawing modes affect text output.
- It is possible to print text outside the current output window, but you can use the WIDTH command to set the maximum line width.
- The maximum line width depends on the current window width and the size of the current text font.
- Setting the 80-column font as the default using Preferences, will make each character 8 pixels wide. This will give 80 characters for a hi-res window and 40 for a lo-res window. (The true width is actually 75 or 76 in hi-res and 37 or 38 in lo-res because the sizing gadget and border take up some room.)
- You must use the operating system routines OpenFont, SetFont, and CloseFont to select your own font. In addition, the textAttr function will allow you to set text attributes.

### SETTING THE MAXIMUM LINE WIDTH

```
WIDTH [line_size][,print_tab]
```

- line\_size: the maximum line width in characters.
- print\_tab: an optional parameter used to set the width of columns. The columns are used when a comma is added to the end of a PRINT statement.

## Opening a Font

To open a font, the following functions are required:

- VARPTR (variable\_name): Returns the address of the first byte of variable specified.
- SADD (string\_expression): Returns the address of the first byte of data in the specified string expression.
- CHR\$(I): Returns a string whose single character has the ASCII value given by I.

- **OpenFont (array&):** The graphics kernel function that opens a font. The array passed to this routine contains two entries:
  - array&(0) = the address of a text string (the font name) ending with an ASCII "0."
  - array&(1) = font height + additional information (height = 8 for 80-column display; height = 9 for 60-column display).

The function of **OpenFont** is to return a pointer to the font descriptor. Because **OpenFont** returns a value, you must use the **DECLARE FUNCTION** statement along with the **LIBRARY** statement to open the graphics library.

## Setting a Font

Once the pointer to a font has been returned, that pointer can be used to set the font for use in a specific window. To set a font, you must call the graphics kernel subroutine *SetFont*.

```
CALL SetFont (RP&, FontPtr&)
```

- **RP&:** The address of the window's raster port structure (**WINDOW(8)**).
- **FontPtr&:** The pointer found by the **OpenFont** function.

## Closing a Font

When you are finished with a particular font, or wish to open a new one, close the open font.

```
CALL CloseFont& (FontPtr&)
```

To see these commands working, try the program **FONT.SBAS**.

```
REM
REM  FONT.SBAS
REM
    DEFLNG a-z
    DECLARE FUNCTION OpenFont LIBRARY
    LIBRARY "graphics.library"
    WINDOW 2,"FONT.SBAS",,12
```

```

REM
REM   Select a character height (8 and 9 are valid).
REM   Choose the system font and print a sample.
REM
    height = 8
    CALL Choosefont ("topaz.font",height)
    PRINT
    PRINT "This is the topaz font at height 8"
REM
    height = 9
    CALL Choosefont ("topaz.font",height)
    PRINT
    PRINT "This is the topaz font at height 9"
REM
    rp = WINDOW(8)
    CALL Move (rp,320,100)
    PRINT "Text positioned with Move"
REM
    WIDTH 5
    PRINT
    PRINT "12345678901234567890"
    WIDTH 60                                'Restore text width "df
    END

REM
REM SUBPROGRAM FONT
REM
REM   This subprogram selects the desired font and font size.
REM
    SUB Choosefont ( fontname$, height) STATIC
REM
    DEFLNG a-z
    rp = WINDOW(8)                            'get raster port address
    TextAttr(0)=SADD(fontname$+CHR$(0))      'set font address
    TextAttr(1)=height*65536&                'set font height
    IF FontPtr THEN CloseFont FontPtr        'close an opened font
    FontPtr = OpenFont(VARPTR(TextAttr(0)))  'set font pointer
    CALL SetFont (rp,FontPtr)                'set the current font
    END SUB

```

## AmigaBASIC Animation

Animation in AmigaBASIC provides two approaches:

- Use of bitmap manipulations.
- Use of sprites.

---

### **BLITTER OBJECTS (BOBS)**

---

“Bitmap” refers to the pattern of memory bits that defines the current color of every pixel on the screen. Monochrome devices use one bitplane of memory, which maps 1 bit to every pixel on the screen. Each bit can contain either a 0 or a 1, corresponding to either black or white on the screen. Color devices use multiple bitplanes. Thus, every pixel has 2,3,4,5,etc., bits defining its color.

Bitmap animation refers to the process in which a specified region of display memory is copied and used in a temporary reassignment of the bit values in other areas of the display memory. Continuous reassignment to new areas of display memory gives the impression of object motion.

In general, the use of bitmap animation requires the following sequence of steps:

1. Save the background image bitmap.
2. Draw the desired object on the screen bitmap.
3. Restore the background image bitmap.
4. Move the object to a new location on the screen.

Every time you want to move a bitmap object, you must perform this sequence of steps. Most computers require assembly language instructions written specifically for this purpose. The Amiga provides you with both hardware and software support for these functions. Included in the Amiga's hardware is a device known as a *blitter* (*block image transferer*). It is designed to move entire blocks of bitmap images at one time. Using the available operating system software support, these *blitter objects*, *bobs* for short, can be identified and moved independently of the rest of the display, even though they are technically just part of the entire screen.

## SPRITES

---

Sprites are animation objects drawn using an entirely different hardware system than that used for the normal bitmap display. They are much easier to move than bitmap images because their location is specified by x and y coordinates stored in a hardware register. Some of the limitations of sprites include:

- Support of only eight hardware sprites. (We'll discuss virtual sprites a little later.)
- Sprites are always displayed in low-resolution mode, even if the current screen has a higher resolution.
- Each sprite can be a maximum of 16 pixels wide, although they can be any desired height.
- Sprites can contain a maximum of three foreground colors and one background color each.
- Each pair of sprites shares a set of color registers.

Although some of these limitations may appear rather serious, the Amiga's operating system overcomes many of them.



---

## VIRTUAL SPRITES

---

The concept of *virtual sprites* allows the use of more than just the 8 hardware sprites. Because the display hardware is capable of changing display characteristics as each line is displayed on the screen, you can reposition a sprite after it has been displayed and show it again somewhere lower on the screen. (Remember that the display is scanned from top to bottom.)

So, although the Amiga supports only eight hardware sprites, each of these can be redisplayed at different screen positions many times. Virtual sprites, *Vsprites*, do have one limitation: No more than four sprites can have the same colors on any particular horizontal line.

---

## SPRITES OR BOBS: WHICH TO USE?

---

To you, the AmigaBASIC programmer, the user-interface to sprites and bobs appears the same. All of the OBJECT commands discussed work with either bobs or sprites. Deciding to use bobs, sprites, or both depends to a great extent on what you want to accomplish. The following section describes the unique behaviors of these special objects to help you decide which is best suited to your application.

### Features of Bobs and Sprites

---

Resolution:	Sprites always exist in low resolution. Bobs have the resolution of the screen on which they are displayed.
Size:	Vsprites are limited to 16 pixels in width, but may be any height. Bobs can be almost any size, but can be limited by the available display memory.
Number:	Only six Vsprites of the same color, and only four Vsprites of different colors can appear on any particular horizontal line. The number of bobs is limited only by display memory.

Speed:	Vsprites are fast. Bobs are somewhat slower. The more bobs you have, the slower they move.
Number of Colors:	Vsprites can have only three foreground colors and one background color. Bobs can use as many colors as available on a screen.
Selection of Colors:	Vsprites can have colors completely different from those used for the current screen. Vsprites use color registers 16-31, therefore, they can be used to add more colors to the screen without allocating additional bitplanes. Bobs can use any of the color registers (0-31). When used in unison with Vsprites, it is not advisable to use a 32-color screen.
Color Priority:	Vsprites will always appear in front of bobs. Bobs can be assigned priorities. The priorities determine which bobs will be displayed in front of others.
Display Considerations	Vsprites can be moved in and out of any window freely. Bobs never move outside the border of their assigned windows. Vsprites stay on the screen even after their windows are closed. Bobs disappear when their windows are either covered or closed.

---

Another consideration regarding Vsprites is the possible color conflict with the pointer. The mouse pointer is actually a sprite, so changing the Vsprite color registers can affect the mouse pointer. Since additional colors are made available for sprites by changing their color registers as they move up and down the screen, the mouse pointer, too, can change colors in different horizontal portions of the screen.

## **CREATING AN OBJECT—USING THE OBJECT EDITOR**

---

Creating an object for animation is easily done by running the AmigaBASIC program ObjEdit, which is in the BasicDemos drawer on the Amiga Extras diskette.

---

To open the Object Editor and start operations:

1. Insert the Amiga Extras diskette.
2. Double click on the Amiga Extras icon.
3. Double click on the BasicDemos drawer.
4. Double click on the ObjEdit icon.

At this point you'll be in the Object Editor and the following prompt will appear:

```
Enter 1 if you want to edit sprites  
Enter 0 if you want to edit bobs>
```

If you don't know which to choose, review the preceding section. When you've decided, enter your selection and press *Return*.

When the Object Editor window appears, press the right mouse button over the *Files* menu to see its contents. To create a new object, select *New*. To modify an existing object, select *Open*.

Choose the way you want to create the image using a *Tools* menu option. You can select free-form, oval, rectangle, or line. *Erase* will remove any part of the drawing.

Move the pointer to the desired starting point on the canvas. Press the selection button down and hold it. Move the pointer to the desired end position and release it. All drawing or erasing will cease if you move the pointer outside the frame, but will resume when reentering the frame.

Select a color by moving the pointer to the color choice bar at the bottom of the screen and clicking on a color. Each new image created on the screen will be outlined with this color.

Select an interior fill color for your object by choosing the desired color from the choice bar, then selecting the *Paint* option from the *Tools* menu. Position the pointer over the region you wish to paint and click the left mouse button. If the region you selected is not completely enclosed the paint will "leak" out into the surrounding area.

The canvas can be enlarged by using the *Sizing* gadget in the lower right corner, but it is treated as an object. To create multiple objects, they must be created on separate canvases and saved in separate files.

When your object is complete, choose *Save As* from the *File* menu. When you are prompted, enter a file name and press return. Use a name descriptive of both the object and its identity as a sprite or bob (e.g., car.sprite or dog.bob).

## Loading the Object Data

When you have completed the design of your bob or sprite and have saved it to a file, the next step is to get the data into your AmigaBASIC program. Do this by opening the saved object file and reading the data into one large string variable. For example, suppose you named the file containing the object "Plane.sprite." The following AmigaBASIC commands load the sprite into the string named "PlaneImage\$."

```
OPEN "Plane.sprite" FOR INPUT AS 1
PlaneImage$ = INPUT$ (LOF(1),1)
CLOSE 1
```

## Creating the Object within AmigaBASIC

Once you load the object into a string variable, use the OBJECT.SHAPE command to create an object of the shape defined by the data. The form of the command is:

```
OBJECT.SHAPE object_num,data_string
```

- object\_num is an integer identifier greater than zero.
- data\_string is the string into which you have loaded the object data (PlaneImage\$ in the example above).

In addition to creating an original object, you can use the OBJECT.SHAPE command to create a new object with exactly the same shape as an existing object.

```
OBJECT.SHAPE new_object,existing_object
```

- new\_object is a unique, positive, nonzero integer identifier for the new object to be created.
- existing\_object is a unique, positive, nonzero integer identifier for the existing object.

Creating a new object from an existing one causes both to share the same image data. This is useful when you have similar objects and want to save memory. We will give you instructions later on how the objects can be given different colors.

---

## Displaying Objects

After the objects have been assigned a shape, the OBJECT.ON command is used to display them.

```
OBJECT.ON [object_num[,object_num]...]
```

For example:

```
OBJECT.ON 2,3,1, (Displays objects 2,3,and 1.)
OBJECT.ON      (All objects defined with
                OBJECT.SHAPE will be displayed.)
```

## Temporarily Removing Objects

To temporarily suspend the display of an object, use the OBJECT.OFF command.

```
OBJECT.OFF [object-num[object_num]...]
```

For example:

```
OBJECT.OFF 3,1      (Suspends display of objects
                    3 and 1.)
OBJECT.OFF          (Suspends display of all objects
                    defined using the OBJECT.SHAPE
                    command.)
```

## Permanently Removing Objects

Objects may be permanently removed by using the OBJECT.CLOSE command.

```
OBJECT.CLOSE [object_num[object_num]...]
```

For Example:

```
OBJECT.CLOSE 2      (permanently removes object 2.)
OBJECT.CLOSE        (permanently removes all objects
                    defined using the OBJECT.SHAPE
                    command.)
```

In addition to removing objects from the display, the `OBJECT.CLOSE` command also frees all the memory associated with the objects.

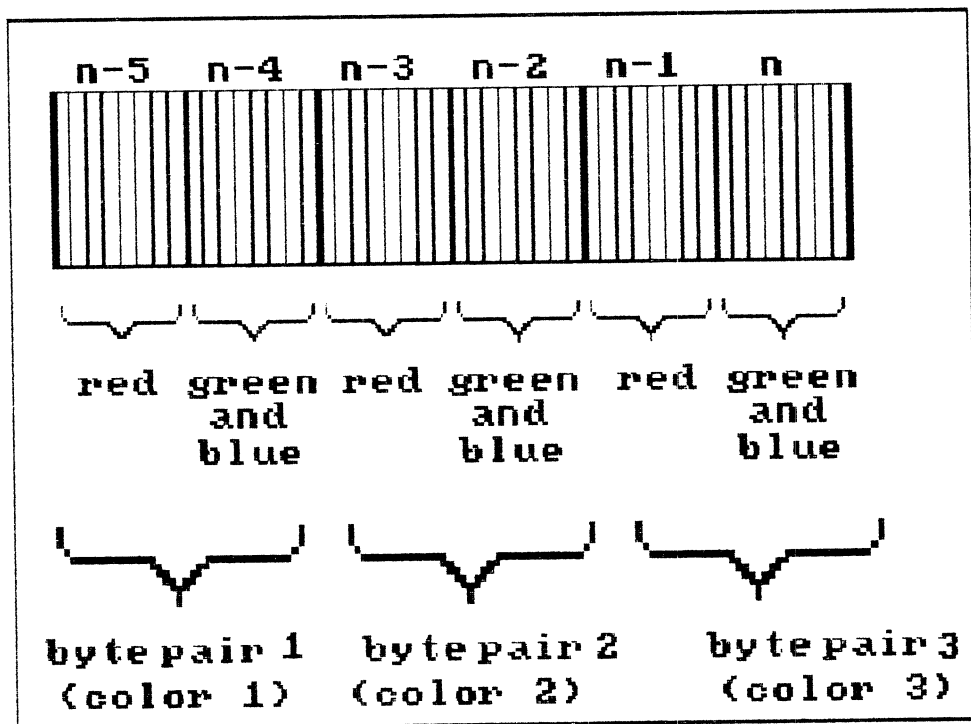
## Selecting Object Colors

Vsprites and bobs use different schemes to determine the colors they will use. Bobs use the colors defined for the current screen, whereas Vsprites use up to four of the colors in the range of 16 to 31.

**VSPRITE COLORS.** The foreground colors displayed by a Vsprite are specified in the last 6 bytes of its ObjEdit data file, as shown in Figure 7.1.

- The last 6 bytes of a Vsprite's ObjEdit file specify its foreground display colors.
- Each color is specified as a byte pair.

FIG. 7.1 The last 6 bytes of a Vsprite's ObjEdit data file.



- The first byte within a byte pair stores the green and blue components of the color.
- All color components are represented by value in the range of 0 to 15 (e.g., red = 15, green = 15, and blue = 15 yields the color white).
- The green-blue value is specified using the result obtained from the equation:

$$\text{greenblue} = 16 * \text{green} + \text{blue}$$

This equation loads the greenblue byte with the green value in the high bits and the blue value in the low bits of the byte.

For example,

```

green = 4
blue = 6
greenblue = 16*4 + 6 = 70

```

$2^6$	----->	Binary								
$+2^2$		1000000								
$+2^1$		+100								
		+ 10								
-----										
$+ 70$		<table border="1" style="display: inline-table; border-collapse: collapse; text-align: center;"> <tr> <td style="padding: 2px 5px;">0</td> <td style="padding: 2px 5px;">1</td> <td style="padding: 2px 5px;">0</td> <td style="padding: 2px 5px;">0</td> <td style="padding: 2px 5px;">0</td> <td style="padding: 2px 5px;">1</td> <td style="padding: 2px 5px;">1</td> <td style="padding: 2px 5px;">0</td> </tr> </table>	0	1	0	0	0	1	1	0
0	1	0	0	0	1	1	0			
		<table style="margin-left: 40px;"> <tr> <td style="border-top: 1px solid black; border-bottom: 1px solid black; padding: 0 5px;">green</td> <td style="border-top: 1px solid black; border-bottom: 1px solid black; padding: 0 5px;">blue</td> </tr> <tr> <td style="padding: 0 5px;">binary</td> <td style="padding: 0 5px;">decimal</td> </tr> <tr> <td style="padding: 0 5px;">(100 = 4)</td> <td></td> </tr> <tr> <td style="padding: 0 5px;">(110 = 6)</td> <td></td> </tr> </table>	green	blue	binary	decimal	(100 = 4)		(110 = 6)	
green	blue									
binary	decimal									
(100 = 4)										
(110 = 6)										

The default colors assigned to Vsprites by the ObjEdit program are the four colors defined in Preferences. If these colors are suitable, you don't need to change them. If you want to select your own colors, you can use the AmigaBASIC subprogram listed below.

```

SUBPROGRAM ColorChange (ImageName$, red,green,blue)
REM
DIM red(2), green(2), blue(2), greenblue(2)
length = LEN (ImageName$)
FOR I = 0 TO 2
    greenblue(I) = 16*green(I) + blue(I)
next I
col$ = CHR$(red(0)) + CHR$(greenblue(0))
col$ = col$ + CHR$(red(1)) + CHR$(greenblue(1))

```

```
col$ = col$ + CHR$ (red(2)) + CHR$ (greenblue(2))
MID$ (ImageName$, length - 5) = col$
RETURN
END
```

## Bob Colors

You can use two methods to change the colors with which your bobs will be drawn:

- Using the PALETTE command
- Using the OBJECT.PLANES command

Because bobs use the current pen definitions to display their colors, they can be changed by redefining pens with the PALETTE command. Remember, however, that all graphics drawn with a particular pen will change color when that pen is redefined.

