

Authors: Susumu Aral

Hisaya Takahashi

Adapted from “Pasocon Anime”

with the permission of Computer Age Co., Ltd.

Copyright © 1983 by OFFKEN Co., Ltd.

FIRST EDITION

All rights reserved, No part of this book shall be reproduced, stored in a retrieval system, or transmitted by any means, electronic, mechanical, photocopying, recording, or otherwise, without written permission from the publisher. No patent liability is assumed with respect to the use of the information contained herein. While every precaution has been taken in the preparation of this book, the publisher assumes no responsibility for errors or omissions. Neither is any liability assumed for damages resulting from the use of the information contained herein.

Printed in Japan

CONTENTS

1. [WHAT IS PERSONAL COMPUTER ANIMATION 1](#bookmark20)
2. FROM HAND-DRAWN ANIMATION TO PERSONAL COMPUTER ANIMATION

Encountering digital images 2

[Why is Leonardo Da Vinci important today? 4](#bookmark0)

1. EQUIPMENT AND TOOLS FOR PERSONAL COMPUTER ANIMATION

Learning about the canvas (The TV Screen) 5

[Learning about the keyboard 7](#bookmark4)

[Learning about the pallette 9](#bookmark5)

[Basics of screen design 11](#bookmark6)

1. DRAWING BACKGROUND PICTURES 13
2. THE BASICS OF BACKGROUND PICTURE CREATION 14

[Drawing with straight lines (Plot, Draw) 14](#bookmark7)

[Drawing with curves (Circle) 16](#bookmark8)

[Drawing with bands (Box, Bar) 18](#bookmark9)

[Painting colors (Paint) 20](#bookmark10)

1. [DRAWING PICTURES: DESCRIPTIONS 22](#bookmark11)

[Improving by practice 22](#bookmark12)

[In the sea (Coral) 24](#bookmark13)

In the sea (Adding seaweed and beach flowers

to the picture) 26

DRAWING PICTURES (2):

In the evening sun (drawing palm trees on the beach) 28

[In the evening sun (drawing a scorching sun) 30](#bookmark15)

[In the evening sun (drawing a setting sun) 32](#bookmark16)

[In the evening sun (the setting sun tints the sea red) 34](#bookmark17)

In the evening sun (night has completely fallen and a

crescent moon shines) 36

[A futuristic city (drawing with free imagination) 38](#bookmark18)

1. DRAWING ANIMATION 40
2. [TECHNIQUES OF PERSONAL COMPUTER ANIMATION 41](#bookmark19)

[Painting colors 43](#bookmark21)

[Moving animation patterns 45](#bookmark22)

1. EXAMPLES (1)

[In the sea (drawing small fish) 47](#bookmark27)

[In the sea (drawing a tropical fish and a shark) 49](#bookmark25)

[In the sea (drawing a turtle) 51](#bookmark26)

In the sea (drawing a diver) 53

[In the evening sun 55](#bookmark28)

[Then draw a couple 57](#bookmark29)

[A futuristic town 59](#bookmark30)

[Now draw a hot-air balloon 61](#bookmark31)

[Add more balloons 63](#bookmark32)

[Burst the balloons! 65](#bookmark33)

1. MAKING MUSIC 67

[Creating music and sound effects 68](#bookmark34)

[Creating a song of the sea 70](#bookmark35)

[Creating a song of the evening sun 72](#bookmark36)

[Creating a song of a futuristic town 74](#bookmark37)

1. WHATS ANIMATION? 76

Making characters move a flying bird 77

[Human movement 79](#bookmark41)

[Moving images across the screen a sailing ship (Loc) 81](#bookmark42)

[An airplane in flight (Move-To) 83](#bookmark43)

[Shooting bullets (Move-Step) 85](#bookmark44)

[Movement of the background (Screen scroll) 87](#bookmark45)

1. MAKE A ONE-SCENE ANIMATED GAME 89
2. [INTRODUCTION 90](#bookmark46)
3. [CRISIS AT SEA/HOW TO PLAY 91](#bookmark47)

[Program technique 93](#bookmark48)

[Inputting the program 95](#bookmark49)

1. [SPACE ADVENTURE/HOW TO PLAY 98](#bookmark50)

[Program technique 100](#bookmark51)

[Inputting the program 102](#bookmark52)

1. MAKE FOUR-SCENE ANIMATED GAMES 105
2. [INTRODUCTION 106](#bookmark53)
3. [CRISIS AT SEA-CONTINUED/HOWTO PLAY 108](#bookmark54)

[Escape from the bottom of the sea 110](#bookmark55)

[Scene 3: Reaching shore 112](#bookmark56)

[Making the game 114](#bookmark57)

[Inputting the game 115](#bookmark58)

1. SPACE ADVENTURE

How to play: Scene 1 (Launching)

Scene 2 (Meteorite shower) How to play: Scene 3 (Invaders) I Scene 4 (The warp))

[Program technique 122](#bookmark59)

[Inputting the game 124](#bookmark60)

1. ADD DRAMATIC MUSIC 127

[Music patterns: How to make music 128](#bookmark61)

[Music for sadness or danger sad music 130](#bookmark62)

[Music expressing danger 132](#bookmark64)

[Music for victory melody 134](#bookmark65)

[Victory music (with harmony) 136](#bookmark66)

[Cheerful, happy music 138](#bookmark67)

1. THE WORLD OF GRAPHICS SIMULATION 140
2. PATTERN GRAPHICS-DRAWING

GEOMETRICAL PATTERNS 141

[Drawing mosaics 143](#bookmark68)

1. [DYNAMIC GRAPHICS USING SPRITES 145](#bookmark69)

[Using characters 147](#bookmark70)

[Using multicolor mode 149](#bookmark71)

1. AFTERWORD

Recording the animation: photographing screen graphics 151

WHAT IS  
PERSONAL  
COMPUTER  
ANIMATION?

1

FROM HAND-DRAWN  
ANIMATION TO PERSONAL  
COMPUTER ANIMATION.

2

EQUIPMENT AND TOOLS  
FOR PERSONAL COMPUTER  
ANIMATION.

FROM HAND-DRAWN  
ANIMATION TO  
PERSONAL COMPUTER  
ANIMATION.  
ENCOUNTERING  
DIGITAL IMAGES.

There is a brand-new image technology called personal computer animation. But before we find out just what this is, let’s first examine examples of conventional technologies.

Images can briefly be classified into cartoons, illustrations, television or movie animation, and microcomputer games, and these are either moving or non-moving.

Cartoons on single sheets of paper do not look like they are moving unless these sheets are turned in rapid succession. For the cartoons to move, one must use a special method to create movement, changing several pictures per second.

One typical method used in television and movies is called cell animation. Pictures showing movement on a number of separate sheets are first traced on clear celluloid boards. The board with the tracing is then overlaid on a background picture, creating a complete scene called a cell picture. These cell pictures are then photographed; by displaying 24 of these cell pictures per second, the human eye detects movement.

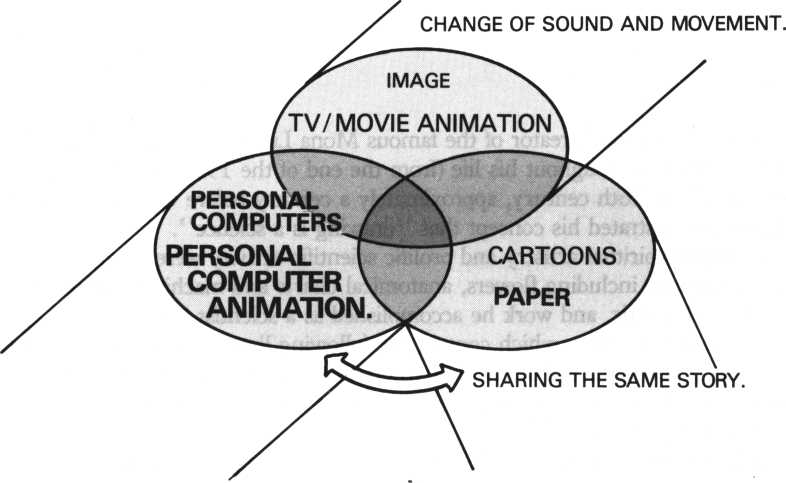
The main difference between personal computer animation and cell animation is one’s ability to communicate with the animation itself. Asking a question of a cell-animated image on the screen is impossible; however, with personal computer animation you can actually talk to your images!

Personal computer animation took its first step with microcomputer games. Although these are still very popular, you cannot draw and therefore create

pictures since the animation is only generated by the hardware (machine). You are limited to merely playing the games. Personal computer animation enables you to enjoy the creation of animated art, as well as actively participating with what is occurring on the screen and communicating with what you have created. You could call it a digital image (pictures consisting of points) encounter, images easily created with our software.

PERSONAL COMPUTER ANIMATION VS. CARTOONS.

■STORY ENDINGS DEPEND ON THE INDIVIDUAL (GAMES). ■MOVEMENTS OF NUMERICAL EXPRESSION GRAPHS (PATTERNS, CURVES).



* RAPIDLY CHANGING COLORS ANO SHAPES.
* INTERACTIVE METHOD STUDIES.

WHY IS LEONARDO DA  
VINCI IMPORTANT  
TODAY?

Leonardo da Vinci, creator of the famous Mona Lisa smile, was scientist as well as artist. Throughout his life (from the end of the 15 th century to the beginning of the 16th century, approximately a century before Galileo), Leonardo faithfully demonstrated his concept that “drawing is a science”.

His highly spiritual artistry and prolific scientific activity were a perfect match. Drawings including flowers, anatomical charts and machinery were both scientific and artistic, and work he accomplished as a scientist is recorded in a copious quantity of notes which contain the following lines. “Where there is no application of mathematical science, or where nothing can be combined with mathematics, there is no certainty”, and “Those who are not mathematicians cannot read my principles”.

Had a youthful Leonardo lived in this era of personal computer animation, his creative contributions would surely have meant even greater benefits for the world.

Science and art are now completely separate and almost no communication exists between them. In addition, science is itself divided into natural science and socio-cultural science, and communication between them is also lacking.

Today, computers can combine science and art in a revolutionary advance beyond traditional limits. The same logic is applicable to various fields in liberal arts which were themselves revolutionized by the technology of computer science, thereby overcoming the boundary between socio-cultural science and natural science, thereby enhancing their values.

New personal computer animation can be regarded as the combination of science and art, and the creation of additional benefits. Put another way, it is Leonardo da Vinci’s idea that “drawing is science”.

Few households lack television today, and many homes have video. Personal computers have become more popular in recent years and have become another video phenomenon. We are now entering an era where personal computer animation featuring an “animation interface” is coming into homes, applicable to a variety of fields, such as entertainment and education.

EQUIPMENT AND  
TOOLS FOR PERSONAL  
COMPUTER ANIMATION

LEARNING ABOUT THE  
CANVAS

(THE TV SCREEN).

You can use your home TV screen as a canvas for personal computer animation, and you can also use the television speaker for sound production.

Creating screen pictures is similar to the cell animation process. Overlaying several cell pictures on a background picture comprises cell animation. Personal computer (M5) animation has two main jobs: creating the background picture and the animation itself.

You can produce a background picture on the “backdrop plane” (boundary face) whose colors are changeable according to your desire. On this side of the backdrop plane, there is a “back plane” (background face) where you can write characters, symbols, and pseudo graphics, or specify the area to be used on the screen (view port). Here you also write BASIC programs.

There are 32 sprite planes or animation faces which correspond to several cell pictures, and you can produce one sprite (animation) on a sprite plane; therefore, 32 sprites can be created on 32 sprite planes. There are three sprite sizes: 8x8, 16 X 16, or 32 x 32 dots, and you can create a bigger sprite pattern overlaying these different-sized sprites.

Display priority numbers are assigned to sprite planes. Putting sprites in the same positions on the number 2 and 10 sprite planes will result in the sprite on the number 10 sprite plane to be hidden behind the number 2 sprite plane. This enables easy expression of each sprite’s movement.

Personal computer animation is called digital imagery, because the screen

consists of 256 horizontal dots and 192 vertical dots.

Characters and graphic patterns are displayed on the screen by coloring the dots. The number of dots representing a character is first determined, and this group of dots is called a pixel.

A pixel is a basic unit used to display a character or graphic pattern, consisting of 8 horizontal dots and 8 vertical dots in graphic mode, and 6 horizontal dots and 8 vertical dots in text mode (for characters). One pixel in semi-graphic mode consists of 4 horizontal and 4 vertical dots, and one screen consists of 64 horizontal and 48 vertical pixels. One pixel in full-graphic mode consists of one horizontal and one vertical dot. By specifying a color for each dot, you can draw a picture on the screen.

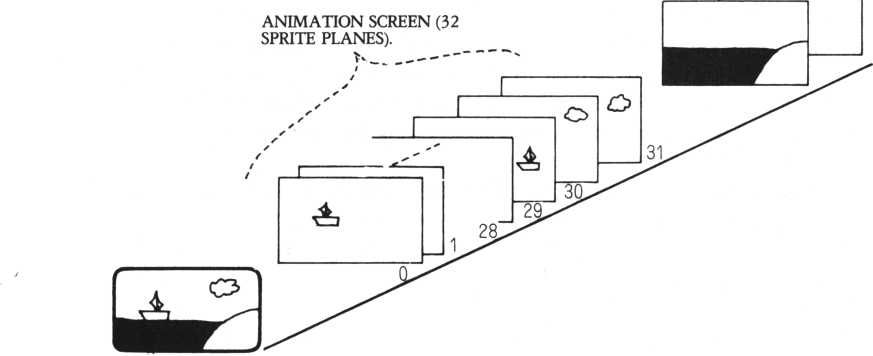
Specifying XY-coordinates of a pixel (in character units) or a dot determines picture location on the screen. The origin of XY-coordinates is located on the top left: X = 0 and Y = 0.

There are two standard screen page displays for convenience: turning on the power displays page zero, and simple key operation either displays page one or returns to page zero from page one. You can create amusing animation by using these various graphic functions.

THE COMPUTER CONCEPT BOUNDARY SCREEN (BACK-

OF CELL ANIMATION. DROP PLANE).

BACKGROUND SCREEN (BACK PLANE).



LEARNING ABOUT THE  
KEYBOARD.

If the screen is the “canvas”, then the keyboard can be compared to a brush. To draw pictures for personal computer animation, the characters input from the keyboard are all characters used by the personal computer.

Let’s look at the keyboard (see the page on the right). Numeric characters from 1 to 0 are arranged on the highest row and alphabetical characters are arranged on the second, third, and fourth rows. Positioned above these larger characters on the same keys are special and graphic characters.

To choose one of these characters, special keys on the right and left are used. Here we explain the basic features of the keyboard.

You cannot directly draw pictures on the screen for personal computer animation. You must use a language called BASIC-G. You must also specify patterns or movements similar to writing sentences on manuscript paper. Here are the methods using the brush to paint on the screen (canvas).

There are two ways of entering characters, a mode that overwrites existing characters and a mode that moves existing characters to the right to insert new characters.

To delete characters written by mistake, you can delete one or more characters at a time using the delete function (DEL key).

To move the cursor, there are several functions to move you quickly to an area which you wish to change. These are: moving back to the home position (top left); moving to the beginning of the current line; moving to the beginning of he next line; moving to the right; deleting 8 characters at a time; or moving to the same position on the next line.

Next, there is the convenient scroll function, which moves the display page up and down, or right to left.

Finally, there is a view port which divides the screen, so the cursor can be moved, or the page processed within the limited range on the screen.

By making full use of these functions, you can use the keyboard as a brush for personal computer animation.

CTRL: CONTROL KEY.

CONTROLS THE KEYBOARD

RETURN: RETURN KEY. SENDS COMMANDS TO THE COMPUTER.

FUNCTIONS.

FUNC: FUNCTION KEY.

DISPLAYS THE CHARACTERS

OR SYMBOLS TO BE WRITTEN.

RESET: RESET KEY. ENDS OR

RESETS THE SCREEN. DON’T

OVERUSE IT!

|  |  |  |
| --- | --- | --- |
| Gl? 0 G5 0 G5 5 0 © © S5 0 G9 | 50BBBDDD CE | ]  3~ |
|  |  |
|  | |

SHIFT: SHIFT KEY. DOUBLES THE NUMBER OF KEYS. SAME -1 AS LEFT

SHIFT: SHIFT KEY. DOUBLES THE NUMBER OF KEYS. SAME AS RIGHT.

SPACE: SPACE KEY. PLACES SPACES ON THE SCREEN.

LEARNING ABOUT THE  
PALLETTE.

The language called BASIC-G is comparable to a pallette for personal computer animation. BASIC-G is BASIC-I with enforced graphic and sound functions.

BASIC-G executes most commands by a high-speed interrupt process to the hardware. Inputs from the keyboard or joypad are accepted, while sounds are produced or pictures are drawn. You can also enjoy personal computer animation for games.

The sprite function used as the animation pattern is operated by the FOR- NEXT command in BASIC-I, but in BASIC-G you can freely move sprites by specifying speed and direction using the MOVE command. The JOINT command easily creates a larger animation pattern. This sprite function makes the still image appear in motion. For general graphic functions, examples are the CIRCLE command which draws circles or ellipses, the LINE command for lines, and the PAINT command for coloring pictures.

There are some unique, convenient graphic functions which other personal computers do not have, such as the BAR command, for drawing bars, and the BOX command for boxes.

By connecting an exclusive printer to the M5, the reverse forms of characters in bit image can be printed out. A screen copy command can print out the image copy on-screen. Considering the price, reproducing a screen image on hard copy is exceptional. You can also save the screen data of your animated creation on a cassette tape and easily redisplay it at any time. Sixteen beautiful colors are available, and movements and commands which require machine language for conventional personal computers can be used in BASIC for BASIC-G.

The sound function necessary for personal computer animation background music is even better. An OUT command is necessary in BASIC-I to generate sound. However, anyone can easily produce sounds using the PLAY command in BASIC-G, and also change the sound length, strength, tempo, major/minor key, rests, and chords of a composition. The music you compose can be automatically played, and by using musical notes as data, even a trio can perform Beethoven or Schubert.

Sound effects from an SG command can generate synthesizer-like sound, and you can even create a wide range of sound effects such as waves, songbirds, the wind, and airplanes.

Another function of BASIC-G is a LIST command which integrates programs. BASIC-G has a larger memory than BASIC-I, which enables you to easily create

computer animation.

In BASIC-G, you can move animation patterns using the sprite function, and easily create personal computer animation on a background picture generated by full graphics. You can also devise sound effects or compose music to match your creations.



BASICS OF SCREEN  
DESIGN

There are two kinds of design paper used to create personal computer animation: screen layout paper and character pattern paper. The former is used for background pictures, and the latter for animation patterns.

The screen layout paper is design paper for the background pictures of personal computer animation. This paper has horizontal and vertical lines in units composed of dots and characters over the screen’s lines of resolution. If these change, the paper changes accordingly. You design the background picture on this paper by finding the dot locations of the XY-coordinates used by the CIRCLE and LINE commands of the program.

The character pattern paper is design paper for character pattern data of 8 X 8 or 16 X 16 dots used for characters or sprites set by changing character patterns to character codes using the STCHR command. Sprites designed on this paper move as animation patterns. Combined sprites can move in animation patterns of 32 x 32 dots. Basic pattern size is 8 x 8 or 16 x 16 dots; 32 + 32 dot patterns are regarded as collections of four 16 X 16 dot patterns. However, there is a hardware limitation; you cannot arrange more than four adjacent sprites at a time. Doing this will cause the sprites to disappear, so remember this when you design sprites.

Now let us explain the process of creating personal computer animation.

1. Sketch the personal computer animation on an ordinary piece of paper.
2. Draw the background picture based on the sketch on the screen layout paper.
3. Write a program of BASIC lines based on the screen layout and display it on the screen. If the image displayed on the screen differs from your own, modify the program.
4. Color the picture based on the screen layout. Try different colors and then choose. Now, the background picture is complete.
5. Decide animation patterns for sprites and design these on the character pattern paper.
6. Write a program for the sprites designed on the character pattern paper.

Modify the sprites if necessary, repeating the process for each sprite. Then move the sprites by simulation.

1. Combine the background picture and animation pattern together, and observe

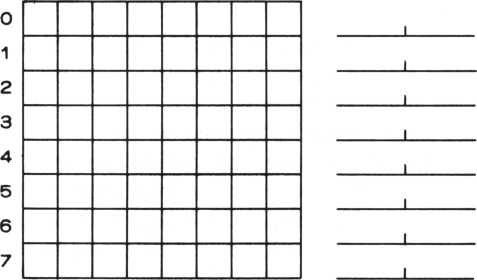
the movements. If necessary, change the location of the animation pattern position or make any other necessary changes.

1. Personal computer animation is now complete. Run the program to see the animation.

Good design is the key to the creation of successful personal computer animation. Even if the background pictures are well made, ill-conceived animation patterns will spoil the whole. Using the design paper is important.

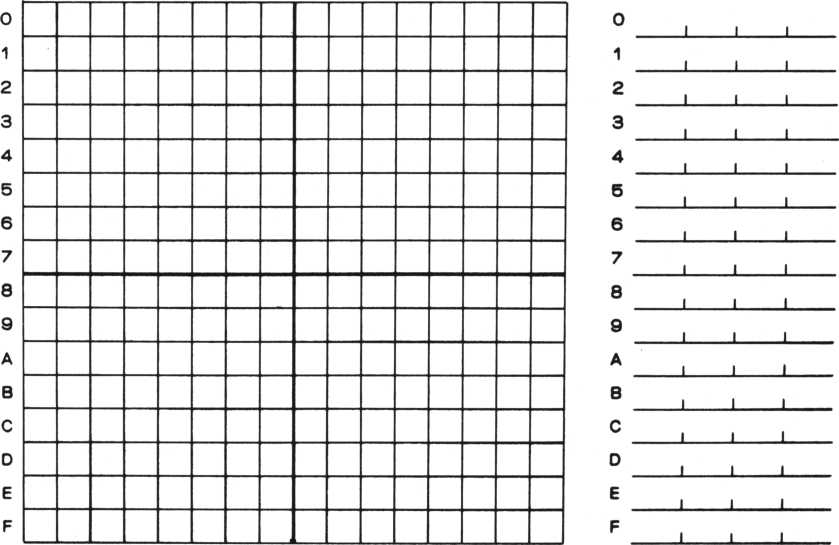
CHARACTER PATTERN PAPER (8x8 DOTS).

0 1 2 3 4 5 6 7



CHARACTER PATTERN PAPER (16x16 DOTS).

01 23456789ABCDE F



II

DRAWING

BACKGROUND

PICTURES

1

THE BASICS OF  
BACKGROUND PICTURE  
CREATION.

2

DRAWING PICTURES.

THE BASICS OF  
BACKGROUND PICTURE  
CREATION.

DRAWING WITH  
STRAIGHT LINES  
(PLOT, DRAW).

PLOT is the command used to display a dot on specific coordinates on the screen. The display of a few dots over coordinates totalling 49,000 or more may not only appear insignificant, but unrecognizable in terms of color. There are other possibilities; as shown in DEMO-1, PLOT can change the display into a Milky Way-like image by displaying many dots, utilizing a few rules. PLOT can be rewarding as a command.

DRAW is a command used to draw a straight line between two specified coordinates. This command is the easiest among the graphic commands. Converting a picture into coordinates draws the picture on the screen. In DEMO-2, coordinates are converted into DATA statements and the picture is drawn by reading these statements. Changing the colors of some dots can show distance. The DRAW command is both practical and useful.

10! ### DEMO-1 CPIot D\*»o> ###

28 lor C 0 = 2 to lSsrandoNtize 38 Print "ISM" s 9 i n i t : b co 1 &81

4 8 for P = 8 to 18 8 step 25 58 for to 1 2 5 s B \*.X / 2 8

68 on B+l 9 o t o 7 0 > 8 8 » 9 8 » 10 0, 118,128? 138 70 V ®r nd< 9) 5 9 o t o 14 8 88 V\*rnd<l4)sgoto 148

90 V = r n d < 2 8 > : g o t o 14 0 10 0 V \* r n d < 3 2 ) s g o t o 14 0 110 V = r n d < 42) i 9 ot o 140 1?0 V = r nd (50) s 9ot- o 140 130 V-rnd < 5 5 )

140 fcoi COS P}ot X, 180-V:X, 180+V? 250-X, 100-V? 250-X, 100 + V

15 0 next X, ft

160 sleep 5,68

170 next CO

180 goto 28

10! ### DEMO-2 (Draw Demo) ###

20 Print "ISMT s g i n i t s b co \ &81

30 fcol &07SXZ=15:VZ\*25:9osub $ D L I N E

40 fcol &0CsXZ=0sV2\*40sgosub 9DLINE

58 fcol & 8 Pi s restore 120 \*XZ«-45sVZ\*100s9osub $ D LIN E 68 goto 68 70$DL I HE

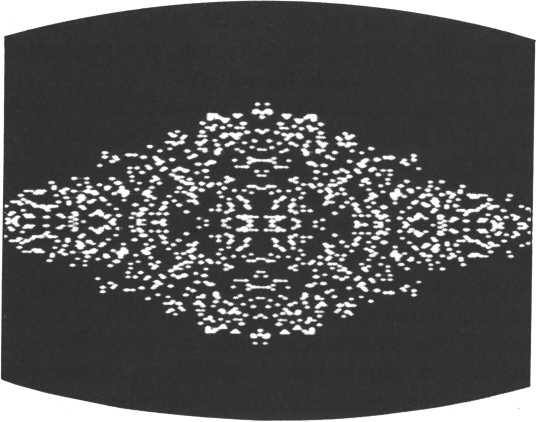
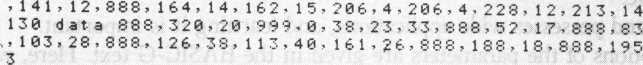
80 read Xfisif X8=888 then X1«X2sV1«V2ire«d X2VV2\*9oto 110 90 i i X ft - 9 9 9 then return 10 0 read V 1 , X 2 , V 2 s X 1 « X Pi

110 draw Xl+XZ,Vl+VZ,X2+XZ,V2+V2:goto $DL1NE

120 data -20,28,17,14,888,40,24,25,13,65,10,888,148,30,99,18

240, 1 0 32,70,34 23,191,2

140 data 198,21,888,233,33,888,255,48,999



DEMO-1

DRAWING WITH  
CURVES (CIRCLE).

The CIRCLE command is the most applicable among the graphic commands. It is basically used to draw circles, but you can specify many parameters such as semidiameters in the X-direction, semidiameters in the Y-direction, increments, starting angles, ending angles, angles of inclination, and sectors. There are many of them, and learning the definitions of these parameters and their uses is a definite challenge.

This command possesses wide applications, and learning it is important. Detailed descriptions of the parameters are given in the BASIC-G text. Here, we explain its use.

When pictures are drawn on paper, most consist of curves. However, pictures generated by DRAW commands consist of straight lines. They look as though they were drawn by a ruler, and feel mechanical, artificial, and cold. Although DRAW provides you a big advantage when creating personal computer animation, it can be disadvantageous according to the pattern.

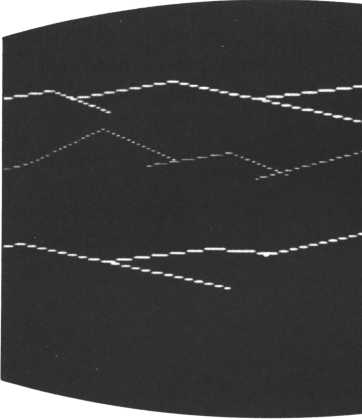
We often desire the expression of natural images in computer-drawn pictures. In these cases, the CIRCLE command is very useful. DEMO-3 shows the sun and clouds. The CIRCLE command enables the expression of natural, soft clouds, and this is accomplished by using parts of ellipses for the clouds. For the beginning and ending angles, we specify ordinary angles rather than the circular measure (pi), which is highly convenient.

If you want to draw a curve closer to a straight line (e.g., the horizon), you can use parts of a large semidiametric circle (300 to 400), or an ellipse.