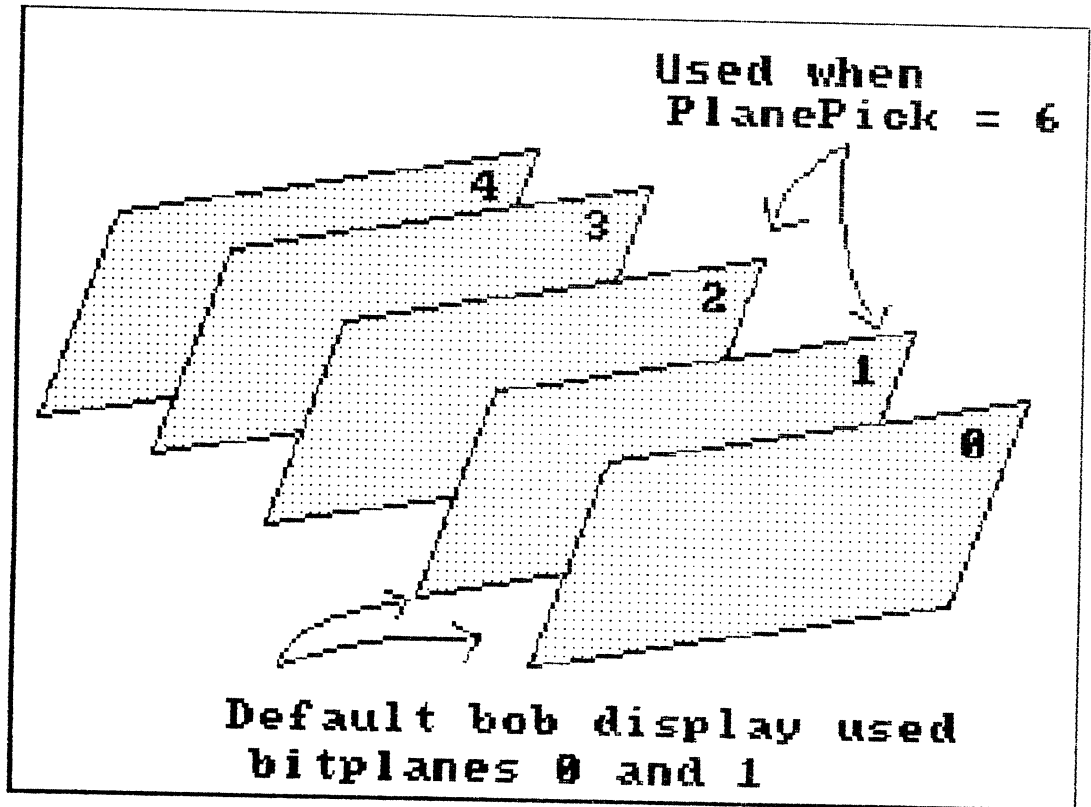
The second way you can redefine bob colors is to use the OBJECT.PLANES command to reassign the bit planes used to display the bob colors. Suppose you have a 2-bitplane bob ( $2^2 = 4$  colors max) and a current screen that can use up to 3 bitplanes (depth = 3). Normally the 2 bitplanes assigned to the bob are 0 and 1. On a screen with depth 3, however, you have the option of using bitplanes 1 and 2 instead of 0 and 1 (see Figure 7.2).

```
OBJECT.PLANES objectnumber [,PlanePick][,PonOff]
```

- Bitplanes are memory locations in which color display definitions are stored. Each bitplane has 1 bit of memory for each pixel on the screen. Each bit is capable of storing a 0 or a 1. A 1-bitplane screen is capable of displaying only two colors (0 = black, 1 = white). A 2-bitplane screen uses two bits per pixel for color definition. The possible values for each pixel are 00,01,10, and 11, for a total of four possible colors.
- The parameter PlanePick in the OBJECT.PLANES command is used to specify which bitplanes are to be used for a particular bob.
- $\text{PlanePick} = 2^n + 2^m$  where  $n$  and  $m$  are the bitplanes you wish the bob to use. That is, to use bitplanes 1 and 2 instead of 0 and 1, use  $\text{PlanePick} = 2^1 + 2^2 = 2 + 4 = 6$ .



FIG. 7.2 Bitplane characteristics.

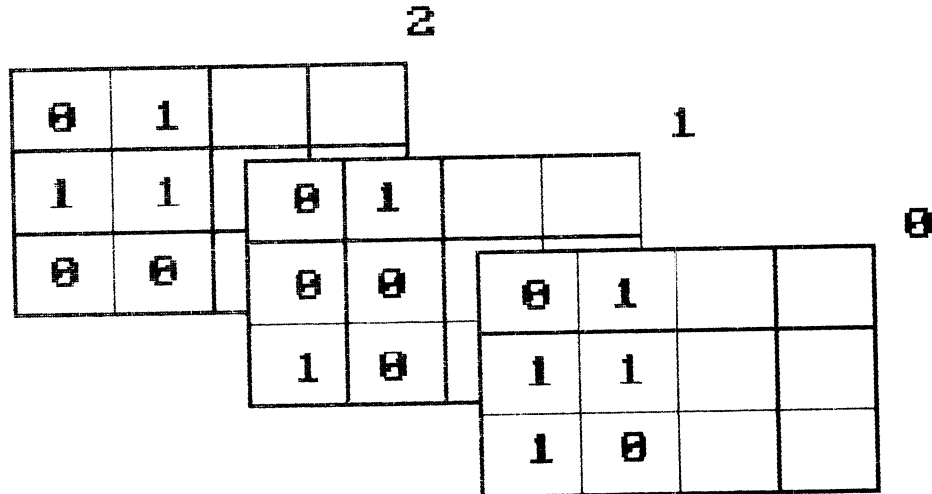


If your bob uses pens 1 and 2 originally, setting the PlanePick value to 6 causes the bob to be displayed with pens 2 and 4. Any part of the bob originally displayed in pen 3 is displayed using the color assigned to pen 6.

The second parameter in the OBJECT.PLANES command specifies whether or not you want to use additional planes to modify the colors displayed. A PonOff value of 1 tells AmigaBASIC to use bitplane 0 in a rather interesting way. If you use a PlanePick value of 6 (pens 2,4, and 6) and then set PonOff to 1, a value of  $2^1 (=2)$  will be added into any bit in the bitplane. Thus, if a bob originally used pens 1,2, and 3, the new pens will be 3,5, and 7. See Figure 7.3 for a pictorial example.

- When PlanePick = 6, bitplanes 1 and 2 will be in use.
- When PonOff = 1, bitplane 0 will be used to change the pens used.

FIG. 7.3 The OBJECT.PLANES command.



- If a 1 is stored in a particular location of plane 1 or 2, then the corresponding bit in bitplane 0 will be set to a 1.
- If zeroes are stored in the corresponding bits of bitplanes 1 and 2, then the corresponding bit in bitplane 0 will be assigned the value 0.

## Assigning Object Parameters

When bobs overlay on the display, the operating system will decide which bob will be displayed on top. The OBJECT.PRIORITY command can be used to force specific bobs to be displayed on top of others.

```
OBJECT.PRIORITY object_num,priority
```

The priority is an integer value from  $-32768$  to  $32767$ . Objects assigned a higher numeric priority are displayed on top of objects with lower priorities. For example, an object with priority 15 will be displayed on top of objects assigned priorities in the range  $-32768$  to 14. This command applies only to bobs. Vsprites will always be displayed on top of all other images.

## Positioning Objects

Vsprites and bobs are positioned using the OBJECT.X and OBJECT.Y commands.

```
OBJECT.X object_num, xcoordinate  
OBJECT.Y object_num, ycoordinate
```

- object\_num is the object identifier.
- xcoordinate and ycoordinate specify where the upper left corner of the object is to be placed.

Valid x,y coordinates for bobs and Vsprites are in the range of -32768 to 32767, but these objects are only visible in a much smaller range. Bobs are not visible anywhere outside the window to which they are assigned. Vsprites are visible from -15 to 639 in x on a high resolution display and from -15 to 319 on a low resolution display. The visibility of Vsprites in the y direction depends on their height.

## Determining Current x and y Object Positions

When used as functions instead of statements, the OBJECT.X and OBJECT.Y commands will return the current x-y coordinate of the specified object.

```
xcoordinate = OBJECT.X (object_id)  
ycoordinate = OBJECT.Y (object_id)
```

## MOVING OBJECTS

---

You may guess that object motion can be performed by continuously positioning an object at a new x-y coordinate. Although this is possible, the easier way you can move objects continuously is by using the object velocity commands:

```
OBJECT.VX object_num, x_velocity  
OBJECT.VY object_num, y_velocity
```

- x and y velocities are in pixels per second.

Once an object has been given a velocity, its motion is initiated by using the OBJECT.START command.

```
OBJECT.START [object_num[,object_num...]]
```

An explicit list will start the objects assigned a velocity. Leaving the object identifiers off will start all objects previously assigned velocities. To stop a specified object, a list of objects, or all objects, use the command:

```
OBJECT.STOP [object_num[,object_num...]]
```

Objects will be implicitly instructed to stop when they collide with a border or with other objects. If you want objects to remain in motion, periodically check for collisions, and restart them when collisions occur (collisions will be discussed further).

## Determining Current x and y Object Velocities

When used as functions, the velocity commands can be used to return the current x and y velocity components of an object.

```
x_velocity = OBJECT.VX (object_num)
y_velocity = OBJECT.VY (object_num)
```

- x and y velocities are returned in pixels per second. The net velocity =  $\sqrt{(x\_velocity)^2 + (y\_velocity)^2}$

## Accelerating Objects

Acceleration is the rate of change of velocity. You can assign an object an acceleration with the following commands.

```
OBJECT.AX object_num, x_accel
OBJECT.AY object_num, y_accel
```

- x\_accel and y\_accel are the x and y accelerations in pixels per

second per second. (The number of pixels per second an object will speed up or slow down in one second.)

Avoid high accelerations because they can cause your object to collide with a border very quickly. At such high velocities, your object can actually go right through the border, and you'll never see it again.

## Determining Object Acceleration

When used as functions, the OBJECT.AX and OBJECT.AY commands can be used to return an object's current x and y accelerations.

```
x_accel = OBJECT.AX (object_num)
y_accel = OBJECT.AY (object_num)
```

- x and y accelerations are returned in pixels per second per second.

$$\text{The net acceleration} = \sqrt{(x\_accel)^2 + (y\_accel)^2}$$

## Detecting Collisions

Whenever two objects collide, AmigaBASIC does an OBJECT.STOP on both of them. The same result occurs when an object collides with a border. To get the object moving again you must issue an OBJECT.START command (see above for syntax and usage). But how will your program know when a collision has occurred so that it can start the object again? The answer is: By using the COLLISION function.

**THE COLLISION FUNCTION.** The COLLISION function can be used to return three important pieces of information:

- COLLISION(0) returns the identifier of the object that has collided.
- COLLISION(-1) returns the window in which the collision occurred.
- COLLISION (Object\_id) returns a collision code indicating what the object collided with.

The collision codes in Table 7.1 describe the code numbers and their meanings.

When the COLLISION (object\_id) command is issued, the information on that collision is removed from the stack (memory) to make room

TABLE 7.1.

Collision Codes	Meaning
0	No collision has occurred
Any positive #	The identifier for the other object in the collision
-1	Object collided with top border
-2	Object collided with left border
-3	Object collided with bottom border
-4	Object collided with right border

for the information on the next collision. This memory location can hold information for up to 16 collisions at a time.

If you want to know the `object_id` and the collision window associated with a `COLLISION (object_id)` function, be sure to use the `COLLISION (0)` and `COLLISION (-1)` functions first.

A second (and more convenient) way to detect collisions is to use the `ON COLLISION` command.

```
ON COLLISION GOSUB sub_label
```

This command diverts control to the specified user subroutine when a collision occurs. (You must also have issued the `COLLISION ON` command to initiate event trapping). Collision trapping can be turned off by using the same command with a statement label of 0. That is,

```
ON COLLISION GOSUB 0
```

To enable the `ON COLLISION` command, the `COLLISION 0` command must be used. If five collisions have occurred before the `COLLISION ON` statement is issued, then the `ON COLLISION` command will have no effect at the time of the collision. Once `COLLISION ON` has been issued, however, program control will be directed to the specified subroutine once for each of the five collisions that had occurred plus any new collisions that occur. Event trapping can be suspended with the command

```
COLLISION STOP
```

and can be turned off entirely by using

```
COLLISION OFF
```

## Selective Collision Detection

In the default state, AmigaBASIC will record all collisions, whether with other objects or with windows. In many cases, you'll want to prevent the detection of certain collisions. You can do this with the OBJECT.HIT command.

```
OBJECT.HIT object_id [,MeMask][,HitMask]
```

- `object_id` is the object identifier.
- `MeMask` is an integer value whose bit pattern defines the collision type of the object.
- `HitMask` is an integer value whose bit pattern defines the collision type of the object with which this one will collide.

This all sounds very confusing, but some examples will clarify the usage. Here are the questions you should ask yourself when determining an object's collision characteristics:

1. Does the HitMask value end with a 1 (is it even)? If yes, then the object will collide with borders.

For example:

HitMask		Collides with borders?
(decimal)	(binary)	
1	0001	Y
2	0100	N
15	1111	Y

2. When I compare each bit of an object's MeMask with its corresponding bit (by position) of another object's HitMask, do any of the positions both contain the value one? If so, then the objects will collide with each other. (This operation is the equivalent of a logical AND.)

For example:

Object id	MeMask	HitMask	Will Collide With
1	(2) 0010	(1) 0001	Borders only
2	(8) 1000	(5) 0110	obj3, obj4
3	(12) 1100	(9) 1001	obj2, obj4, borders
4	(10) 1010	(10) 1010	obj2, obj3

Object 2 will collide with Objects 3 and 4 because the fourth digit of Object 2's MeMask and the fourth digit of Object 3 and 4's HitMask contain ones.

Object 1 collides with borders only because none of the other objects have a one in their first position.

It is important to specify the collisions consistently. For example, if it is possible for Object 2 to collide with Object 3, then you should also make it possible for Object 3 to collide with Object 2. To make sure, compare the MeMasks against the HitMasks in both directions (obj2 to obj3 and obj3 to obj2).

## Sample Program BOUNCE.BAS

Program BOUNCE.BAS integrates the concepts discussed above. It requires that you create an object (either a bob or a sprite) using ObjEdit and save the object to a file. The sample file used for this program was in the basicdemos directory and was called "ball.sprite." Specify your directory and file name accordingly. The program assumes you have created a multicolored object and makes multiple copies of the object.

```

REM
REM   BOUNCE.BAS
REM
DEFINT a-z
DIM ball$(4)
WINDOW 2,"BOUNCE.BAS", (0,0)-(400,186),4
OPEN "basicdemos/ball.sprite" FOR INPUT AS 1
ball$(1) = INPUT$(LOF(1),1) 'Read the object data into a string
CLOSE 1

REM
FOR obnum = 2 TO 4
  ball$(obnum) = ball$(1) 'Copy data for three more objects
  READ red(0),green(0),blue(0)
  READ red(1),green(1),blue(1) 'Give objects 2,3, and 4 new colors
  READ red(2),green(2),blue(2)
  GOSUB ColorChange
NEXT obnum

REM
      'col 1   col 2   col 3
DATA 15,0,0,  0,15,0, 0,0,15  :'object 2
DATA 0,15,0,  0,0,15, 15,0,0  :'object 3
DATA 0,0,15,  15,0,0, 0,15,0  :'object 4

```



```
REM
  FOR obnum = 1 TO 4
    OBJECT.SHAPE obnum,ball$(obnum) 'Build objects from their data
  NEXT obnum
REM
REM   Initialize positions and velocities
REM
  OBJECT.X 1,300: OBJECT.Y 1,20 : OBJECT.VX 1,100: OBJECT.VY 1,1
  OBJECT.X 2,200: OBJECT.Y 2,100: OBJECT.VX 2,130: OBJECT.VY 2,2
  OBJECT.X 3,100: OBJECT.Y 3,50 : OBJECT.VX 3,160: OBJECT.VY 3,2
  OBJECT.X 4,0  : OBJECT.Y 4,10 : OBJECT.VX 4,190: OBJECT.VY 4,4
REM
  FOR obnum = 1 TO 4
    OBJECT.AY obnum,obnum      'Initialize accelerations
    OBJECT.HIT obnum,2,1      'Only detect collisions with borders
  NEXT obnum
REM
  OBJECT.CLIP (0,0)-(384,200) 'Define the borders
  OBJECT.ON          'Display all objects
  OBJECT.START      'Initiate object motion (all objects)
REM
  ON COLLISION GOSUB collide 'Define action on collision
  COLLISION ON      'Turn on collision detection
REM
  PRINT "Press left mouse button to end"
  Pause: IF NOT MOUSE(0) THEN Pause 'Waste time while objects move
  WINDOW CLOSE 2
  END

collide:
  obnum = COLLISION(0)      'Object involved in collision
  IF obnum = 0 THEN obstart 'Restart if no more collisions
  wall = COLLISION(obnum)  'The wall the object collided with
  vy = OBJECT.VY(obnum)    'The y velocity at collision time
  vx = OBJECT.VX(obnum)    'The x velocity at collision time
REM
  IF (wall = -1 AND vy < 0) OR (wall = -3 AND vy > 0) THEN
    OBJECT.VY obnum,-vy 'Reverse the y velocity
  ELSEIF (wall = -2 AND vx < 0) OR (wall = -4 AND vx > 0) THEN
    OBJECT.VX obnum,-vx 'Reverse the x velocity
  END IF
  GOTO collide
```

```
obstart:
  OBJECT.START 'Restart all objects
RETURN

ColorChange:
  length = LEN(ball$(obnum)) 'return the length of the string
  FOR i = 0 TO 2
    greenblue(i) = 16*green(i) + blue(i) 'calculate greenblue values
  NEXT i

REM
REM   Change the object data to define new object colors
REM

  col$ = CHR$(red(0)) + CHR$(greenblue(0))
  col$ = col$ + CHR$(red(1)) + CHR$(greenblue(1))
  col$ = col$ + CHR$(red(2)) + CHR$(greenblue(2))
  MID$(ball$(obnum),length - 5) = col$
RETURN
```

# Sound Programming, Software, and Hardware

No other personal computer has the power and flexibility of the Amiga for producing sound and music. Would you like your computer to talk to you? Great. The Amiga is capable of speech synthesis, as discussed in Chapter 9. Maybe you want to write musical scores, play complex digitized sounds, or attach a dedicated synthesizer? If so, you'll be interested in the software discussed on pages 194–197. Maybe you'll want to add sounds and musical notes to your AmigaBASIC programs, games, and animations. If so, read on.

---

### DIGITAL SOUND

The Amiga produces *digital* sound. Traditional tape recorders use *analog* sound. They record the sound as a continuously varying magnetic field that mimics the pressure variations of the original sound waves. Replaying a taped recording uses the reverse process. The tape player converts the magnetic variations stored on tape into an electric current

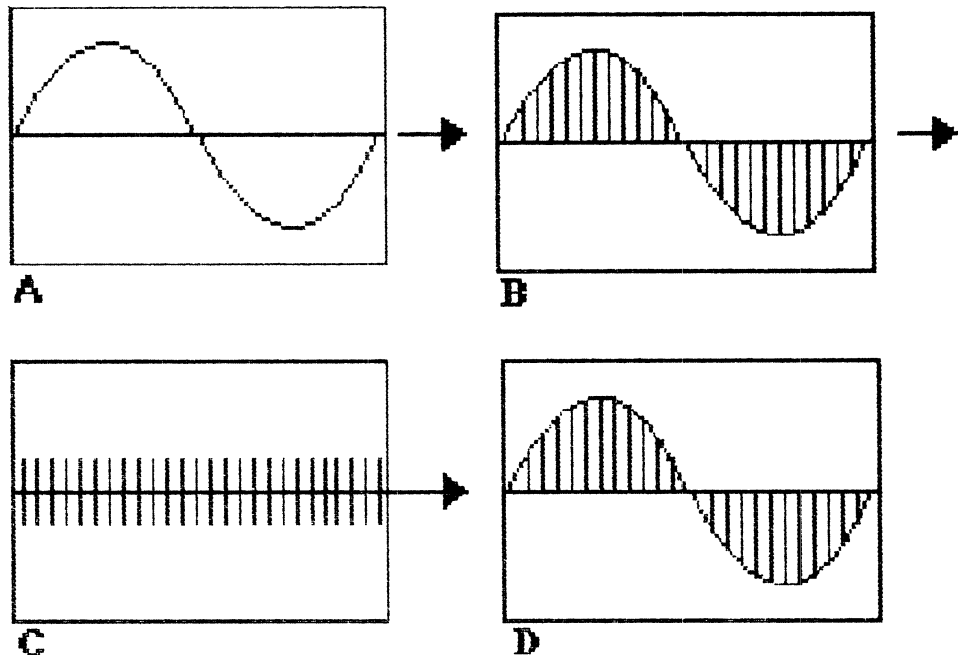
sent to the speakers. The speakers convert the current into audible pressure waves.

In digital sound creation, all sounds are recorded as a series of numeric values. The amplitude (height) of a sound wave is measured, called *sampling*, at constant intervals (usually thousands of times a second), and translated into numeric values. A digital sound player, such as the Amiga's sound chip, reads these numbers and reconstructs them as analog sound waves to output through a speaker. With the Amiga you can sample real-world sounds using a *digitizer*, or you can create these tables of numbers yourself using AmigaBASIC.

## Digitizing

When you record a lion's roar on a tape recorder, you can speed it up or play it in reverse. Basically, however, you have one roar. When you record a sampled sound on the Amiga, it is stored as a set of numbers. By

**FIG. 8.1** In sound digitizing, the (A) analog wave is (B) rapidly sampled, (C) converted into a stream of numbers by the Amiga, and (D) then converted back into analog sound waves when the Amiga plays the music back.



altering the numbers you can change the pitch, timbre, speed, modulation, direction, and length of the sound. Make the roar faster, slower, or higher in pitch. A digital sample of a sound is more than a recording. It is a tool you can use to create new and unusual sounds on the Amiga.

You can purchase libraries of sounds on disks from software companies and in computer stores. Some of available digitized sounds you can get are dripping water, screeches, booms, opera singers, car engines, and space sound effects for use in games or educational programs. These sounds can be modified however you wish. Or you can make samples yourself using the Amiga and some special equipment previously available only in recording studios. You'll need a digitizer, which is an *analog-to-digital convertor*, a microphone or tape recorder, and software to control the sampling.

The microphone inputs the sound into the digitizer, which rapidly samples the stored magnetic analog wave of sounds and converts it into digital data (a stream of numbers) that the Amiga sound chip can read and store as shown in Figure 8.1.

Several digitizers are being marketed for the Amiga with software that lets you control the rate of sampling and the subsequent manipulation of sounds. You can also use the sampling data in your AmigaBASIC programs. See the section on "Sound Software and Hardware" later in this chapter for more information.

## PROGRAMMING SOUND WITH AMIGABASIC

---

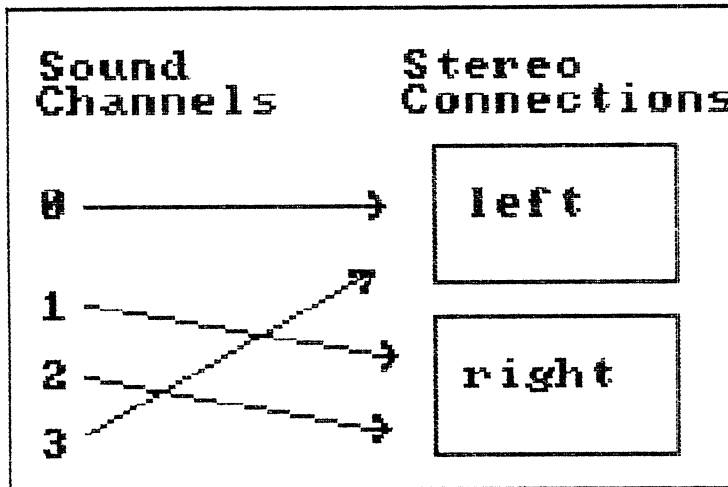
Creating your own sounds and musical notes using AmigaBASIC involves the use of two central sound commands:

- SOUND (sets the frequency, duration, volume, and channel)
- WAVE (assigns a waveform to any channel)

Before you examine these commands, note that the Amiga has four sound channels attached to two stereo outputs. Each channel can play a table of numbers independently of the others. This means that you can program a song in four-part harmony. Just as easily, two channels can play music, one can play an announcer's voice, and the other can play a barking dog—all simultaneously.

When you program, keep in mind that channels 0 and 3 are attached

FIG. 8.2 The Amiga's four sound channels and stereo combinations.



to the left stereo output and channels 1 and 2 are attached to the right stereo output. You'll want to designate which sounds are sent through which output, depending on whether you have one or two speakers connected to your Amiga. The program CHANNEL.BAS shows how to designate various channel combinations. If you're writing a program for distribution, you might want to send all sounds through the left speaker for users who don't have external speakers attached. Figure 8.2 shows the stereo connections.

```

REM
REM CHANNEL.BAS
REM
REM This program uses multiple channels to play
REM a four note chord. It starts by using only
REM channel 0, then channels 0 and 1, then channels
REM 0,1, and 2, and finally uses channels 0,1,2 and 3.
REM Unless your sound output is connected to external
REM speakers, all channels will be played on the speaker
REM in your monitor.
REM
    PRINT "Channel 0 only"
    SOUND 261.63,20,255,0 ' C note
    PRINT "Click left mouse button to continue"

```

```
pause1: IF NOT MOUSE(0) THEN pause1
        CLS
        PRINT "Channels 0 and 1"
        SOUND 261.63,20,255,0 ' C note
        SOUND 329.63,20,255,1 ' E note
        PRINT "Click left mouse button to continue"
pause2: IF NOT MOUSE(0) THEN pause2
        CLS
        PRINT "Channels 0, 1, and 2"
        SOUND 261.63,20,255,0 ' C note
        SOUND 329.63,20,255,1 ' E note
        SOUND 392!,20,255,2    ' G note
        PRINT "Click left mouse button to continue"
pause3: IF NOT MOUSE(0) THEN Pause3
        CLS
        PRINT "Channels 0, 1, 2, and 3"
        SOUND 261.63,20,255,0 ' C note
        SOUND 329.63,20,255,1 ' E note
        SOUND 392!,20,255,2    ' G note
        SOUND 523.25,20,255,3  ' C note (middle C)
```

## The Sound Command

*Frequency* defines the pitch of a sound: the number of waves that repeat in a time period. The Amiga can reproduce frequencies in the range of 20 to 15,000 Hz (cycles per second). The AmigaBASIC manual shows the frequency for the octaves on the diatonic scale.

Suppose you have a program setting the note for middle C (value 523). If you tell the sound chip to scan the data twice as fast, it will play C one octave higher on the musical scale. Thus, by changing the frequency at which the table of numbers is scanned, you change the pitch of a sound. Program SCALE.BAS plays each note of the Equal Tempered Scale for one second.

```
REM
REM   SCALE.BAS
REM
REM   This program plays each of the notes of the Equal
REM   Tempered Scale for a duration of one second.
```

```
REM
  FOR note = 1 TO 19
    READ frequency#
    SOUND frequency#,18.2
  NEXT note
REM
DATA 27.500 : 'Note A, Octave 0
DATA 55.000 : 'Note A, Octave 1
DATA 110.000 : 'Note A, Octave 2
DATA 220.000 : 'Note A, Octave 3
DATA 233.068 : 'Note A#, Octave 3
DATA 246.928 : 'Note B, Octave 3
DATA 261.624 : 'Note C, Octave 4
DATA 277.200 : 'Note C#, Octave 4
DATA 293.656 : 'Note D, Octave 4
DATA 311.124 : 'Note D#, Octave 4
DATA 329.648 : 'Note E, Octave 4
DATA 349.228 : 'Note F, Octave 4
DATA 370.040 : 'Note F#, Octave 4
DATA 392.040 : 'Note G, Octave 4
DATA 415.316 : 'Note G#, Octave 4
DATA 440.000 : 'Note A, Octave 4
DATA 880.000 : 'Note A, Octave 5
DATA 1760.000 : 'Note A, Octave 6
DATA 3520.000 : 'Note A, Octave 7
```

For musical notes, the frequency value is usually between 130 and 2,000. The larger the number, the higher the pitch of the sound or note. Other sound effects such as beeps and sirens, can be on the edges of the 20 to 15,000 range.

*Duration* is a number between 0 and 77, which defines how long the specified frequency will be played. The larger the number, the longer the note plays. The value 77 is equal to about 4 seconds; 18.2 equals about 1 second. To play a sound longer than 4 seconds requires executing successive SOUND commands.

*Volume* is a number between 0 (silent) and 255 (loudest). If you don't include this parameter, a volume near the middle of the range will automatically be chosen as a default value. In programming sound it is important to adjust the volume parameter so that sounds fade in and out to make interesting listening. A song played all at one volume is not very dynamic. So, make sure to vary the volume of your sound effects.



Program VOLUME.BAS changes the volume of a single note.

```
REM
REM   VOLUME.BAS
REM
REM   This program changes the volume of a single note.
REM
REM   freq = 370.04   'note is f#
REM   duration = 18.2 'one second
REM   FOR volume = 0 TO 255 STEP 25.5   'max volume is 255
REM     SOUND freq,duration,volume :
REM   NEXT volume
```

*Channel* defines which of the four sound channels will play the sound. The values are the same as the channel connections shown in Figure 8.2. A way to make sounds that are aurally interesting is to send different notes, or one note at various pitches, simultaneously through different channels. Your ear will blend them all into one rich sound. If you don't define a channel, the default is 0, the left speaker.

## The WAVE Command

*Channel number* sets the channel that will play the sound. Play it on all four channels, or just on one, two, or three. You can create four different sounds and play them all at once by assigning each to a different channel.

*Wavearray* is the name of the array that holds values describing the shape of the wave to be played. You define this waveform array prior to the WAVE command. If you do not specify a custom wave array, the default is a simple sine wave as shown in Figure 8.3. Or, you can assign

FIG. 8.3 A simple sine wave used in the SIN command.

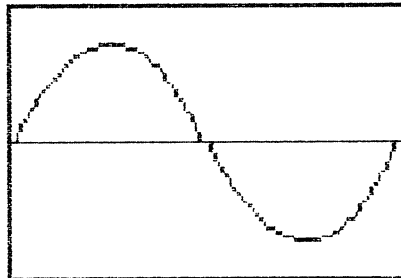


FIG. 8.4 A simple sine wave.

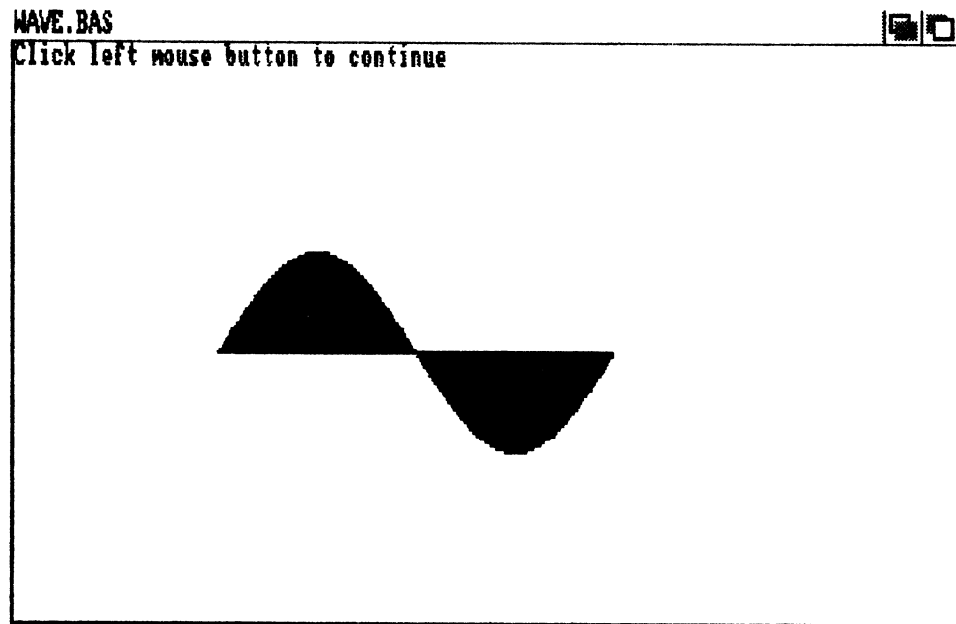


FIG. 8.5 Sine waves with harmonics.

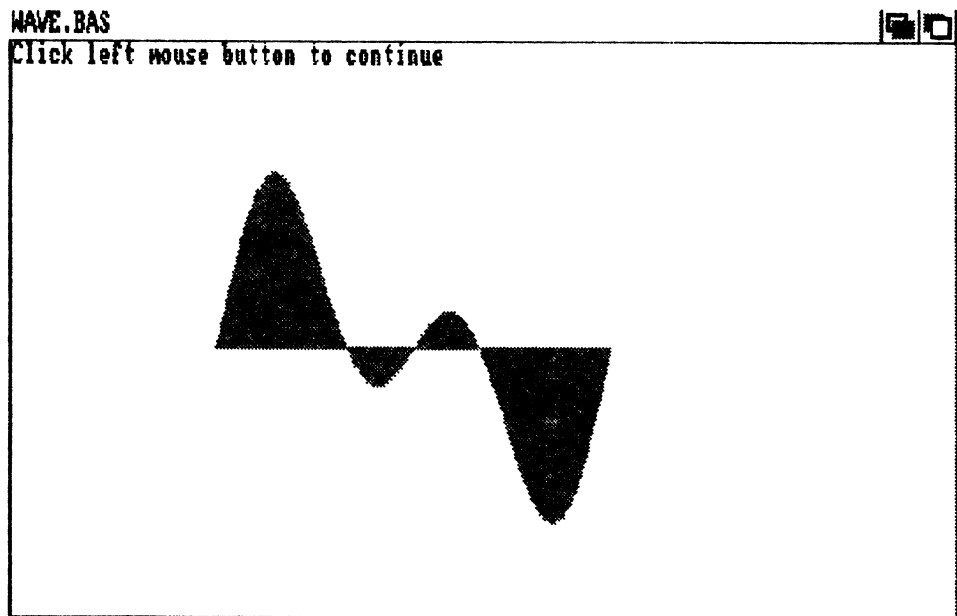
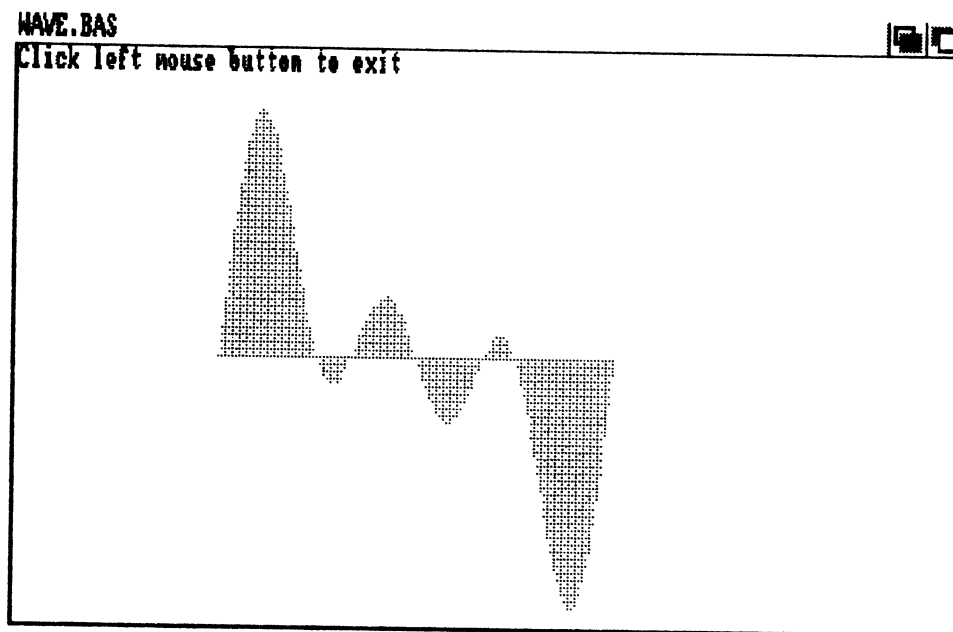


FIG. 8.6 Complex sine waves with harmonics.



a sine wave using *SIN*:

```
WAVE, channel number, SIN.
```

Defining your custom wave requires an array of at least 256 elements; it can contain more but not less. The array must be an integer array, indicated by %. A larger value will give you a better sound.

Program *WAVE.BAS* uses the *Wave* command to illustrate the construction and execution of simple and complex sine waves composed of various harmonics (Figures 8.4 to 8.6).

```
REM
REM  WAVE.BAS
REM
REM  This program uses the WAVE command to illustrate the
REM  construction and execution of simple and complex sine
REM  waves composed of various harmonics.
REM
DIM wav%(255)
WINDOW 2, "WAVE.BAS", (3,11)-(613,185),4
```

```

COLOR 2,1 'white foreground color, black background
CLS 'clear screen to see new background color
twopi# = 2 * 3.1415927# 'initialize constant
inc# = twopi#/256# 'calculate angular increment
freq = 261.624: duration = 77:
volume = 255: channel = 0

```

```

GOSUB first
COLOR 0
PRINT "Click left mouse button to exit"
pause1: IF NOT MOUSE(0) THEN pause1
COLOR 2,1: CLS

```

```

GOSUB second
COLOR 0
PRINT "Click left mouse button to exit"
pause2: IF NOT MOUSE(0) THEN pause2
COLOR 2,1: CLS

```

```

GOSUB third
COLOR 0
PRINT "Click left mouse button to exit"
pause3: IF NOT MOUSE(0) THEN pause3
WINDOW CLOSE 2
END

```

'Create a simple sine wave using the first harmonic frequency

```

first:
FOR i = 0 to 255
  wav%(i) = 30 * SIN(i*1*inc#)
  x = i + 132: y = 93 - wav%(i)
  LINE (x,93)-(x,y),0
NEXT i
WAVE 0, wav%
SOUND freq,duration,volume,channel
RETURN

```

'Create a complex sine wave composed of first and second harmonics

```

second:
FOR i = 0 TO 255
  wav%(i) = 30 * (SIN(i*1*inc#) + SIN(i*2*inc#))
  x = i + 132: y = 93 - wav%(i)
  LINE (x,93)-(x,y),2
NEXT i
WAVE 0,wav%
SOUND freq,duration,volume,channel
RETURN

```

'Use the first, second, and third harmonics

```

third:
FOR i = 0 TO 255
  wav%(i) = 30 * (SIN(i*1*inc#) + SIN(i*2*inc#) + SIN(i*3*inc#))
  x = i + 132: y = 93 - wav%(i)
  LINE (x,93)-(x,y),3
NEXT i
WAVE 0,wav%
SOUND freq,duration,volume,channel
RETURN

```

Then, program MOREWAVES.BAS shows you the use of rectangular (Figure 8.7), triangular (Figure 8.8), sawtooth (Figure 8.9), and random waves.

```

2REM
REM   MOREWAVES.BAS
REM
REM   This program illustrates the use of rectangular
REM   triangular, sawtooth, and random waves.
REM
DIM wav%(255)
WINDOW 2,"MOREWAVES.BAS",(3,11)-(613,185),4
COLOR 2,1 'white foreground color, black background
CLS 'clear screen to see new background color
twopi# = 2 * 3.1415927# 'initialize constant
inc# = twopi#/256# 'calculate angular increment

GOSUB rectangular
COLOR 0
PRINT "Click left mouse button to continue"

```

FIG. 8.7 Rectangular waves.