We have explained dots, straight lines, and curves, which are used to draw pictures and their corresponding graphic commands, using several examples. There is an M5 utility program called P-Editor which draws pictures without writing programs; however, you have to work a bit harder with the dot units.

10! ### DEMO-3 (Circle Demo) ###

20 Print **"BMP's\*** in it sf col &8F **30** groove 40,160!circle 30,25,,30,388 40 groove 7 0 , 14 5 s c i r c 1 e 30,20?,210,330 50 9 m o v e 70,165scircle 30,20,,10,120 60 groove 118,150scircle 25,20,,218,480 78 groove 150?70Scircle 38,25,,30,300 80 groove 188,55scircle 30,28,,218,330 90 groove 180,75scircle 30,28,,10,120 100 groove 228,60:circ1e 25,28,,210,488 110 fcol & 0 8 s groove 78,25scircle 25 120 goto 120



DEMO-2



DEMO-3

DRAWING WITH BANDS  
(BOX, BAR)

The BOX command draws a rectangle (box) using two specified XY-coordi- nates as diagonal points. The BAR command draws a band (bar). These two are closely related, and the relationship between them is shown by the expression, BAR = BOX + PAINT.

Using those commands enables you to draw very abstract pictures. «DEMO-4» is a picture of the M5.

The BAR command can express colors and shapes which were not clearly expressed by the PLOT command; the BAR command can be used instead of PLOT.

10! ### DEMO-4 (Box Demo) ###

20 Print "lap” s 9 i n i t ! b co 1 £01 if cot 80 B

30 for 1 = 1 to 10:read Xliif XI =999 then fcol 803:read XI 40 read V1>X2,V2:box X1,V1,X2,V2inextifco1 8BF

50 N=13 s Xfi=14:XX=10i VV=12:X= 1 4 iV=105i 9osub \*DB0X:X = 5i V\*120:9 osub $D B 0 X

60 X = 9i V=135!9osub $DBQX!N =11i X = 3 3i V«150!9 osub $DBOX 70 N = 4! Xfi=10: XX = 8: V V = 4 : X = 6! S' = B4! 9 osub \*DBOX 30 9oto 80

100\*DBOXi for 1 = 1 to NiX = X + X fl 110 box X,V,X+XX,V+VYinext:return

120 data 4, 10,2 50, 190 ,5, 12,24 8 ,65, 125,43,23 3,58,7.90.24 5, 175

130 data 999.216,165,226. 1 17,23,150,39, 162,204. 120,226, 132,2 10,135,226,147,205,150,221,162,198,95,225,99

10 1 ### DEMO-5 (Bar Demo) ###

20 Print "HM" i 9 i n i t ! b co 1 801!VV=8

30 V = 20!repeat i VV = VV+10:V = V + VV:C = C+1 46 for X=10 to 250 step 8

50 if C mod 2 = 0 then fcol &0fi else fcol 806

60 bar X,V,X + 2,V + 3:nextiunti1 V> = 188 X = 0iXO = 0:X1=80: X2= 1 80 70 for V=180 to 10 steP-5:X=X+7iX0=X0+12!Xl=Xl+5:X2=X2+3 88 fcol 807! if V> = 110 then bar X0,V- 100,X0 + 2,V-97 90 fcol 803! if V> =38 then bar X,V-26,X + 2,V-17

100 fcol 8 6 D s b a r X1,V,X1+2,V + 3: fco1 60Fibar X2,V.X2 + 2,V + 3:ne

':(t

110 fco 1 &03:bar 10,85.38, 1 45:fco1 8.85:bar 45,28.75.185« <co1 8,07 i b ar 1 95,78,215,1 18Mco1 80C

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 120 | b ar | 140, | 10,155, | 4 5 M c o 1 & Q 4 : b a r | 230. 148,255, 198: < co 1 |
| b a>\* | 5 , 25 | , 20, | 65sfcol | S:04: bar\* 230, 140 | ,255•198 |
| 130 | f col | &06 | :bar 5, | 25,20,65 |  |
| 140 | 9 O t O | 1 4 0 |  |  |  |

PAINTING COLORS  
(PAINT).

The PAINT command paints inside the boundary of the picture, including the coordinates specifying the boundary. Specifying a color for the boundary paints the interior of the boundary with that color.

Painting a picture with the most suitable colors will make what you draw appear real, or clarify shape. Shading colors expresses distance. DEMO-6 shows distance by colors used to paint the mountains drawn with straight lines. The mountains are clearer than before their painting.

In DEMO-7, the sun and clouds drawn by curves are painted; the sky is blue and the clouds appear clearer.

«DEMO-5» is a city at night viewed from a hill. Coordinates used by the BAR command are changed gradually by the FOR-NEXT statement, so that small rectangles are aligned to light the path.

To express distance, the width of the horizontal path is wider on the near side, and the directions of the vertical paths are changed to cross at the far end. These expressions can be accomplished only by changing the increments of the variables, displaying the unique technology of the personal computer.

DEMO-8 is a painted picture of the M5 drawn by the BOX command. The M5 appears realistic. Here, the keys were painted using the PAINT command but using the BAR command would have produced the same results. (Execute each program by adding it to each DEMO program by command.)

40 9oto 140

120! ### DEMO-6 < Pa i n t Demo 1 ) #\*#

130! ## with Draw Demo ##

140 fcol &0?s Paint 28,10 150 fcol &0C s P a i n t 20,45 168 fcol &02s Paint 20,80 170 fcol &8B sPaint 28,158 180 restore 9 8 s 9 o t o 38

120 9 ot o 150

130! ### DEMO-7 < Pa i n t Demo 2> ###

140! ## with Circle Demo ##

150 fcol &65«Paint 1,1 160 fco1 &0F:Paint 180,60 170 Paint 60,150 180 fcol &88sPaint 70,25 190 9 ot o 190

80 9 o t o 150

130! ### DEMO-8 (Paint Demo 3> ###

140! ## with Box Demo ##

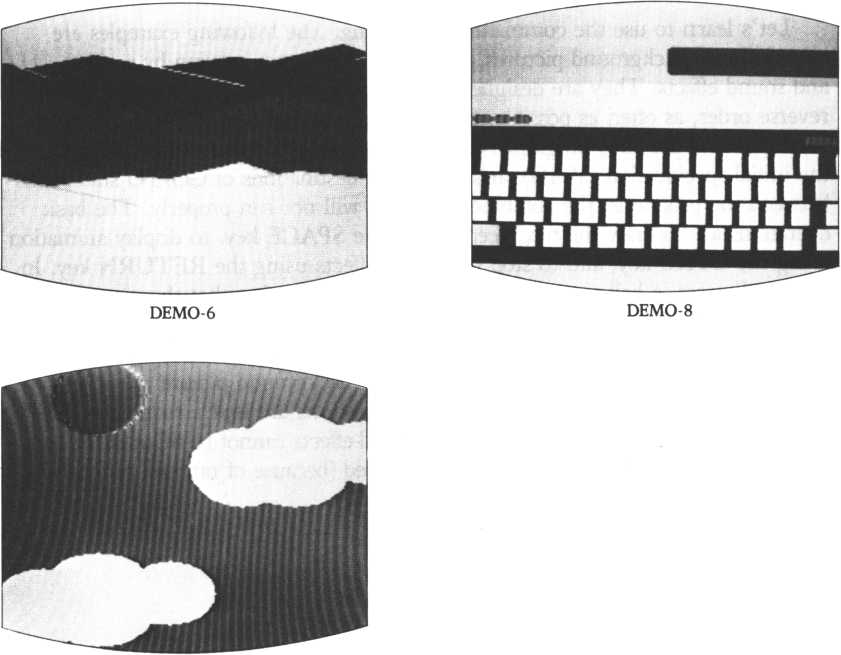
150 fcol &8ft: Paint 10,285 fcol fcOCs Paint 126,44s fcol &8Espain t 10,80

160 Paint 217, 186 s Paint 24,151s Paint 285,121 s Paint 211,13 6 s P aint 286,15lsfcol &06spaint 199,96

170 fcol &0FsN=13s Xfl«14iXX\*105VV»12\*X»148Y\*18559 osub fPBOXs X = 5 s V—1205 9 o s u b $ P B 0 X

180 X = 9 sV=135 5 9 osub fPBOXs N = 1 1 5 X = 33 5 V\*158s9osub SPBOX 190 fcol &045N=4;Xfi=10sXX=85VV=4sX=6sV=845gosub \*PB0X 280 9 ot o 200

218$PB0X5 for 1 = 1 to Nsx = X + XR 220 Paint X+l,V+ls next s return



DEMO-7

DRAWING PICTURES:  
DESCRIPTIONS.

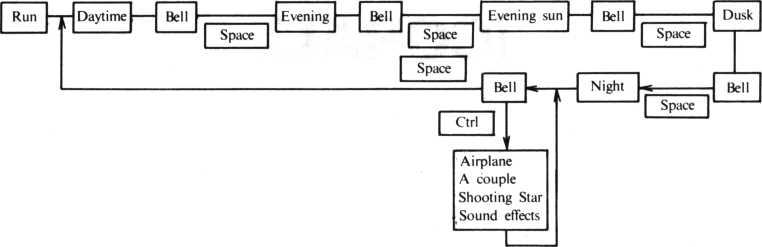
IMPROVING BY  
PRACTICE.

Let’s learn to use the commands by drawing. The following examples are classified into background pictures, animation (characters moving by sprites), and sound effects. They are designed so that they can be input separately or in reverse order, as often as possible, on the condition that they are added to the corresponding previous programs. In such cases, unless you carefully perform the initial specification (MAG), and define the destinations of GO/TO statements between the programs, the combined program will not run properly. The basic operations are to draw the next scene using the SPACE key, to display animation using the CTRL key, and to stop the sound effects using the RETURN key. In almost all cases, a bell rings (CTRL + G sound) indicating that the computer is awaiting input,

1. In the sea:

This program is very long, so a part of the background picture must be deleted when animation is input. Before sound effects are input, redundant REM statements must be deleted. However, all sound effects cannot be input at the same time. The shark and turtle may not be displayed (because of random numbers?).

1. The evening sun:



1. A futuristic city:

After you finish drawing the background picture, the bell rings. By pressing the CTRL key, the animation and sound effects are produced in order. There is no need to key-in until the bell rings again.

The pictures of this program use up all memory capacity. To input the sound efects, all REM statements must be deleted, but the sound effects still cannot be input together at the same time.

Pressing the RETURN key stops the sound of waves (here the timing is  
difficult) and rings the bell. Move to the next page by pressing the SPACE key.

IN THE SEA  
(CORAL).

The sea is a clear blue and coral stretches endlessly. A white belt of breaking waves surround a south sea island. Having departed the airplane, you strip off your clothes, run down to the white hot beach and enter the blue sea. Around the table-shaped coral, a group of small fish swim. Some red beach flowers are on the steep cliff, but you realize that you are a long way from the beach. When you decide to return to shore, something dark moves beneath you. It’s a shark!

Let’s draw the picture of the mysterious south sea. It looks rather dark, because the background color is black and there are neither seaweed nor flowers. Let’s add them to the picture.

10' ##### DIUER #\*###

1. © Print M l?W : 9 i n i t t b c o 1 St © 1 s m a 9 2
2. 0 re st ore 310: tc o1 & 8 ft \* 90s u b $D LINE:Paint 8,4 58ft : f o \*u b $D

LINE\* fcol &8D:Paint 45>2>s \*coJ I6F:9osub $D L I N E s fcol &8E 40 Paint 78 \* 2 ,&8ft , &8D ,&8F : 9 osub IDLIHEi fcol &8D: Paint 55,98, & 8 ft > & 8 E < & 8 F : 9 osub \* 0 L i HE : f col 0,8 E : P a i n t 8 8 , 178,&0D

50 f c o 1 & 8 D s 9 o s u b $ D L I N E s P aint 2 4 8, t 8 5 , & 8 ft , & 80s f c o 1 £ 8ft: 9 os u

b $DL I NE:pa i nt 2 4 8, 1 3 5,& 8 ft,% 8 D 6 0 restore 3 5 8:9 os ub i0 0 I R 78 9osub tSEftWEEO 88 Print "WP

9 8 J » i n k e y < 1 > : i 4 J = 1 then 9 o t o 188 else 9 o t. o 9 8 1 8 81 ##### 0 Rft W CIRCLE #####

1. 9 8 $ D C I R
2. 8 8 read C1: i f 01=999 then return 218 read X,V,C2,C3

228 icol C1:9 m ou e X,V:c i r c1e 17, 16, > 188,360, , is fcol 6 s P a i n t X,V-2,C1: f co1 C2 8circle 14,13,, 188,368, , 1 238 fcol C 3 s Paint X,V-3,02:9 o t o 288 248! ##### D R ft M LINE #####

2 5 0 $ 0 L I N E

268 read Xflsif Xft = 888 then X1=X2:V1=V2:read X2,V2:9oto 298

2 7 0 if X ft \*9 9 9 then return

288 read V1,X2,V2:X1=Xfl

290 draw X1,V1,X2,V2:9oto 268

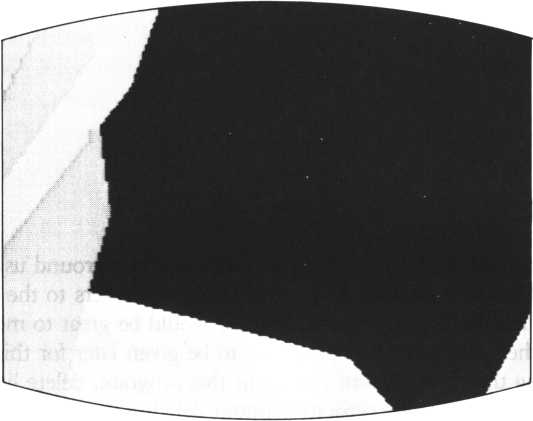
—, it, m i ti « a a « I T N F riQTQ a a a a a

310 data 30,0,36,16.888,12.72.888,0,184,999,52,0,60,25,888,0 ,120,999,84,0,72,40,888,0,145,999

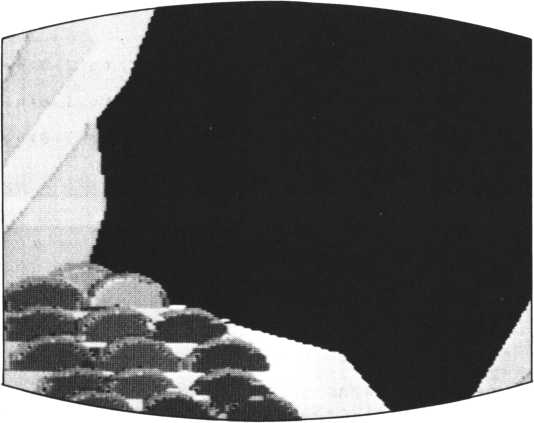
320 data 68,57,64,96,888,55,130,888,44,199,999,55,130,155,16 0,888,184,199,999

330 data 2\*1 0 . 1 99, 160.236.888,244, 155,888,255 ,140,999, 195, 199 '232,136,888,255,115,999 340! \*\*\*\*\* CIRCLF DATA #####

350 data 6,56,136,3,3.8,40,142,2,2,2,72-140,3,3,8,39-155,2,2 ,12,68, 155,2,2, 1 1,95, 155,3, 12,8,44, 168,2,2, 1 2,76, 1 6,8.2,2 360 data 11,110,168,3,12,8,54,182,2,2,12,77,182,2,2,11,111,1 82,3,12,12,60,192,2,2.11,91,192,3,12,12,123,192,3,12,999



IN THE MIDDLE.



HALF-COMPLETED.

IN THE SEA (ADDING  
SEAWEED AND BEACH  
FLOWERS TO THE  
PICTURE).

To lighten the sea, add list 2. Here, we paint the background using a STCHR command. Lists 3 and 4 are used to add seaweed and flowers to the picture. You can easily draw fine lines using sprites. Since it would be great to move the fish in the picture, there are additional programs to be given later for this purpose, as well as animating the diver. Before you input this program, delete lists 2 and 4. (There is not enough memory capacity without deleting them.)

1. erase:HI=HI+1:if HI=5 then HI=1
2. on HI goto 30-24,25,26

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 24  38 | for | I =0 | t. 0 | 255sstchr | H040C040C040C040CM | t 0 | I»8Snexti9oto |
| 25  30 | for | 1-0 | t o | 255s stchr | “0401040104010401“ | t o | I> 8\* n ext s 9 ot o |
| 26  30 | for | 1\*0 | t o | 255s stchr | “0785870507050705” | t o | 1» 8s next s 9oto |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 70 9osub \*SEPWEED | | |  |  |
| 1000\*SEAWEED | | |  |  |
| 1020 | st ch r | M 180807c0380?0800,< | t o | 148 |
| 1030 | stchr | “ 7 f 80000 f 38400801 | t 0 | 1 4 1 |
| 1040 | stchr | “ 8 0 6 0 1 0 8 8 6 4 l 2 c a 3 5 “ | t o | 142 |
| 1050 | stchr | “1b 7 f 0 7 ff8f748810“ | t o | 143 |
| 1060 | st ch r | “710887c038177800“ | t o | 144 |
| 1070 | s t c h r | ”7f88380f30478833“ | t o | 1 45 |
| 1080 | stchr | “8864148c6412ca35“ | t o | 146 |
| 1090 | stchr | “lb7f07f f0f748810“ | t o | 147 |
| 1180 | stchr | “0204881324485851“ | to | 148 |
| 1110 | stchr | “5?a4a4a8cad4fbfc“ | t o | 149 |
| 1120 | stchr | “1068830cl027c810“ | t 0 | 158 |
| 1130 | Stchr | «214e70883fc83884“ | t o | 151 |
| 1140 | stchr | “0204881324485051“ | t o | 152 |
| 1150 | stchr | “52a4a4a8cad4fb fc“ | t 0 | 153 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1168 | stchr | "■ 1 060838c 1 027 c8 10 \* | | 1 to 154 |
| 1170 | stchr | "214e?0803fC03804 | | \* to 155 |
| 1188 | SC-28: | for | CD3514 8 to 152 | steP 4 |
| 1 1 98 | S C = S C + 1:scod SC,CD:scol | | | SC,& 8 C |
| 1288 | next |  |  |  |
| 1210 | loc 21 | t 0 | 212,140 |  |
| 1228 | 1 oc 22 | t o | 198,175 |  |
| 1230 | loc 23 | t o | 151,147 |  |
| 1248 | loc 24 | t 0 | 58,52 |  |

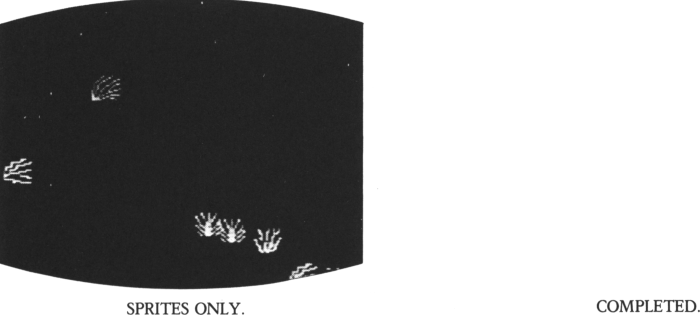
|  |  |  |  |
| --- | --- | --- | --- |
| osub SFLOWER i$FL0WER  i stchr •‘000888444422221 1 “ | | t. o | 200 |
| i stchr | ■ 1 149251309251309” | t o | 20 1 |
| i st ch r | ”0088839098a2a244” | t. o | 282 |
| i stchr | ”c54952e449d264c8“ | t 0 | 283 |
| i stchr | ”0008084444222211” | t. o | 284 |
| i stchr | " 1 149251389251309” | t. 0 | 285 |
| i stchr | ”0838889090a2a244” | t o | 286 |
| i stchr | “ c54952e449«i2 64c8” | t o | 287 |
| i stchr | ”00000 f 080979435e” | t 0 | 288 |
| i stchr | ”909f f0c080fC0706” | t o | 28 9 |
| i stchr | ”7868c31ef008008f" | t 0 | 218 |
| i stchr | ”f880407c0708c07f” | t 0 | 21 1 |
| i stchr | ‘\*08888 118097 9435 e ” | t 0 | 212 |
| i stchr | “909 ff0c880fC0706” | t 0 | 213 |
| stchr | “7860C31ef000800f" | t 0 | 214 |
| stchr | “ f 880407c0708,c07f ” | t. o | 215 |
| SC\*24: | for C D « 2 8 8 to 212 | step | 4: |

S C = S C +l:scod SC,CD:scol

SC , $=088 next

2180 loc 25 to 128, 142:1oc oc 28 to 10, 185!return

26 to 1 H5» 1 38 s 1 OC 27 to 178, 1 72:1



DRAWING PICTURES (2)

IN THE EVENING SUN (DRAWING PALM TREES ON THE BEACH).

The former pictures do not change the scenes; only the background colors change. Here, let’s create a picture which changes gradually.

List 1 is sprite DATA on three palm trees on the beach. The central palm tree (palm tree 1) is displayed on the screen by the MOVE command using 4 sprites: the leaves (SCOD 0) using a LOC command; and the trunk on 3 sprites (SCOD 1-3) using the the JOINT command. 1

1290 stchr "\*2\*8dcd\*©\*622880" to 119 1300 ?cod 4,116:scol 4, 1 1310! \*\* PALM TRUNK 2-A \*\*

1320 stchr H 0 0 00 000 0 0 0 0000 0 0" to 120 1330 Stchr " 00000000000000 00 11 to 121 1340 stchr “0203030303 07 07 06 \*\* to 122 1350 stchr "060606068©0e0c0c" to 123 1360 scod 5» 120:scol 5,1 1370! \*\* PALM TRUNK 2-B \*\*

1380 stchr “0000000008000000" to 124

1390 stchr “0000000000000000" to 125

1400 stchr "0c0c0c0c0clclcl8" to 126

1410 stchr “1818181818181818" to 127

1420 scod 6,124:scol 6,1 1430! \*\* PALM LEAF 3 \*\*

1440 stchr "01lb3\*7\*\*\*a\*8b 1 \* " to 128

1450 stchr “1\*7\*4\*1\*38383010" to 129

1468 stchr “c09cbe\*e\*6el\*0\*c“ to 130

1470 stchr “bc7elel\*87clc0c0" to 131

1480 scod 7,128:scol 7,1 1490! \*\* PALM TRUNK 3-A \*\*

1500 stchr "0000000000000000" to 132

1510 stchr “0000000080000000“ to 133

1520 stchr “6070303038381818“ to 134

1530 stchr "181818181 clc0c8c“ to 135

1540 scod 8, 132:scol 8, 1 1550! \*\* PALM TRUNK 3-B

1560 stchr "0080000000080000" to 136

1570 stchr "0008000000000000“ to 137

1580 stchr "0c8c0c0c8c0e0e06" to 138

1598 stchr "0606860606060606“ to 139

1680 scod 9,136:scol 9,1

1. 10 m a \*3 3 : J o i n t 3 to 2,3 : J o i n t 2 to 1,3 : 1 o c 8 to 1 5 8,3 8 : rr« o o e 1 in 8 on 166,65

16 28 Joint 6 to 5,3: 1oc 4 to 190,67:mooe 5 in 1 on 177,95 16 30 Joint 9 to 8,3: 1oc 7 to 9,55:moue 8 in 5 on 9? 87:return

IN THE EVENING SUN  
(DRAWING A  
SCORCHING SUN).

The scorching sun is the symbol of the south seas. The sun is about to set, but here, let’s have it shine brightly.

List 5 is the sprite for the sun (subroutine $SUN1). The sun alsways uses sprite number 11. The shape of the sun is changed by the STCHR command, and the color, by a SCOI command. List 2 is the main routine which calls the subroutine ($YASHI, IGROUND, $BSET). List E is comprised of subroutines IGROUND and IBSET: IGROUND draws the beach, IBSET adds one to variable RE at every execution, specifies the background color DATA (pattern data, character code, and character set number) using an ON-RESTORE statement, and then reads the data by a READ statement, and paints the background color using a STCHR statement. List 4 is the DATA.

10! #«##\* PALM «###\*

20 e o e r« t 4 8,78

30 on eoent 9osub SLIGHT

4 0 Print " !«M“ s 9 i n i t : b c o 1 & 0 1

50 RE\*0 \* 9 osub $ B S E T s 9 o s u b $BSET\*9osub SBSET 60 9 o s u b S G R 0 U H D s 9 o s u b \* P 8 L H s 9 o s u b IS U N 1

140S6ROUND: fcol &8ls9rooue 106,2065 circle 356, 140:Paint 125,1 8 6 , Sc 6 1 s r e t u r n

150SBSET:RE = RE + 1s on RE restore 170,230,316,180,240,320,250,1 90,260,336,278,200,210,280,290,346,220,386

16 0 read I I \* I 1,I 2,C H $: f or 1 = 11 to 12 steP-1:stchr CHS to I , I Ii next : return

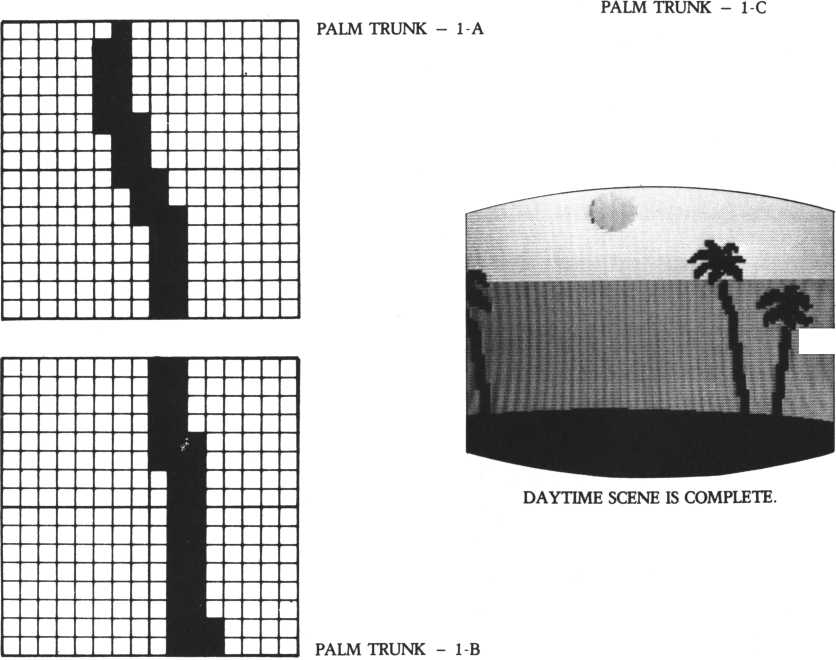
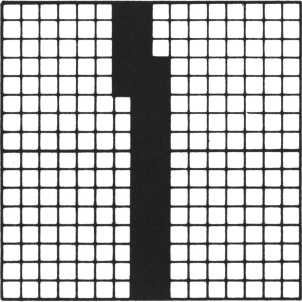
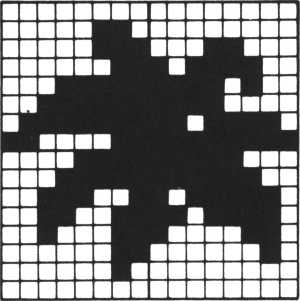
176 data 4,255,8,878i078i870f070i 238 data 5,255,8,0785070587050785 318 data 6,255,96,8785078567850705

1. 1o c 011 to 108,20:Print "W"
2. 9 o s u b $SEfiSHORE s i \* inkey<0><>8 then 9oto 1721 else Prin

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1660 | stch r | n | 00030\*1\*3\*3\*7\*7\*- | t o | 196 |
| 1670 | stchr | H | 7\*7\*3\*3\*1\*0\*0300" | t o | 197 |
| 1688 | st ch r | ii | 8 0 c 0 \* 0 \* 8 f c \* c \* e \* e " | t 0 | 198 |
| 1690 | st ch r | it | \* e \* e \* c \* c \* 8 \* 0 c 0 8 814 | t o | 199 |
| 1700 | ma9 3 |  |  |  |  |
| 1710 | scod | 1 1 | , 196 s sco1 11,609 |  |  |

t "W"

1. 38 I = i n k e y < 0 ) 5 i \* 1 = 7 then erase 11:r e t ur n else 9ot o 1730



IN THE EVENING SUN  
(DRAWING A SETTING SUN).