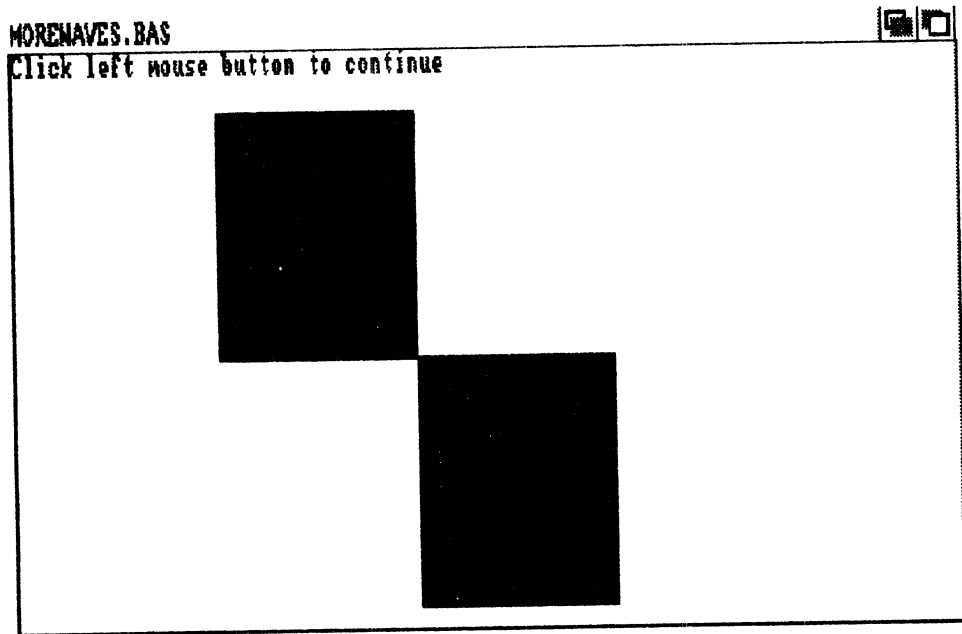


FIG. 8.8 Triangular waves.

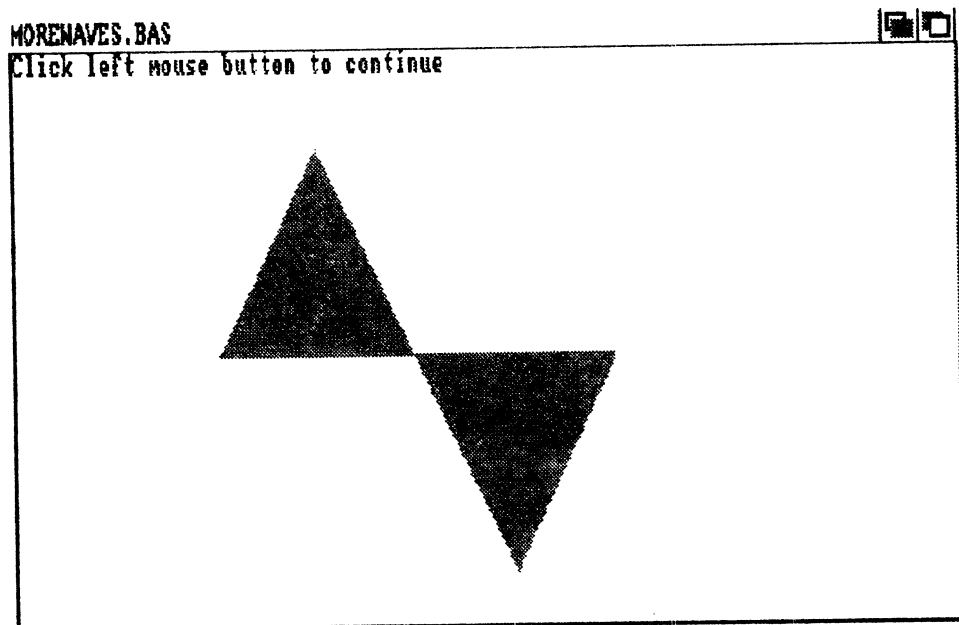
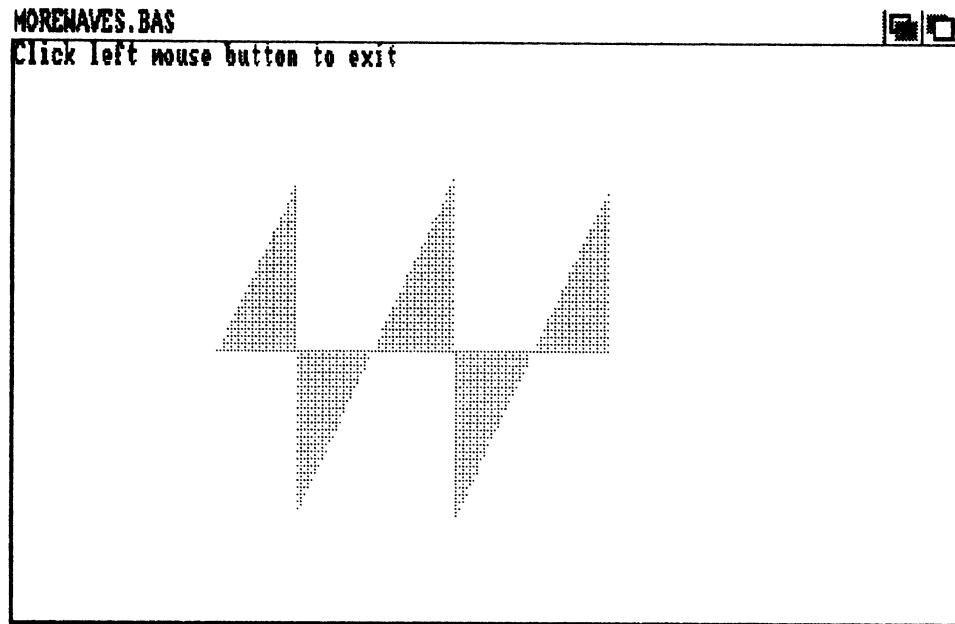


FIG. 8.9 Sawtooth waves.



```
pause1: IF NOT MOUSE(0) THEN pause1
COLOR 2,1: CLS

GOSUB triangular
COLOR 0
PRINT "Click left mouse button to continue"
pause2: IF NOT MOUSE(0) THEN pause2
COLOR 2,1: CLS

GOSUB sawtooth
COLOR 0
PRINT "Click left mouse button to exit"
pause3: IF NOT MOUSE(0) THEN pause3
WINDOW CLOSE 2
END

makesound:
freq = 261.624: duration = 77:
volume = 255: channel = 0
```

```

WAVE 0,wav%
SOUND freq,duration,volume,channel
RETURN

```

'Create a simple rectangular waveform

```

rectangular:
FOR i = 0 TO 255
  wav%(i) = 75
  IF i > 127 THEN wav%(i) = -wav%(i)
  x = i + 132: y = 93 - wav%(i): LINE (x,93)-(x,y),0
NEXT i
GOSUB makesound
RETURN

```

'Create a simple triangular waveform

```

triangular:
j = 63
FOR i = 0 to 255
  IF (i <= 63) THEN wav%(i) = i
  IF (i > 63 AND i < 192) THEN
    j = j - 1: wav%(i) = j
  END IF
  IF (i > 192) THEN
    j = j + 1: wav%(i) = j
  END IF
  x = i + 132: y = 93 - wav%(i): LINE (x,93)-(x,y),2
NEXT i
GOSUB makesound
RETURN

```

'Create a sawtooth waveform

```

sawtooth:
j = -1
FOR i = 0 TO 254
  IF (i = 51 OR i = 153) THEN j = -j
  j = j + 1: wav%(i) = j
  x = i + 132: y = 93 - wav%(i): LINE (x,93)-(x,y),3

```



```
NEXT i
GOSUB makesound
RETURN
```

## SOUND SOFTWARE AND HARDWARE

---

The Amiga's sound capabilities have inspired many developers to design software and hardware products to let you create and play sound effects and music on your Amiga. These products are suitable for serious musicians and for just having fun.

### Hardware

There are two hardware pieces that, in combination with the Amiga, are the basis for a professional-style recording studio.

**MIDI.** Musician Instrument Digital Interface (MIDI) is an international standard that makes all electronic musical instruments compatible through a computer. For the Amiga, the MIDI interface is a small box that plugs into the serial port and comes with three cables, MIDI IN, OUT, and THRU. These cables are attached to a synthesizer. Other synthesizers can be daisy-chained to the first. The cables carry the machine language instructions between the devices, and the computer controls the exchange of information. This computer-controlled MIDI environment is widely used by professional musicians.

Commodore is marketing its MIDI interface through third-party developers. With MIDI, synthesizers available at various prices, your Amiga, and your stereo, you can set up a powerful music production studio very inexpensively.

**SOUND DIGITIZERS.** As part of your music studio or on its own, a sound digitizer is a tremendous tool that lets you sample real-world sounds and, using software, save and manipulate those sounds.

**SOUND SAMPLERS.** As described earlier in this chapter, you can program tables of numbers into the Amiga and it will convert the numbers (digital information) into sounds (analog information). Or, you can plug an analog-to-digital convertor into the Amiga, plug a tape

recorder or microphone into the digitizer, and input any sounds you choose. The sounds are converted from analog waves to digital tables. The Amiga records the numbers and can then convert them back into sounds at any time. You can alter the numbers to alter the sounds to create interesting effects or different tunes. Sampled sounds take up a lot of memory, but you could include them in your programs in limited amounts.

## Software

You have a wide choice of music software. Some products require nothing more than your Amiga and your imagination. Others include or benefit from hardware such as MIDI and a sound digitizer. In all cases, you get the full enjoyment of the Amiga's capabilities if you add stereo speakers.

*The Music Studio* from Activision is a music composition program. It lets you compose music, print it out, and add lyrics. If you know music or want to learn, you can use standard musical notation. For non-musicians, there are patterns that represent notes.

You can use the mouse or the keyboard to input your notes and listen to your composition as you create it. The compositions can be played through the Amiga alone or with MIDI through external devices.

*The Music Studio* comes with built in instruments that you can modify, or you can create new instruments and sounds. There are several tunes on disk that you can play and modify.

*Instant Music* from Electronic Arts is a fun program that allows you to jam with the Amiga. The Amiga plays three instruments and you, using the mouse, play the fourth. You can choose which instrument you want to play, and just play anything you want. The screen displays all four parts so you can see what your instrument is playing. The lively screen display does not use standard musical notation, but you can still save your compositions. (You are able to print them with *Deluxe Music Construction Set*.) Create your own scores or use those included.

*Musicraft* from Commodore-Amiga is a composition program. It will play Amiga-generated sounds or sampled sounds and music. You use the mouse and the note editor to write music, which can be saved and played back. There is a synthesizer feature which lets you alter sounds and see the waveforms changing on the screen. You can also use the Amiga keyboard as a musical keyboard. *Musicraft* comes with some musical scores on disk and can use other compatible scores.

*SoundScape* from Mimetics is a professional, MIDI-based studio system for the Amiga. It includes the operating system, a sequencer, and a musical editor. You can use any sounds in IFF format and or sounds from external devices (synthesizers, sound digitizers, etc.).

*SoundScape* is system of modules. You can use one or many. There are tutorials available for learning on your own and there are personalized teaching tools, so teachers can customize a teaching program in their own style. You could plug in a piano-style keyboard or use the Amiga keyboard with *SoundScape*. And students can practice playing instruments by playing with the Amiga using *SoundScape's* match mode at any tempo. You can write and edit music using the keyboard or mouse for input (use real-time sequencing).

Mimetics also markets a digital sampler that can be used in the *SoundScape* system or alone. It comes with software for recording and playing back.

*Deluxe Music Construction Set* from Electronic Arts is a professional music composition program. You can create scores using musical notation or patterned symbols, the music can be printed, and lyrics added. Music and sounds can be played through the Amiga or by using MIDI-operated devices. *Deluxe Music* will print out the music you create in *Instant Music*.

*Pitchrider* from IVL Technologies is a musical education tool. It synthesizes notes you (the student) play into a microphone. The notes are shown on the screen so you can see if you're playing the correct notes or not. Correct yourself and learn at your own pace.

*FutureSound* from Applied Visions is a digital sampler. It plugs into the parallel port but includes a printer connector so you can have both working. You can sample and playback at varying speeds. *FutureSound* includes a microphone, microphone jack, cables, and software.



# AmigaBASIC Speech

The Amiga's speaking ability can add a new dimension to the programs you write. You can have your Amiga give you verbal prompts in interactive programs you write, read you a letter from Mom, give verbal descriptions of scenes and situations in game programs, or verbally present educational information while it illustrates the discussion with remarkable graphics, too.

Though you can program speech on the Amiga using machine language, AmigaBASIC gives you a much easier way using convenient commands.

## **THE SAY COMMAND AND THE TRANSLATE\$ FUNCTION**

---

The SAY command instructs the Amiga to use its speech capabilities on the string of phonemes returned by the TRANSLATE\$ function. *Phonemes* are the smallest units of speech that distinguish one utterance or word from another. Every word in a language is composed of a series of sounds, and each discrete sound within a word is a phoneme.

Enter this simple one-line AmigaBASIC program and run it:

```
SAY TRANSLATE$ ("Hi there. I can talk.")
```

The TRANSLATE\$ function translates the contents of the string you type within the parentheses. In addition to explicit strings, TRANSLATE\$ operates on string variables as illustrated by the program below:

```
char$ = "Hi there. I can talk."  
SAY TRANSLATE$(char$)
```

To see the string of phonemes returned by the TRANSLATE\$ function alongside its English text equivalent, modify your program to look like this:

```
char$ = "Hi there. I can talk."  
PRINT char$  
PRINT TRANSLATE$(char$)  
SAY TRANSLATE$(char$)
```

The third line returns these characters:

```
/HAY4 DHEH1R. AY4 KAEN TA03K.
```

Although this string looks peculiar, it's the form required by the narrator program executed by the SAY command. In fact, you can directly enter the string above to get the same result.

```
SAY "HAY4 DHEH1R. AY4 KAEN TA03K."
```

You can study Appendix H in the Amiga User Guides for an excellent discussion on how to write phonetically for the SAY command. But in most cases, it's much easier to use the TRANSLATE\$ function rather than learning the phonemes yourself.

## VOICE ARRAY SETTINGS

---

The *voice array* is an optional integer array that you can pass to the SAY command. You can set each of the nine array elements to control voice characteristics like pitch, inflection, rate, gender, sampling frequency, volume, channel assignment, synchronization mode, and synchronization control. Table 9.1 lists the array elements, what they

TABLE 9.1. VOICE ARRAY ELEMENTS

Element #	Controls	Valid Values
0	Pitch	65-320
1	Inflection	0-1
2	Rate	40-400
3	Gender	0-1
4	Sampling Frequency	5000-28000
5	Volume	0-64
6	Channel Assignment	0-11
7	Synchronization mode	0-1
8	Synchronization control	0-2

control, and their valid values. See the SAY command write-up in Chapter 8 of the *Amiga User Guides* to learn about each of the elements in detail.

The program TALKER.BAS gives you the opportunity to experiment with speech. Your Amiga will repeat any words you type at the keyboard or speak the contents of any ASCII disk file you specify. The file-speaking option lets you listen to messages your friends send you, and the keyboard option lets you quickly change voice parameters and try them on typed strings.

```

REM
REM   TALKER.BAS
REM
REM   This program will read a file to you or repeat the
REM   characters you type in at the keyboard.
REM
DIM voice%(8),min%(8),max%(8)
GOSUB initialize

prompt1:
PRINT "Enter F to read a file or K to read the keyboard"
SAY TRANSLATE$("Enter F to read a file or K to read the keyboard")
INPUT answer$
IF answer$ = "F" OR answer$ = "f" GOTO readfile
IF answer$ = "K" OR answer$ = "k" GOTO readkeyboard
PRINT "Not a valid input parameter"
GOTO prompt1:

```

readfile:

```
PRINT "Do you want to set voice characteristics? (y/n)"
SAY TRANSLATE$("Do you want to set voice characteristics")
INPUT answer$
IF answer$ = "y" OR answer$ = "Y" THEN
    GOSUB setvoice
END IF
PRINT "Enter a file name"
SAY TRANSLATE$("Enter a file name")
INPUT filename$
CLS
OPEN "I",#1,filename$
WHILE NOT EOF(1)
    INPUT #1, chars$
    PRINT chars$
    SAY TRANSLATE$(chars$),voice%
WEND
CLOSE #1
GOTO endprogram
```

readkeyboard:

```
CLS
PRINT "Do you want to set voice characteristics? (y/n)"
INPUT answer$
IF answer$ = "y" OR answer$ = "Y" THEN
    GOSUB setvoice
END IF
PRINT
PRINT
PRINT "Enter a string, Press RETURN to repeat the last"
PRINT "string entered, or type EXIT to end the program"
INPUT tempchars$
IF tempchars$ = "exit" OR tempchars$ = "EXIT" GOTO endprogram
IF tempchars$ = "" GOTO saystring
chars$ = tempchars$
saystring:
PRINT
PRINT chars$
PRINT
SAY TRANSLATE$(chars$),voice%
GOTO readkeyboard:
```



```
setvoice:
  GOSUB display
  PRINT "Enter the element number you wish to modify"
  INPUT number
  IF number < 0 OR number > 8 GOTO setvoice
  newvalue:
  PRINT "Enter the new value"
  INPUT value%
  IF value% < min%(number) OR value% > max%(number) THEN
    PRINT "Invalid value, try again"
    GOTO newvalue
  END IF
  voice%(number) = value%
  PRINT "Do you want to change more voice elements? (Y/N)"
  INPUT answer$
  IF answer$ = "y" OR answer$ = "Y" THEN
    CLS
    GOTO setvoice
  END IF
  GOSUB display
  RETURN
```

```
display:
  PRINT "0. Pitch (65-320): ",voice%(0)
  PRINT "1. Inflection (0-1): ",voice%(1)
  PRINT "2. Rate (40-400): ",voice%(2)
  PRINT "3. Gender (0-1): ",voice%(3)
  PRINT "4. Sampling Frequency (5000-28000): ",voice%(4)
  PRINT "5. Volume (0-64): ",voice%(5)
  PRINT "6. Channel Assignment (0-11): ",voice%(6)
  PRINT "7. Synchronization mode (0-1): ",voice%(7)
  PRINT "8. Synchronization control (0-2): ",voice%(8)
  RETURN
```

```
initialize:
  voice%(0) = 110
  voice%(1) = 0
  voice%(2) = 150
  voice%(3) = 0
  voice%(4) = 22200
```

```
voice%(5) = 64
voice%(6) = 10
voice%(7) = 0
voice%(8) = 0
min%(0) = 65:   max%(0) = 320
min%(1) = 0:   max%(1) = 1
min%(2) = 40:  max%(2) = 400
min%(3) = 0:   max%(3) = 1
min%(4) = 5000: max%(4) = 28000
min%(5) = 0:   max%(5) = 64
min%(6) = 0:   max%(6) = 11
min%(7) = 0:   max%(7) = 1
min%(8) = 0:   max%(8) = 2
RETURN

endprogram:
END
```

# Video Applications

It has been said that the Amiga represents the dawn of “desktop video”: video productions at affordable prices for professional or home users. You can use this capability by combining the sound and graphics created on your Amiga with a VCR, genlock device, digitizer, or all three together to design handsome video productions for sales presentations, training videos, educational products, music videos, or personal videos enhanced with sounds, titles, and animation. This section introduces the products and techniques you’ll need to know about to create desktop videos for home or business use.