The sun begins setting. The burning sand is still very hot on which to walk. As the day draws to an end, the sea begins to turn red.

List 6 is the additional part to the main routine. List 7 is DATA on the background color. List 8 is composed of sprites for the sun and the sun’s reflection on the sea (HANSHA). The reflection combines sprites 18 and 19; it changes shape and color with every scene.

The scenes are changed by the INKEY command at the end of the subroutine used to specify the sprite for the sun. The subroutine determines whether the SPACE key was pressed (key scanning), and if pressed, erases the sprites for the sun and the reflection by the ERASE command, then returns to the main routine.

70 9 o s u b TBSETi 9 o s i .< b \*BSETs9osub #BSETS9osub \*BSETi9osub \*SU N2

180 data 4,255,8,8b898b898b898b89 240 data 5.255,0,0704070407040704 250 data 5,255,192,8707060807070608 320 data 6,255,96,0704078407040784

1. 40 $SUN2

1768 stchr "030F1F3F7F7FFFFF" to 208 1778 stchr “FFFF7F7F3F1F7F80" to 281 1788 stchr “C8F0F8FCFEFEFFFF" to 282 1790 stchr "FFFFFEFEFCF8Fe08" to 283 i860 seed 1 1 > 200s scoi 1 1 > & 8 8.

1818 1 o c 8 1 1 to 108»5H9osub ^REFLECTIONS P r i n t "W

1811 9osub $SEflSHORE s i f inkey<8><>8 then Soto 1811 else Prin

t

1. 20 I\*inkey<8)sif I-? then erase 11>18. 19s return else Soto 1820

2100$REFLECTION!

2110! ## REFLECTION 1 ##

2120 stchr "7 400001408348088” to 176

2130 stchr ”3444883487807480” } o 177

2140 stchr ”4e88007c004\*4800" to 178

2150 stchr ”804e001ef8084c88" to 179

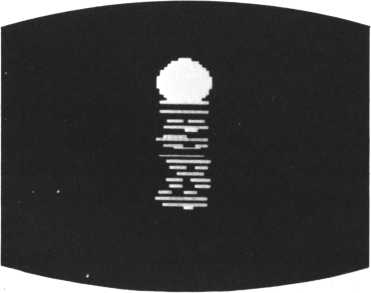
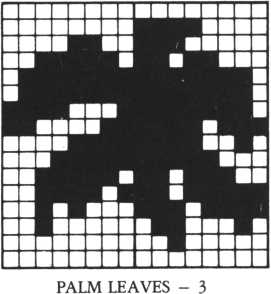
2160 stchr M1400040803001480“ to 188

2170 stchr ”8400780844003408^to 181

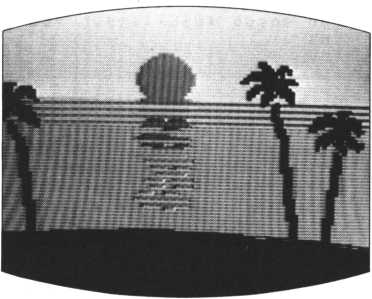
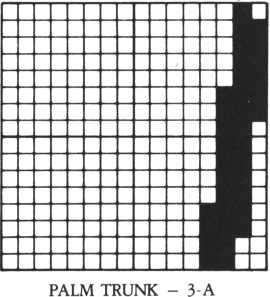
2188 stchr "44009e004c003400” to 182

2190 stchr ”140040804\*004ee0” to 183

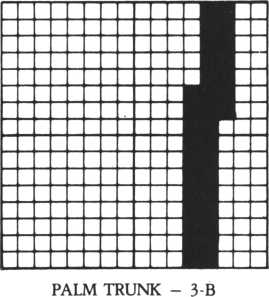
22 8 8 scod 18?176:scod 19? 180: scol 18>&86:scol 1 9 ? & 0 8 : u o i r» t 1 9 to 1 8 ? 3 : m o u e 8 1 8 in 6 on 108?85:return



THE SUN AND THE REFLECTION



THE COMING OF DUSK.



IN THE EVENING SUN  
(THE SETTING SUN  
TINTS THE SEA RED).

The setting sun shines brightly on the sea, ushering in the night, and the red-tinged sea signals the final moments of day. People stop and watch this timeless scene, continuously repeated since the creation of earth yet still filling us with wonder. Let’s replicate this scene using the M5.

List 9 is an additional part to the main program. List 10 is DATA on background color. List 11 is composed of the sprites for the sparkling sea and the evening sun.

80 Josub \*BSET!9osub \*BSETi9osub \*BSET« 9osub \* BSET s 9 osub \*SU N3

198 data 4,96,8,8b8b89098b8b8989 260 data 5.255,0,0908090809088908 270 data 5,255,128,0908080908090988 330 data 6,255,96,0908098809080908

1. 3 0 $ S U H 3

1850 stchr "0000000000000000" to 204

1860 stchr "0301317171711111" to 205

1870 stchr "0000000000000000" to 286

1888 stchr "c0101clelelel11 1 " to 287

1890 scod 11,284!sco1 11.&86

1900 1oc311 to 108,48:9osub $ R E F L E C T 10 H 2 s Pr int "M"

1918 I\*inke9<0)8if 1 = 7 than erase 11,18,19s return else 9oto 1910

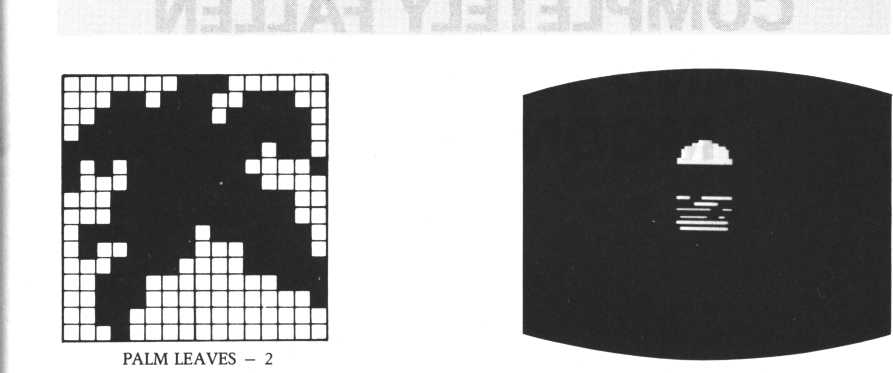
2218\*REFLECTI0N2

2220! \*« REFLECTION 2 \*#

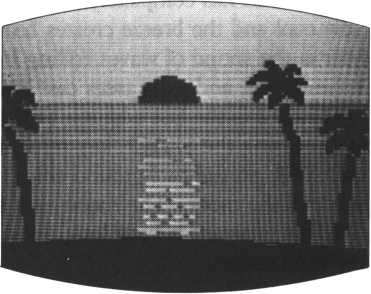
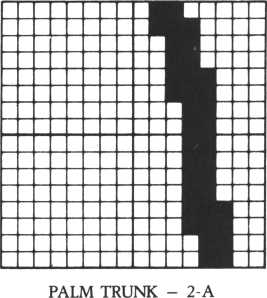
2230 stchr "0000001 10007001 1" to 184 2240 stchr "0071001100710000" to 185 2250 stchr "00000811001000c?" to 186 2268 stchr "080C001e881e0080" to 18?

2270 stchr "3\*003\*0000\*e0873\* to 188 22a© stchr " 8 0 0 0 6 7 9 0 3 c 0 0 \* \* 3 8 “ to 189 2290 stchr ”\*3@0\*©80807\*00c\*\* to 190 2300 stchr "00007600\*800\*\*18\* to 191

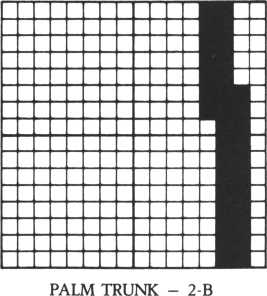
2310 scod 18? 184s scod 19? 188\* s c o 1 1 8 ? ?< 0 fl » s c o 1 1 9 ? & 0 B \* m om © 8 1 8 in 6 on 108? 95 s return -



THE SUN AND THE REFLECTION



THE SEA IN THE EVENING SUN.



IN THE EVENING SUN  
(NIGHT HAS  
COMPLETELY FALLEN  
AND A CRESCENT  
MOON SHINES).

The sun has completely set and the sea darkens gradually. The hot shore is now cool and the breeze creates beautiful patterns on the fine sand. All you can hear is the sound of waves rolling in repeatedly. The phospherescent crests of waves shine, and palm trees cast long shadows on the beach in the moonlight.

List 12 is the last addition to the main routine. Keys are scanned at line 120 and pressing the SPACE key returns the main routine to the beginning. List 13 is DATA on the background color. List 14 contains the sprites for the setting sun and the moon. The moon uses sprite number 21.

This program is very long. How were your results? We’re sure you drew a beautiful sunset.

90 9osub IBSET : .9 osub IBSET s 9 osub IBSET : 9 osub IB S E T s 9 o f u b IBS ETs9osub \*5UN4

100 9 osub IBSET: 9 osub IBSET: 9 osub I MOON: Print- \*\*MM

12 0 I=i n k e y<0> : i f 1=7 then erases9oto 40 else 9 o t o 110

200 data 4,255,0,0b080b080b0^:0b08

220 data 4,255,0,8105010581050105

280 data 5,255,0,040c040c040c040c

290 data 5,159,0,0401040184010401

380 data 5,255,0,0401840104018401

348 data 6,255,96,0401040104010481 1928ISUN4

1940 stchr ‘'0008000000080000“ to 208 1958 stchr “ 08008800838 f 3 HC to 289 I960 stchr "0000000000000000" to 210

1970 stchr “ 1980 scod 11 1990 locSli 2080 I = i n k ey 0

2010 $ M 0 OH 2030 stchr " 2040 stchr \* 2050 stchr H 2060 stchr “ 2070 scod 21 2080 1o c 9 2 1 2090 return 2320$REFLECT 2330\* ## REF

2340 stchr “ 2350 st ch r " 2360 stchf " 2370 stchr " 2380 scod 18

00000000C0404c44“ to 211 > 2 0 8 s s c o 1 1 1 » & 8 6

to 108,48:9osub $REFLECTI0N38Print MCT"

< 8) 8 i 4 1 = 7 then erase 11>188 return else

0001000000000000“ tO 212 0000000000010000" tO 213 0080C06070303838" to 214 38307060c0800800" to 215 >212:sco1 21>

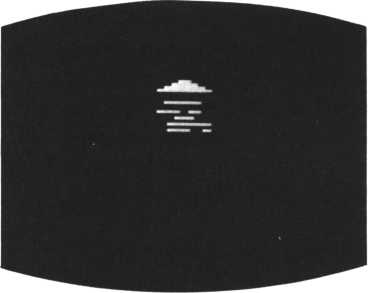
to 38\* 38! print “W"

I O W 3

LECTION 3 ##

080000003f8000fe" to 192 0007001f001f0000“ tO 193 0000000048000078“ to 194 0 0 e 0 0 0 44 0 0 c 7 8 8 0 0" to 195

> 1 92 8 sco1 18 >§06



THE SUN AND THE REFLECTION



THE NIGHT SKY AND THE CRESCENT MOON.

goto 200

1oc 818 to 188\*788retu rn

THE SETTING SUN.

: .



A FUTURISTIC CITY (DRAWING WITH FREE IMAGINATION)

Super high-rise buildings are neatly arranged and highways radiate in all directions like a spider’s web. So far our subject has been natural scenery, which you would normally not associate with a personal computer. We would like you now to draw a world more suitable to personal computers using your imagination.

List 1 is a program used to draw a futuristic city. Five colors for the background picture are provided between lines 30 and 90, so that each color is used in order. (After the picture is drawn and the SPACE key is pressed, the program determines whether to return to line 20 at line 270, and if it determines so, it again starts from the beginning. At this time the program changes the background color depending on the value of H.)

Subroutine IDCIR draws the bases of the buildings. Subroutine ISHOOT draws curved lines. Drawing highrises is easy if you use the BAR command.

How are you coming? Does your picture simulate a future city? What are the curved lines on this side?

10! \*\*\*\*\* TOWN \*\*\*\*\*

28 Print " ISW" s 9 i n i t. : b c o 1 & 0 1 30 H = H + 1 s i f H = 6 then H=1 40 on H 9oto 50,60,70,80,90

50 for 1=0 t o 2 55 ! stchr " 0 4 0 c 0 4 0 c 0 4 0 c 0 4 0 c " to I , 8s next s 9ot-o 100

60 for 1=0 to 255s stchr H.0480008400080c00 " to I , 8 ! n e x t : 9 o t o 100

70 for 1=0 to 255! stchr " 0 4 0 9 0 4 0 9 8 4 0 9 8 4 0 9 '\* to 1 > 8! next! 9 o t o 100

88 for I “ 0 to 255: stchr « 0408048884888400 « to I , 8: r» e x t : 9 ot o 108

98 for 1=8 to 255!stchr »8608068086800600" to I ,8:next:9oto 100

10 0 9 m o u e 8 8, 118!V = 8!X S = 2

110 fcol &0fl:Xft = 4iXB=16i9osuh $DGIR

128 fcol &09!V=5!XR=18!XB=30!9osub fDCIR

130 fcol & 0 C•V = 9!X fi = 3 2 sX B = 4 4!9 o s u b \*DCIR

148 fcol ?<84 : V = 1 4 : X 0 = 4 6 ! XB = 52! 9 osub \* 0 C I R

158 fcol &0P: Xft = 52! XB = 68! 9 osub t-DCIR

160 fcol &85: bar 27,16,51,184:fcol &84:bar 51, 1 7,57, 184s fco1 & 8 7:bar 1 12,32,128,188: 4 col §84:b ar 1 28,33, 1 32, 188 178 fcol 8«85:bar 72,8 , 1 88,98 : f co 1 &8C:bar 188,9,187 188 f co1 &8C:bar 48,48,75,112:fco1 &84:bar 75,41,81 &87: bar 48,56,65 , 1 28 : f co 1 &83sbar 65,57,78, 128 198 fcol &0D:b ar 96,58,1 16,1 17: fcol &86:bar 117,51,

288 9move 28,18:C=8:V=0:S=2 218 fcol &8B:X 1 =0:X2= 1 4:9 osub SSHOOT 228 fcol &04:Xl=17:X2=31:9osub $SHOOT 238 f co 1 &83: Xl=34: X 2 = 5 8 : 9 o s u b $SHOOT 248 fcol &06: X 1 = 53: X2-75: 9 osub $SH00T

**, 98**

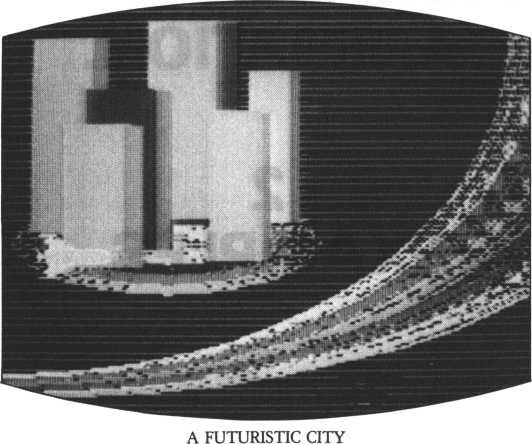
,1 12:fcol 120,116

258 fcol &87:Xl=78:X2=186:S=4:9osub \*SHOOT:Print "W"

268 J = ir»ke9<l>:if J = 1 then 9 o s u b $ S P RI T E 278 I = i n k e9 ( 8>:if 1=7 then 9oto 28 else 9oto 268 288$DCIR

298 for X = X fi to XB step X5:V»V+i:circle X,VJ next s return 380tSHOOT

318 for X = X1 to X2 step S:C=C+1 320 if C mod 4 = fl then ^=V+l 330 circle 280 + X,160 + V:next J return



**DRAWING**

**ANIMATION**

1

TECHNIQUES OF  
PERSONAL COMPUTER  
ANIMATION.

2

EXAMPLES.

TECHNIQUES OF  
PERSONAL COMPUTER  
ANIMATION.

A butterfly’s shape can be briefly classified into two parts: a butterfly spreading its wings, and one closing them. When you have decided which kind of butterfly you are going to draw, take a sheet of character pattern paper and draw a 16 x 16 square. Draw a butterfly within the square.

Here we made two kinds of butterflies as shown on the next page. These butterflies are flying from left to right (patterns 1 and 2).

We have drawn two kinds of butterflies. If you want to express more delicate movements, make many patterns with different wing shapes.

Now we also want to create a butterfly landing on a flower. The butterfly holds its wings upright when it is on a flower, so we made three patterns for the butterfly as shown on the right page.

Making shapes one at a time and thinking of the movements of what you desire to draw accomplishes the task best.

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |

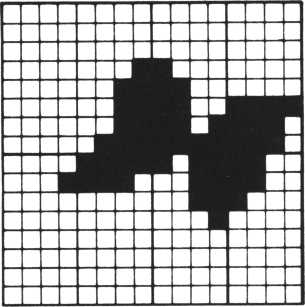
|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

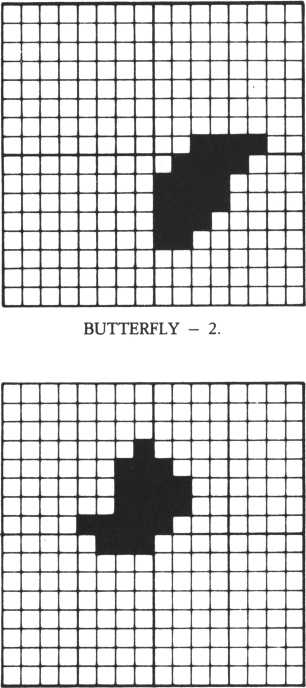
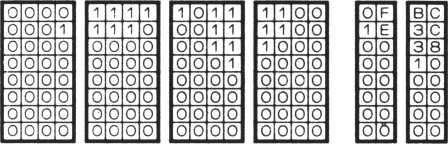
|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 0 | p | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 |

|  |  |
| --- | --- |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 0 | 1 |
| 0 | 3 |
| 0 | 3 |
| 0 | 3 |
| 0 | 7 |

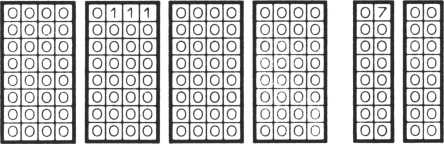
|  |  |
| --- | --- |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 8 | 0 |
| c | 0 |
| c | F |
| D | E |
| F | E |



BUTTERFLY - 1.



BUTTERFLY - 3.



BINARY.

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | o | 0 |
| 0 | c | 0 | 0 |
| 0 | 0 | o. | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 |

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | c |

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 |

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |

|  |  |
| --- | --- |
| 0 | 0 |
| 0 | [0 |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
|  | |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |

|  |  |
| --- | --- |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 0 | c |
| 3 | c |
|  | |
| 7 | 8 |
| F | 0 |
| F | 0 |
| E | 0 |
| C | 0 |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |

|  |  |
| --- | --- |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 0 | 1 |
| 0 | 3 |
| 0 | 3 |
| 0 | 3 |
| 0 | F |

|  |  |
| --- | --- |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 8 | 0 |
| c | 0 |
| c | 0 |
| 8 | 0 |

HEXADECIMAL.

TECHNIQUES OF  
PERSONAL COMPUTER  
ANIMATION

PAINTING COLORS.

You can paint only one color on a sprite. For more than one color for one character, specify other sprites for the parts you want colored differently. Combine these characters using the JOINT command. A yacht is used here as an example. Overlaying three sprites for the body, sail, and mast makes the yacht three colors. You can create a character with up to four colors, since up to four sprites can be overlapped simultaneously.

You can create beautiful characters by overlapping more than one color, however, since we can’t arrange more than four sprites at a time, it would be better to use one sprite for a one-color character. The spaceship is an example of a one color character.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 10 1 | ! LfiLt’ | s color |  |  |  |  |  |  |
| 20 | Print | "ISM" : n | a \*3 3 |  |  |  |  |  |
| 30 1 | ! \* \*\*\* | \* V ach t | s\* \* |  |  |  |  |  |
| 40 ! | ! \* \*\* | sail \*♦ | \* |  |  |  |  |  |
| 50 | stchr | "00000 | .0000000 | 02 | 06 » | t. | 0 | 100 |
| 60 | stchr | " 0 e 1 e 3 | e 1 e 0 8 0 0 | 00 | 0 0 " | t | 0 | 1 0 1 |
| 70 | stchr | " 8 0 0 0 0 | 0 2 8 2 8 3 0 | 3 0 | 3 8 " | t | 0 | 102 |
| 80 | s t. c h r | " 38 3c 3 | c 3 e 0 0 0 8 0 8 | | 00" | t | 0 | 183 |

90! \* \* \* hull ♦ + \*

10 0 Stchr "00 00000000 000000" t © 104

110 stchr "0000000000\*f?4\\*« to 105 12 0 Stchr "0 00 00000000 00000" t© 106 130 Stchr "000 0 0 00000\* \* f €\* \* C " to 10? 14@! \* \* \* mast \*\*\*

150 StChr "0000000101010101" to 108

160 stchr ,,0i010l0l01000000" to 109

170 stchr "0040404040404040“ to 110

180 stchr " 4040 4040488 08008" to 111

190 scod 0,180sscod 1, 10 4 5 seed 2/108

208 sco 1 0 , & 0 F s s c o 1 1 , &87 s sco 1 2>&0B

210 Print cursor (8? 5) ‘ "\*•\*\* sail of yacht- \* \* \*

220 loc 0 to 180,80

230 Print cur sor <8,20) \* “Hit- Return Key \* ! M 240 if inkeylOchr $< 13) then 9oto 248 25 8 c 1 s s p r i n t cursor<8?5);"\*\*\* hull of yacht 260 erase!loc 1 to 188,88

:4c \* "

\* \* \*

278 Print cursor<8,28>5"Hit Return Key !!

288 if inkeylOchr $< 13) then goto 280

29 8 c1s s P r i n t cursor(8,5)! "\*\*\* mast of yacht

388 erase!loc 2 to 108,88

310 Print cursor <8,20> 5“Hit Return Key !!” 320 if inkey$Ochr$<l3>then goto 328 33 0 Joint 2 to 1, 1 s J oi n t 1 to 8,1 340 clssPrint- cursor (9,5) 5 "\*\*\* yacht \* \* \* M 358 moue 0 in 8 on 180,88

368 Print cursor(8,28)!"Hit Return Key M" 378 if inkeM lOchrK 13) then goto 378 388! \*\*\*\*\* UFO \*\*\*\*\*

390 stchr "0001030387070707“ to 112 480 stchr "870f8f3fff3f0705" to 113 418 stchr "08008080c8c0c0c0“ to 114 428 stchr " c8e8e0f8fef8c848“ to 115 438 scod 3»112!scol 3,18

448 clssPrint. cursor (8,5)?" \*\*\* UFO \*\*\*“

458 erasesloc 3 to 100»88send

MOVING ANIMATION  
PATTERNS.

Move the yacht described on the previous page from right to left. You can only specify XY-coordinates on the screen using the MOVE-TO command, in order to prevent the yacht from disappearing. Use a MOVE-STEP command here; you cannot, however, stop the yacht.

Next move the butterfly. You can’t express the butterfly’s movements in the same way as you did those of the yacht. The two patterns of the butterfly appear alternately on the screen and movements in the XY direction are irregular.

How did you do? Is the butterfly moving well? Both drawing and moving animation patterns are equally important. By doing both well, you will then be able to create superb animation.

This program loops endlessly. To end the program, press the SHIFT and RESET keys simultaneously.

|  |  |  |  |
| --- | --- | --- | --- |
| 10 ! | Let's move |  |  |
| 20 | Print " I3W" : m a 9 2 |  |  |
| 30 ! | \* \* \* \* \* yacht- \*\*\*\*\* |  |  |
| 40 ! | \* \* \* sail \* \* \* |  |  |
| 50 | stchr ”8800000000008206" | t 0 | 100 |
| 60 | stchr “0ele3ele80800088“ | t 0 | 101 |
| 70 | stchr “0000002020303038" | t 0 | 102 |
| 80 | stchr “383c3c3e00088000" | t- 0 | 103 |
| 90 ! | \*\*\* hull \* \* \* |  |  |
| 100 | StChr “0000000000000000" | t- 0 | 1 8 4 |
| 110 | stchr "0000000000 f f 7f 1 f “ | t o | 105 |
| 120 | Stchr "0 8 0 0 0 0 0 8 8 0 0 0 0 0 0 8" | t- o | 186 |
| 130 | stchr "0 0 0 8 8 8 0 0 00 f f f e f c " | t o | 107 |
| 140 | 1 \* \* \* mast \* \* \* |  |  |
| 150 | StChr "8000000101010181 “ | t 0 | 108 |
| 160 | StChr “0101010101000080“ | t 0 | 189 |
| 170 | stchr “0040404840404040“ | t 0 | 1 10 |
| 180 | stchr "4040404040088000" | t o | 1 1 1 |

190 scod 0> 1005 scod 1j1845 scod 2 j 10 8 200 scol 0 ? & 0 F : s c o 1 l,8<0?sscol 2>&0B 210 Joint 2 to 1»1\*Joint 1 \o 0 > 1

22 0 moo© 0 in 0 on 22 5»65s move 8 in 1 steP-1 > 8>?8 230! ##### butterfly #####

2 4 0! \* \* \* butterfly 1 \* \* \*

250 stchr "0000000103838387" to 144 260 StChr "0f10000000000000" to 146

270 stchr ”00000080c0c<defe" to 147 280 stchr "b c 3 c 3 810 8 0 0 8 0 0 8 0" to 147 29 8! \* \* \* butterfly 2 \*\* \*

300 stchr " 0800000000000000 " to 148 310 Stchr H0000000000000000 M to 149 320 stchr u 0 0 0 0 00 0 0 00 00 0 0 3cM to 150 330 stchr M 7 8 f 8 f 0 e 0 c 0 0 0 0 0 8 0" to 151

34 0 scod 4? 144!scod 5 > 14 8• sco1 4«&0ft\*scol 5 ? & 0 fl s X \*0

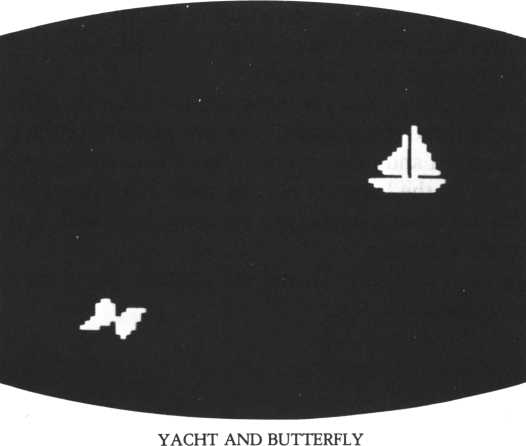
350 V=150!randomize

360 if r n d( 1 )= 1 then B=-rnd<15>

1. 0 1oc 4 to X ? V + B s sleep 2 > 5:erase 4
2. 0 randomize:if r n d < 1 ) ® 8 then $ = - r r» d < 1 5 >

390 X = X + 3! 1o c 5 to K?V + ft:sie e P 2>5\*erase 5 400 X»X+10

410 if X = > 2 5 5 then goto 340 else 9o t o 3 50



EXAMPLES (1)

IN THE SEA (DRAWING SMALL FISH).

In order to draw a group of small fish swimming, you must first draw seven fish on one character. A group of such size is inadequate, so the same character pattern was put on two sprites, and the sprites were then combined horizontally using the JOINT command. A group of fish is prepared in two 8x16 dot charac ters. Another pattern of small fish differ slightly from the first pattern. These two character patterns appear alternately using the SLEEP command. One character pattern is different in color from the other. Both colors change every time the fish move. In order to make the fish swim in the specified area, the position of the fish is checked by the SPRITE function. If the X-coordinate of this group of fish is 50 or less, and the Y-coordinate is 110 or more at line 6290, the sprites will be erased by the ERASE command. The program then shows the sprites in a specific area whereby the fish swim.