---

### USING A VCR

You can use your Amiga to generate special effects for “videos.” The videos could be Amiga-generated graphics only or Amiga graphics overlaid onto preexisting videotape. At its simplest, creating a video requires connecting the video jack on the Amiga back to your VCR and taping the Amiga-generated graphics or animation onto the video tape for a portable video you can play on any VCR. When you’re taping to a VCR, if you have an analog monitor keep using analog output. For a composite monitor, connect it to the monitor output jack on the VCR.

For tips on VCR editing and creating home videos, the “Deluxe Video Users Guide” by Electronic Arts is very useful (see “Vendor Reference”). It

gives information on what hardware to buy, how to do single- or two-deck editing, and how to use genlock devices and digitizers in videos. It also suggests strategies such as “black striping” video tape to help hide edit flaws. When you add Amiga-generated animations or graphics onto video tape, you can use fadeouts to black to close the first “splice” or edit, and fadeins to black where the next edit begins. (You used fades to and from black in the animation and Impact tutorials.) If you then black stripe by simply prerecording nothing on the entire tape, and use black fadeins and fadeouts, editing is easy because edit imperfections are almost unnoticeable.

## VIDEO DIGITIZERS

---

A *video digitizer* (or *frame grabber*) is a way to put the whole world into your Amiga: just point a video camera at an object and in seconds it appears on your screen. This capability at low cost is one of the most exciting features of owning an Amiga.

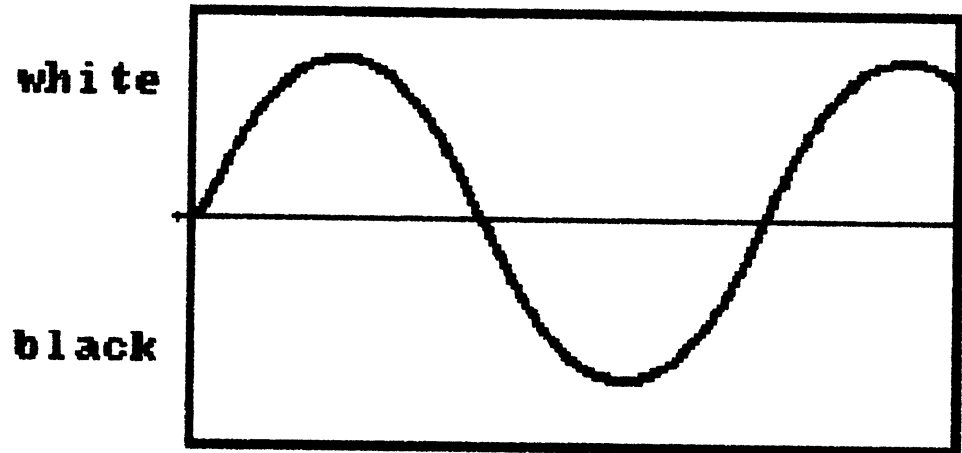
The object could be a live person, a street scene, or an image from a book or photograph. Once you have the digitized image on the screen you can put it into a painting program, enhance it with color, add objects by painting them in, include text, or resize the image. The uses for such a system might include digitized artwork, home or business video productions with titles, advertisements mixing video images and computer-created objects, and scientific applications.

### The Technology

*Video digitizing* is the conversion of images created with analog video signals by a video camera (or other video hardware such as a laser-disk player, VCR, or computer), into digital signals the Amiga can utilize.

Video cameras produce analog signals that are waves. The digitizer samples these waves and uses numerical data to describe their high and low points (see Figure 10.1). Sampling only the highs and lows of a video signal produces a black-and-white image on the screen. Sampling at faster speeds and at more levels between the highs and lows (black and white) will give you levels of gray (Figure 10.2). To get a color image, simply assign a color to each level being sampled. Then, add software

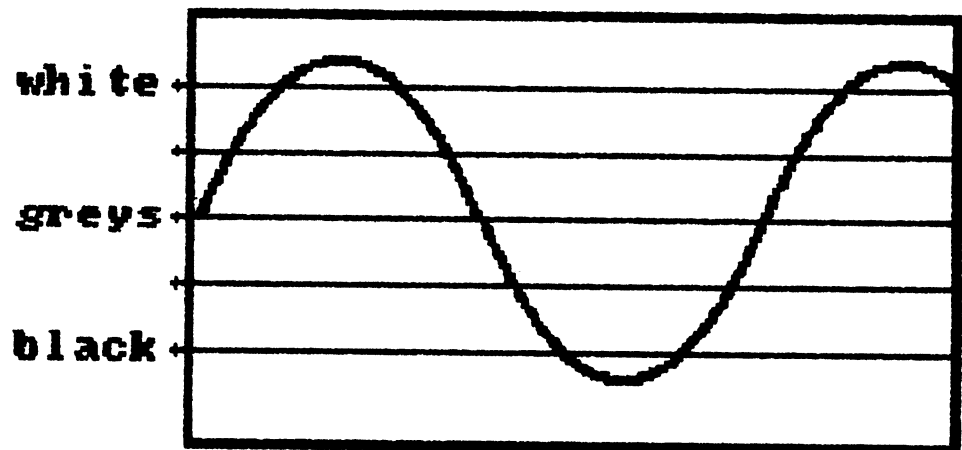
**FIG. 10.1** Digitizing black and white images by sampling to determine whether the wave is above (white) or below (black) the sampling line.



that controls the relationships of the color levels to the real-world colors.

To digitize a picture into your Amiga you'll need three items: (1) A video camera to input the image. (2) The digitizer module. This is usually a small box that plugs into the Amiga's parallel port and has a cable that attaches it to the camera. (3) Software to control the type of sampling (black-and-white or color) and the enhancement of the image after it is

**FIG. 10.2** Digitizing gray-scale images by sampling at various levels of gray between black and white.



captured on screen. Currently, digitizer module and software are sold together in Digi-View by NewTek.

## USING DIGI-VIEW

---

Digi-View by NewTek allows you to capture images of outstanding clarity. The system includes software, three color filters, and the digitizer, which is a matchbox-sized module that plugs into an Amiga port, all at a low price. You'll also need a black-and-white video camera.

With Digi-View and a monochrome camera, you can generate an image with resolution of up to  $640 \times 400$  resolution in 128 shades of gray. It takes about 20 seconds to digitize this exceptionally clear, hi-res, black-and-white image. The lo-res mode takes only 10 seconds to capture.

The software then has easy-to-use pulldown menus that allow you to enhance the image with different shades of gray, or you can save it in IFF format for input into any paint program and add color, titles, or special effects.

You can also digitize broadcast-quality color images at  $320 \times 200$  resolution. To do this, aim at your subject and use three color filters held in sequence in front of the camera lens for 10 seconds each. Unfortunately, this requires that your human subjects hold very still for 30 seconds to avoid blurring the image. Moving scenes are not appropriate material but all static objects are excellent subjects. After making the three passes in red, green, and blue, the software combines the images and relates them as one, using all of the Amiga's 4,096 colors resolved at up to 21 bits per pixel—over 2 million shades. These full-color hold-and-modify images cannot, however, be used in a paint program (though Digi-View is planning to remedy this). It is easier than it sounds and very powerful.

With a slide projector, you can also input 35 mm camera images. This avoids the problem of keeping the subject or scene still. Suppose you wanted to create a house-for-sale advertisement. Take a slide of the house and project it onto a small viewing screen. Aim the video camera at the screen, hold the three filters in front of the camera for 10 seconds each, and wait a moment for the full color slide image to appear on the screen. You could then use the mouse and software to adjust the color,

brightness, and contrast (make the lawn greener). This color image cannot be manipulated by a paint program. It can, however, be printed out or photographed from the screen and combined with other graphics and text in your ad.

The possible uses for this system are endless. A local artist uses his digitizer to reproduce the artwork of the masters from his art library onto the screen. These images can then be saved, enhanced, and printed on canvas, t-shirts, or paper.

The 600 lines of resolution produced by monochrome cameras provide excellent high-quality images with Digi-View. If you use color cameras with Digi-View, you'll get unattractive streaking through the image. Digi-View suggests using an RS-170 monochrome camera with 2:1 interlace sync. Most newer cameras are fine, but those older than five years, use a random interlace system instead, which results in inferior quality.

Although Digi-View produces high-quality images, they take between 10 and 30 seconds to digitize. Speed stands second to quality. If you don't need to digitize live video or moving objects, however, you can use Digi-View to capture images within a few seconds, with excellent resolution, and at low cost.

## **USING GENLOCK**

---

Simply described, genlock is a process for mixing two independent video signals into a single signal. The following description from the "Deluxe Video Users Guide" (see "Vendor Reference") gives an idea of how complex the process really is:

Imagine yourself holding a can of spray paint. Your job is to spray the inside of a glass faced tube completely once every 1/60 of a second. A control panel hangs in front of your face, telling you when to spray each of the three colors you have in cans, and where to spray them. Now imagine two control panels, with conflicting information on each. When you mix two video signals, each tries to control the electron beam, your spray paint. Unless they are perfectly in sync, the result is chaos. Genlock is a means of synchronizing two discrete video signals.

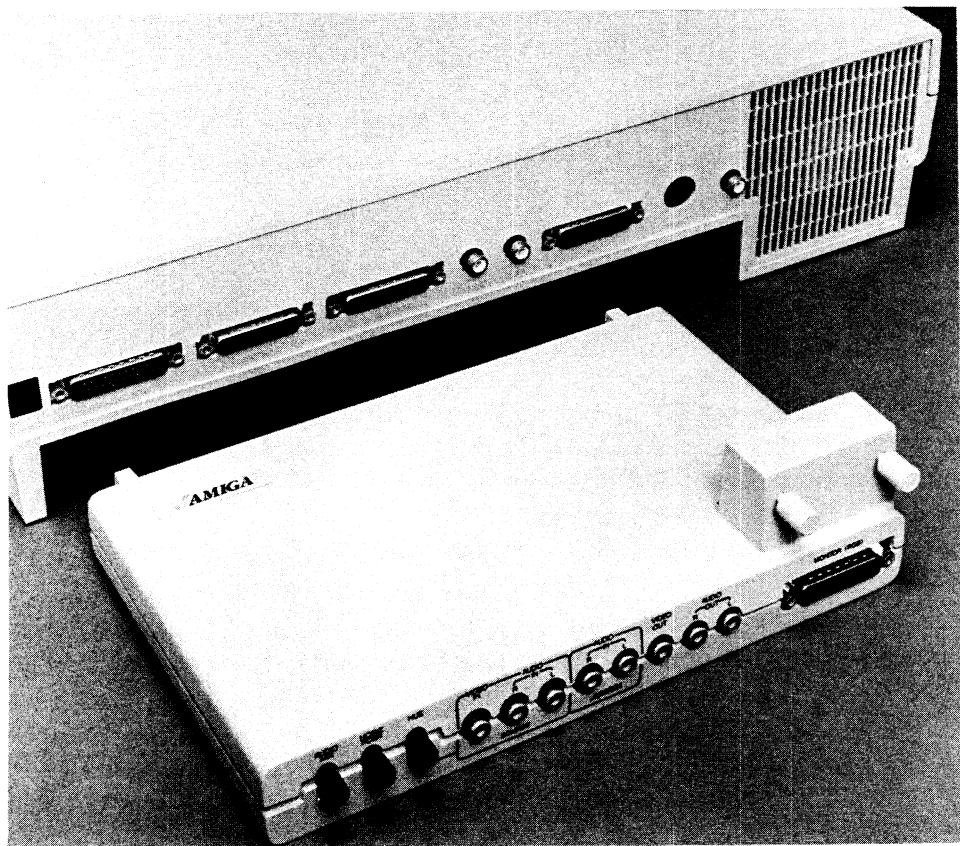
## Using Amiga Genlock 1300

Amiga Genlock 1300 by Commodore-Amiga was designed to let you overlay the graphics and audio created on the Amiga onto videotape, laser disk, or other computers. Outputs include composite and RGB videos as well as a stereo audio signal (Figure 10.3).

You can use Genlock 1300 to create professional effects such as fades, wipes, scrolling titles over live action, superimposing animation onto live video, and mixing live audio. In its simplest use: just combine your VCR camera with the Amiga and Genlock 1300 and record everything you make on your Amiga onto the VCR.

Suppose you want to overlay an animated title sequence onto the beginning of a business videotape showing your office facilities to clients.

FIG. 10.3 The Amiga Genlock 1300.





You create the title animation with Aegis Animator or Deluxe Video Construction Set. Then you hook up two VCRs with genlocking input to the Amiga. You then play the videotape on one machine, play the animation on the Amiga screen, and tape both onto the second VCR. With appropriate software (such as the animation programs) you can include television-like dissolves, blackouts, and other special effects in your video production.

### **Other Genlocks**

Other genlock devices are available for use with the Amiga. An example is the Telecomp 1000 from Universal Video. It allows you to place the live video in a special window on the screen (see "Vendor Reference").



# File Management

After you've used the software discussed in this book, you might suddenly find your diskettes filled with files in no apparent order. If a great deal of time has passed since you started using the diskette, you might forget what the files are. It is extremely useful to organize your files into logical groupings that are assigned meaningful or descriptive names. This will help you to find files efficiently without losing any. This chapter outlines using the *Command Line Interface (CLI)* for comprehensive file management.

## THE CLI

---

There are two ways to interact with the Amiga. One is by selecting icons to perform predetermined functions, as you do with the Workbench software. The other method uses the CLI to interpret commands typed in from the keyboard. The language you use to type commands is called AmigaDOS, and it has just 45 commands. These 45 commands are all you need to manage your files, and the following outline makes it easy to learn.

### Opening the CLI

First, activate the CLI icon by choosing *Preferences* from the Workbench and clicking the CLI to on. This step does not put you into

the CLI, it simply allows you to select the interface later on. Now, choose *Save*. You won't have to repeat this step the next time you boot Workbench; the CLI will stay on.

Next, open the CLI icon from the System drawer. When you get the `1>` prompt, you are ready to use AmigaDOS to command the machine.

To see a list of all 45 AmigaDOS commands, type the following string at the `1>` prompt:

```
1> list c
```

To get a printed list of all 45 AmigaDOS commands, type at the `1>` prompt:

```
1> list c to prt:
```

Most of the commands are, as you'll see, extremely straightforward.

## Data Disks

As noted in Chapter 2, before you start any work on the Amiga, it is often necessary (and always wise) to prepare (*initialize or format*) a disk on which to print and save the files (paintings, programs, songs, . . .) you create. A disk will be ready to store information after you use the initialize procedure from the Workbench (instructions are in the Amiga Users Guide) or the *Format* from the CLI (more on this follows in this chapter).

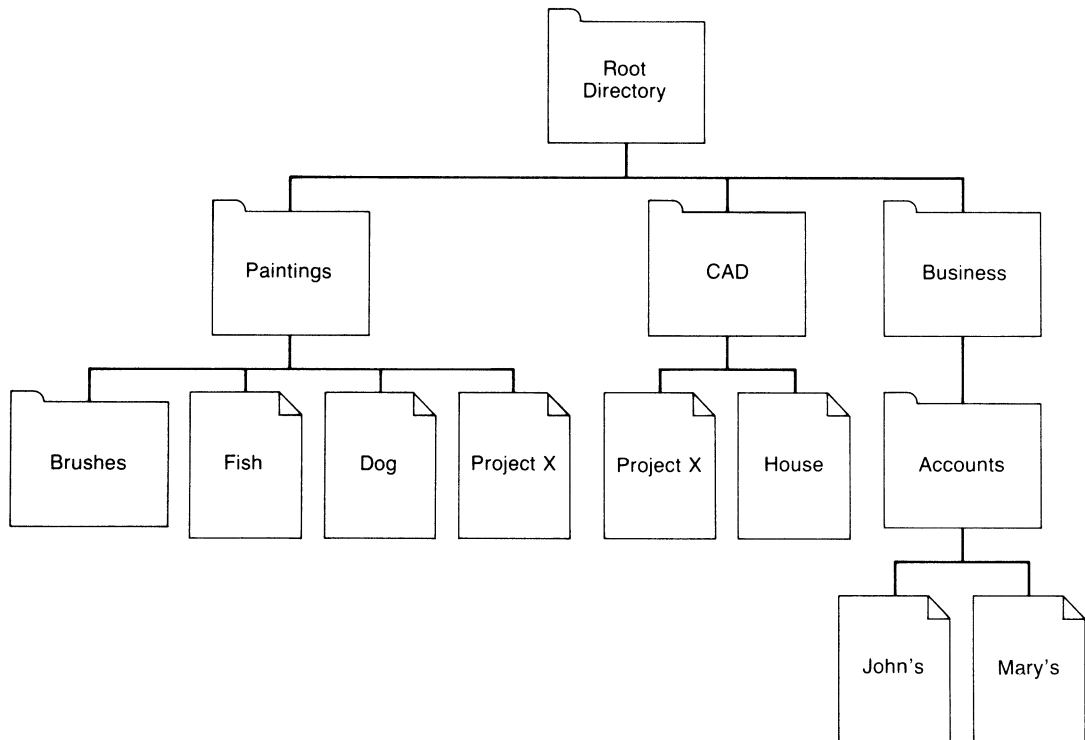
## Files

When you create a painting, program, or song, it is saved as a series of symbols. The entire collection of symbols make up a *file*. Think of a file as you would a piece of paper. An empty file is like a blank piece of paper, while a file containing characters, numbers, spaces, etc., is like a typed page.

Files are stored by copying them onto a disk. If you simply save everything to a disk in random order, eventually you will have a huge conglomeration of files without logical associations. Issuing the *dir* command at the CLI to get a directory (a listing) of a disk's contents is a lengthy and frustrating procedure when files are not grouped logically.

AmigaDOS, however, allows you to divide a disk into *directories* and

**FIG. 11.1** A schematic drawing of a disk organized with a Root Directory and various levels of Subdirectories.



*subdirectories*. These directories and subdirectories are also called *drawers*. (You've used them on the Workbench. The *Preferences* and *Trashcan* icons, for instance, represent drawers.)

Just as real drawers can be used to store things like paper, an AmigasDOS directory (drawer) is used to store files. In the paper analogy, a directory is to a file as a folder is to a sheet of paper. Taking the analogy one step farther gives a clear concept of a subdirectory. Imagine a folder labeled "expenses" into which you place two additional folders labeled "medical" and "car." Just as the "expenses" folder can contain the "medical" and "car" folders, a directory can contain subdirectories. Any directory that is contained within another directory is a subdirectory and any directory or subdirectory can contain files.

Schematically, a disk might be structured as shown in Figure 11.1.