A shark appears in this animation, so you must input the SSHARK list as well. The list here is programmed for the fish only. To execute the list, add it to the main program.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 150 | 9os ub $ FISH | |  |  |
| 168 | U F 2 = 1 | then 9 os u b $S H ft RK |  |  |
| 6140$FISH | |  |  |  |
| 6158 | 1 ## FI | SH 2 ## |  |  |
| 6168 | stchr | “ 8 0 7 0 e 8 7 0 8 7 0 e 0 7 8 0 ,: | t. o | 188 |
| 6170 | st ch r | “3874388083078300M | t 0 | 1 89 |
| 6180 | stchr | '• 00801c 3a 1c 808878“ | t o | 1 98 |
| 6190 | stchr | “ © 8 7 8 8 0 8 e 9 d 4 e 8 8 0 8 “ | t o | 191 |
| 6200 | stch r | “80 1 c2e1c 8 3058380“ | t o | 192 |
| 62 10 | st ch r | “ Ic2eic0305030080" | t. 0 | 193 |
| 6228 | s tchr | “88081c2©9cc08078“ | t o | 194 |
| 6230 | stchr | “ b 8 7 0 8 e 9 7 c © 8 8 8 8 8 0 “ | t. o | 195 |

6 2 4 8 scod 1 8 > 1 8 8 : s c o 1 1 8 \* 5 : s c o d 19\* 188s scol 1 9 > 51 >\* o i n t 19 to 18\*2:1oc 18 to 180\*58

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 6258 | m | oue | 1 | 8 i n 6 | s | t ©P | - | 1 \* 1 | i \* 1 | 0 |  |  |  |  |  |
| 6260 | si | eeP | 4 | \* 5:scod |  | 18\* | 1 | 92: | i sc | o 1 | 18\* | 8: | s c o d | 19\* 192:scol | 1 9 \* 8 : m |
| o o e 18 | | i n | 6 | s t e P - 1 \* | 2 | \* 5 |  |  |  |  |  |  |  |  |  |
| 6278 | s 1 | e©P | 4 | \* 5 s s c o d |  | 18\* | 1 | 88: | > s c | ol | 18\* | 1 0 | : sco d | 19 \* 18 8:scol | 19\*18 |
| 6280 | mo | u© | 18 | in 6 s | t | © P - | 1 | \* 2 \* | 18 |  |  |  |  |  |  |

6290 TM\*sPrite<18,0>: TM\*sPr ite<18, 1) : i f TW>118 and TM<58 the n 9 o t o 6306 else 9oto 6 2 5 0

6300 erase 18,19\*scod 18,183:scol 18,7:scod 19,1885scol 19,7 6310 joint 19 to 18,2:1oc 18 to 58,118

6328 wove 18 in 6 step l,-2,5ssleeP 4,5:scod 18,192sscol 18, 8 s scod 19,192 5 sco1 19,8

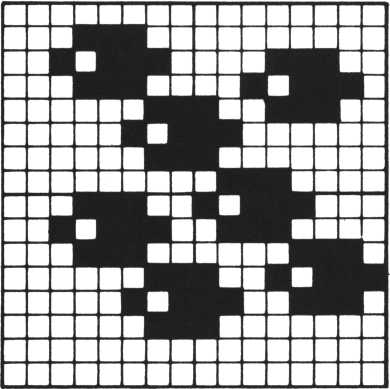
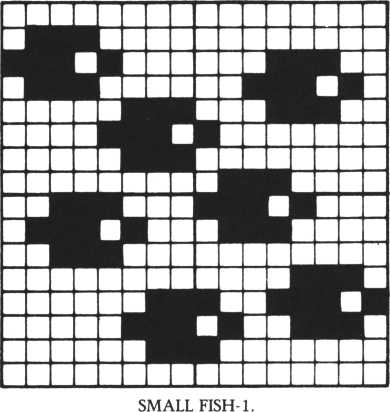
63 38 woMe 18 in 6 step 1 , - 1 , 1 0:s1eeP 18,10:scod 18,188:scol 18,7:scod 19,188sscol 19,7

6348 T T = s P r i t e <18,0 > :TF-sPrite( 18, 1> s i f TT<58 and TF>100 the

n FZ \*r n d < 2)s 9 o t o 63 53 else 9 o t o 6328

63 58 if F 2 < > 1 then m o m e 18 in 6 step 2,1,4\* return else scol

1 3,4:s c o1 19,4:wove 18 in 6 steP-1,2, l\*return



SMALL FISH-2.

IN THE SEA

(DRAWING A TROPICAL  
FISH AND A SHARK).

This is an undersea scene. Try drawing a tropical fish and a great white shark hunting for game.

The tropical fish is one character size (8x8 dots), and two character patterns are prepared: one for the tropical fish fanning its fins, and the other for it spreading its fins. The two patterns are alternated using the SLEEP command, so the fish appears to be swimming.

This list and sprites are used for the tropical fish and the shark only. To execute this program, don’t forget to add this list to the main program.

20 Print MISM" 5 9in it ibcol &81 : wa9 2

98 J \* i n k e 9 < 1 > : if J \* 1 then -3oto 100 else goto 90

10 0 g os u b \* T R O P

110 if r n d <1> =0 t h e n 9 os u b \*• S H ARK

1 7 8 r a n d o w i z e : 9 o t o 1 8 8

5000\*SHARK

5001 9osub \*SHARK

5010 on event 9 o s ub I JAMS\* event 40,10

50 2 0 if st atus< 2)«1 then scol 13,5:move 13 in 2 s t e P ~ 1,2,2 5030! ## SHARK ##

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 5040 | stchr | M 0 0 0 0 0 0 8 0 0 8 0 0 0 7 0 3 " | t o | 112 |
| 5050 | stchr | \* 0 1 0 0 000 0 0 0 0 0 0 0 0 0 " | t- o | 1 1 3 |
| 5060 | stchr | "00000000 00 0 0 f f O f " | t 0 | 1 14 |
| 5070 | stch r | " f f a a 0 0 5 5 f f 0 0 8 0 0 8 " | t o | 1 1 5 |
| 5080 | scod | 10, 112\*scol 10? 12 |  |  |
| 5090 | stchr | "0000008 103 f f f f f f " | t o | 1 1 6 |
| 5100 | stchr | " f f f f f f 0880000000" | t o | 1 17 |
| 5110 | stchr | "00e@c8c0c0f f f f f f" | t. o | 1 18 |
| 5120 | stchr | " f f f f ff381C0C0484" | t 0 | 1 19 |
| 5130 | scod | 11,116:scol 11,12 |  |  |
| 5140 | stchr | "000803071 f f f f f f f " | t. o | 128 |
| 5158 | stchr | "e7c3800008000080" | t 0 | 121 |
| 5160 | stchr | "■ 6 0 e 0 e 8 e 0 e 0 c 0 c 8 c 0 " | t o | 1 22 |
| 5170 | stchr | Me8e0e08800800800" | t o | 123 |
| 5189 | scod | 12,120:scol 12,12 |  |  |
| 5190 | stchr | "0000000000000703" | t 0 | 1 24 |
| 5200 | stchr | "0100080088000800" | t o | 125 |
| 5210 | stchr | "000000000000 f f ef " | t o | 126 |
| 5220 | stchr | » f f aaf f 0000000000" | t 0 | 127 |

12 to 11,23 Joint 11 to 1 8 > 2 3 e v e n t on move 10 in 0 on 2 5 5 , r n d < 5 0 ) + 6 0 • m o u e 10 in 1 step i \* status(1)=9 then e vent offsreturn else 9o t o 5 5340$TROP

53 504 ## FISH 1 ##

|  |  |  |
| --- | --- | --- |
| " | t o | 128 |
| II | t o | 129 |
| II | t. o | 1 30 |
| II | t o | 131 |
| " | t 0 | i 32 |
| II | t 0 | 133 |
| II | i o | 134 |
| II | t o | 135 |
| 11 | t o | 10 |

"0000000103\* f f f f f

5238

5240

5250

5260

5270

5280

5290

5300

5310

5320

5330

stchr

stchr’

st ch r

stchr

stchr

stchr

stchr

stchr

Joint.

-1,0,5

330

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 5360 | stchr\* | it | 000103860c!f3f77" | t 0 | 1 8 8 |
| 5370 | stchr | " | f f7f3f1f0C86030 1 H | t o | 181 |
| 5380 | stchr | " | 00 f 008000080c8ec" | t o | 1Q2 |
| 5390 | S t C h r | it | f e e c c 8 8 8 8 8 8 8 8 8 f 8 “ | t 0 | 183 |
| 5408 | s c o d | 13 | ,188!sco1 13,6 |  |  |
| 54 18 | s t. c h r | " | 080000878c1f 3 f 77" | t o | 184 |
| 5428 | s t. c h r |  | f f 7 f 3 f 1 f 0 c 0 7 8 8 0 0 “ | t o | 185 |
| 5430 | s tchr | H | 0 0 0 0 8 0 f f 0 0 8 0 c d e 8 “ | t. o | 1 8 6 |
| 5448 | s t ch r: | It | fee 8 c 8 8 0 8 8 f + 8 8 8 8 | t 0 | 1 07 |
| 5458 | r a n d o | m i | 2 e! 1o c 13 to 255, | r n d | «: 5 0 > + 2 8 |
| 5460 | if F < | > 2 | t h e n m o e 13 in | 2 s | t e P- 1 < 1,8 |
| 5478 | i f F = | 2 | a n d r n d < 2 > \* 2 t h e n | wove 4 3 in 2 st | |
| 5488 | si eeP | 4 | ,15!s c o d 13,184!s | CO 1 | 1 3 < 8 |
| 54 98 | if F < | >2 | t- hen m o»,«e 13 in | 2 steP-2,1,158 | |

i \* F « 2 a n d r r« d < 2 > = 2 t h e n rr» o u e 1 sleep 4,8\* scod 1 3 < 1 8 8 ' s c o 1 13,6

S P = s p r i t. e < 1 3,8 > : i \* S P > 7 0 then m o u e

i:

1 , 2 : F = 1

3 in 2 s t e P- 2, 1,8:F = 1

s t e P - 1 , - 1,6 • e n F = 1

5 5 8 0 55 10 5520 F = 2 5530 5540

S S\*sPrite<13,0)8 randomize S S• i \* F = 2 and S S< 3 8 i \* s P r i t e(13, < 6 0 then return else 9 oto 54 60 6360$JfiWS!if J-1 then 9oto 6380 6370 TC\*9«for TD=124 to 132 step \*1 Jreturn

t- h \*

**4**:**TC** = **TC+1**:scod **TC,TD:n ext:J scod T C** > T **0 3 n e x 15 J**

6380 T C - 9 i for TD=U2 to 120 steP 4\*'TC\*TC+i = 2 • retur-r»

IN THE SEA  
(DRAWING A TURTLE).

The turtle is one character size (8x8 dots), and two character patterns are used for the turtle showing its legs in different positions. The interrupt process is performed using the EVENT-ON command, and the character pattern of the sprite is changed to the other at the point of destination. The SPRITE functions, and MOVE to STEP commands are often used, making the turtle move comically: the turtle’s position is checked by the SPRITE function; its moving direction and speed are changed by changing the MOVE-STEP commands when the turtle arrives at a certain area, and the MOVE-STEP commands again change when the turde reaches another area. The turde swims at the bottom of the screen, where the X-coordinates are between 50 to 110, and it moves by MOVE-STEP commands. The position of the turde is checked by the SPRITE function at line 6080. If the Y-coordinate is 160 or lower, the MOVE-STEP commands change and the moving direction and speed change accordingly. The turtle’s position is again checked at line 6100 by the SPRITE function. If the Y-coordinate decreases to less than 70, the moving direction and speed then change again.

The commands, IF-STATUS (5) = 0 — THEN —, at lines 6070, 6090, 6110, 6130, are used to avoid error 20 (sprite error). A sprite error occurs when the coordinates of a sprite which has disappeared from the screen were requested (SPRITE function) or the sprite was requested to move (MOVE command). This IF statement sets the interrupt process to off and returns to the main program, if the sprite plane of the turde character pattern has disappeared from the screen at post 5.

This list and these sprites are for the turtle only. To execute this program, add it to the main program.

mmm j \*

140 i 1r n d < 1 > = 1 then 9©sub fTURTLEl 5940#TURTLE1son event 9osub $TURTLE2:event 15,15

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 5950 1 | ! ## TURTLE ## | |  |  |
| 5968 | stchr | "0131180407060506" | t 0 | 180 |
| 5970 | stchr | ”0506070119300107" | t o | 181 |
| 5980 | stchr | Mc0c68cd870b050b0" | t o | 182 |
| 5998 | stchr | "50b0?0f8cc868000" | t o | 183 |
| 6000 | scod | 17,180:scol 17,5 |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 6010 | stchr | "0101000f1f362506" | t o | 1 84 |
| 60 2 0 | s t c h r | M25361f0f81080800" | t. o | 185 |
| 6030 | s tchr | " c 8 c 8 8 0 f 8 7 c b 6 5 2 b 0 11 | t. 0 | 1 86 |
| 6840 | stch r | " 5 2 b 6 7 c f 8 c 8 8 8 c 8 7 0 " | t o | 187 |
| 6050 | loc 17 | to rnd< 68 >+58, 198: | e u e n t | |

move 1'

i n 5 steP 1

6060 6070 6080 i , 20 6090 if 6100 , 20 6110 6120 6130

i f i f

i f

s t a t. u s < 5 > = 0 t h a n

s P r i t e < 1 7,8 > < 1 6 8

on

-1» UsleeP 1 ? 4 0 e u e ri t- o f f ' r e t u r n t h e ri ri\*i o v e 17 in 5

S t €■ P - i \* - 1

status(5) = 0 t- hen e u e n t off s P r i t- e < 1 7 » 8 ) < 7 0 the n m o u e

if status<5> move 17 in 5 if statuf<5) 6390$TURTLE 6400 if F K = 1 6418 return

• 0 then steP-1 = 0 then

e m e n t o f f 1? 3:sleep e y e n t o f f

\*• ret u r n 17 in 55

r e t- u r n 1,20 ret ur n

step 1,1

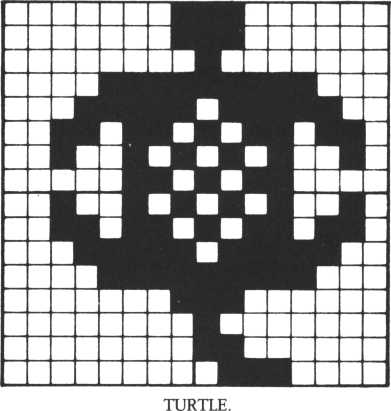
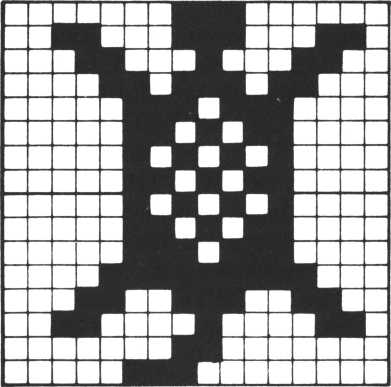
else \*3 o t o

then seed 17,184:FK=2 else seed 1'

180:

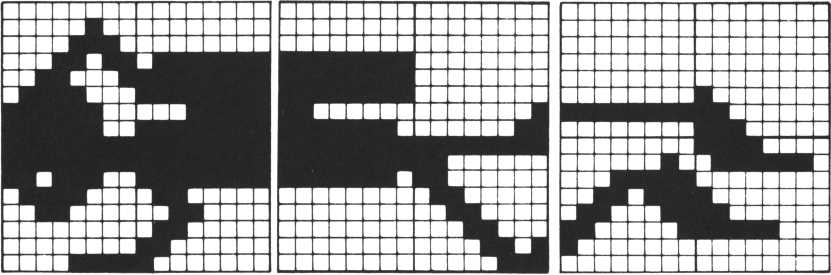
, 2 ! s 1 e e P : s 1 e e P 1

6066 F K = 1



IN THE SEA  
(DRAWING A DIVER)

In this listing, the position of the diver is checked by a SPRITE function at line 5910. If the X-coordinate indicating the diver’s position is 100 or less, a random number is substituted in the variable FS by the RND function. If the substituted random number was 2, the color of the diver changes to blue and the MOVE-STEP commands change so that the diver moves toward the top left of the screen, gaining speed. Only when FS equals 2 does the shark appear and the diver’s color lightens as he swims away from the shark. To execute this, input the ($SHARK) listing.



DIVER

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 120 •; | 3 OS | uh | $ | 0 I U | ER |  |  |  |  |  |
| 130 : | i + | F S = | 2 | th | e n 9 © | sub | $ SHARK | : r | ar | i d o m |
| 5 5 5 0 $ D I | | UER |  |  |  |  |  |  |  |  |
| 5560 1 | 1 # | # D | I | UER | ## |  |  |  |  |  |
| 55 7 0 | S t- | c h r |  | 11 08 | 0 4 8 e 1 | b 3 1 | 78+c+c" | t | o | 156 |
| 5580 | St | c h r |  | H + + | + f d c 7 | 830 | 0 0 0 0 0 0 u | t | 0 | 157 |
| 5598 | St | c h r |  | " 0 0 | 0 00 07 | + + + | 7 + 1 8 + + M | t | 0 | 158 |
| 5608 | St | ch r |  | " + + | + + + + 6 | 03 + | 000000» | t | 0 | 1 5 9 |
| 56 18 | sc | od | 1 | 4 , 1 | 56 s sc | Ol | 14,3 |  |  |  |
| 5628 | St | c h r |  | \*' 080008 + | | + + + | + + C0+ + " | t | o | 1 6 8 |
| 5638 | St | chr |  | 11 + + | + + + + 3 | ce0 | 0 0 0 0 0 0 " | t. | 0 | 161 |
| 5640 | St | c h r |  | “00 | 0 0 00 0 0 0 0 | | 008088" | t | o | 162 |
| 56 5 8 | St | c h r |  | '\* + + | + +000 | 000 | 8 0 0 0 0 0 •' | t | o | 163 |

scod 15,160:scol 15\*3 S’tchr " 0 0 0 0 0 4 0 e 1 + 3 7 6 3 c 0 " to 164 stchr M80c060341e0f0303” to 165 Stchr tt 0 0 0 00 0 0 0 0 0 8 0 +00 0" to 16 6 stchr " 0 00 0 00 00 0 0 0 08 0 f 0" to 167 scod 16,164:scol 16,3 stchr "00040elb3178\*cfc" to 168 stchr " + + ++dc783000008f" to 169 stchr "0900007HHUf H" to 17 0 stchr " + + f i + + 6 0 3 8 6 0 c 0 8 0 M to 171 stchr "800000+ + + + Hc8fe" to 172 stchr " + + + + +00000800080to 173 stchr "0000080000000103" to 174 stchr " + +C0C06030180d07" to 175 stchr " 0000000000 00+ + 08" t o 176 stchr "00021+3763c18000" to 177 stchr " 8 0 O 0 0 0 0 0 0 0 8 0 c 0 e 0" to 178

stchr " + 8 7 e 0 0 8 0 c 8 + 8 0 0 8 8" to 179

.j o i n\*t 16 to 1 5 \* 2 \*o i r» t 15 to 14,2

mo m e 14 i n 3 o n 2 5 5,18 0:mo u e 14 in 4 s t e P-1,0,4:F1= 0:F S

i + F S-2 then sleep 2,5 else sleep 5,5

T C \* 13 s for T D =15 6 to 164 step 4:TC = TC+1:scod TC,TD:next i + F S = 2 then sleep 2,5 else sleep 10,5

TC=13:+or TD=168 to 176 step 4:TC=TC+1:scod TC,TD:next if F1< > 0 then goto 5930

i + sPrite<14,l)<100 then Fl=2sFS=srnd<2)

i + F 1 = 2 and F S = 2 then scol 14,4:s c o1 15,4:s c o1 16,4:mo w in 4 steP-1,-1,1

5660

5670

5680

6690

5700

5710

5720

5730

5740

5758

5760

5778

5780

5790

5880

58 10

5820

5830

5840

58 5 0

X 0

5860

5870

5880

5890

5900

5910

5920

e 14

5930

i + status<4>=0 then return else goto 5860

IN THE EVENING SUN

—Draw a star and an airplane

An airplane flies through a night sky with shooting stars, its lights blinking. To make the airplane’s lights blink, change the color of the sprites from amber to no color and back to amber again. This program needs the program that follows also input in order to run.

LIGHTS

AIRPLANE

STAR



19! \*\*«#« PALM #####

2 0 e v e n t 4 0,7 0

30 on e ment 9 o su b \* LIG HT

110 J»inke9<l>:if J = 1 then erase 7,8,9 s 9 o t o 130

12 0 I = i n k e 9 < 8) : i f 1=7 then erase!9ot o 40 else 9 o t o 110

1 3 0 9o s u b $ P Lfl NE:9os u b $ft U E C:s1e e p 2 ,6 0: ement offs erase 14.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 5 > 16? | 17 s 90su b $P ft L M s 9 os u b tM 00 N• | | | P r i n t | "mn:9010 118 |
| 5 0 0 0 $ P L ft N E : | | t. = 0 s e m e n t o n |  |  |  |
| 5010! | \* \* P L ft N E \* \* | |  |  |  |
| 5020 | s t c h r | \* 80068cfi 18306080“ | t o | 144 |  |
| 5 0 3 '0 | s tchr | " 800000 0 00 00 0 0 0 0 0 " | t 0 | 145 |  |
| 5040 | stchr | "0 000 0 0 0 0 0 0 00 0000 M | t 0 | 146 |  |
| 5 0 5 0 | stch r | “ 0 8 0 0 000 0 0 0000 00 0 “ | t o | 147 |  |
| 5660 | st ch r | " 0080020808080800 | t o | 148 |  |
| 5070 | s t c h r | " 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 “ | t o | 149 |  |
| 5080 | s t. c h r | " 0 0 0 0000 0 0 0 0 000 00 " | t 0 | 1 5 8 |  |
| 5090 | stch r | “0800008 0 88808800“ | t 0 | 151 |  |
| 5100 | scod 1 | 2,144:scod 13,148 |  |  |  |
| 5110 | s c o 1 1 | 2,1!scol 13,8 |  |  |  |
| 5120 | •j o i n t | 13 to 12, i : w\*oue 12 | i n | 4 o n | 0,3 5 : m o • e 12 |
| 1,8, | 5!Sleep 2.60 | |  |  |  |
| 5130 | 1f sPr ite(12)1>- 48 t heh | | r e t’ | ur n el | S e 9 o t o 5 1 3 0 |

5480 if sPrite<12> 1) \*250 then goto $ S T R R else goto 5480 5490$STfiR

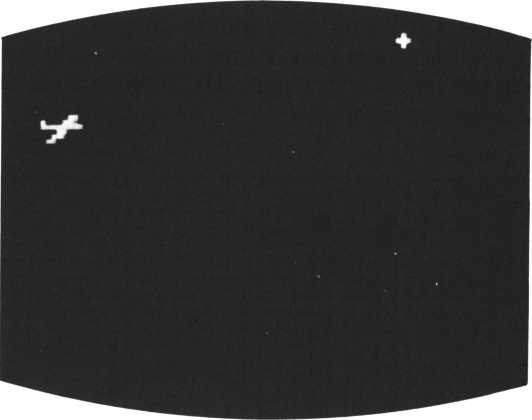
5510 stchr M 4 0 e 0 4 0 0 8 8 0 0 0 0 0 0 8" to 140 5520 stchr "8000808000000000" to 141 5530 stchr M8880880000080800" to 142 55 4 8 stchr "8 0 8 8 8 0 0 0 0 0 0 0 8 8 0 0" to 143

5558 scod 22 > 148:scol 22?&8B:loc 22 to 240,0:moue 22 in eP -2 ? 1 ? 1

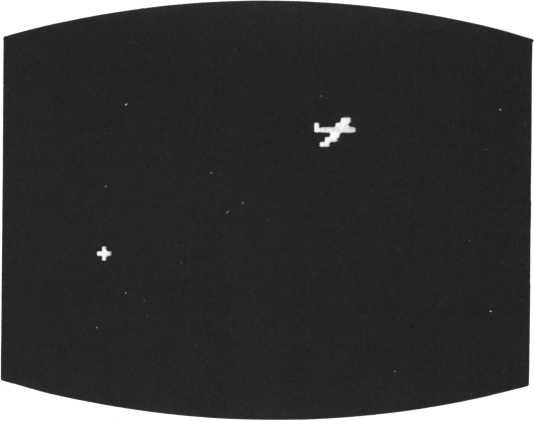
7 st 5568

55 68 if s P r i t. e < 2 2,8 > = 6 2 then erase1 22\* return else goto 5570$LIGHT s L = L + 1

5 5 88 if L mod 2 = 8 then scol 13 > & 0 ft else scol 13,8 5598 return



PLANE AND STAR (I)



PLANE AND STAR (2)

THEN DRAW A COUPLE

The boy steps up to the girl and holds her hand. When the people appear, the palm trees disappear; that’s because no more than four sprites can appear at once on a single line.

54 50 Joint 15 to 14,35 wove 14 in 2 on-30,95:move 14 in 3 to 68,95

10! ##### PALM #####

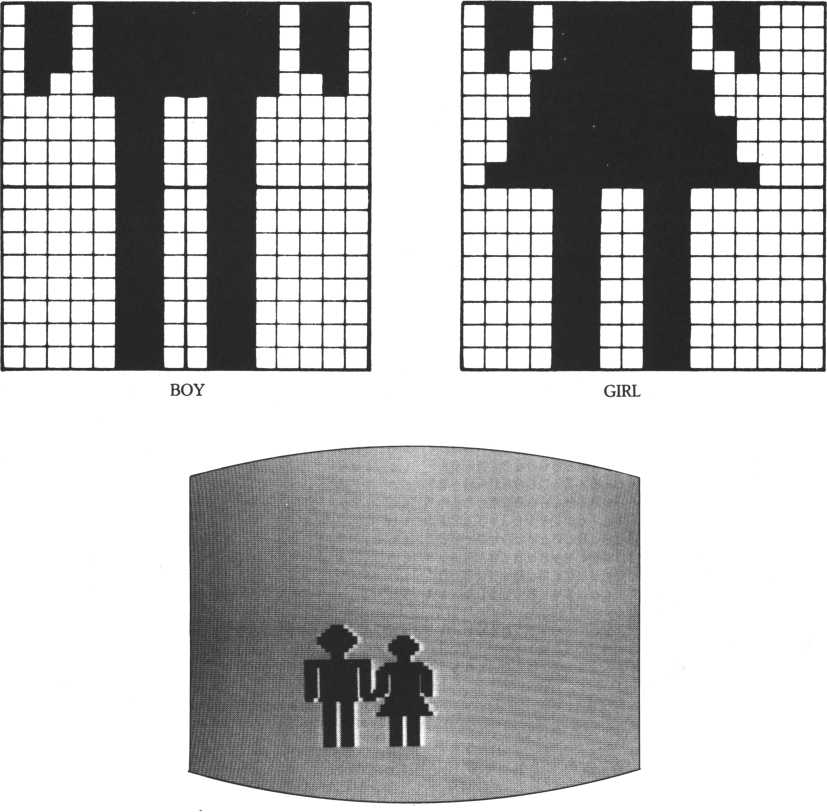
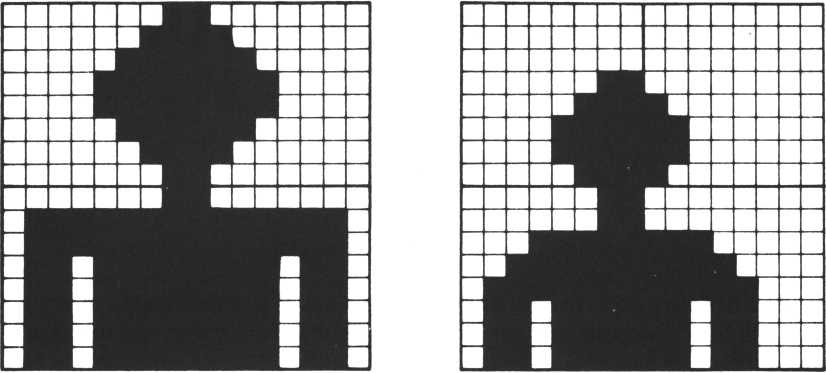
5140\*COUPLE 5150! \*\* COUPLE \*\*

5160! \* \* BOV \* \*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 5170 | stchr | "0163078f8f070381" | t 0 | 152 |  |
| 5180 | stchr | \* 8 1 7 f 7 f 6 f 6 f 6 f 6 f 6 f " | t. o | J |  |
| 5190 | st ch r | H 8 0 c 0 e 8 f 8 f 8 e 8 c 8 8 8 " | t. o | 154 |  |
| 5280 | stchr | " 8 8 f e f e f 6 f 6 f 6 f 6 f 6 " | t- o | 155 |  |
| 52 10 | stchr | ” 6 f 6 f 6 f 4 f 8 6 8 6 8 6 0 6 " | t 0 | 156 |  |
| 5220 | st ch r | " 06 0606060 6860606” | t 0 | 157 |  |
| 5230 | stchr | " f 6f 6f 6f 260606860'1 | t 0 | 158 |  |
| 52 4 8 | stchr | "6868686060606068 " | t 0 | 159 |  |
| 5250 | scod | 14,152:scod 15,156: | scol | 14, | 1:scol |
| 5260 1 | ! \* \* GIRL \* \* | |  |  |  |
| 5270 | st ch r | "80 08 000 3070 f 8f 87" | t o | 168 |  |
| 5288 | stchr | "03831f3f7f6f6f6f" | t. o | 161 |  |
| 5298 | stchr | H0000000080C0C080" | t 0 | 162 |  |
| 5308 | stchr | " 8 8 8 0 e 8 f 8 f 8 d 8 d 8 d 8 " | t 0 | 163 |  |
| 5318 | stchr | " 6 f 6 f 4 f 1 f 1 f 3 f 3 f 7 f " | t. 0 | 164 |  |
| 5320 | stchr | " 0 C 0 C 0 C 8 C 0 C 8 C 0 C 0 C " | t o | 165 |  |
| 5338 | stchr | Hd8d8c8e0e0f8f8f8" | t 0 | 166 |  |
| 5340 | st ch r | "C0C0C0C0C0C0C0C0" | t o | 167 |  |
| 5350 | scod | 16,168:scod 17,164: | S C 0 1 | lb. | 1:s co1 |
| 5368 1 | 1 \*\* JOIN HANDS \*\* | |  |  |  |
| 5378 | stchr | "6f6f6f4f06060606" | t o | 168 |  |
| 5380 | stchr | "8606060606860606" | t o | 169 |  |
| 5398 | stchr | " f6f6f7f368606060" | t 0 | 178 |  |
| 5480 | stchr | "6060606068686060" | t o | 171 |  |
| 5410 | stchr | " 6 f e f c f 1 f 1 f 3 f 3 f 7 f " | t o | 172 |  |
| 5420 | st ch r | "0c8c8c8c8c8c.8c0c" | t o | 173 |  |
| 5430 | stchr | " d 8 d 8 c 8 e 8 e 0 f 8 f 8 f 8 " | t o | 174 |  |
| 5440 | stchr | "C0C0C0C0C0C0C0C0" | t o | 175 |  |

1. Joint 17 to 16,3:wove 16 in 4 on 100,95:sleep 10,18
2. 9osub SSUNSET:sieeP 3,38

5470 if sP r i t. e < 1 4 , 1 > =68 then SCRE8DV , 1 68 : scod 17,172 else 9 o to 5470



COUPLE

A FUTURISTIC TOWN

—Drawing spaceships

Spaceships of different sizes fly across the screen. The large spaceship is made up of four sprites. To smoothly erase the sprites from the screen, MOVE-STEP commands are used.

18! ##### TOWN #####

26 0 J«inke9<l):if J =1 then 9osub $ 5 P R I T E

278 I = i n k e V <. 0 > • if 1=7 then 9oto 20 else 9oto 260

5000$SPRITEsma9 2

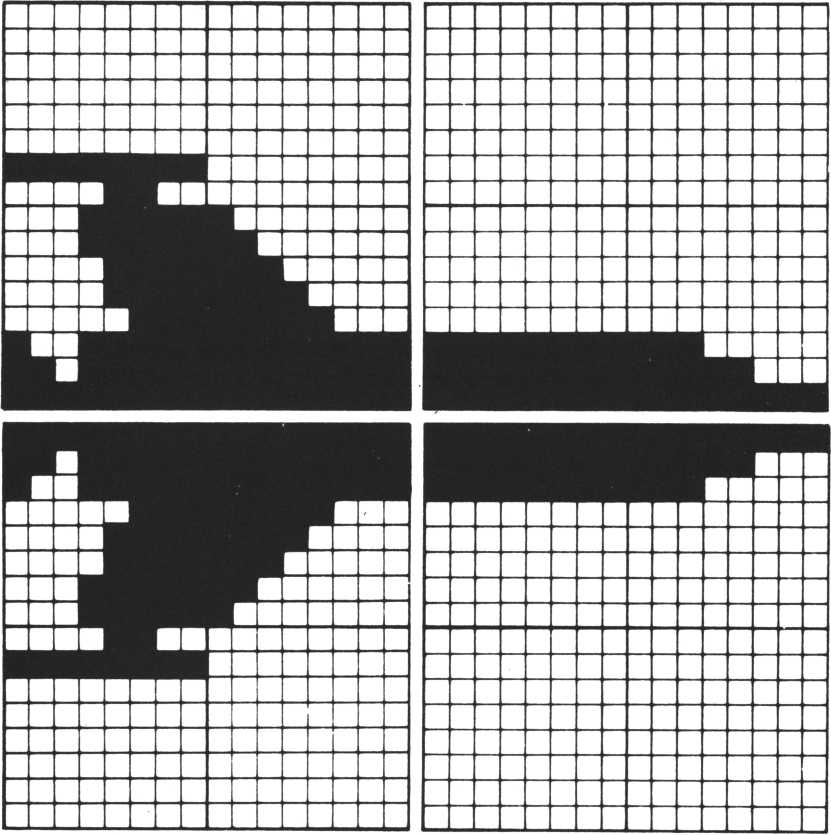
5220 scod 0 » 10 01 s c© d 1,104:scod 2» 108s scod 3 > 1 1 2 : s c o d 4,116 5230 scol 8,9 : s c o 1 1 j 9s scol 2 > 9:sco1 3,9 : s c o 1 4,10

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 5010 ! | ! \*\*\*\*\* | UFO \*\*\*\*\* |  |  |
| 5020 | stchr | 11 000000000000 4 i 0C” | t 0 | 100 |
| 5030 | stchr | "1H f 0\* 0f 079 f d i i i " | t 0 | 101 |
| 5040 | stchr | ”0000000000000000" | t o | 102 |
| 5050 | stchr | ”80c0e0f0f8fH f H" | t 0 | 103 |
| 5060 | stchr | ”0000000000000000” | t o | 104 |
| 5070 | stchr | " 0000000000f f HfP | t 0 | 185 |
| 5080 | stchr | M0000000080000080M | t o | 186 |
| 5090 | stchr | M0000008000e0f8f f “ | t 0 | 187 |
| 5100 | stchr | MfMff8000000800" | t o | 188 |
| 51 10 | stchr | ”0000000000000000” | t o | 109 |
| 5120 | stchr | ” i t f 8e00080000000" | t o | 1 18 |
| 5130 | st ch r | "0000000000000000" | t 0 | 1 1 1 |
| 5140 | stchr | "Hdf91070t0Hf H" | t 0 | 1 12 |
| 5150 | stchr | "0Cf i 000000000000” | t o | 1 13 |
| 5160 | stchr | "HHf H‘310e0c030" | t o | 1 14 |
| 5170 | stchr | ”0000008000000000" | t o | 1 15 |
| 5180 | stchr | "8001030307070707" | t o | 1 16 |
| 5190 | stchr | "070f013fi <3f0705" | t o | 1 17 |
| 5200 | stchr | "00008080C0C0C0C0" | t o | 1 18 |
| 5210 | stchr | "C0e0e8f8ief8c040" | t. o | 1 19 |

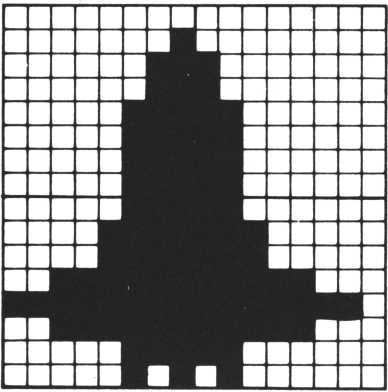
5240 joint 3 to 2,4 : j o i n t 2 to 1?3s Joint 1 to 0,2 5250 move 0 on-30,100:1oc 4 to 150,180

1. move 8 in 1 step 1 > 0,2: m o u e 4 in 2 step 0,-1
2. 9 osub $RUSMINSTRftIN 5263 9 osub $UFOTRKEOFF

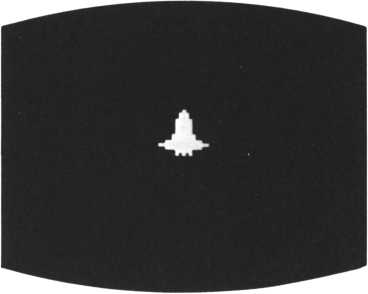
5270 i i s t a t u s < 1) = 1 or status(2)=1 then 9oto 5270 612 0 Print "Wiret u r n



SPACESHIP (LARGE)



SPACESHIP (SMALL)



SPACESHIP

NOW DRAW A HOT-AIR  
BALLOON

A hot-air balloon flies skyward from the screen’s bottom left to top right. The balloon is drawn in three colors. Four sprites of different colors were used to form the balloon. MOVE-STEP commands are used to create movement.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 ! | ##### TOWN \*#### |  |  |  |  |
| 268 | J-1 | t h | en | 9 o s u b | fSPRIT£ |
| 270 | I=inke9<0>\*i\* 1=7 | t h | en | 9 o t o | 20 else 9 o to 260 |

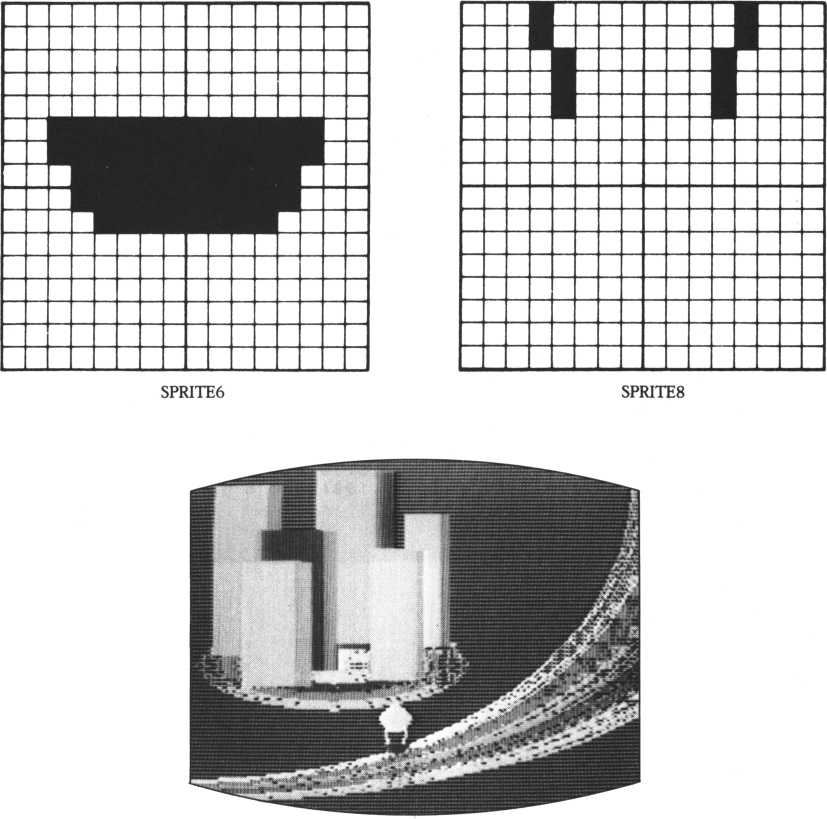
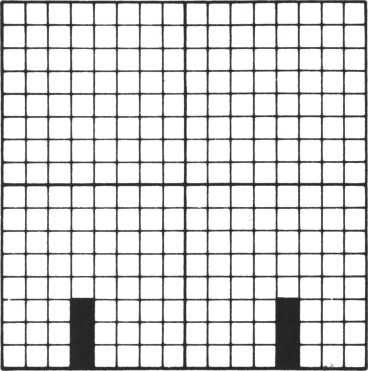
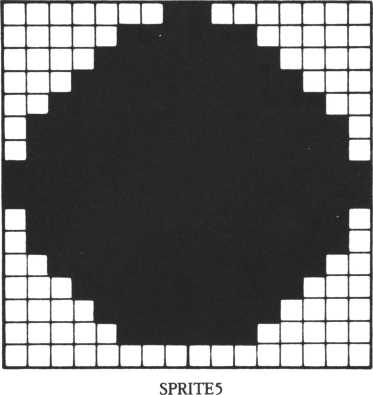
5450 scod 5 > 12 0:s c o d 6,124tscod 7 >12 8:scod 8,132 5460 scol 5, 10 s sco1 6, 12\* scol 7,15\*sco1 8,15

5288! \*\*\*\*\* BIR BALLOON \*\*\*\*\*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 5290 | stchr | ”01070\*1\*3\*7\*7\* \* \*” | t o | 120 |
| 5300 | st ch r | ”\* \*7\*7\*3\* 1 \*0\*0700" | t 0 | 121 |
| 5310 | st ch r | ”80e0\*0\*8\*c\*e\*e\* \*" | t O | 122 |
| 5320 | st ch r | " \* \* \*e\*e\*c\*8\*0e000” | t 0 | 123 |
| 5330 | st ch r | ”00000000003\*3\* 1 \*” | t 0 | 124 |
| 5340 | st ch r | " 1 \*0\*000000000000" | t O | 125 |
| 5350 | st ch r | "0000000000\*C\*C\*8” | t O | 126 |
| 5360 | st ch r | "\*8\*0000000080000" | t O | 127 |
| 5370 | stchr | "0000000080000000" | t O | 128 |
| 5380 | stchr | ”0000800000101010” | t O | 129 |
| 5390 | st ch r | ”0000000000000000" | t 0 | 138 |
| 5400 | stchr | "0000000000080808” | t O | 131 |
| 5410 | stchr | ” 1010080808000000" | t 0 | 132 |
| 5420 | stchr | "0080000000000000" | t 0 | 133 |
| 5430 | stchr | "0808101010000000" | t O | 134 |
| 5440 | stchr | "0000000000000000" | t. o | 135 |

5470 Joint 8 to 7»3\*Joint 7 to ? 5' Joint 6 t o 5,3

1. sleep 9,9? wove 5 in 3 on 50, 190!move 5 in 4 step 2,-1,8
2. 9 o s u b fRAILRQADCROSS 6120 Print " M" i return



ADD MORE BALLOONS

A multicolored array of balloons fly into the sky. All balloons are the same size; two patterns for string form part of the program. The balloons’ flying times ar controlled by the SLEEP command, which flies the balloons in order and prevents more than four sprites from occupying the same line.

10! \*•\*\*» TOWN ##\*\*\*

260 j\*ink«y < D : if J = 1 then 9osoJ> iSPRITE

27 0 I = i n k e y < 8 > : i f 1=7 then goto 20 else goto 260

5000\*SPRITE: mag 2

58 10! \* \* \* \*♦ BALLOON \* \* \* \*\*

5820 stchr " 00000 1 070 fl f 3 f 3 f " to 156

5830 stchr “ 7 f 7 f 3 f 3 fl f 0 f 070 1 '\* to 157

5840 stchr "000088e0f0f8fcfc " to 158

5850 stchr " f e \* e f c f c f 8 f0e888\*

to 159 t o 1 6 0 to 161 to 16 2 t o 1 6 3 t o 1 6 4 to 165

58 6 0 StChr "0000 00010100 8000'

5870 StChr "0000000000000001•

58 80 st ch r "8080800000804048\*

5898 stchr "4040404040408800\*

5 9 0 0 StChr "010100 00080© 0101\*

5910 stchr "0101810101010100'

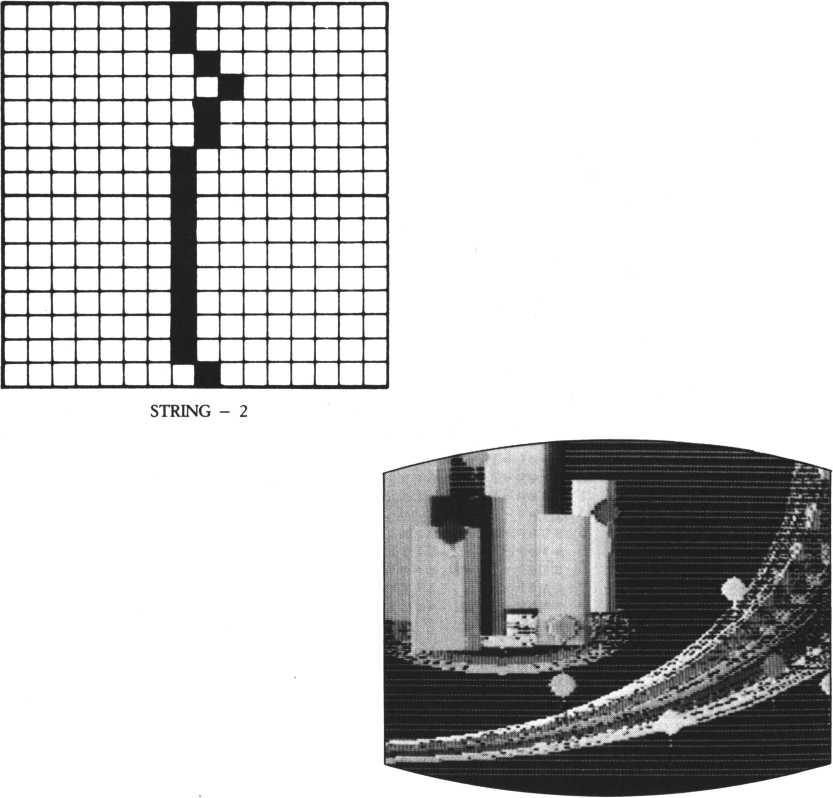
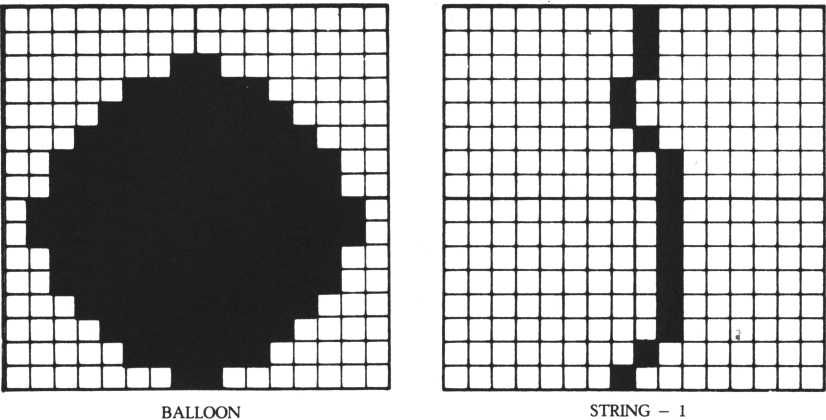
5920 stchr "0000884088880000" to 166

5930 stchr "0000000088080080" to 167

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 5948 | for I | = 12 | t 0 |  | 2 1 !s c o d I? 156!next |  |
| 5958 | for I | = 22 | t 0 |  | 26!scod I >160!next |  |
| 5960 | for I | = 27 | t 0 |  | 3 1 ! scod I >164!next |  |
| 5978 | 1 = 12! | for | J = 2 | | to 1 1 :sco1 I , Ji scO1 1 + 10 | |
| 5980 | for I | = 12 | t 0 |  | 2 1 i • j o i n t 1 + 18 to I) | ' 3i next |
| 5998 | K = 0 \* for | | 1 = 12 | | to 21s move I in K | on 258 |
| 6000 | if s t a t us(9 > | | | | =1 then goto 6000 |  |
| 6010 | nfoue | 12 | i n | 0 | s t e P - 1 , - 1 , 6 : s 1 e e P | 2,5 |
| 6020 | m o u e | 14 | i n | 2 | s t. e P - 2 > - 1 > 6 i s 1 e e P | 6,5 |
| 6030 | move | 13 | i n | 1 | step-1,-2,4!sleeP | 6,5 |
| 6840 | move | 15 | i n | 3 | steP-l,-3»7isleep | 2,5 |
| 6050 | move | 16 | i n | 4 | s t e P - 2 > - 5,8 ! s 1 e e P | 3,5 |
| 6860 | move | 17 | i n | 5 | s t e P - 5 > - 2,5 s s 1 e e P | 2,5 |
| 6070 | move | 18 | i n | 6 | s t e P-3,~2> 6 is1eeP | 4,5 |
| 6080 | move | 20 | i n | 8 | step-3,-l,5isleep | 3,5 |
| 6090 | move | 19 | i n | 7 | step-1,-1,4!sleep | 2,5 |
| 6100 | move | 21 | i n | 9 | s t e P - 2 , - 3,7 |  |
| 6181 | g o s u b | ♦TELEPHONE | | | |  |
| 61 10 | if s t a tu | | s < 0 > | | =1 then goto 6118 |  |

6128 Print "M"!return

on 258 ? 1 98 s K \* K + 1 : n ext. I



BURST THE BALLOONS!

A bird flies alongside a balloon and bursts it with its beak.

Two patterns of the bird are prepared as shown below; one with its wings up, the other with its wings down. To move the bird, its sprite number is given as 10 and sprites are alternated using the SLEEP command.

To burst the balloon at the moment of sprite collision, the program skips to the ON COINC GOSUB statement subroutine and changes the sprite of the balloon to a burst-balloon image.

\*\*\*\*\* BIRD RHD BftLLOOM \*\*\*\*\* if status(4)=l then 9oto 5580 stchr 11 000000800000 00 f f " to" 136 ”7fffif If If Iflcl8" to 137 " 00 0 0000 0 0 0 00 00 f 0 11 tO 138 "d8f8808000080000“ to 139 "181 clf1f1f1fff7f" to 148 "ff08000000000000“ tO 141 " 000000008880 f 0d 8\*\* to 142 Mf000000080080080" to 143 “000001070f1f3f3f " to 144 "7f?f3f3f1f8f8781" to 145 “8 © 8 8 8 8 e 8 f 8 f 8 f c f c “ to 146 tt f ef ef cf cf 8f 0e88©‘\* to 147 " 000000000 0000000" tO 148 "0000000000000000" t-O 149 "8 0 8 0 8 8 c 8 6 8 2 8 2 8 2 0" to 150 “2860404840484040" to 151 “8088181800000177" to 152 "0100081818000880" to 153 M80808c8c88S8c0f7“ to 154 "€ 8 8 88 88 c 8 c 8 0 8 8 8 8" to 155 11?144!scod 12?148

1. 1 ? $88! scol 1 2 ? St82 2$ to 12! joint 12 to 1 1 ? 3
2. in 8 on 255?1 4 8 ! m o v e 11

10' \*\*«#\* TOWN #####

268 J\*inkey<l>iM T = 1 then 2 7 0 I \* i n k e 9 03 > 5 i f 1=7 then 5000$SPRITE: ma9 2

9 osub 9 o t o

$ S P R I T £

20 else 9 o t o

260

5490 ! 5500 5510 5520 5530 5540 5550 5560 5570 5580 5598 5608 5618 5620 5638 5640 5650 5660 5670 5688 5690 5700 5710 5720 5730 5748 5750 5760 5770 5780 5790 6120 6122 6124

stchr stchr st ch r stchr st chr stchr stchr stchr stchr stchr stchr stchr stchr stchr stchr stchr stchr stchr stchr scod

SCO 1 Joint move scod move sleep on i f

in 1

1 8 ? 1 3 6 i s c o 1 1

18 in 2 step 1? 8 2 ? 5!scod 18? 148 c o i n c 9os ub 5888 s t a t u s < 2)\* 0 then

? Z< 8 F : 1 o c 10 t o \* 11

3!coinc sleep 2 ?

o n

5!scod

t o 58

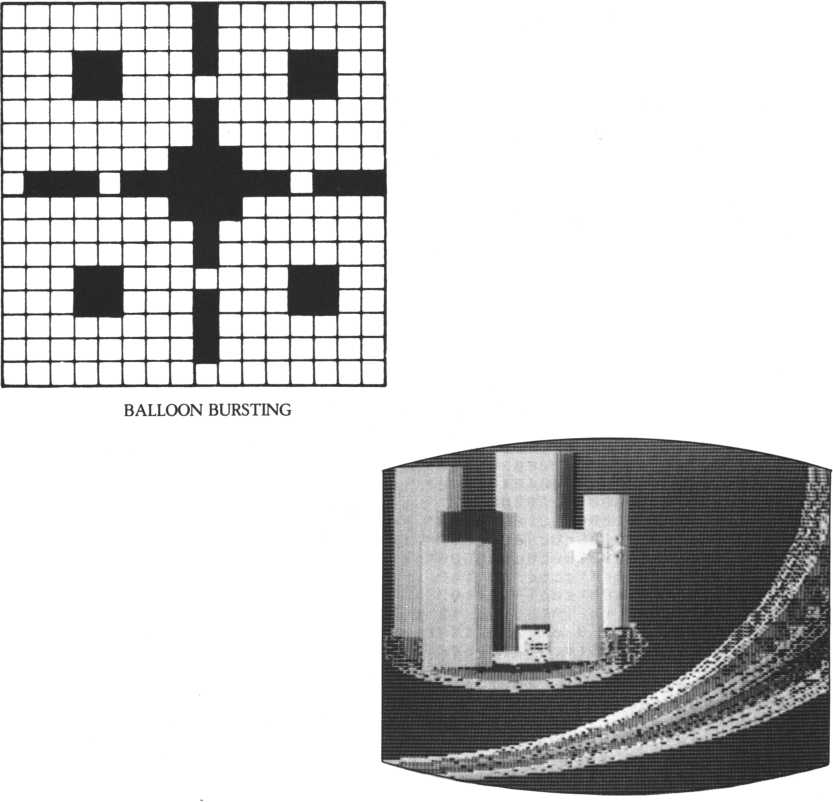
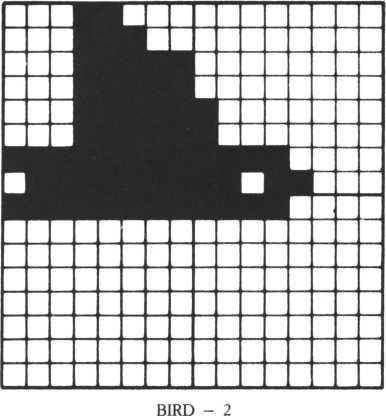
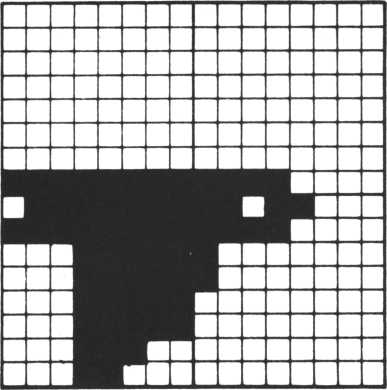
189?51 ?5

18?136

9 o t o 5 818 else 9ot o 5778

Print Joint s 1 e©P

"W" s r et Or n 12 to llserase 12:scod 11?152 3?5serase 11:coinc offsreturn



**IV**

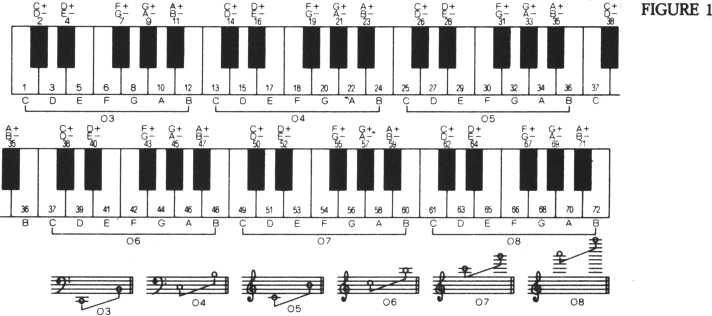
**MAKING**

**MUSIC**

CREATING MUSIC AND  
SOUND EFFECTS

To compose a piece of music, you must choose an interval and a sound length. These are specified with a PLAY command. Here the PLAY command will be explained in detail.