Rather than scrolling through the entire contents of an unstructured disk, you can specifically request the directory Paintings, subdirectory

Brushes, and read the names of the files saved there. Note that you can use the filename "Project X" under two separate subdirectories, but you can't have two identically named files within the same directory. Also you can't have two subdirectories named Brushes within Paintings. This would cause a lot of confusion.

## Creating Drawers

To create the Paintings drawer listed in Figure 11.1 follow the procedures described for single or double disk drive systems.

**SINGLE DRIVE SYSTEMS.** To format and initialize a disk named "Graphics," type the following command at the 1> prompt. (Make sure the disk doesn't have any files you want to save because formatting a disk completely erases its contents.)

```
1> format drive df0: name "Graphics"
```

When prompted, remove the Workbench disk and insert the disk to be formatted. Press the Return key to start the formatting. When complete, replace the Workbench disk.

To create a directory named Paintings on the formatted disk, type:

```
1> mkdir Graphics:Paintings
```

The system will prompt you to insert the disk labeled "Graphics" before making the directory. Follow the rest of the steps below to make directories named "CAD" and "Business."

Insert Workbench

```
1> mkdir Graphics:CAD
```

Insert "Graphics" disk

Insert Workbench

```
1> mkdir Graphics:Business
```

Insert "Graphics" disk

Insert Workbench

Use the following commands to create the "Brushes" and "Accounts" subdirectories:

```
1> mkdir Graphics:Paintings/Brushes
```

```
1> mkdir Graphics:Business/Accounts
```

To verify the directory structure of the disk, type the commands shown below at the 1> prompt. You should get directory listings like those shown below.

```
1> dir Graphics:
    CAD <Dir>
    Paintings <Dir>
    Business <Dir>
1> dir Graphics:Paintings
    Brushes <Dir>
1> dir Graphics:Business
    Accounts <Dir>
```

**TWO DISK DRIVES.** If you have an external disk drive, file management is much easier. To format a disk, put it in the external drive and type:

```
1> format drive df1: name "Graphics"
```

To put the Paintings drawer listed in Figure 12.1 onto the initialized disk, type:

```
1> mkdir Graphics:Paintings
```

or

```
1> mkdir df1:Paintings
```

The first command will look for the disk labeled "Graphics" no matter which drive it is in. The second command assumes a disk with a directory named "Paintings" is in the external drive.

Verify the disk directory structure by typing the commands shown at the 1> prompts.

```
1> dir df1:
    CAD <Dir>
    Paintings <Dir>
    Business <Dir>
1> dir df1:Paintings
    Brushes <Dir>
1> dir df1:Business
    Accounts <Dir>
```

## Other AmigaDOS File Management Commands

The *list* command is similar to the *dir* command, but provides you with more information. For each file, *list* displays the file name, size, protection status, creation date, and a comment if one was specified with the *filenote* command.

- 1> list Graphics: Lists information on all files in the main directory disk named "Graphics."
- 1> list df1: Lists information on all files in the main directory of the disk currently in the external drive.
- 1> list df0:s/Startup\_Sequence: Lists information on the file named "Startup\_Sequence" in the "s" directory of the disk in the internal drive.

To change the default directory, use the *cd* command. Once you have changed the default directory, all file and directory operations are performed relative to the new default. For example, typing:

```
1> cd Graphics:Business/Accounts
1> list
```

will give you a listing of the files in the Accounts subdirectory of the Business directory on the disk named Graphics. The *cd* command is particularly useful when you wish to copy multiple files into a directory. Using *cd* can often save you significant typing and minimizes the chances of errors. For example, the following two blocks of commands provide exactly the same information but with different degrees of effort.

```
1> list Graphics:Paintings/Fish
1> list Graphics:Paintings/Dog
1> list Graphics:Paintings/"Project X"
1> cd Graphics:Paintings
1> list Fish
1> list Dog
1> list "Project X"
```

(Note: File or directory names containing embedded blanks must be enclosed in double quotes.)

To stay organized, you'll occasionally need to delete old files and reorganize files by copying to new disks and/or directories.



```
1> copy Graphics:Paintings/Fish to Backup:Paintings/Fish
```

Copies the file named “Fish” from the “Paintings” directory of the disk labeled “Graphics” to the “Paintings” directory on the disk labeled “Backup.” (The “/Fish” specification on the destination disk is unnecessary because the filename is identical to the one being copied.)

```
1> copy df0: to df1: all
```

Copies all files from the main directory of the disk in the internal drive to the main directory of the disk in the external drive.

Removal of files from a disk or directory is accomplished with the delete command.

```
1> delete Graphics:Paintings/Fish
```

Deletes the file named “fish” from the Paintings directory on the disk named Graphics.

```
1> delete df0:fun
```

Deletes the file named “fun” on the disk currently in the internal drive.

```
1> delete df1:Painting/dog, df1:CAD/house
```

Deletes the file named “dog” in the Paintings directory of the disk in the external drive, and the file named “house” in the CAD directory of the same disk.

Often you’ll want to give an existing file a different name. One method for doing this is to copy the existing file to a new file, then delete the old file. A much easier method uses the rename command.

```
1> rename fish to red_fish
```

Renames the file named “fish” in the current default directory to a file named “red fish” in the same directory.

```
1> rename df1:Paintings/dogs df0:bigdog
```

Renames and moves the file named “dog” in the Paintings directory on the disk in the external drive, to a file named “big dog” on the disk currently in the internal drive.

## File Protection

The `protect` command prescribes the operations that a file is capable of undergoing. Any operations not specifically prescribed in the protection status will not be possible.

```
1> protect fish status w
```

Allows the file named “fish” to be written to, but not read, executed, or deleted.

```
1> protect fish status rw
```

“fish” can be read and written to, but not executed or deleted.

```
1> protect fish status r
```

“fish” can be read, but not written to, deleted, or executed.

```
1> protect color.bas status rwd
```

The program named “color.bas” can be read, written to, executed, and deleted.

## Backing up Entire Disks

It is extremely important that you frequently back up your important disks so you don't totally lose your programs and data if an original disk is lost or damaged. When possible, it's best to save your original disks as master copies and use only back-ups to work with.

To copy the entire contents of a disk to a new disk, use the `diskcopy` command. This command formats the destination disk, so make sure that disk doesn't contain any files you want to keep. For one-drive systems, the following command will back up a disk onto a new disk named “Graphics backup.”

```
1> diskcopy from df0: to df0: name "Graphics backup"
```

One disk-drive systems require three complete swaps of the original and back-up disks. You'll be prompted automatically when the system is ready for the second disk.

---

Two disk-drive systems simplify disk back-ups significantly, because the original and destination disks need to be inserted only once.

```
1> diskcopy from df0: to df1: name "Graphics backup"
```

This command prompts you to insert the disk you want to copy from in the internal drive (df0:), and the disk you want to copy to in the external drive (df1:).

## CREATING A RAM DISK

---

*Random Access Memory (RAM)* is the memory space inside the computer used to store programs and data temporarily. When the power switch is turned off, the contents of RAM are lost.

RAM can be addressed as an AmigaDOS device by using the ram: device name. For your purposes, think of the ram: device as you would a disk drive containing a disk. You can store programs and data files onto this "RAMdisk" using all of the same file and directory management commands that apply to standard disks. The advantage of a RAMdisk, however, is its much greater speed.

When working extensively in the CLI, you save a lot of time by storing the CLI on a RAMdisk. To create a RAMdisk containing the CLI commands, enter the CLI from Workbench and type the following commands:

```
1> mkdir ram:c
1> copy sys:c ram:c all
1> assign c: ram:c
```

The first command makes a directory named "c" (for command) on the RAMdisk. Copy transfers the entire contents of the directory named "c" on the system disk to the "c" directory on the RAMdisk. (The system disk is the one that was booted in the internal drive. In this case it is the Workbench disk.) The *assign* command names the "c" directory on the RAMdisk as the new command directory.

You can get information on the size of the RAMdisk by typing:

```
1> info
```

The ram: device is allocated only as much memory as it needs, so it will always be listed as 100% full no matter how many files it contains.

After creating the RAMdisk, you can remove the Workbench disk from the internal drive. Because the CLI commands are executed from the RAMdisk, you can use the internal drive to hold other disks you want to modify.

To see the commands you copied onto the RAMdisk, type:

```
1> dir ram:c
```

# Screen Reproduction Techniques

The Amiga creates a new link between traditional artistic media and the technology of the computer. You've learned how to create greeting cards, professional advertisements, CAD drawings, and charts and graphs, not with paper and pens, but on the computer screen. The remaining step is determining how you'll transfer the image on your screen onto the card, canvas, or annual report.

The Amiga supports many output devices, offering you a selection that covers whatever personal or business needs you have. This section briefly introduces the alternatives. You must evaluate what the final destination of your art will be, how fast and attractive the output should be, what quantity of output you'll have, and how much money you can spend. We suggest that you read this section, evaluate your needs, and then get further information and demonstrations from local computer stores, friends, magazines, and user group newsletters and meetings. Information, products, and prices fluctuate enough to warrant some shopping around.

---

## PRINTERS

At some time, you'll want to transfer your Amiga images to paper. The needs fall into three basic categories: text capabilities, color printing for graphics, and plotting for CAD work.

Workbench 1.1 supports the following common printers:

Alphacom Alphapro 101

Brother HR-15XL

CBM MPS 1000

Diablo 630

Diablo Advantage D25

Diablo C150 (color inkjet)

Epson (dot matrix)

Epson JX80 (dot matrix)

HP Laserjet

HP Laserjet Plus

Okimate 20 (color thermal transfer)

Qume Letterpro 20

Custom-default setting to a simple generic printer with no character translation

Workbench 1.2 also supports the Imagewriter II and Okidata.

“Supporting” these printers means that on your Workbench disk is a driver for each one. A driver is a program that tells the Amiga how to interact with a specific printer. Commodore-Amiga is constantly updating the list and each new version of Workbench will contain drivers for more and newer printers.

Many other printers can be set to one of the above standards. Just because your printer isn't on the list doesn't mean you can't use it. Table 12.1 is a list of printer-and-driver combinations for a variety of printers. It shows specifically which printers can be set to emulate one of the above supported printers and which driver will run it.

If your printer isn't on either list, start your search for a driver at local computer shops, user groups, and bulletin board services to see whether the driver you need is on a public domain program (a low-priced or free program created and distributed by Amiga users to help other users).

**TABLE 12.1.**

**Color Printer Compatibility List\***

Printer	Driver	Comments
Diablo C150	Diablo C150	
Epson JX-80	Epson JX-80	
Apple Imagewriter II	Imagewriter	Requires custom driver
Juki 5510	Epson JX-80	Requires a color upgrade
Okimate 20	Okimate 20	Requires Amiga Plug-n-Print

**Dot Matrix Printer Compatibility List**

<i>Printer</i>	<i>Driver</i>	<i>Comments</i>
Alphacom Alphapro 101	Alphapro 101	
Apple Imagewriter	Imagewriter	Requires custom driver
Brother 1509	Epson	
Citizen MSP series	Epson	
CIE CI-3500	Epson	Requires IBM-compatible module
Epson FX series	Epson	
Epson LX series	Epson	
Epson MX series	Epson	
Epson RX series	Epson	
Epson LQ-800	Epson	
Epson LQ-1500	Epson	Supports text only
IBM Graphics	MPS-1000	
Juki 5510	Epson JX-80	
Okidata 192	MPS-1000	Set in IBM-compatible mode
Okidata 193	MPS-1000	Set in IBM-compatible mode
Panasonic 1080	Epson	
Panasonic 1091	Epson	
Panasonic 1092	Epson	
Panasonic 1592	Epson	
Star SG-10	Epson	All features not implemented
Star SD-10	Epson	All features not implemented

**Daisy Wheel Printer Compatibility List**

<i>Printer</i>	<i>Driver</i>	<i>Comments</i>
Brother HR-15XL	Brother HR-15XL	
Citizen Premiere 35	Diablo 630	
Comrex CR-II	Brother HR-15XL	
Comrex CR-III	Brother HR-15XL	
Commodore 6400	Diablo 630	Requires Centronics option
Diablo Advantage D25	Diablo Adv. D25	
Diablo 630	Diablo 630	
Dynax	Brother HR-15XL	
Epson DX-20	Diablo 630	
Juki 6000	Diablo 630	

TABLE 12.1 (continued)

**Daisy Wheel Printer Compatibility List** (continued)

<i>Printer</i>	<i>Driver</i>	<i>Comments</i>
Juki 6100	Diablo 630	
Juki 6300	Diablo 630	
Qume Letterpro 20	Qume Letterpro 20	
Qume Sprint 11	Diablo 630	
Panasonic KXP 3131	Diablo 630	
Panasonic KXP 3151	Diablo 630	
Tec F10	Diablo 630	

**Laser-Jet Printer Compatibility List**

<i>Printer</i>	<i>Driver</i>	<i>Comments</i>
Canon Laser	Epson	Set in FX-80 mode
CIE Lips 10	Epson	Set in Epson mode
HP Laser Jet	HP Laser Jet	
HP Laser Jet Plus	HP Laser Jet Plus	

\*All of the printers listed above require the appropriate printer cable.

Many home and business programmers are writing custom drivers for their own machines and releasing them as public domain software through user groups and good computer shops. For example, the Canon PJ1080A we used throughout this project worked via a driver that a local programmer wrote and donated to the user group software library. The drivers available in the public domain include:

- Alphacom Alphapro 101
- C Itoh 8510A
- CBM MPS 1000
- C Itoh Prowriter
- Diablo 630
- Diablo C-150
- Epson 2
- Epson JX-80
- Generic
- HP LaserJet PLUS
- NEC 8025A
- Okidata-92
- Okimate 20
- Qume LetterPro 20



---

Star Gemini 10  
Star SG-10 (text only)  
Brother HR-15XL  
Canon PJ1080A  
CGP-220  
Diablo Advantage D25  
Epson  
Epson 3  
Epson LQ-800  
HP LaserJet  
Imagewriter II  
NEC P6  
Okidata m192  
Panasonic K-P10xx  
SmithCorona D300  
Star SG-10 (IBM Mode)

If you're an experienced programmer, try writing your own driver.

If none of the above methods do the trick, try calling the printer manufacturer to ask whether your printer can be adjusted to emulate one of the supported printers. This can often be accomplished by adjusting dip switches in or on the printer. If this doesn't work, and you are very persuasive, try getting the printer manufacturer or dealer to write a custom driver for your printer.

## Shopping for a Printer

Whether you'll require one printer for text and one for graphics depends entirely on your needs. Inexpensive color printers give you home-quality graphics and okay-for-a-letter-to-Mom black-and-white text. If you need professional quality graphics for presentations and letter quality text printed fast and in large quantities, you'll probably need two different printers or a laser printer. The increased in-house printing capabilities could, however, save enough money to quickly recover your investment.

**PRINTOUTS.** When buying a printer, be sure to look at several printouts (of your own work, if possible) on various papers. The most-advertised printers aren't always the best. For instance, the well known Okidata is reasonably priced and produces bright colors, but it leaves horizontal bars across the page whereas the little-known Radio Shack CGP-20 gives a bright, solid print on the proper paper.

**PAPER.** All papers are not made alike. The type of paper you use will have a tremendous impact on the print quality. Don't write off a Canon or Radio Shack when you see their mediocre printouts on cheap roll paper. Try printing something out on better-quality roll paper, bond, and composition-type paper for mechanical impact (available at paper stores). Prints on composition-type paper are stunning. Do this for any printers you test.

**INK.** Talk to people or read the printer manual to find out how long the ink cartridges or ribbons last. Remember that ribbons look great at first, but some lose quality later. How many pages will they print? It can get expensive.

Consider an ink jet printer that uses cartridges of ink that shoot dots onto the page (no keys). We're still using the same cartridge that our printer came with, after hundreds of pages.

**NOISE POLLUTION.** Listen to the printer. Remember that there is background noise in the computer shop. How loud will it be in your home or office?

## **PLOTTERS**

---

The drivers for plotters are included in the software rather than on Workbench because plotters are really only suited to a specific type of graphics. It is not practical for a plotter to reproduce all types of graphics. In fact, normal "raster graphics" is not well suited at all.

Plotters are designed for software packages using "vector graphics." This type of graphics defines its data as a series of line segments, and plotters are very good at drawing line segments. In fact, plotters with a resolution of 1000 points per inch are quite commonplace. Thus, when appropriate, the quality of plotters is actually superior to that of most printers.

To see why plotters work better with vector graphics than with raster graphics, consider the way a plotter works.

- Most plotters use ink-filled pens that are selected by the plotter driver whenever a new color is desired.

- Plotters must be given specific instructions to:
  - (i) position the pen
  - (ii) put the pen on the paper
  - (iii) pick the pen up off the paper.

To plot a  $640 \times 200$  screen as a collection of dots requires 128,000 positions. At each position, the pen must be put down and picked up, and whenever a new color is desired, the plotter must perform a very time-consuming pen exchange. All of this would very quickly add up to one afternoon per plot if you were using raster graphics.

Your CAD program will describe the data that you need to include about your plotter in order to run it with the program. If your program still doesn't run your plotter, you could try some of the above options of calling the plotter manufacturer, the software company, and checking other sources.

## SCREEN SHOTS

---

Using a camera, you can photograph the screen to make slides or prints of an image. This is a fairly inexpensive way to get presentation-quality reproductions quickly.

You'll need to set a 35 mm camera (though for fun without copies a Polaroid works great) on a tripod in front of the screen. The room should be completely dark to avoid reflections. If your room can't be completely darkened, you can buy a box to hood the screen. Align the center of the lens with the center of the screen. Use a fast black-and-white film, or daylight film for color. Shutter speeds should be set to about 1/15th to 1/8th of a second. Shutter speeds faster than 1/30th of a second will often cause a dark band across the picture. (It takes about 1/30th of a second to refresh the screen.) Focus so that the screen fills the frame of the camera.

Until you're experienced, it's best to take several exposures at various f-stops to make sure you get one excellent shot. You can use a light meter or just the automatic setting.

These shots can be developed in your own darkroom or taken to a local developer. Also, if you use color slide film, the slides can be reproduced and enlarged on a Xerox 6500 Color Copier. This service is available at many copy centers.

## FILM RECORDERS

---

Film recorders are especially useful in business. They accept video output signals from the Amiga and use a three-color exposure technique to create 35 mm film of the image. The internal monitors of film recorders are flat to prevent distortion and curvature. They'll accept various film formats, including instant-developing film. Prices range from about \$1,500 to \$15,000.

## STRAIGHT-OFF-THE-DISK SLIDES AND SEPARATIONS

---

Various companies will take your disk and transfer your work to slides or film separations. You can find printing shops and companies that specialize in disk transfers listed in computer magazines and in the Yellow Pages. Such services are relatively expensive, but the product is professional quality with no scan line or screen curvature distortion. You can order film separations for offset or halftone gravure; hard dot laser; negative or positive; emulsion up or down; with negative or positive proofs; and with prepress services such as stripping, color correcting, and air brush work. The service used for the color plates in this book was Imageset of San Francisco (see "Vendor Reference"). You simply send them your disk with prepayment and they return color slides to you.