Th^re are two ways of selecting intervals: by the letters O and A to G; and by the letter N and numbers 1 to 72. The range of intervals is six octaves, the same as a 72-key keyboard (see figure 1).



Notes and how to specify them FIGURE 2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| note name | Dotted whole note | Whole note | Dotted half note | Half note | Dotted quarter note | Quarter note |
| Musical symbol | o. | © | i | J | J. | J |
| PLAY | 1. | 1 | 2. | 2 | 4. | 4 |
| specification |  | 64! | 48! | 32! | 24! | 16! |
| note name | Dotted eighth note | Eighth note | Dotted sixteenth note | Sixteenth note | Dotted thirty-second note | Thirty-second note |
| Musical symbol | J) |  | J> | J | ~T~ | > |
| PLAY | 8 | 8 | 16. | 16 | 32. | 32 |
| specification | 12! | 8! | 6! | 4! | 3! | 2! |

Example:

|  |  |  |  |
| --- | --- | --- | --- |
| Half-note triplet | r-9-\*  J J J | 32=114-11-1-10 | Ex.: C11 I C11 I C10 ! |
| Half-note quintuplet | i—5—»  nm | 32=74-7-1-6 + 6 + 6 | Ex.: C7IC7IC6IC6IC6I |
| Quarter-note triplet | /—3—\  JTJ | 16=5 + 5 + 6 | Ex.: E 5 ! E 5 ! E 6 I |
| Quarter-note quintuplet | /—5—\ j j j j j | 16=4 + 3 + 3 + 3 + 3 | Ex.: A4! A3! A3! A3I A3I |

There are two ways of setting sound length: normally it’s done with numbers, but for triplets or quintuplets an exclamation mark (!) is added to the number (see Figure 2).

R is specified instead of a letter from A to G for a rest. Now you are ready to compose a piece of music. Key in as follows:

PLAY‘ ‘05C4C + 4D4D + 4E4F4F + 4G4G + 4A4A + 4B406C4’ ’

[RETURN]

The notes go higher as the line progresses. These sounds are all enclosed by 05 in Figure 1. Key in the following to do the same as the above example:

To produce the sound of a chord, key in two or three symbols enclosed by quotation marks, inserting commas between them.

PLAY “C1”,“E1”,“G1” [RETURN]

This produces the chord CEG. For volume, envelope and staccato check your manual.

Sound effects are sounds other than music, such as waves, explosions, or even the sound of a UFO—or other sounds essential for games or animation. An SG command is used by the M5 to produce sound effects. The general format of an SG command is as follows:

SG a,b,c

A number between 0 and 3 is specified for “a”. If you specify 3, noise is produced. If you specify one of the other numbers, the same sound is produced as that of the PLAY command. You can create all sounds simultaneously by using SG four times. If you specify “a” to be other than 3, specify a number between 1 and 1023 for b as a frequency division ratio. If “a” is specified as 3, specify a noise number from 0 to 7. Specify a number fom 0 to 15 for “c” as volume. 0 is loudest; 15 is inaudible. You may omit specifications for “b” and “c”, but not for “a”. Leave all the commas in, however.

CREATING A SONG OF  
THE SEA

Here is a piece of frightening music for the sea scene: sounds of scuba equipment, sonar signals, and the shark appearing.

The shark’s incidental music is a duet between low and high notes. The high notes gradually become louder and the overall speed increases. Finally, underwater noises are produced.

• IF YOU USE THE SOUND OF SCUBA EQUIPMENT:

5861 GOSUB IPutterputter

20000$PUTTERPUTTER

20010 for 681=500 to 300 steP-40:for BflJ=BfiI to BfiI-200 step -5:s9 0,BflJ,<B8I-BfiJ>/20 20040 nextis9 0» >0?nextsreturn

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 2 0 1 0 0 $ S U B M R I N E S 0 U N D |  |  |  |  |  |
| 20110 s9 0,120,IStsleeP 2,2:s9 B  50: s9 0 ? > 1 5-PI I 1/30  20150 next: sleep 1,6 0 s r e t u r ri | .? 0s sieeP | 3 ? | 2: for | 811=0 1 | t o 4 |

• IF YOU USE THE SOUND OF SONAR:

6261 GOSUB $ Sound of submarine 6321 GOSUB $ Sound of submarine 5001 GOSUB $ Shark

Note: To produce sounds, delete all REM (!) statements in the main program before inputting the subroutines’ corresponding sounds.

To input a musical piece for the shark, use multi-statements as much as possible. If you input it as is, the memory capacity will not be adequate.

?0000$SHRRKSOUND

20010 for 861=0 to 50 step 5: $ 9 0,1017,15:®\* 1,438,5 + 86 I/5i f

or 8 6 K = 0 to 2580-861\*45!next

20060 s\* 0,960,15ts\* 1,419,5 + fi'SI/5Mor 8 6 K = 0 to 2500-861\*45:

next: next: 5- g 0 , , 0 : $ 9 1 > , 8

20130 s:9 3,7, 15: for 861\*10 to ?0:for 866 = 0 to 100:r»etft:s\* 2, 86 I,0:n ext

20190 for 861=0 to 30 0: s 9 3 , \* 1 5 - 8 61 / 2 0 : n e x t : % 1 e e !> 1,3 0: r e t u r n

CREATING A SONG OF  
THE EVENING SUN

This musical piece gives the feeling of an airplane flying at twilight. We also wrote code for an ambulance and the sound of a UFO taking off, so that you can use them for some other programs.

This piece is a duet run by a PLAY command. The sound of the airplane here is quite different from that used in the town sequence, because the mood is different. We’re setting the atmosphere for the appearance of the couple. Delete the REM(!) statements on the main program before inputting the subroutines concerning sounds because of insufficient memory capacity.

**22200\*SUNSET**

**22210 Pla« "t139sld4f4a4a2e4","s1d4d4d4d2c4" 22220 Play "c4\*494y2M“> “c4c4c4d2d4"**

**22230 Play ” d4f4a4a2e4",“d4d4d4d2c4 “**

**22240 Play Bc4e4c4d2r4"> oc4c4c4M.el0.dl6.r8. “ 22230 Play “ Ma4o6c4c2o594n,ud4d4d4d2c4"**

**22260 Play "»494b-4b-2a4"," c4c4c4<4.e16.d16.r8. 22270 Play “d4Ma4a2«4", “d4d4d4d2c4“**

**22280 Play "c4#4c4d2r4","c4c4c4♦4.#16. d16. r4 “ 22290 return**

X

**22000\*PLftNE 22018 s9 2,44,0 22020 s9 3,7,0 22030 for PR I \* 0 to 60 22040 s 9 3, ,PRI/7 22050 next 22060 sleep 9,60 22070 for PRI«0 to 68 22080 s9 3,,8-PRI/7 22090 next 22100 return**

85H\*5SB8V?mi5

22320 s9 1,102,15

1. sleep 1,40
2. s9 0,127,15

22350 s 9 1,128,15

22360 sleep 1,40

22370 if ink©9<8>«8 then s9 22380 9oto 22308

0»,0:s9 1 , , 8:r et ur n

22700\*UFOTAKEOFF

22710 for ABIM88 TOO 1 steP-1

22720 s 9 2,ABI,0

22730 s9 3,7,15

22740 sleep 1,5

22750 next

22760 for ABI = 1 to 5 22770 s9 3,7, 15 22780 sleep 1 , 18 22790 next

22800 for ABJ =15 to 0 steP-1

22810 s9 3,7 > AB J

22820 s1eeP 1,15

22830 next

22840 return

* IF YOU USE THE AIRPLANE SOUND:

5130 GOSUB $Airplane2 [RETURN]

* IF YOU USE THE SONG:

5461 GOSUB $Eveningsun: SLEEP 3,30

Note: To stop the sound of the waves press the RETURN key. To go to the next step, press the SPACE key. If both the sound of the waves and the sound of the airplane are input, memory is filled to capacity. If you want the song, input it only, without the sound effects.

CREATING A SONG OF A  
FUTURISTIC TOWN

We’ve written code for six sounds: a telephone, a railroad crossing, a jet plane, spaceships landing and taking off, and balloons bursting. The jet, UFO and balloon sounds match the visual scene. Modifications to the program are listed below.

• IF YOU USE THE JET SOUND:

5261 GOSUB $ Airplane overhead

|  |  |  |
| --- | --- | --- |
| 24420 | S9 1 , | 208,15 |
| 24430 | sleep | 3, 1 |
| 24440 | S9 1 , | , 0 |
| 24450 | sleep | 1 , 1 |
| 24460 | next |  |
| 24 47 9 | sleep | 1 |
| 24480 | if in | k ey C 0> \*8 then retur n |
| 24490 | goto | 24400 |

24400$TELEPHGNE 24410 for 8 H I - 1 to

20

24500SCROSSING 24510 for 8X1=0 to 150 steP s9 0,180,15-8X1/10 S 9 1,180, 15-8X1/18

24528

24530

24540

24550

24560

24578

S9 2,180,15-8X1/10 next

2 4 0 0 0 $ P L 8 N E

if ink e y < 8 > » 8 then return goto 24500

24010 s9 3,7, 15

24020 for 8WI«0 T 500 step 2

24030 s9 2,8W I . 0

24040 s9 3, , 15-ab s < 8MI/48-8>

24850 sleep 1 , 1

24060 next

24070 s9 3,,0

24880 return• IF YOU USE THE SOUND OF SPACESHIPS

5263 GOSUB ISpaceships landing and taking off

24680$UFQLPNDIN6 24618 for PMK=18 to 488 step 4 24628 PMP=PMK:PMB=PMP+50:PMC = 4 24638 for PMI=0 to 1

24648 if PMI = i then PMW = PMP s PMP=PMB : PMB=PMW : PMC=-4

24650 for PMJ=PMP to PMB steP PMC

24660 s9 0 , PMJ,4 + PMK/48

24678 next PMJ

24688 next PMI

24698 next PMK

24700 s 9 0 , » 0 • s 9 1^0

24710 return 24100$UFOTPKEOFF

24110 for PLK~400 to 10 steP-4 24128 PLP«PLK:PLB=PLP+50:PLC=4 24130 for PL I-0 to 1

24140 if PL I -1 then PLW^PLPiPLP = PLB:PLB = PLW:PLC=-4

24 150 for P L J = P L P to PLB steP PLC

24160 s 9 0,PLJ,4 + PLK/48

24178 next PLJ

24188 next PLI

24190 next PLK

24200 s9 0, , 05 S9 1 , , 8

24218 return

• IF YOU USE THE SOUND OF BALLOONS BURSTING:

5801 GOSUB $Bursting balloons

24300$BPLLOON 24310 s9 2,10,8 24328 s 9 3,7,15 24338 for PV1=8 to 68 24348 s 9 3, , 15-PVI/4 24358 next 24368 return

**WHAT'S ANIMATION?**

The word “animation” comes from the Latin word “anima”, which means “life.” Animation is the art of showing the natural movements of living things and objects in the world.

Animation wouldn’t he possible if movement wasn’t governed by laws of physics and natural patterns. This means that there is a certain amount of order and repetition in movement, which makes the art of replicating it easier. The earth revolving around the sun, the alternation of night and day, the alternate motion of arms and legs, waves rolling in and out—these are all repeating patterns. Once you’ve reproduced the pattern in animation once, you can repeat the pattern as many times as you need to tell a story or to set a mood. Acceleration of a falling object, or inertia of a rolling object, are examples of laws at work.

You can use simple math to express these laws in movement on the computer screen.

Most movements that you’ll want to show consist of a cycle that includes several patterns. These cycles make up the basic structure of personal computer animation.

To make a movement look realistic, you’ll use “insertion”—that is, you’ll insert several pictures that bridge the movement’s beginning and end. To show rapid motion, you’ll use fewer pictures; to show slow motion, you’ll use more.

MAKING CHARACTERS

MOVE

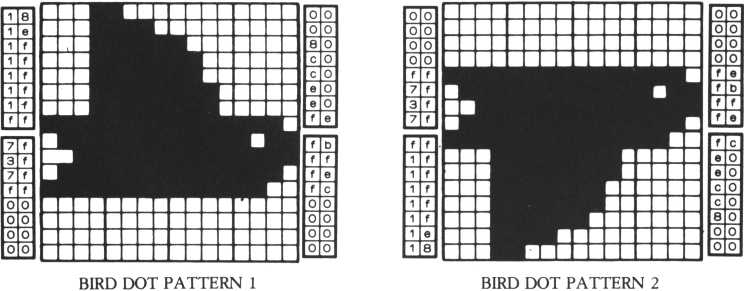
A FLYING BIRD

A flying bird

Here’s a simple example of a moving character: a bird composed of two sprites. The two sprites will move together across the screen simultaneously; but first one sprite, then the other will “melt into the background’’ by taking on the color of the rest of the screen.

The program for wing movement is listed below. A dot pattern for half the bird is made sprite 0; the other half is made sprite 1. At line 90, sprite 0 is specified as green, while sprite 0 is left colorless.

The bird moves from left to right by the MOVE command at line 120. Line 130 changes sprite color alternately during the bird’s flight. In this IF statement, if sprite 0 is colorless it is made green and sprite 1 is made colorless. If sprite 0 is green, it is made colorless and sprite 1 is made green. This alternation is repeated via a loop between lines 130 and 160.



10i \*\*\*\*\*\*\* bird \*\*\*\*\*\*\*

20 Print "HIM" : ma9 2

1. restore 180
2. for 1=128 to 135 50 read ft $

80 stchr ft $ to I 70 next I

8 0 scod 0? 128\*scod 1 > 1 3 2 90 scol 0\* 2\* scol 1 » 0 10 0 Joint 1 to 0 ? 1 110 loc 0 to 0 » 80

1. 0 move 0 in 0 s t e P 1> 0 >2
2. 0 if s P r i t © < 8 > 3)\* 0 then scol 0 > 2 s s c o1 1 » 8 else scol 8 > 8 s s c

ol 1,2

1. 0 if s P r i t. e < 8 » 1 > > 2 4 8 then goto 110 150 sleep 10?1

160 9oto 138 178 end

188 d at a 181 © 1f i f1f1f1f f f > 7f3f 7f f f08 808008,088088c0c0e0e0fe, fb f f f ef C08000000

198 data 08080080f f7f3f7f, f f1f t f1f1f1f1e18,80 000080fefb f f f©, fce8e8c0c0808000

If the loop is repeated too quickly, the movement is too fast to be noticeable. That’s why line 150 is there to slow the movement down.

When you add this program to the main program, you can add an another process instead of a SLEEP command. Move another bird by changing data at lines 180 and 190. Two dot patterns are used here. If you use three or more dot patterns, the movement becomes more natural.

HUMAN MOVEMENT

/

Here’s another way to show a character’s movement: switching the screen location of a sprite, or making one sprite seem to disappear “beyond” the screen. In this example, we’ll use the switching of locations.

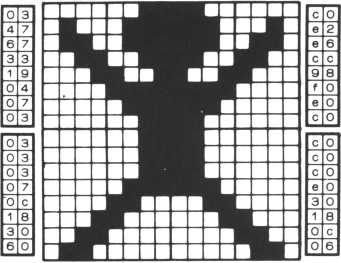
In this program, a man moves his arms up and down. Sprite 0 is a dot pattern of he man raising his arms up; sprite 1 is another dot pattern of the man with his arms down. Sprite 0 is displayed in the upper part of the screen (line 100); sprite 1 is displayed at the bottom of the screen (line 110). In lines 130 and 140, sprite 0 is displayed below (instead of above), and sprite 1 is displayed above (instead of below). Repeating this process gives the illusion that the man’s arms are moving.

If this process is performed too quickly, the movement is too fast to see.

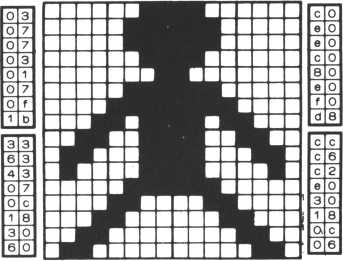
Lines 120 and 150 provide waiting time. Or: when you add this program to the main program, you can input another process, instead of the SLEEP command.

The movement in this program is made by the loop between lines 100 and 160. If you use an IF statement (as in the previous flying-bird program), you can decrease the number of LOC statements from four to two and decrease SLEEP statements from 2 to 1, thereby shortening the program.

We’ve written the program this way so that you can clearly see the principle involved. When you run the program and watch it on the screen, cover the upper part of the screen with a piece of paper. The man in the lower part appears to be moving. Later, revise this program by “hiding” one sprite “outside” the screen.



MAN DOT PATTERN 1



MAN DOT PATTERN 2

scod scol 1 oc 1 oc 1 sleep 1 oc 0 1 oc 1 s 1 eeP 9 ot o

**end**

10 20 **30 40 50 60 7 0 80** 98 100 110 120 **130 140 150 160 170 180**

\*\*\*\*\*\*\* man \*\*\*\*\*\*\* Print \*mm\* i ma9 2 restore 188 for 1\*128 to 135 read fit stchr fit to I next I

0 > 128s scod 1 0 > 4:sco1 1,4

1. to 108,48
2. to 100,120 18,1

to 180,128 to 100,40 18, 1 100

**132**

**data 83476733198f0703> 038303070c183060,C0e2e6cc98f8e0c8 C0c0c0e038180c06 198 data 0387070381070 I 1b: ccc6c2e038180C06**

**•336343870c183860,c8e0e0c080e0f0d8**

.

MOVING IMAGES ACROSS THE SCREEN A SAILING SHIP (LOC)

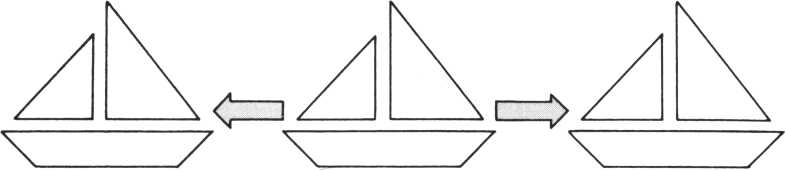
There are two commands we could use to show a sailing ship: LOC and MOVE. Here we’ll show you how the LOC command works.

Type in the program listing below and run it. Pressing the right-arrow key moves the ship to the right; pressing the left-arrow key moves it to the left.

Look at the program listing. At lines 130 to 150, the program goes to a subroutine in lines 500 to 560 that is based on the LOC command. Press either arrow key once. The ship will move by one dot to the right or left and then stop. If you hold the key down, the ship will keep moving as long as the key is being pressed. Unless the LOC command is in repeated use, no movement results. To move the ship across the screen with a single command, you can use MOVE.

This program, based on LOC moves the ship by changing the LOC coordinates; when you press the right-arrow key, the program adds 1 to the X-coordinate, and when you press the right-arrow key, it subtracts 1.

MOVE is a good command for showing continuous movement, or for the general movement of a large picture. LOC is good for suddenly moving an image so that it “pops up” somewhere else on the screen, because no intervening process of movement is required.



□

KEEP PRESSING THE fZTI KEEP PRESSING THE

LEFT-ARROW KEY I 1 RIGHT-ARROW KEY

10! \*\*\*\*\*\*\* LOC \*\*\*\*\*\*\*

20 Print “ HUM" • m a 9 2

30 restore 1000 40 f or 1 = 128 to 143 50 read fi$

60 stchr fi$ to I 70 next I

80 scod 0>1285scod 1,132 90 scod 2, 1365 scod 3,148 100 scol 0,15s scol 1,15 110 scol 2,8sscol 3,8 120 FX=100sgosub 500 130 K = i nkey<0)

140 if K=55 then FX=FX+1sgosub 500 150 if K=54 then FX=FX-159osub 588 160 goto 138 170 end

500 if F X > 210 then FX = 218 510 if FX< 8 then FX = 0 520 1oc 0 to FX,129 530 1oc 1 to FX+16,129 540 1oc 2 to FX+16,145 550 1oc 3 to FX,145 560 return

1000 data 0000000000888801,0187071f1f7f7f80,868606Iele7e7ef > fefefefefefefe8e

1010 data 00C0C0C0f0f8fcfc,ffffffffffffffc0,000000000808000 , 0000C0C0 f 0f 0fC00

1020 data f f f f f f f f f f f f0000,8000000000000 000 , f8f8C0C00000000I ,0000000000088000

1030 data 3f3f 0 f 8 f 03018088,8000000000800000, ffffffffffff000i ,0000000000000000

With MOVE, four sprites can be moved simultaneously, thanks to an auxiliary command called JOINT. However, the same task requires four LOC commands, one for each sprite. The four LOC commands in this program are at lines 520 to 550.

AN AIRPLANE IN FLIGHT  
(MOVE-TO)

When you move an airplane or UFO from one point on the screen to another, you’ll use a MOVE-TO command. The general format of a MOVE-TO goes like this:

MOVE A IN B TO C,D,E A represents a sprite number (from 0 to 31).

B represents a post number (from 0 to 11).

C represents an X-coordinate of the screen, or screen pattern (because when you create a viewport with the VIEW command, the coordinates available change).

D represents a Y-coordinate of the screen or screen pattern.

E represents the speed of movement, from 255 (slowest) to 1 (fastest).

The post number in B is used to move the sprite. If you give no value for the post number, the post with the same number as the sprite is used.

If the screen is not divided by a viewport, specify a number from 0 to 255 as an X-coordinate (Q and a number from 0 to 191 as a Y-coordinate (D). However, if the presence of a viewport has changed the position of “top left” on the screen [[1]](#footnote-1) [[2]](#footnote-2) [[3]](#footnote-3) [[4]](#footnote-4) [[5]](#footnote-5)

(for example, a line like VIEW, 1, 0, 30, 23), the position of top left on the screen shifts to the right by one sprite size. In such a case, a number from 0 to 247 must be specified for C. When the cursor’s home position is moved with a VIEW command, the range of C and/or D is decreased by the number of sprites moved x 8.

In this program, the sprites are created in lines 30 to 170. The JOINT statement at line 170 combines sprites 1 and 0 at the position of sprite 0. Then both sprites can be moved by the MOVE-TO command that follows in lines 180 to 210.

When using a MOVE command, don’t forget to write MOVE ON (as at line 190). Without this line, the sprites won’t move. The MOVE-ON at line 180 puts the sprites on the screen, and the next MOVE-TO sets the plane flying.



SHOOTING BULLETS  
(MOVE-STEP)

To show the motion of a bullet or a missile, which will either reach its target or fly harmlessly out of the screen, a MOVE-STEP command is used. MOVE- STEP moves a sprite from an original point in a specified direction, without any ending point being specified. The general format is:

'•MOT-

30 view 1,0,30,23 4 8 r e m sprite

50 stchr “000002030705051f“ to 40

60 stchr \*8680" to 41

7 0 stchr "0 0©04 0 c 0 e 0a 0 a 0f8 ” to 4 2

80 stchr 11 (8 f 961 6H H f 3 1 01 " to 43

98 stchr “0181010000020280“ to 44

180 stchr “800809090080128c" to 45

118 stchr “8080888000484081“ to 46

128 stchr “0100909800084830“ to 47

138 stchr “0880000808000008“ to 48

148 stchr “0008800001810101“ to 49

150 stchr “ 8 800 808800 000 00 8" to 5 8

160 stchr “8880000088888080“ to 51 170 Mft 6 2

1. 0 scod 8,48
2. 8 scol 8,14

280 scod 1,44

210 scol 1,8

228 scod 2,48

238 scol 2,15

248 Joint 1 to 0,1 2 5 8 r e m m o ue. . .step 26 0 m o u e on

278 move 8 in 8 on 118,175

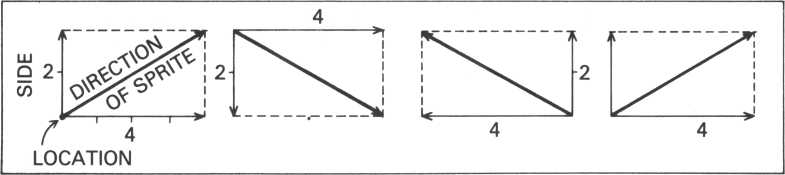
288 move 2 in 1 on 118,168

298 if inkey <0> 07 then 9oto 298

38 8 mooe 2 in 2 step 0,-5, 1

310 if st at-us^2) = 1 then 9 o t o 318

328 -3ot o 288



MOVE A IN B STEP C,D,E A REPRESENTS A SPRITE NUMBER (0 to 31).

B REPRESENTS A POST NUMBER (0 TO 11).

C SPECIFIES DISTANCE IN THE DIRECTION OF THE ABSCISSA.

D SPECIFIES DISTANCE IN THE DIRECTION OF THE ORDINATE.

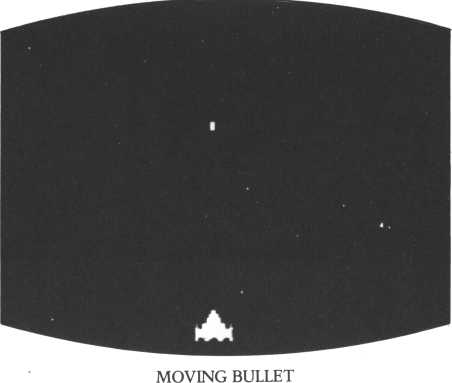
E REPRESENTS THE SPEED OF MOTION—FROM SLOWEST (255) TO

FASTEST (1).

In this program, A, B and E are given the same values as in the MOVE-TO command. C and D, however, are completely different. In the MOVE-TO command, the XY = coordinates of the pattern were specified and the pattern was moved to these coordinates. Here, the direction and distance are specified by the component vector of the abscissa and the ordinate.

Look at the diagram; it will show you how to go about specifying a component vector. Make a rectangle based on the axes of the abscissa and the ordinate. The sprite’s direction of movement will be the diagonal of the rectangle. The longer the diagonal is, the faster the sprite moves. Either the value of the axis of the abscissa or the ordinate can be zero.

Now take a look at the listing for this program. Lines 10 to 230 set up three sprites; sprites 0 and 1 are combined at line 240. Unless a MOVE-ON command is used, the MOVE-STEP command will not take effect. This program shoots bullets from the beam cannon when the space bar is pressed.



MOVEMENT OF THE  
BACKGROUND  
(SCREEN SCROLL)

So far we’ve only displayed and moved sprites on the screen. Now let’s do something more advanced—move the background as well.

The M5 has four scroll functions: up, down, right and left. Using these four scroll functions, we can move the background in four directions. But to give the appearance of movement, the screen must also display something. This is done by printing and scrolling in an endless cycle. You’ll print something at the bottom of the screen when you scroll the background up; at the top when you ‘scroll it down; at the left when you scroll it to the right and at the right when you scroll it on the left. Unless you do this, the screen may be empty much of the time.

In this program listing, the sprites are created in lines 30 to 160. The program combines two sprites and displays them at the center of the screen at line 180. To change the direction of background motion depending on keyboard input, the program goes to the subroutine in lines 190 to 250. After line 250 are the subroutines that scroll the screen in different directions. All these subroutines also print a dot. The program also changes the airplane’s direction of flight.

|  |  |  |  |
| --- | --- | --- | --- |
| 18 Pr int | 11 ram11 |  |  |
| 28 view 1 | ,0 ?38,23 |  |  |
| 3 8 r e m sPr | i t e |  |  |
| 48 stchr | " 1 818 0 8 3 c 7 e ffdb00” | t 0 | 40 |
| 50 stch r | "88001 8 0000008818" | t. 0 | 41 |
| 6 0 s t c h r | " 88dbii7e3c0018 18" | t o | 42 |
| 70 stchr | " 1 888000808180888" | t o | 43 |
| 30 stchr | " 06Qelcdedelc0e 8 6\*" | t 0 | 44 |
| 98 stch r | " 8 0 8 8 0 0 2 12 1 08 0 8 00 " | t 0 | 4 5 |
| 18 0 s t c hr | " 6 8 7 0 3 8 7 b 7 b 3 8 7 8 6 8 | " t. 0 | 4 6 |
| 110 stchr | ” 0 0 0 0 0 0 8 4 8 4 00 0 0 0 0 | " t 0 | 47 |
| 1. 8 m a 9 1 2. 8 sco d | 8,48 |  |  |
| 1 4 6 s c o d | 0,2 |  |  |
| 15 8 s c o d | 1 > 4 1 |  |  |
| 16 8 s c o 1 | 1 i 8 |  |  |

17 6 joint 1 to 8,1

180 move 0 in 8 on 110,160

196 0=1

208 B=inke9<8>

210 if B = 51 then 0=1

228 if B = 46 then 0=2

236 if B = 54 then 0 = 3

240 if 6=55 then 0=4

256 on 0 9 osub 270,336,390,458

266 9 ot o 288

276r em

286 scod 8,48 298 scod.1,41

30 8 Print cur sor < r n d <29> ,8 > $ \*,"? 316 Print HWM 326 return 3 3 0 r e m

346 scod 8,42 358 scod 1,43

366 Print cursor(rnd<29),22)?", ” ; 376 Print 38 8 ret urn

1. 9 8 r e ro

406 scod 8,44 410 scod 1,45

426 P r i n t cursor <0,rnd(21)> 5",

438 Print \*Mm 446 return

1. 5 6 r e m

468 scod 0,46 476 scod 1,47

4 8 8 P r i n t c u r s o r < 2 9,r n d < 21> > \*". " s 49 6 Print "W“

508 return 518 end

**VI**

**MAKE A  
ONE-SCEIME  
ANIMATED  
GAME**

1

INTRODUCTION

2

CRISIS AT SEA

3

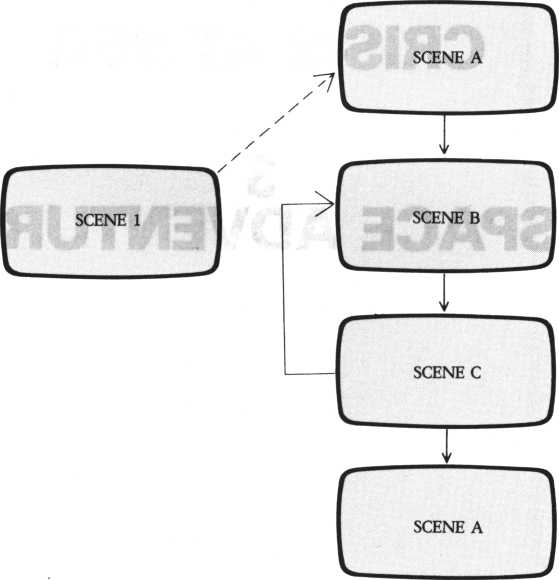
SPACE ADVENTURE

INTRODUCTION

Cartoonists who work on paper typically work within two forms: the one- panel cartoon and the comic strip, which tells a story across a number of panels. There’s a reason for bringing this into animation. Most microcomputer games have just one scene. However, the M5 is an animation personal computer: with it you can easily change scenes by switching backgrounds. In the games we’re about to introduce, story development across several scenes is the basic concept. However, to explain the basics of game animation, we’ll start by choosing one scene from these games and going through it in detail.

PROGRESSION OF SCENES

ONE-PANEL CARTOON # COMIC STRIP



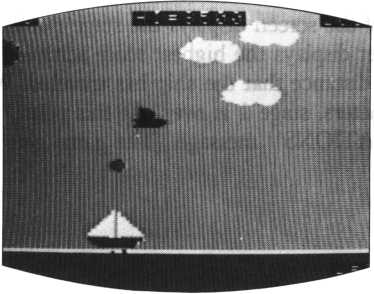
CRISIS AT SEA HOW TO PLAY

You are afloat far out on a dangerous sea. The object of the complete game is to sail back to shore. For now, we will only work with one scene of the game, however.

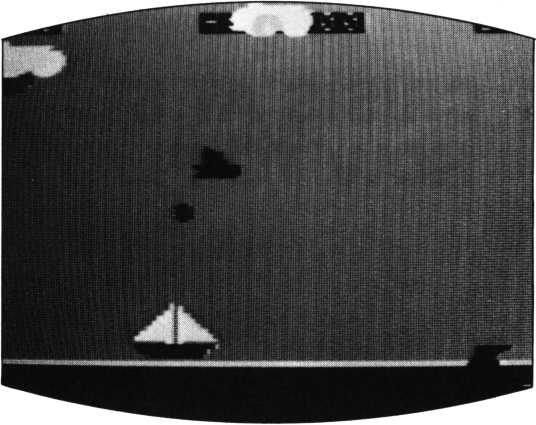
Type in and RUN the program. You can move the ship to the right by pressing the right-arrow key, and to the left by pressing the left-arrow key. But you can’t move out of the screen altogether.

The bird in the sky aims droppings at you; the sharks in the water are trying to get you. When droppings hit you or the shark eats you, the game is over. Some sharks attack from the right, others from below. When a shark attacking from blow gets to the mast of your ship, the game is over.

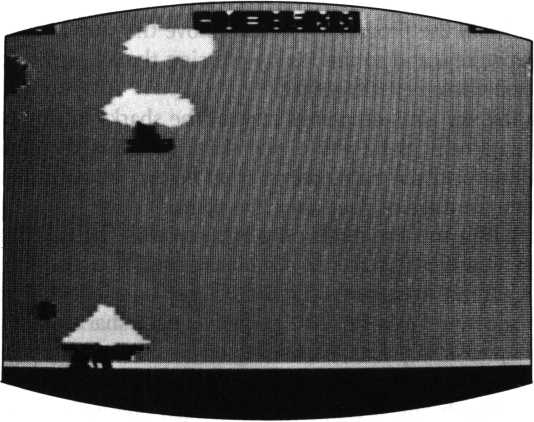
To attack the bird and the sharks, fire at them with the CTRL and SHIFT keys. The CTRL key fires upward; if you hit the bird, you get 100 points. To shoot at a shark on the right, press the SFIIFT key on the left. If you hit a shark attacking from below you also get 100 points. A dead shark attacking from the right gives you up to 150 points, depending on its distance from the ship. The nearer the shark is, the more points you get. If the distance is very far, you may even lose points. Watch out: it may take more than one bullet to kill a shark.



THE BIRD DROPPED DROPPINGS.



THE SHARK APPEARED.



THE SHARK ATE YOU UP.

The bottom right of the screen displays the stage number; the top left displays the score; the top center displays the highest score achieved to date; and the display at top right is the distance that the ship has traveled. When the distance covered reaches 500 meters, stage 1 is over. At this point, you’ll hear music and see a “CONGRATULATIONS” message. (The game gets more difficult at higher stages.)

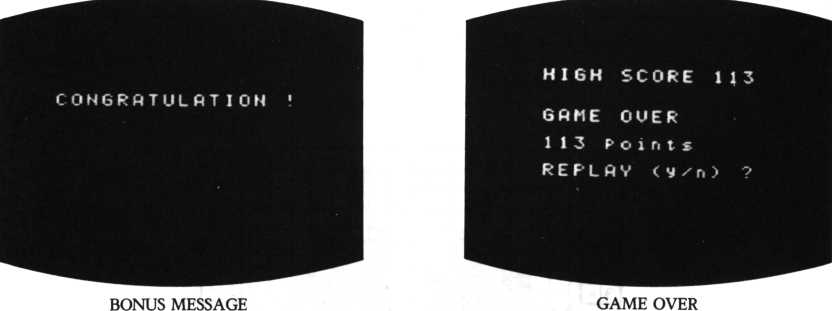
PROGRAM TECHNIQUE

The program gives the bird the appearance of flight by changing the colors of its wings alternately, as described in section 1(2) above. When one of its wings is green, the other is colorless, and vice versa.

Everything except the ship is controlled by MOVE commands. The program reads keyboard input by DMKEY(0) and moves the ship by the subroutine at lines 500 to 520 using a LOC command. The program reads your keyboard input for shooting by INKEY(1).

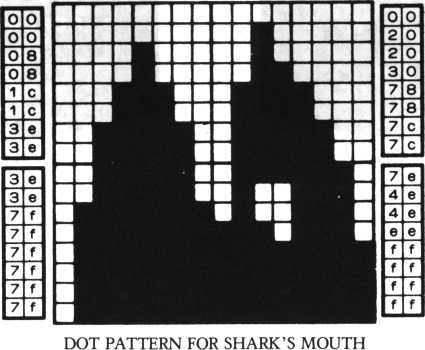
The shark attacking from the right and the one attacking from below don’t appear simultaneously because they are put on the same post. A random number determines which shark to display. The constantly moving clouds are three identical sprites on three posts.

To change the game’s level of difficulty, change the bird and sharks’ speed and the bird’s altitude over the ship.



Sprite list

|  |  |  |
| --- | --- | --- |
| Sprite number | Contents | Post number |
| 0 | Droppings from bird | 4 |
| 1 | Bullet shooting to the right | 6 |
| 2 | Shark’s mouth | 5 |
| 3 | Shark | 5 |
| 4 | Bird(l) | 3 |
| 5 | Bird(2) | 3 |
| 6 | Cloud(l) | 0 |
| 7 | Cloud(2) | 0 |
| 8 | Same as that of sprite no.6 | 1 |
| 9 | Same as that of sprite no.7 | 1 |
| 10 | Same as that of sprite no.6 | 2 |
| 11 | Same as that of sprite no.7 | 2 |
| 12 | Bullet shooting upwards | 6 |
| 28 | Ship(l) |  |
| 29 | Ship(2) |  |
| 30 | Ship(3) |  |
| 31 | Ship(4) |  |



INPUTTING THE  
PROGRAM

Input the program by copying in the listing given here. Do it slowly and carefully. Even one mistake can make it impossible to run the program; or at the least, it won’t run properly.

By using the FUNC key, you can speed the input process. For example, instead of typing PRINT, press the FUNC and P keys together. As you get used to the M5 keyboard, you’ll pick up speed.

If you made a mistake in input and want to delete the whole line, type the corresponding line number and press the RETURN key. To delete from a point in the middle to the end of the line, move the cursor to the appropriate point and press the CTRL and X keys at once.

When you have two lines (say, 10 and 20) that are similar, follow this procedure:

Input line 10 and press RETURN.

Change line numbers from 10 to 20 and press RETURN.

Now modify the new line 20 so that it’s not an exact match of line 10.

By using these and other shortcuts that your M5 makes available, you can be inputting program listings in much less time than you thought it would take you.

10! \*\*+ SEA ADVENTURE GAME <1> \*\*\*

20 Print " MSI" J view 1 > 8» 3 8 » 23\* console 8 s on error 9 osub 30 30 i f err-20 then resume:return 40 Print Hi\*serase

5 0 for 1=4 to 6\* stchr r P t $<16 >“? M > t o !2 8 >1:stchr r P t $<8 > " f 4 ”

> t o 241 \* 1 s stchr r P t $( l 6 > " 4 " )t o 139? I s next

60 for 1=252 to 255: st-ch r rPtl(8? "8?"> to I ? 4: next

78 for 1=8 to 1 8:P r i n t cursor <8? I)\* rPtl(30> "i")?:next

80 Print cur sor ( 8 , 1 9> ? r P 11 (3© ? ) ! : f or 1=2© to 2 2 : P r i n t cur

sor <©,I> 5 rPt\*<30,”+H> 5:next

9© Print cur sor (3? 4) ? " jflk” ? cursor (3? 5> 5 "W?":

10© restore 1888sC\*«r\*t\*<I 6» "8 \*}s4 or 1 = 172 to 21i:read fi$:st chr R$ to I:next

118 stchr C$ to 22 4:s t c h r C t to 2 2 6:s t c h r C♦ to 2 2 7: f o r 1 = 15 2 to 154:stchr C $ to I :n e x t

12 8 stchr Ct to 1 3 2:s t c h r C $ to 1 34:stchr C $ to 135 13© scod 6?224:scod l?152:for 1=0 to 3:scod 2 +1 » X?2 + 4\*1 in ext

s scod 6 ?132: f orv 1=7 to 11 step 2:scod I\* 188:scod I +1 » 1 9 2: n e x t 140 for 1=0 to 3 : scod 28+ 1 » 1 96 + 4\* I : r. ext : sco 1 ®?13:sco1 1,14:

scol 2 > 4:scol 3 ? 4 : s c o 1 4 > 2:s c o1 5\*0

158 scol 6,148for 1\*7 to 12:scol 1, 1 5snext 8sco1 28»15!scol 2 9,15sscol 30,8sscol 31,8

168 Jointsjoint 5 to 4,lsJoint 8 to 7,2sJoint 18 to 9,2sJoin t 12 to 11,2

178 ma9 2 s L1\*L mod 6sprint cursor<25,20) 5 ML="5num$<L +1>8FX\*7 2i 9osub 5068 randomizesmoue on

180 for 1\*7 to 11 step 2 s 1 o c I to rnd(248),rnd(55)s move I in 1/2-3 steP- 1,8,38next

190 1oc 4 to-16,rnd<50)+20+L1\*8smove 4 in 3 steP 1,0,2

208 Print cursor(2,2) 5 n u m $ < P ) 8c ursor<24,2)5 n u m $<PP)8"m"8 c u r s

or <12,2)? HHI = M 8 numf(HI)

21 0 lor 1\*0 to 28 if status(I)\*0 then loc(I+3)\*2+l to 248,rnd (55)8 move(1 + 3)\*2+1 in I steP-1,8,3

228 next:i I sPrite<4,1)>240 then 1oc 4 to-16,rnd<50)+ 28 + L1\*8 'wove 4 in 3 step 1,8,2

238 if statu s < 4)\* 1 then 9 o t o 250 else BX=sPrite(4,l)sBV\*sPri t e < 4,0)

240 if B X < ▼ 1 2 8 and BX>\*40 then loc 8 to BX,BVsmoue 0 in 4 to BX,152,2

250 S\*rnd<180)sif S\*>97 and status<5)=8 then SAX»rnd<80)+48s loc 2 to SAX,1 5 1 s m o u e 2 in 5 to SAX,138,80-10\*L1 260 if S<2 and status(5)=8 then erase 2sloc 3 to 248, 146s moo e 3 in 5 st eP- 1,0,2

278 K\*inke9(l)8if K\*4 then s 9 3»5,15ssleeP 3, 18 s \*3 3,,0sloc 1 to s P r i t e < 3 0,1 > > 8,13 6 8 m o u e 1 in 6 step 3,8,2 288 if K \* 1 then s 9 3,4,158sleeP 3,18 s9 3,,8 sloe 6 to sPrite< 28,1 > +1 2 > 1048moue 6 in 7 step 8,-1,2 298 K 1 = i n k e 9 < 8 )s if Ki\*55 then FX=FX+1sgosub 508 380 if \*1\*54 then FX\*FX-1:9osub 580

310 BO0\*coinc<0,28,31) 8 i f B08>\*28 and 8O8<\*3\*10! \*\*\* SEP ADI

28 Print ‘‘HIST ! view 1,8,38,23s console 8s on error 9 osub 38 38 if err\*28 then resumes return 48 Print "I" 8 erase

58 for 1\*4 to 6 s stchr rPt$(16,"7")to 128,lsstchr rPt$<8,Mf4“

>to 241, is stchr rPt$<16,'\*4M)to 139,l8next

68 for 1=252 to 255sstchr rPt$(8,"87H)to I,4snext

78 for 1\*0 to ISsprint cursor(8, I)!rPt$(38,“■")Is next

88 Print cur sor (0, 19) 5 rPt$<38, ) » s for 1=28 to 22sprint cur

sor(0,I)?rPt$<30,"+")5 snext

98 Print cursor (3,4) i "A" ?cursor<3,5) ? " W" 8

100 restore 1888sC\* = rPt\*<16, " 8 " )s for 1 = 172 to 2U8read A$sst chr A $ to I snext

110 stchr 0$ to 224sstchr C$ to 226sstchr C $ to 227sfor 1\*15 2 to 154 8 stchr C $ to Is next

128 stchr C $ to 132sstchr C$ to 1348 stchr C $ to 135 130 scod 8,224sscod 1,152s for 1=8 to 3s scod 2+I»172+4\*Isnext sscod 6,132sfor 1=7 to 11 step 2!scod I,188sscod I+i,192snex

t

148 for 1=8 to 3!scod 28+1 , 196+4\*Isnexts scol 8,13sscol 1,14s scol 2,4sscol 3,4s scol 4,2!scol 5,8

150 scol 6,14 s f o r 1=7 to 12s scol I,15s next s scol 28,15sscol 2 9,15sscol 38,8sscol 31,8

168 Joint s Joint 5 to 4, is Joint 8 to 7,2s Joint 18 to 9,2sJoin t 12 to 11,2

178 ma9 2 s L1= L mod 6!Print cursor<25,28)\\*L = M?numf<L+ 1)sFX\*7 2!losub 580s randomizesmoue on

188 for 1=7 to 11 step 2s1oc I to rn d(248),r n d < 5 5) smove I in 1/2-3 step-1,0,3s next

198 loc 4 to-16,rnd<58)+20+Ll\*8srooue 4 in 3 steP 1,8,2

208 Print cursor(2,2)J nuwl(P)5 cursor(24,2)? numt(PP)5"m"\c u r s

or <12,2) 8 MHI = M 5 r.um$(HI)

210 for 1=8 to 2 s i f status(I)=0 then loc<I+3)\*2+l to 248,rnd <55)smoue<I+3)\*2+i in I steP-1,8,3

228 next!if sPrite(4,l)>248 then loc 4 to-16,rnd<58)+28+Ll\*S smoue 4 in 3 step 1,0,2

238 if status(4)=l then 9oto 258 else BX = sPrite(4,l)sBV = sPri ted,0)

240 if B X < = 1 2 8 and BX>=40 then 1 oc 0 to B X » B V s rr» o u e 0 in 4 to BX,152,2

250 S = r n d O 88) sif S = >97 and status(5)«0 then SRX = rnd<86) +40 : loc 2 to SOX, 151s m o u e 2 in 5 to SOX, 13 8,80-10\*11 268 if S<2 and status(5)=0 then erase 2! 1oc 3 to 248,1465mou e 3 in 5 step-1,0,2

270 K \* i n k e y <1> s i f K = 4 then s9 3,5,15:sleeP 3,l!s9 3,,8sloc to sPrite<38,l)+8\*136smove 1 in 6 step 3,0,2 280 if K =1 then s 9 3,4,15ssleeP 3, 1 5 s 9 3,,8s loc 6 to s P r i t e < 26,1) +12, 1045 move 6 in 7 step 0,-1,2 290 K 1 = i n k e y < 0 X \* if Kl-55 then FX=FX+1:9osub 500 300 if K1= 5 4 then FX=FX-1s9osub 500

310 BO0\*coinc<0,28,31>:if B08>«28 and BG8<\*31 then 9oto 440 320 BQl=coinc<2,28,29) : i f B0i=28 or 601=29 then 9oto 448 330 if c o i n c < 3,3 8,3 8 > = 3 8 then 9oto 448

348 B 0 2 = c o i n c < 6,4,5 > s if B02O4 and B02O5 then 9ot o 368 358 for 1=30 to 80!s9 8 , I , 15snexts s9 8,,8:loc 4 to 241,885 To c 5 to 241,88s erase 6iP = P + 1 8 0

368 if coinc< 1 , 2,2) =2 then s9 3,6 , 1 5 : s 1 e eP 18,l;s9 3 , , 8 : P r i r» t "W ? J erase 2,lsP=P+188

1. 8 if coined , 3,310 3 then 9 o t o 39 8 else P r i n t c u r s o r ( 0 , 15 >

? r P t $ < 3 8 , " ■" >

1. 8 s 9 3,4 , 1 5 5 s1e e P 18, 1 s $ 9 \*3, ,8 s P 0 = 1 5 8-<s P r i t e <3, 1>-s P r i t e <

38,1>)sprint cursor($Pr it#<3, 1>/8 » 15) ? num$<P 0 >s er a s e 3, 1 : P = P

+pi:i

398 if s P r i t e < 4,3) = 0 then scol 4,2:sco1 5,8 else scol 4,0:sc

ol 5,2

408 PP = PP+1 5P = P+1

410 if PP<588 then 9oto 288 else PP«8s cls? erase

420 Print cursor<5, 10) ! "CONGRRTULOTI ON \* "Jcursor(5, 12) ? "BONU

S GOME STGRT ! M s s L = L+1

430 Play "Sltl70o59898a898o6co5b8a8998989r4d8d8d8e8fe8d8cc8c 8o6c"ssleeP 2 2 9 oto 48

440 for 1=8 to 15ss9 3,5,15-l2sleeP 18,1s n e x t

458 cls!S' = V+ ispp = 0!erasesif V<3 then Print cursor <8, 10) ? "STfl RT !M?V+lssleeP 3!9oto 48 460 if P> HI then HI=P

478 Print cur sor < 8,7) 5 " H I GH SCORE M \*, H I 2 cur sor < 8 , 1 8) ? H GOME OUE R"Jcursor <7, 12)? P ?"Points"? cursor(8,14)5 "REPLRV <y/n) ?M

488 Z$=inkey$ if Z$<>MyH RUTOND 2 $ < > ” n H then 9oto 488 490 if 2 $ =M y ” then P = 82pp\*0SL = 0sV\*0;yoto 48 else end 508 if F X > 18 4 then FX=184 518 if FX< 40 then FX = 40

528 loc 28 to FX, 129s loc 29 to FX +16, 129s loc 30 to FX+16,145 sloe 31 to FX,145!return

1000 data 000088081clc3e3e,3e3e7f7f7f7f7f7f,0020203078787c7c »?©4©4eeef f f ff f f f

1810 data 0f8fIfIf3f3f7fff,8000088088080800,fffefSe0c0c8e0e0 ,0000008000000000

1020 data 181e1f1f1f1f1f f f,7f3f7f f f80000008,8l8880c8c0e0e0fe , f b f f fef C00000000

10 38 data 00000008f f7f3f7f,f f1f1f1f1f1f1e18,80000080fefb f f fe ,fce0e0c0c0808088

1048 data 800000870f0f3f7f,3f1f7f1f81000088,78f8f f f f f f f f ff f f ,f f f f f f f f f f3f 0000

1850 data 8808f f f f f f f f f f f f,f f f f f f f fcf830808,0808e0c8fcfcfefe ,fefefcfcf8e08000

I860 data 0088080080000001,8187871f1f7f7f88,8686861e1e7e7efe ,fefefefefefefe8e

1078 data 80c8c0c0f8f0fcfc, f f f f f f f f f f f ff fc0,0080080808800000 ,0080C0C0f0f0fC00

1080 data ffffffffffff0080,0000008000000000,f8f8C0C000000000 ,0000000000880080

1090 data 3f3f0f0f03010800,0000000000080000, ffffffffffff 0000 ,0000000000000000

SPACE ADVENTURE  
HOW TO PLAY

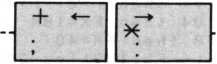
This game is a scene from a space war. Defend your spaceship by either dodging meteorites or blowing them up with missiles. There are four ships and four stages in the overall game. Every time you finish playing one stage, energy is added to the ship’s total supply as a bonus and the ship locates a little higher on the screen. Dodging meteorites becomes more difficult as the game advances.

Don’t attack the meteorites just because it’s easier than dodging them. It costs you 20 points in energy to launch a missile, bu you only gain 10 points by blowing up a meteorite. It’s a better strategy to dodge unless you are in urgent danger.

RIGHT

**LEFT**

**SHIP MOVES TO THE LEFT**



**SHIP MOVES TO THE RIGHT**

SHIFT

**MISSILE LAUNCH**



When energy drops to zero or the ship has been destroyed, the game is over. If the score reaches 5,000, you gain another ship. Use the keys as shown on the previous page to move the ship right or left or to launch a missile.

*w*

HI-SCORE 0

SCORE

8

ENER6V

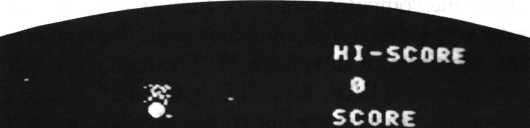
6970

**\***

**\***

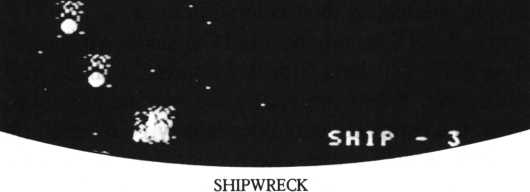


**METEORITE CLUSTER**



ENERGY

697©



PROGRAM TECHNIQUE

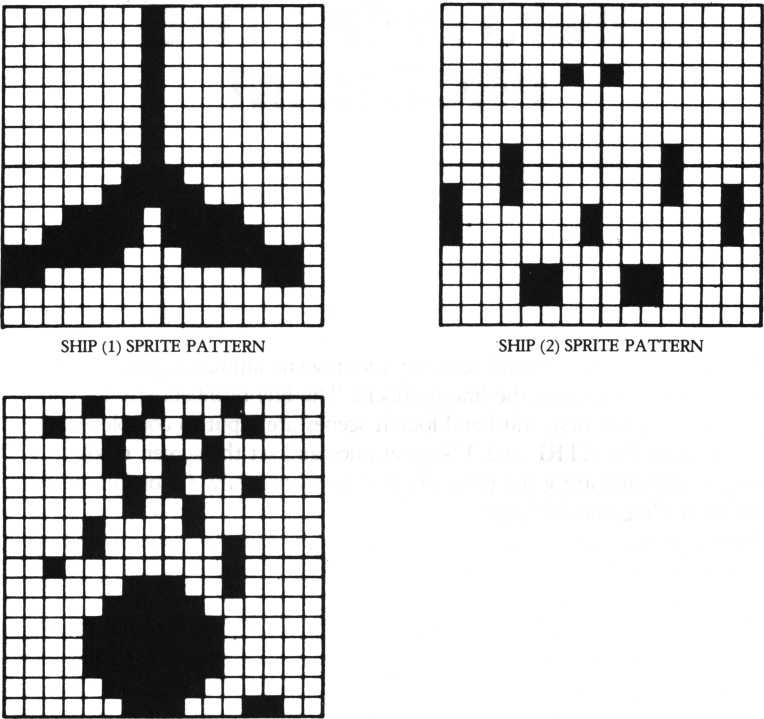
This program is written largely in GII mode. It can only run in GI mode but (as mentioned earlier) it’s part of a four-scene game. The stars and meteorites descending the screen behind the ship are moved by the down scroll of the control character. The computer displays the meteorites by counting the number of times scroll is performed, and then does the end processing. Scrolling ten times subtracts ten points from the energy reserve.

After each game the computer scores the highest score in 8FFE and 8FFF in low and high order respectively. The computer reads the high score from the above address and substitutes it in variable H. The high score will not be cleared when the program runs if CLEAR, &H8FFD is declared.

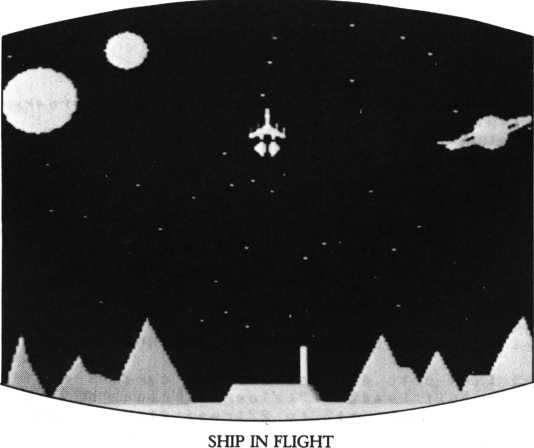
Line numbers of this program are not consecutive, but input it as is, because this is only part of the four-scene program. The code for the other three scenes will be added later.