## SLIDE SHOWS

---

For home and business purposes you might actually want to keep the images in the computer in the form of a slide show. Software such as the animation programs and Impact offer you the alternative of arranging your pictures in the sequence you prefer, fading from one to the other, and even including music and text in the presentation. There are also simple programs for slide shows available in the public domain libraries of computer stores and user groups. The most popular one allows you to load your slides in any order, the computer shows one after the other, and you can use the keyboard to pause the show at any point.

Slide shows work well on the Amiga monitor for individual viewing.

For large groups, use a large-screen monitor or television. These are available at various prices in sizes suitable for use in a large conference room or even an auditorium. The effect can be quite striking.

## **VCRS**

---

Also suited for home and business use is the VCR. See Chapter 10 for descriptions of the hardware and software needed to take a presentation created on the Amiga and record it onto a videotape. By doing this, you can create a portable video tape that can be shown anywhere on any VCR and TV, no computer needed.



# Appendix

## VENDOR REFERENCE

---

Activision  
2350 Bayshore Frontage Rd.  
Mountain View, CA 94043  
415-960-0410

Aegis Development  
2210 Wilshire Blvd.  
Suite 277  
Santa Monica, CA 90403  
213-306-0735

Applied Visions  
15 Oak Ridge Rd.  
Medford, MA 02155  
617-488-3602

Commodore Business Machines Inc.  
1200 Wilson Drive  
West Chester, PA 19380  
215-431-9100

New Tek  
701 Jackson B3  
Topeka, KS 66604

Electronic Arts  
PO Box 7530  
San Mateo, CA 94403  
800-245-4542

ImageSet  
555 19th Street  
San Francisco, CA 94107

IVL Technologies Ltd.  
#3-3318 Oak Street  
Victoria, BC V8X 1R2 Canada  
604-383-4320

Mimetics Corp.  
PO Box 60238 Station A  
Palo Alto, CA 94306

Universal Video  
(A division of Valiant IMC)  
195 Bonhomme Street  
PO Box 488  
Hackensack, NJ 07602





# INDEX

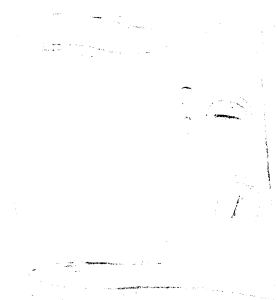
- Acetate tracing, *see* tracing images
- Activision, 196
- Aegis Animator
  - Advanced tips, 106–108
  - Comparison to Deluxe Video, 89–94
  - Editing the script 108–114
  - Overview 88, 94–96
  - Tutorial 96–106
- Aegis Development 88
  - See also* Aegis Animator, AegisDRAW, and Aegis Impact
- Aegis DRAW
  - Graphics primitives, 63
  - Drawing strategies, 64
  - Tutorial, 64–86
- Aegis images, 40
- Aegis Impact
  - Tutorial, 46–61
- Airbrush, Deluxe Paint, 30
- Alley, Jim, 32, 35
- Amiga
  - chips 5
  - central processing unit (CPU), 6
  - disk drive, 7
  - in art 3. *See also* Paint programs
  - in business 4. *See also* Aegis Impact
  - in CAD 3. *See also* AegisDRAW
  - in education, 4
  - in entertainment, 5
  - in music 4. *see also* Sound in science, 5
  - keyboard, 7
  - microprocessors, 5
  - monitor, 6
  - mouse, 7
- AmigaBASIC, *see* Programming
- Anakin Research Inc., 37
- Analog-to-digital converter, 183
- Analog mode, *see* Amiga, monitor
- Animation, *see* Aegis Animator, Deluxe Video, and Programming
- Animator-Deluxe Video
  - comparison table, 89–94
- Ani-Panic function, Aegis Animator, 108
- Antialiasing, Deluxe Paint, 27
- Appear, Deluxe Video, 120
- Applied Visions, 197
- Backer tool, AegisDRAW, 76
- Backgrounds, importing to Aegis Animator, 96
- Background track, Deluxe Video, 126
- Back ups, *see* file management
- Bar chart, Aegis Impact, 48
- Basic demos drawer, 165
- Bitmap, 161
- Bitmap animation, 161
- Blend, Deluxe Paint, 31
- Blitter, 162
- Blitter objects, 162
- Bloop, Deluxe Video, 122
- Borders,
  - Aegis Impact, 51
  - Deluxe Paint, 16–17
- Brawn, Gene, 88, 129
- Brush menu, Deluxe Paint, 14
- Brush modes, Deluxe Paint, 19
- Brush selection tool, Deluxe Paint, 14, 21
- Business graphics, *see* Aegis Impact
- Camera images, 36
- Change directory (CD) command, AmigaBASIC, 218
- Changing background color, Aegis Impact, 49
- Charts, creating, *see* Aegis Impact
- Chips, 5
- Christman, Ian, 33
- Circle tool, Deluxe Paint, 24, 26
- Clip art, 40–41, 96
- Clone, AegisDRAW, 66
- COLLISION command, AmigaBASIC, 175
- Collision detection, 175, 177
- COLLISION function, AmigaBASIC, 175
- Color Bars, Deluxe Video, 127
- Command line interface (CLI), 213
- Commodore-Amiga, *see* Graphicraft and Musicraft

- Computer-Aided Design
  - comparison to paint programs, 11–12
  - overview, 11
- Cook, Janet, 33
- Cubes, 3D drawing, AegisDRAW, 64
- Cut, Deluxe Video, 129
- Cycle draw, Deluxe Paint, 22
- Cycle range, Deluxe Paint, 22
  
- Data disks, 214
- Data drawers, setting Deluxe Video, 118
- Deluxe music construction set, 197
- Deluxe Paint
  - advanced examples, 32–35
  - multitasking with, 39–40
  - screen resolution with, 39
  - tutorial, 12–32
- Deluxe Print, 41–43
- Deluxe Video construction set
  - advanced tips, 128–129
  - overview, 88, 114–115
  - tutorial, 126–127
- Digital modes *see* Amiga, Monitor
- Digitizers
  - sound, 206
  - video, 38, 185
- Digitizing Pens, *see* Graphics tablets
- Digi-View, 208
- Dir command, AmigaBASIC, 218
- Directories, 214
- Disappear, Deluxe Video, 124, 12
- DISKCOPY command, AmigaBASIC, 220
- Disk drive, 7
- Disks, *See* File management
- Documentation, 9
- Donnelly, Jerry, 35
- Drawers, *see* File management
  
- Easy!, 38
- Editing scripts, Aegis Animator, 108–114
  
- Electronic Arts, *see* Deluxe Music, Deluxe Paint, Deluxe Video, and Instant Music
- Empty effects, Deluxe Video, 118
- Empty track, Deluxe Video, 118
- Engel, Robynne, 3
- Enlarging text, Deluxe Paint, 15
- Eraser option, AegisDRAW, 66
- Extras disk, Amiga 8, 164–165
  
- Fade in, Deluxe Video, 118
- Fade out, Deluxe Video, 126
- Fade-to-black, Aegis Animator, 104
- Fast feedback, Deluxe Paint, 24
- Fast menu, Aegis Animator, 96, 97, 102, 107
- Fetch, Deluxe Video, 122
- File management
  - backing up disks, 220
  - command line interface (CLI), 213
  - creating drawers, 216–218
  - data disks, 214
  - file protection, 220
  - files, 214–216
  - opening CLI, 213
  - other file management commands, 218–219
- Files menu, object editor, 165
- Fill
  - AegisDRAW, 69, 72
  - Deluxe Paint, 13
- Fill patterns, Deluxe Paint, 18
- Film recorders, 230
- Fonts, Deluxe Paint, 14
- Foreground track, Deluxe Video, 216
- Foreign language, 5
- Four-channel sound, 4
- Framegrabbers, *see* Video Applications, video digitizers
- Future sound, 197
  
- Genlock, 209
- Genlock 1300, 210
- Global speed, Aegis Animator, 106
  
- Glow on, Aegis Animator, 96
- Grid snap, AegisDRAW, 69, 76
- Grid tool
  - AegisDRAW, 65, 80
  - Deluxe Paint, 17, 25
- Group, AegisDRAW, 73, 83
- Graphicraft, 40
- Graphics library, 156–158
- Graphics primitives, *see* Programming and Aegis Impact
- Graphics Programming, *see* Programming
- Graphics tablets, 36–38
- Graphs, creating, *see* Aegis Impact
  
- Hexidecimal values, 145–146
- HitMask, AmigaBASIC, 177
- Hook tool, Aegis Animator, 102, 108
  
- Icon, Amiga, 2
- Icon builder, Aegis Impact, 56
- Icon charts, Aegis Impact, 55–57
- Icon graphs, Aegis Impact
  - Automatic Graphing, 57–58
- Icons, Aegis Impact, 46
- Instant music, 196
- Into, Aegis Animator, 96
- Intuition 6. *See also* Kickstart IVL Technologies, 197
  
- Kernal graphics, 159–160
- Keyframe animation, 94
- Kickstart, 7
- Kurta command line interface (CLI), 36–38
  
- Legends, Aegis Impact
  - change color of, 49
  - moving, 50
- Line graphs, Aegis Impact, 53–54
- List command, AmigaBASIC, 218
- Load effects, Deluxe Video, 118
- Load fonts, Deluxe Paint, 14

- Lock colors, Deluxe Video, 126
- Loop tool, Aegis Animator, 102, 108
- Magnify, Deluxe Paint, 14, 31
- Main unit (CPU), 6
- MeMask, AmigaBASIC, 177
- Merge-in-back, Deluxe Paint, 19
- Metamorphic animation editor, 88
- Microprocessors, 5
- Mimetics, 197
- Monitor, 6
- Morph tool, Aegis Animator, 102
- Mouse, 7
- Multitasking with Deluxe Paint, 39
- Musician instrument digital interface (MIDI), 195
- Musicraft, 196
- Music studio, 196
- New script, Aegis Animator, 96, 106
- New Tek, 208
- OBJECT.AX command, AmigaBASIC, 175
- OBJECT.AY command, amigaBASIC, 175
- OBJECT.CLOSE command, AmigaBASIC, 167
- OBJECT.HIT command, AmigaBASIC, 175
- OBJECT.OFF command, AmigaBASIC, 167
- OBJECT.ON command, AmigaBASIC, 167
- Object editor, *see* Programming AmigaBASIC Animation
- Object-oriented graphics editor, 12
- OBJECT.PLANES command, AmigaBASIC, 170
- OBJECT.PRIORITY command, AmigaBASIC, 172
- OBJECT.SHAPE command, AmigaBASIC, 166
- OBJECT.STOP command, AmigaBASIC, 175
- OBJECT.X command, AmigaBASIC, 173
- OBJECT.Y command, AmigaBASIC, 173
- Operating system, Amiga; *see* Intuition
- Outline rectangle tool, Deluxe Paint, 16
- Output primitives, 142
- Oval tool, Deluxe Paint, 25, 26
- Paint option, object editor, 165
- Paint programs
  - comparison to computer aided design, 11-12
  - overview, 11
  - see also* Deluxe Paint, Aegis Images, and Graphicraft
- PALETTE command, AmigaBASIC, 170
- Paste, Deluxe Video, 129
- Path tool, Deluxe Video, 122
- Pattern brushes, Deluxe Paint, 22
- Patterns, creating Deluxe Paint, 17, 19. *see also* Fill patterns
- Peripherals, overview, 9
- Perspective drawing, AegisDRAW, 76-80
- Pettibone, Nancy, 34
- Phonemes, 199
- Pick, Deluxe Paint, 15
- Picture requestor, Deluxe Video, 118
- Pie charts, Aegis Impact, 52
- Pitchrider, 197
- Pixels, 11
- PlanePick, AmigaBASIC, 170
- Play, Deluxe Video, 122
- Player program, Aegis Animator, 106
- Play video, Deluxe Video, 126
- Plotters, 228-229
- Plotting graphs, Aegis Impact, 51
- Preferences menu
  - AegisDRAW, 66
  - Deluxe Paint, 15, 24
  - Workbench, 213-214
- Printer compatibility, 224-226
- Printers, 223-229
- Programming AmigaBASIC
  - animation
    - Blitter objects (BOBS), 161-162
      - colors of, 170-172
    - Comparison of sprites and BOBS, 163-164
  - Object editor
    - creating an object, 164
      - assigning object parameters, 172
      - displaying objects, 167
      - loading object data, 166
      - permanently removing objects, 167
      - selecting object colors, 168-170
      - temporarily removing objects, 167
    - moving objects, 173-174
      - accelerating objects, 174
      - detecting collisions, 175
      - determining object acceleration, 175
      - determining x and y object velocities, 174
      - selective collision detection, 177
    - sprites, 162
    - virtual sprites, 163
- Programming AmigaBASIC
  - graphics
    - AmigaBASIC screen, 132
    - colors
      - resolution, 138-139
      - selecting, 139-141
  - Comparison of screens and windows, 137-138
  - Graphics library subroutines, 156-158
  - Screens
    - default, 133
    - defining, 133

- Programming AmigaBASIC
  - graphics (*Continued*)
  - Selecting kernel graphics routines
    - closing a font, 159–160
    - opening a font, 158–159
    - setting a font, 159
  - Windows
    - attributes, 135–137
    - defining, 134
  - Output primitives
    - drawing arcs, circles, and ellipses, 150–151
    - drawing filled areas, 151–156
      - filling with patterns, 155–156
    - drawing lines, 144
      - line patterns, 145–147
    - drawing Points, 142–143
    - drawing shapes, 148–150
    - windows
      - attributes, 135–137
      - defining, 134
      - skeleton program, 132
- Programming AmigaBASIC sound
  - channels, 184–185
  - digital sound, 181
  - digitizing, 182
  - SOUND command, 185–187
  - WAVE command, 187–195
- Programming AmigaBASIC speech
  - SAY Command, 199
  - Translate\$ function, 199
  - voice array settings, 200–204
- Project menu
  - Aegis Animator, 96
  - AegisDRAW, 76
  - Aegis Impact, 48
  - Deluxe Video, 118
- Random access memory (RAM)
  - creating a RAM disk, 221–222
  - definition of, 6
- Raster graphics, 228
- Removing menu bars, Deluxe Paint, 16
- Resolution, *see* Screen resolution
- Retry, Deluxe Video, 117
- RGB Color Model, 141
- Root directories, 215
- Rotator tool, AegisDRAW, 86
- SAY command, AmigaBASIC, 199–200
- Savannah College of Art and Design, 3
- Scaled drawings, AegisDRAW, 80–86
- Screen reproduction techniques
  - film recorders, 230
  - plotters, 228–229
  - printers
    - compatibility, 223–226
    - shopping for, 327
    - ink, 228
    - noise pollution, 228
    - paper, 228
- Screen shots, 229
- Slide shows, 230
- Straight-off-the-disk slides and separations, 230
- Scene script, Deluxe Video, 118
- Screen shots, 229
- Screen resolution, 39
- Screens, *see* Programming AmigaBASIC graphics
- Scripps Institution of Oceanography, 5
- Script editing, *see* Aegis Animator
- Script Window, Deluxe Video, 118
- Shade tool, Deluxe Paint, 31
- Shading, Deluxe Paint, 30
- Shadows, creating with Deluxe Paint, 28, 30
- Slide builder, Aegis Impact, 51
- Slides
  - Aegis Impact, 46, 51–52
  - photographic, *see* Screen Reproduction Techniques
- Slide shows
  - Amiga, *see* Screen reproduction techniques
  - Aegis Impact, 47, 59–61
- Sound, *see* Programming AmigaBASIC sound
- SOUND command, AmigaBASIC, 18
- Sound hardware
  - digitizers, 195
  - MIDI, 195
  - samplers, 195
- Sound requestor, Deluxe Video, 122
- Sound samplers, 195
- Soundscape, 197
- Spare screen, Deluxe Paint, 185–187
- Speech, *see* Programming AmigaBASIC speech
- Sprites, 162
- Stamp tool, Deluxe Video, 128
- Storage, Aegis Animator, 96
- Storyboard
  - Aegis Animator, 96
  - Deluxe Video, 114
- Subdirectories, 215
- Swap menu, Deluxe Paint, 18
- Table, Aegis Impact, 86
- Text, AmigaBASIC, 158
- Text tool
  - Aegis Impact, 50
  - Deluxe Paint, 14, 49
  - Deluxe video, 119
- Three-dimensional objects, creating
  - AegisDRAW, 64–76
  - Deluxe Paint, 25
- Tic marks, Aegis Impact, 49
- Time menu, Aegis Animator, 99
- Tools menu, AegisDRAW, 65
- Tracing images onto screen, 36
- Translate\$ Function, AmigaBASIC, 200
- TV icon, Deluxe Video, 119
- Tweens, Aegis Animator
  - adjusting, 97
  - definition, 94
  - AegisDRAW, 66
  - Deluxe Paint, 13, 29
- User guides, Amiga, 9

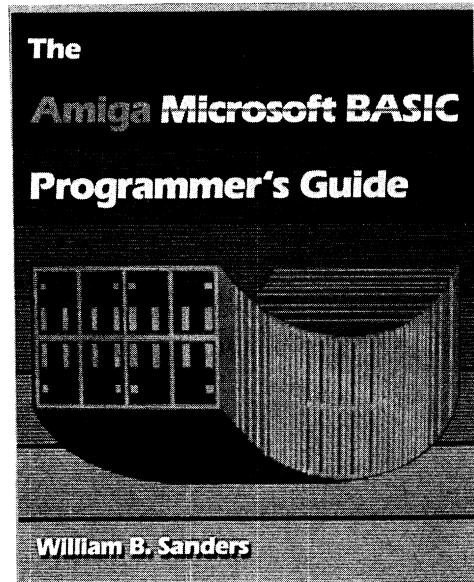
- Vector graphics, 228
- Video applications
  - Genlock, overview, 209
    - Using Amiga Genlock 1300, 210–211
    - Other Genlocks, 210
  - using a VCR, 205
  - video digitizers, overview, 206–208
    - using Digi-View, 208–209
- Virtual sprites, 163
- Voice array settings, 200–201
- Warhol, Andy, 3
- Warner Bros., 94
- WAVE Command, AmigaBASIC, 187–195
- Windows, *see* Programming
  - AmigaBASIC graphics
- Workbench, Amiga, 8
- x-y coordinates, AmigaBASIC, 162
- x-y display
  - AegisDRAW, 65
  - Deluxe Paint, 18
- x-y object velocities, 174
- Zoom tool, Deluxe Paint, 31





**ANOTHER AMIGA BOOK FROM SCOTT, FORESMAN AND  
COMPANY**

**THE AMIGA MICROSOFT BASIC  
PROGRAMMER'S GUIDE**



By William B. Sanders, 384 pages, **\$19.95**, 18523-0

Master Amiga Microsoft BASIC with this easy-to-understand book. You'll find a complete, step-by-step guide to Microsoft BASIC for the Amiga, and numerous, interesting examples of advanced features, both designed for beginning and intermediate users. *The Amiga Microsoft BASIC Programmer's Guide* pays special attention to the color graphics and voice synthesizer—two of Amiga's most unique features. It also provides extensive coverage of pull-down menus and special mouse control, using practical program examples, including a special artificial intelligence program. This book

- Shows you how to create and customize your own software for your individual needs
- Explains how to write simple database programs
- Teaches you how to integrate different programs to work together
- Shows you how to write mult-tasking programs in BASIC
- Gives instructions on how to call up your favorite electronic information using the book's terminal program



**HERE'S HOW TO ORDER:**

Contact your local bookstore or computer store, or send the handy order form below to:

**Scott, Foresman and Company**  
**Professional Publishing Group**  
1900 East Lake Avenue  
Glenview, IL 60025

**In Canada, contact**  
Macmillan of Canada  
164 Commander Blvd.  
Agincourt, Ontario  
M1S 3C7

YES, please send me \_\_\_\_\_ copies of THE AMIGA MICROSOFT  
BASIC PROGRAMMER'S GUIDE, \$19.95, 18523-0

Please check method of payment:

Check/Money Order       MasterCard       VISA

Amount enclosed \$ \_\_\_\_\_

Credit Card No. \_\_\_\_\_ Exp. Date \_\_\_\_\_

Signature \_\_\_\_\_

Name (please print) \_\_\_\_\_

Address \_\_\_\_\_

City \_\_\_\_\_ State \_\_\_\_\_ Zip \_\_\_\_\_

Add applicable sales tax, plus 6% of total for shipping.

Full payment must accompany your order.

Mail order form to: Scott, Foresman and Company  
Professional Publishing Group  
1900 East Lake Avenue  
Glenview, IL 60025

A18527

**HERE'S HOW TO RECEIVE YOUR FREE CATALOG OF THE  
LATEST COMPUTER BOOKS FROM SCOTT, FORESMAN AND  
COMPANY**

Simply mail in the coupon below to receive your free copy of our latest catalog featuring computer and management books, and find out how they can benefit you.

YES, please send me my **free** catalog of your latest computer and management books! I am especially interested to learn more about your books on

- Programming
- Business Applications
- Networking and Telecommunications
- Other \_\_\_\_\_

Name (please print) \_\_\_\_\_

Address \_\_\_\_\_

City \_\_\_\_\_ State \_\_\_\_\_ Zip \_\_\_\_\_

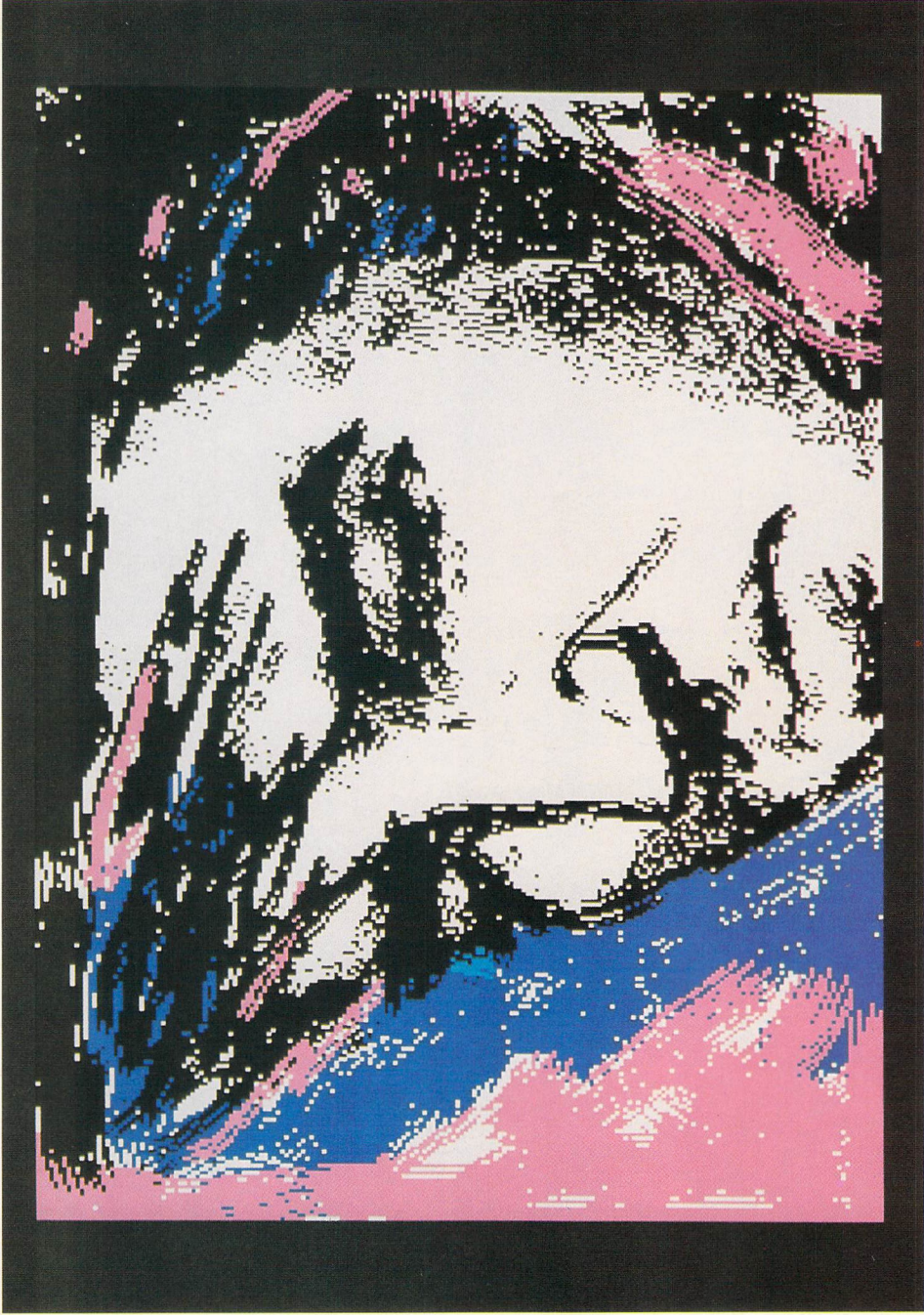
**Mail coupon to:** Scott, Foresman and Company  
Professional Publishing Group  
1900 East Lake Avenue  
Glenview, Illinois 60025



Tina by Tina Boyles, The Savannah College of Art and Design. (Lo res, Deluxe Paint)



Seahorse by Cathy Howard, The Savannah College of Art and Design. (Lo res, Deluxe Paint)



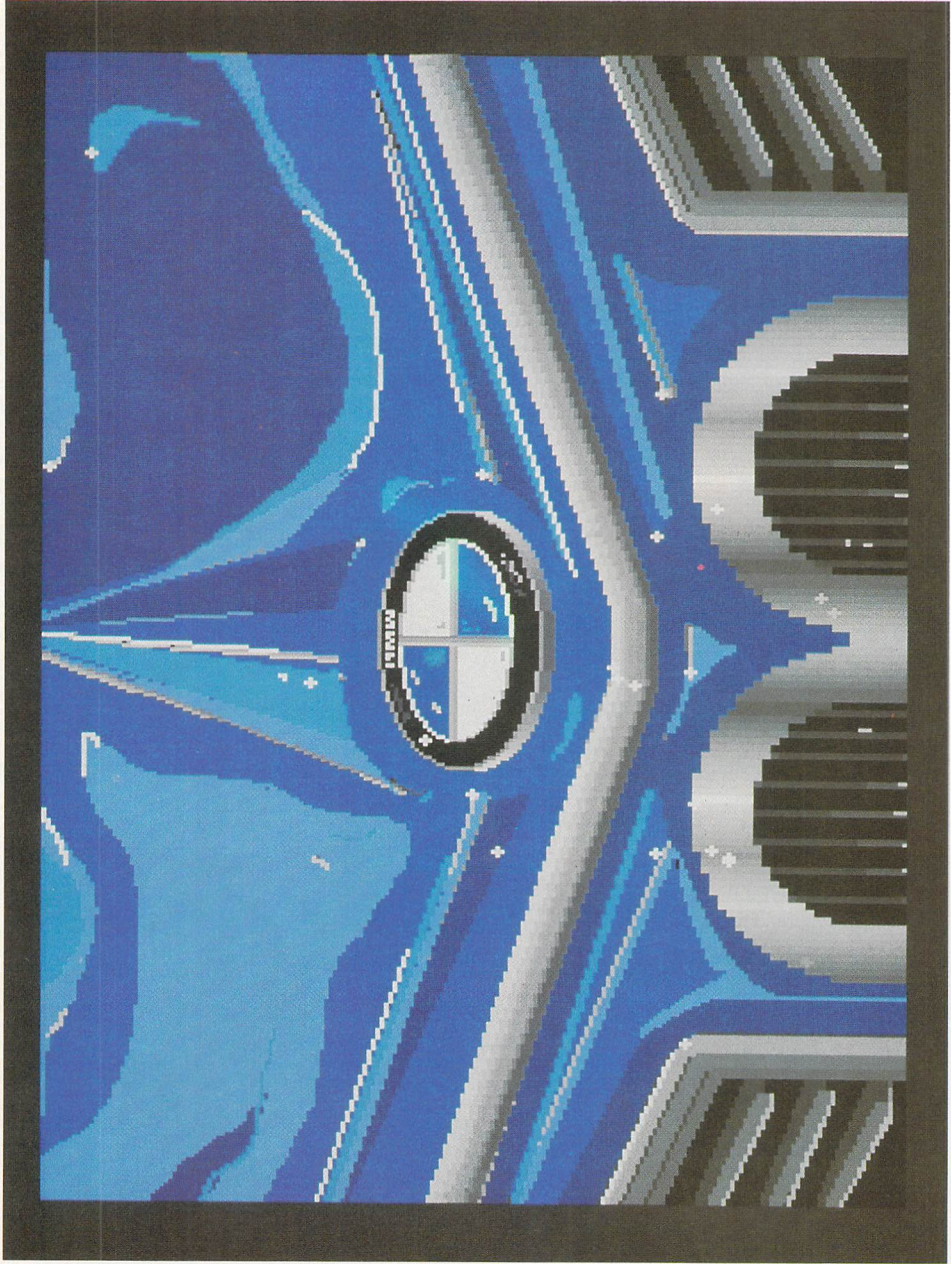
Elvis by Mark Streeter, The Savannah College of Art and Design. (Lo res, Deluxe Paint)



Dara by Les Walker, The Savannah College of Art and Design. (Lo res, Deluxe Paint)

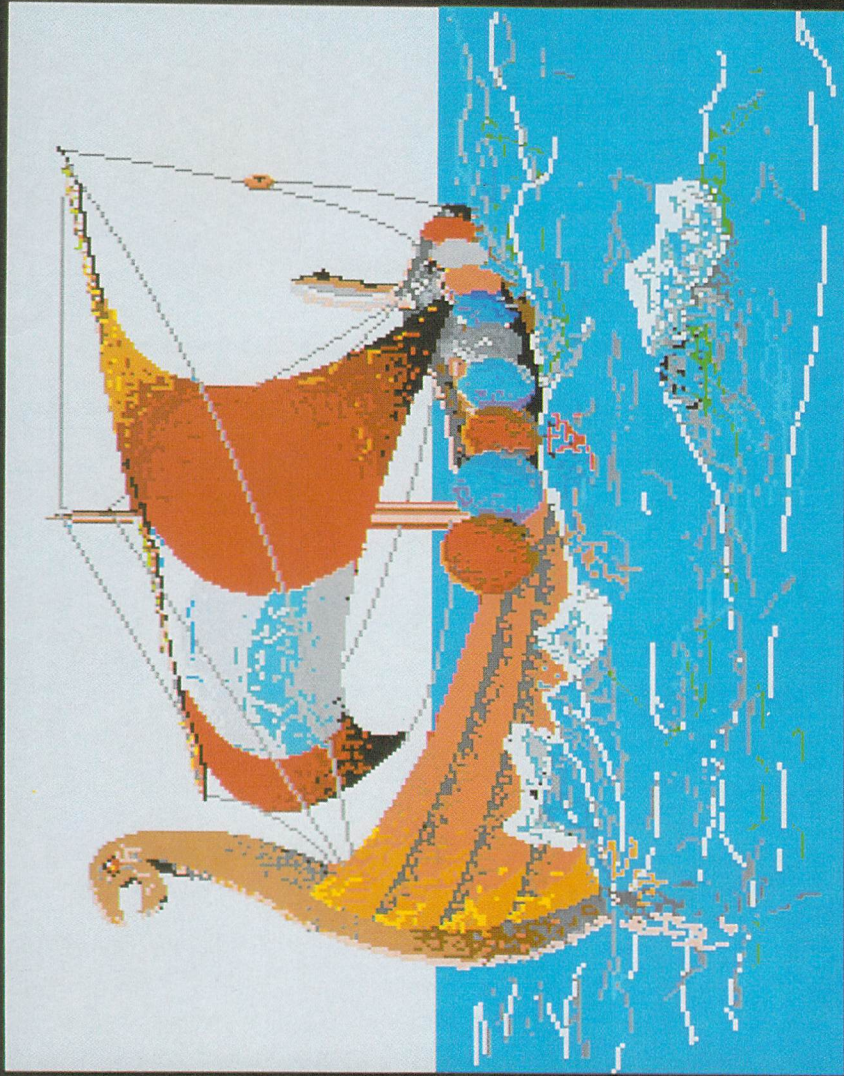


Beach House by Mark Rand, The Savannah College of Art and Design. (Lo res, Deluxe Paint)

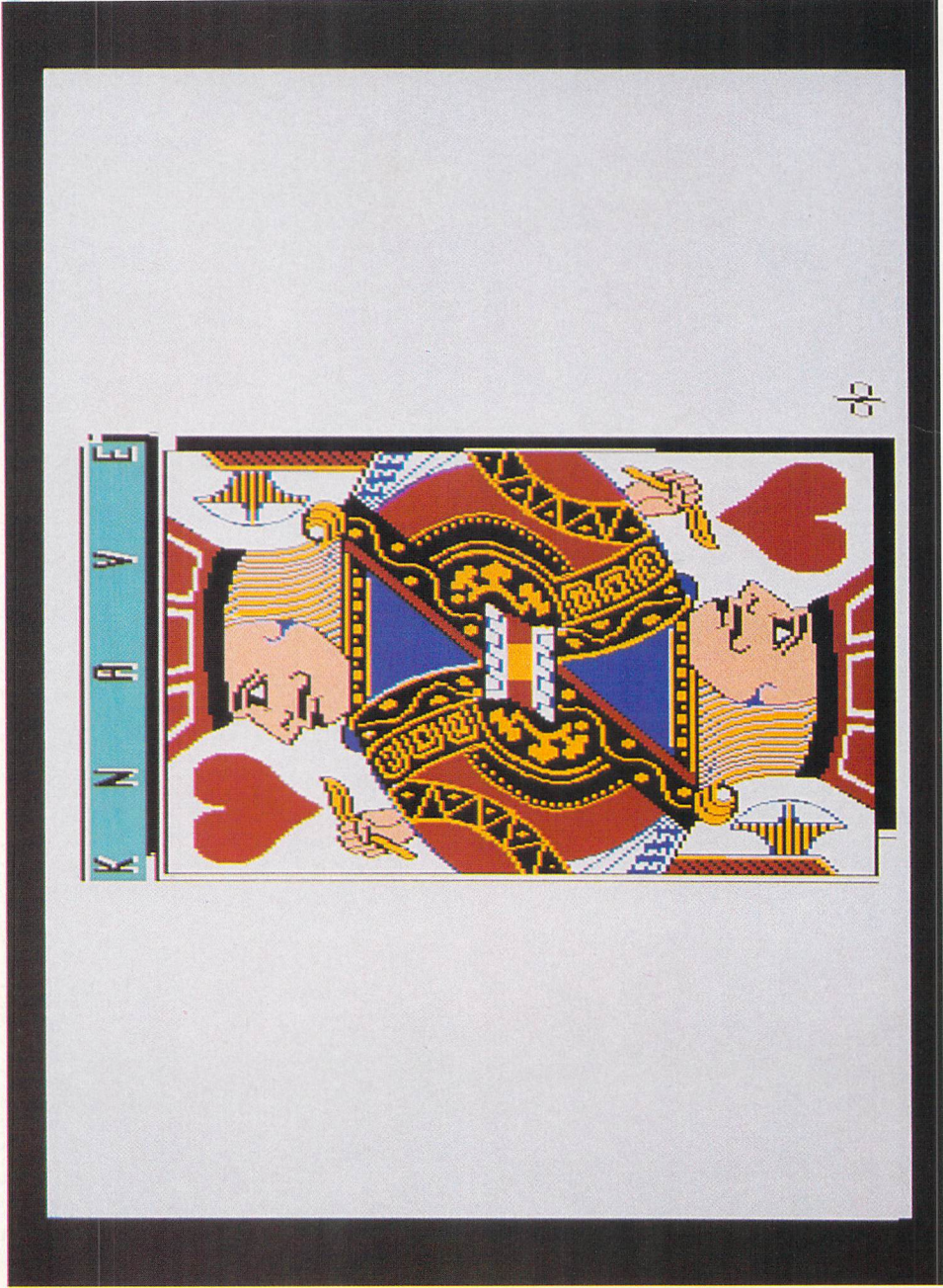


BMW by Gary Coccoluto, The Savannah College of Art and Design. (Lo res, Deluxe Paint)





Viking Ship by Jon Erikson, The Savannah College of Art and Design. (Lo res, Deluxe Paint)



Knave by Gene Brawn, graphic designer. (Med res, Deluxe Paint)

Barnard  
Cafe

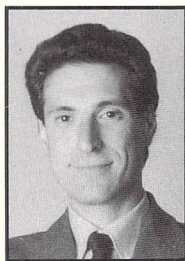
## Create Amiga Graphics You've Only Imagined Until Now

Whether you're using the Amiga for business or personal use, **Becoming an Amiga Artist** will help you achieve the results you want, quickly and easily.

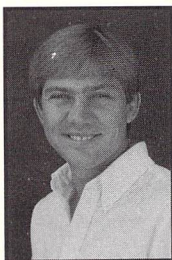
If you're using commercial software the illustrated, step-by-step instructions will help you create outstanding graphics with **Deluxe Paint, Deluxe Video, Impact!, AegisDraw,** and **Aegis Animator.** And, if you'd like to learn to do your own programs in AmigaBASIC, **Becoming an Amiga Artist** teaches you how to program graphics, sound, speech, and animation.

Learn to transfer images from the screen to paper, film, and videotape, create desktop videos, use AmigaDOS for file management, and many more tips, techniques, and shortcuts not found anywhere else.

With **Becoming an Amiga Artist** and your Amiga, you'll be able to give your artwork, advertisements, business reports, technical animations, and videos the professional look you've always wanted.



**Vahé Guzelimian** is President of EduComp, a computer service company providing instruction and consulting services to businesses. A resident of Cardiff-by-the-Sea, California, he has written several books including **Becoming a MacArtist**, and has contributed articles to numerous publications including **MacWorld, A+ Magazine,** and **Nibble Mac.** He is also the author of more than a dozen technical manuals on both hardware and software topics.



**Norbert K. Kuhnert** has worked for 4 years in the development and support of computer graphics software. A resident of San Diego, California, he developed 3D graphics software for the real-time display of vectorcardiographs. He currently works at Megatek Corporation.



**Gia L. Rozells** is a freelance writer and programmer. A resident of San Diego, California, she has authored or co-authored four books, and published articles on a wide range of computer applications. She is also a literary consultant providing writing, editing, design, and marketing assistance for corporations and individuals.

SCOTT, FORESMAN AND COMPANY