Look at the sprite patterns on the next page. The program JOINTs sprite patterns 1 and 2 to make the spaceship. White is specified for SHIP (1) and red for SHIP (2); they’re overlaid so their colors interlock.

When you use a JOINT statement, the LOC statement for SHIP (1) will not automatically bring SHIP (2) along with it. However, a single MOVE command can be used to this effect. When two sprites are overlaid and they are both moved by LOC statements, the computer may occasionally go through the processing for a sprite collision. If the program uses interrupt processing for sprite collisions, the computer tends to check the state of the just-erased sprite, bringing about an error. This program is designed to prevent such an interrupt-based error.



METEORITE SPRITE PATTERN



INPUTTING THE  
PROGRAM

This program is the second scene of a four-scene animated game. Input the program without changing the line numbers. The line numbers are not consecutive because the first, third and fourth scenes are input to complete the game.

If you press the CTRL and T keys at once and set the screen mode to TEXT before you start inputting the program, it’ll go more easily. GI or GII are difficult modes for reading and debugging.

After you finish inputting the program, save it. Run the program and debug it. Save it frequently on the cassette tape; accidental power outages or other problems could erase the program.

This program uses only the front screen. The fourth scene of the space game will use the back screen for a “space warp” effect. Consequently, in this scene the back screen is always in text mode. You can specify GII mode for the front screen and GI for the back on the M5; but you can’t specify GII mode for both screens. There are many control characters in this program that are used by the full four-scene game; input them all.

1650 scod 1 , 0 • scol 1 , 1 5 s s c o d 2', 6: scol 2,6 s .j o i n t 2 to MJsco d 0,125 scol 0,8s scod 3 8,2 4 : s c o 1 30,13

1060 for 1=21 to 2 7:scod I, 16s scol I,next s for 1=28 to 2 9 : s cod I ,20« scol I, 14s next

10 70 Print cur sor < 2 3 , 1 >5 " H 1 -SCORE ” 5 cur sor < 2 3,5> ? " SCORE "5 cur s

or<23,9)S”ENER6V“5 cur sof<23,22) 5 “SHIP - “SUS

1080 locate 23,3:if H = 0 then Print H else Print H 5 '\* +\*8"

1090 stchr 1100 0 000100 0 000000“ to 1 27,7 5 for 1=0 to 30:Print c ursor<rnd<21> ,rnd<22>> ? "4" 5 snext\*

1100 view 23,13,38,20sginitsbox 8,8,63,63s fcol 12s 9 m o v e 8,63 scircle 16,,,270,360spaint 1,62

1110 fcol 9sgmove 32,47:circle 4 s Paint 32,47 sgwove 18,7scirc le 2 : P a i n t 10,7:9move 50,28scircle 3 s p a i n t 58,28:view 1120 coinc on:X = 8 0 s V=184-T\*16s1oc 1 to X,V:loc 2 to X,Vsloc 38 to 232, 184 s s9 3,6,6 1130 if E< 1 then goto 4008

1148 Print cursor(23,11) 5E5 slocate 23,7:if 5 = 0 then Print S else Print S 5 "+-0 " 5

1150 if<S‘>=500>\*(Us8>then lJ = U+ lsprint cursor<29,22)SUS:U\*i 1160 groove 184,167sfcol ©scircle 8/50+17,,,315,315sfcol 4sci rcle C/50+17,,,315,315:B=C

1178 vieu 8,8,21,23sPrint ”W”:Print cursor<rnd<21),8)5 " r5sC =C+l:vieu:if C rood 18=8 then E = E - 1 8 1189 if C< 2508 then goto 1218

1198 for 1 = 1 to 7 s i f statu s < I ) = 1 then goto 1250 1280 next:event off s 9oto 1888

1218 if 01380 then if C<1325 then goto 1258 else N=10:R=4:e vent on

1228 Q=Post<l,7):if<Q=-l>+<C rood <R\*2+1)O0> then 9oto 1250 1230 M=<rnd<19> + l)\*8s if C mod<R\*7~l>=0 then M = sPrite<1,1>

1240 loc Q + N to M,0smove Q + N in Q step 8,1,R

1250 R=inkey<0>sl=inkey<l)sif 1=4 then gosub 1388 else if<I= l>\*<U<3)then gosub 38

1260 if ft = 54 then X = X-(T+1>/2\*4sgoto 1288 1270 if ft=55 then X = X +<T +1>/2\*4

1280 if X>168 then X=168 else if X<8 then X = 8

1298 wove 1 in 8 to X,V:9oto 1130

1380 P = Post<10,ll>:if P = -1 then return

1318 loc P +1 8 to X,V-16S move P+18 in P steP 0,-l,l:E=E-20:re turn

1328 if coinc(1>=-1 then goto 1368

1330 event of f: loc 8 to sPr ite(1, 1),Vs U^U-l:views Pr int curso r <29,22)5 US

1348 for 1=8 to 14 s s 9 3,6,14-lssleeP 15, 1s next s sieeP lserase 13 50 if U = 8 then goto 4088 else return 1128

1368 D=coinc<28,N+l,N+7>:if D<>\*1 then erase 2 8 : 9 o s u b 1390sr et ur n

1370 D = coinc<29,N+l,N + 7>:if DO-1 then erase 29sgosub 1398 1388 return

1398 loc 8 to sPr ite(D,1),sPr ite<D,8) s erase D:s9 3,6,15sslee P 3,l:s9 3,,6:erase 0:5\*5+1s return

1480 data 4884824122449729,17255f9288154201,10049820d258a411 ,f468f62854908402

1410 data 0922040402020888,2003878f0f0f8703,104828a080484010 ,1890c8e0e0e0c08c

1428 data 0800000000000000,0000000001010101,0000000080000000 ,0808000000000000

1438 data 8000080008242f2b,1f8f881000080008,000000002848e8a8 ,f000281000000000 4000r em END ROUTINE 4810 event off:coinc off

4820 Print “1AMCIM" 5 cur sor < 8,9) 5 \*' == BftME OUER ==M

4030 locate 7,3:if<U=8)+<E=8)then goto 4040 else E=E-1080spr

int “CONGRfiTURfiTION MIH1IIMMBQUNUS —" $ ESS goto 4058

4048 if U = 0 then Print “SHIP FILL DESTROVED !!“ else Print “E

NERSV OUT ! ! “

4050 Print cursor<9>13)i “SCORE f S = 8 then Print S else

Print S 5 \*' «\*\*0 H

4060 if H<\*S then Print cursor(6,17)!"\*\* HI - SCORE \*\*":Poke & 8 F F E \* S mod 256,S/256

4870 Print cursor(9»21)»"One More RePlay << y / n >>“?

4080 fl=inkey<0>:if fl = 2 2 then run 36 4090 if A = 3 8 then end else 9 o t o 4888

® S^iFt 'C %#\* ^ ^ /\* \*\*84 ^ \* \*

**VII**

**MAKE**

**FOUR-SCENE**

**ANIMATED**

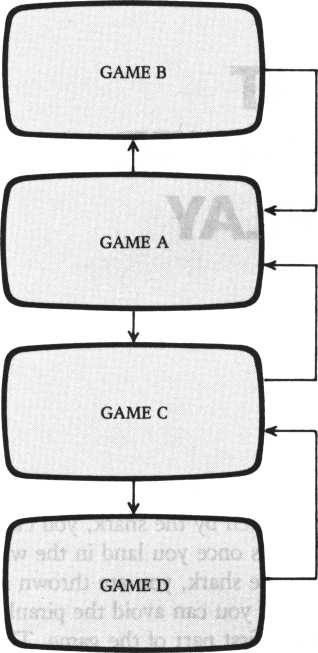
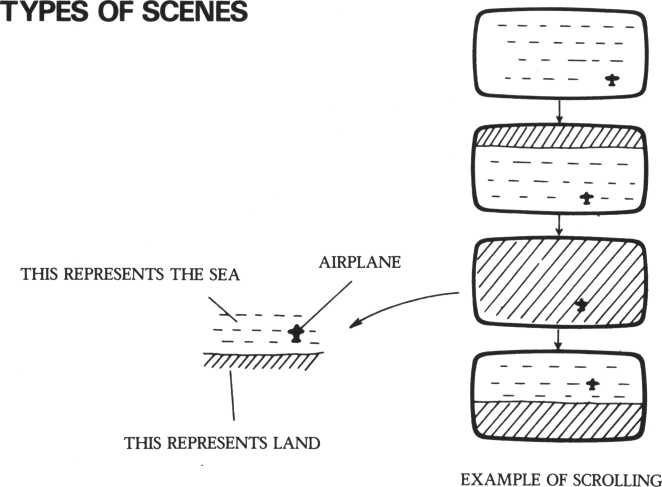
**GAMES**

INTRODUCTION

Now that you have one scene from each game, you’re ready to start building the full games, which are much more detailed and have more story development.

In “Crisis at Sea,” if you can dodge the sharks and the bird droppings, you’ll be able to reach the treasure island. If the droppings or the shark get to the ship, the man is thrown out into the sea and a new segment of the game begins. The man fights with an ever-growing school of piranhas, catching up with the ship which goes sailing on. When the man reaches the ship the former game resumes. Combining two or more games which are consecutive makes the story more amusing and exciting.

In “Space Adventure”, attractive animation scenes and game scenes are combined. The first scene is purely animation: a spaceship leaves its launching pad on Earth and heads out toward space. In the next game scene, you contend with meteorites and attacking aliens. Then you can select the scene in which you have a warp available to try to rapidly reach the star that’s your final destination.



EXAMPLE OF COMBINING GAMES

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | | |
| > | | f | | |
| L\_" | | | ^ | |
| > | | f | | |
| r— —^  ANIMATION SCREEN B k. ^ | | | | |
| y | | r | | |
| r-— | GAME 2 | | | i |

EXAMPLE OF COMBINING ANIMATION SCREENS AND GAMES

2

CRISIS AT  
SEA-CONTINUED  
HOW TO PLAY

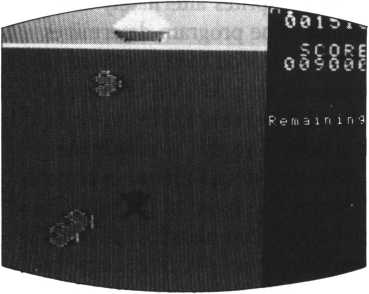
In this expanded version, you can play a bonus game when the ship has moved 500 meters. Even if you’re hit by droppings or bitten by the shark, you can return to play again—if you win the game that begins once you land in the water.

Once you’re hit by bird droppings or bitten by the shark, you are thrown into the sea. There, the piranhas are waiting for you. If you can avoid the piranhas and get back to the ship’s mast, you can resume the first part of the game. The distance that the ship has moved remains what it was when you were thrown into the water. If you are eaten by a piranha the ship goes all the way to the right. You start the game over with 0 meters distance. If you’re bitten three times the game is over and the display at the bottom right indicates the number of men eaten. Pressing the right arrow key moves the man to the right; pressing the left arrow key moves the man to the left; the up arrow key and down arrow key move the man up and down.

The bonus game is the third scene. The scene shifts after the ship has sailed 500 meters; you now see the ocean from above. There are three giant crabs in the upper part of the screen. You must land without being caught by the crabs.

Control the ship’s location with the right arrow and left arrow keys. After you decide where to position the ship, press one of the keys from 1 to 8. The number determines the ship’s sailing speed: 1 is fastest, 8 is slowest. But the slower you go, the higher the possible bonus points. Bonus points are determined by multiplying the pressed key’s number by 500. When the ship sails, you’ll hear as many beeps as the number you pressed. If you crash into a crab, that’s zero bonus points for you. Get high points by sailing slowly for the beach, aiming between the crabs.

The second scene works like the first, in that the higher the stage the more difficult the game becomes. The rules for bonus games are the same.



MIDDLE OF SCENE 2

|  |  |
| --- | --- |
| 0i M. | |  SCORE  ■■H 0 0 4 7 2 0 |
|  | R ■=> fii dinin'? |

STATE OF INITIALIZATION IN SCENE 3



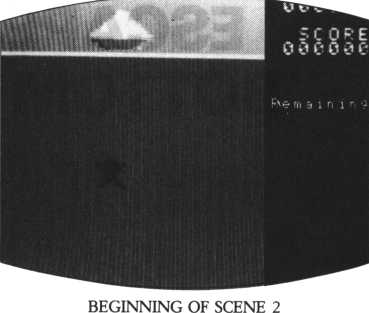
BONUS 580

ft•..Missile

\*- Left

R i 9 h t

Hit h n y Rny ' i i

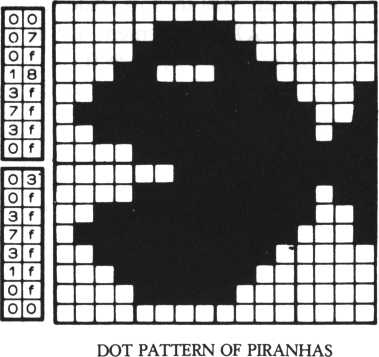


If you’ve already input the program of the first scene, you don’t have to input the program from the beginning; some parts are the same. Modify the program and input the necessary parts only. Modify the destination of the GOTO statements at lines 290 and 300 from 500 to 450. Add the latter part of line 420. Change the destination of the GOTO statement at line 430 to 820. Add “GOTO 480” to line 440. Change lines 450 to 490, 710 to 750, 500 to 520 and 450 to 470. Change data at line 1000 to line 1020. Input the rest of the listing to complete the program.

ESCAPE FROM THE  
BOTTOM OF THE SEA

As we’ve shown you before, the program “hides” sprites alternately in order to give the man’s arms the appearance of movement. The program determines which sprite to hide at line 760 and does so at line 810. The man is moved by a LOC command; the piranhas and the ship are moved by a MOVE command. The program adds 1 to the X-coordinate when the right-arrow key is pressed, and subtracts 1 when the left-arrow key is pressed; it adds 1 to the Y-coordinate when the up-arrow is pressed and subtracts 1 when the down-arrow is pressed.

To make the game easier or more difficult, change the speed of the piranhas or ship by changing the speed values given in the MOVE command. Changing the value shown with STEP can also change the speed. For example, to change the ship’s speed: if a lower value than 15 is given in “MOVE 28 IN 0 STEP 1,0,15” the ship sails faster; if a larger value than 15 is given, it sails more slowly. Giving a higher value than 1 after STEP will also make the ship sail faster. If a lower value is given than 3 in “MOVE I STEP —2,0,3” at lines 580 and 600, the piranhas move faster; if a larger value is given there they move more slowly. Likewise, if the absolute value following STEP is greater than — 2, it speeds up the piranhas; if the value is less, it slows them down.



|  |  |
| --- | --- |
| 0 | O |
| 8 | O |
| c | O |
| e | O |
| f | 1 |
| f | 9 |
| f | t> |
| f | f |
|  |  |
| f | f |
| f | b |
| f | 9 |
| f | 1 |
| e | O |
| c | O |
| 8 | O |
| O | 0 |

List of sprites in scene 2

|  |  |  |
| --- | --- | --- |
| Sprite number | Contents | Post number |
| 0 | Man(l) |  |
| 1 | Man(2) |  |
| 2~ 11 | Piranha | 2~ 11 |
| 28 | Ship(l) | 0 |
| 29 | Ship(2) | 0 |
| 30 | Ship(3) | 0 |
| 31 | Ship(4) | 0 |

Unlike scene 1, the ship in scene 2 travels across the screen by a MOVE command. A JOINT statement combines the ship’s four sprites. If the man is eaten three times the game is over. If you want to change this limit, change the value “3” at line 710 (Y< 3) to a different number.

Lines 590 to 690 constitute a loop. The program checks for a possible sprite collision between man and piranha at lines 650 and 660. It also checks whether the man makes it back to the ship (line 670). If the man does make it, a PLAY command sounds a beep. Lines 770 to 800 constitute a check that the man does not go beyond the screen.

The ship’s four sprites are all put on post 0; the piranhas are on posts 2 through 11. At higher stages the number of piranhas increases, so not all post numbers are in constant use. (“IN-” is omitted in the MOVE command for the piranhas because the sprite numbers and the post numbers are the same.)

SCENE 3: REACHING  
SHORE

To move the ship to the left or right, the program reads keyboard input at INKEY (0). If the right-arrow key is pressed, the program adds 1 to the X-coordinate, and if the left-arrow key is pressed it subtracts 1. From line 990, the porgram goes to the subroutine at line 1010 and moves the ship by a LOC command. The program uses an IF statement at lines 990 and 1000 to check that the ship does not go outside the screen. When the ship sails toward land, the program moves it by a MOVE command; the crabs likewise are shifted left and right by a MOVE command.

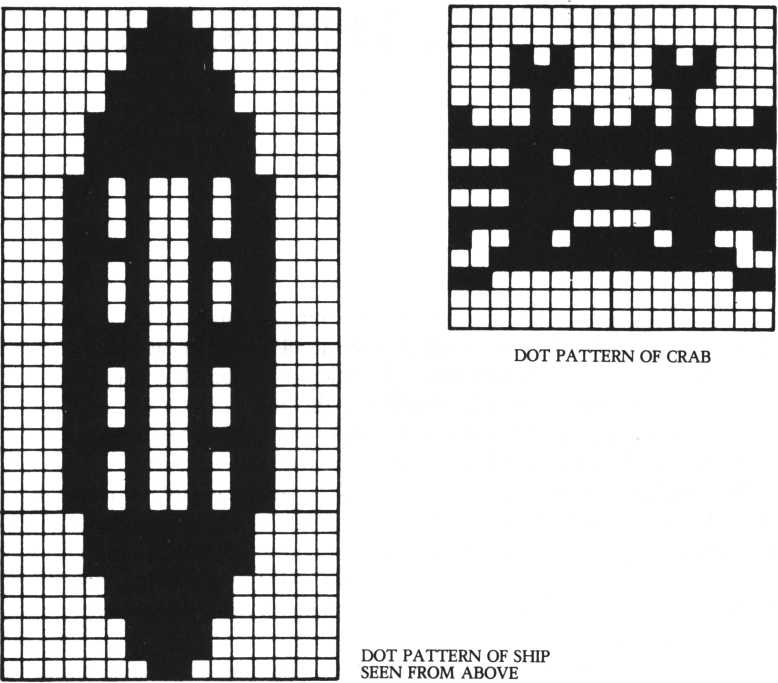
To change the game’s level of difficulty, alter the crabs’ speed by changing the value “3” at lines 880 and 890. A larger value will slow the crabs down, a higher one will speed them up. Or: by giving a larger value than 1 after STEP, the crabs can be moved faster.

The ship’s speed can be changed also. Raising the value of “K3-8” in “MOVE 0 STEP 0, — 1, K3-8” will slow the ship; lowering the value will speed it up. “K3-8’’ is the number of the key pressed to start the ship, so the range of values must be changed as well. Raising the absolute value of “ -1 ” in the STEP statement will speed up the ship also; in this case, you don’t have to change the range of values.

Lines 890 to 970 constitute a loop. The check for possible sprite collision between the ship and a crab comes at line 940.

List of sprites in scene 3

|  |  |  |
| --- | --- | --- |
| Sprite number | Contents | Post number |
| 0 | Ship seen from above(I) | 0 |
| l | Ship seen from above(2) | 0 |
| 2—4 | Crabs | 2-4 |



If you want to change the bonus points, change “500” in “(K3-8) \*500” at lines 960 and 980 to another value. The sound made when bonus points are gained is cued by a PLAY command.

Sprites of the ship seen from above are all linked to post 0. The number of the crabs doesn’t change when the stage changes; sprite and post numbers 2 through 4 are used throughout. You can increase or decrease the number of crabs, thereby changing the difficulty of the game. “IN-” is omitted in the MOVE command for the crabs because the sprite numbers and the post numbers are the same; the expression “MOVE 2 IN2-” is unnecessary.

MAKING THE GAME

In designing a game, the story is the first thing to think about. A game without a sense of drama is not really a game at all. Consider also how best to fit your game idea into the characteristics of a microcomputer. It’s important to try to use all the possibilities given by the M5 to the full.

Once you have a story, lay it out on paper. Draw the screen design in units of dots; having these plans makes creating the game much easier.

Once you’re ready to think about movement, plan the game in terms of sprites. The full use of sprites is the key to developing an exciting, feist-moving game. Without sprites, you would have to constantly use WRITE and DELETE statements to produce movement on the screen. Along with sprites, MOVE statements give you additional flexibility; the computer can do other processing while it moves patterns.

Depending on the game, you must choose the appropriate screen mode. The big choice is between GI and GII mode. If you use full graphics, you must set the mode to GII. If not, you can use GI. If you use both front and back screens, both screen modes cannot be set to GII. GI must be used instead. This is GI’s advantage: both screens can be brought into play. GI mode occupies less memory, so more data can be stored in VRAM. Color specification is easy, but you have a wider choice in GII mode.

A good game is neither too easy nor too difficult. If your points don’t increase with repeated playing of a game, it’s too difficult to be interesting. The best way to make a game interesting is to increase the level of difficulty as the game proceeds.

Once the game is completed, you can add sound. Music is easier than sound effects, although if the computer isn’t doing other processing at the same time sound effects pose no problem. Usually though, sound effects that occur in the midst of games happen simultaneously with other processing. Think out clearly before you write the program where you want the sounds to be.

INKEY commands are used for inputting data. Include these in loops so that input data is read. If INKEY$ is used initial input is performed, which means that keyboard input may go unread. In such cases use INKEY (0) or INKEY (1).

INPUTTING THE GAME

If you’ve already input a part of this program, now input the rest. If you haven’t input any, start from the beginning. Yes, it’s long—but input it all! We couldn’t include REM statements for reasons of memory. There are many multi-statements, which are a little difficult to get used to at first, but input the program as it is. Don’t change any line numbers: the GOTO statements will also have to be changed if you do.

10! \*\*\* SEfi BDUENTURE SOME <1> \*\*\*

28 Print. " HIST s view 1?0>36?23(console 81 o n error 9osut> 38 38 if er r \*2 8 then resumes return 48 Print "MM erase

58 for 1=4 to 6 s stchr rPt$< 1 6?"7 \*)to 128>Isstchr rPt$<8?”f4H

)to 241? isstchr r P t $ < 1 6 ? "4")t o 139? I!next

68 for 1=252 to 255sstchr rPt$(8» "37")to I ? 4 s n e x t

78 for 1=8 to 18 sprint cursor(0?I)? r P t $ < 3 8 > " M" > 5 snext

88 Print cursor <0?19)?rPt$< 38\* 1 ! for 1=28 to 22sprint cur

sor(0,I)SrPt$(38?"+,,>!Sriext

98 Print cursor (3?4) 5 5 cursor (3? 5) ! "W"?

100 restore 1828 s C\* = r P t \* < 1 6, " 8 " > s f or 1 = 172 to 21isread flUst chr fi$ to Is next

110 stchr C$ to 224 s st ch r C$ to 226sstchr C$ to 22?sfor 1 = 15 2 to 15 4s stchr C $ to I !next

120 stchr C $ to 132 s stchr C $ to 134sstchr C $ to 135

138 scod 0> 224 s scod l?152sfor 1=8 to 3sscod 2 + I?172 + 4\*Is next

sscod 6?132sfor 1=7 to 11 step 2s scod I?188sscod I + l?192snex

t

148 for 1=8 to 3s scod 26+It196+4\*Isnexts scol 8?13sscol 1 \* 14 s scol 2>4sscol 3?4!scol 4?2ssco1 5?8

150 scol 6?14s for 1=7 to 12sscol I?15snextssco1 28?15sscol 2 9 ? 15s scol 30 ? 8 s sco1 31?8

160 Jointsjoint 5 to 4?lsJoint 8 to 7?2sJoint 10 to 9 ? 2 s J o i n t 12 to 11,2

178 m a 9 2 s L1=L mod 6sprint cursor<25?28)!ML = Mnum$<L+l)!FX = 7 259osub 450s randomizeswove on

180 for 1=7 to 11 step 2sloc I to rnd<240)?rnd<55)smove I in 1/2-3 steP-1?8?3snext

198 loc 4 to-16?rnd<58)+28+L1\*8smove 4 in 3 steP 1?8?2

280 Print cursor <2> 2) Jnuwl (P) 5 cursor (24? 2) JnuwKPP) 5 "isMcurs

or <12? 2> 5 HHI = H •? nuwtCHI)

210 for 1=8 to 2 s i f statusCI>=8 then loc<I+3)\*2+l to 248?rnd <55)8moue<I+3>\*2+l in I steP-l?8?3

228 next!if sPrite<4?l>>248 then loc 4 to-16?rnd<50)+28+Ll\*8 smoue 4 in 3 step 1?8?2

238 if status<4> = 1 then 9 o t o 25 8 else BX=sPrite<4?l>sBV=sPri t e < 4 ? 8>

248 if BX< = 128 and BX> = 40 then loc 8 to BX?BVsmoue 8 in 4 to

BX,152,2

258 S=rnd<108)sif $ = >97 and st-at us ( 5) =8 then S8X = r n d < 88) +40 s 1 loc 2 to SfiX,151smoue 2 in 5 to SfiX,158,88-10\*L1 260 if S< 2 and status<5>\*8 then erase 2: loc 3 to 240,146!mou e 3 in 5 steP-1?8,2

270 K=inkey<l)!if K = 4 then s 9 3,5,15ss1eeP 3 , 1 ! s 9 3,,051oc 1 to sPrite<30,l)+8,136!move 1 in 6 step 3,8,2 288 if K=1 then s9 3,4,15ssleeP 3,l!s9 3,,0!loc 6 to sPriteC

28,1) + 1 2 , 18 4 s m o u e 6 in 7 step 8,-1,2 298 Kl=inkey<8>5if K1\*55 then FX=FX+1:gosub 450 308 if K135 5 4 then FX=FX-1!9osub 458

318 BO0\*coinc<0,28,31):if BO0>=28 and B08<=31 then goto 440 320 B01=coinc<2,28,29)!if B01=28 or B01=29 then goto 448 330 if co i n c(3,30,30)—30 then goto 440

348 B02sscoinc<6,4,5>5 if B02O4 and B02O5 then 9oto 368 358 for 1=38 to 88ssg 8 , I , 15snext ! s9 0,,0!loc 4 to 241,80!lo c 5 to 241,80!erase 6sP=P+100

368 if cpincd , 2,2) =2 then sg 3,6,15ssleeP 10,l!s9 3,,0:prin t "W" 5 s erase 2,l!P=P+100

378 if coined ,3,3)03 then goto 398 else Print cursor<8,15>

? r P t $ < 3 8 , M■">

388 s9 3,4,15!sleeP 10,1:s9 3,,0:PO=15 0- <sPrite<3,l>~sPrite<

38, 1>)sprint cursor(sPr ite(3,l)/8,15)J nuw$(PO)!erase 3,1!P = P + P0

398 if sPrite<4,3>=8 then scol 4,2 ! s c o 1 5,0 else scol 4,8ssc

o 1 5,2

400 PP=PP+1!P = P+1

418 if PP<568 then goto 288 else PP=0!cls!erase

1. 8 Print cursor (5, 18) 5 “CONGRATULATION ! \*« 5 c ur sor < 5 , 1 2) 5 " BONU S GftME STfiRT ! “s s L = L+1
2. 8 Play “si t178o59 89 8a898o6co5b8a89g898gr 4d8d8d8e8fe8d8cc8c 8 o 6 c "5 sleep 2\* goto 82 8

448 for 1=8 to 15:s g 3,5,15-IssleeP 10,1s next!9oto 4 88 458 if FX>18 4 then FX=184 468 if F X < 4 8 then FX = 48

470 loc 28 to FX,1295loc 29 to FX+16,129!1oc 30 to FX+16,145 !1oc 31 to FX, 145!return 488 Print "1" !erase

4 9 8 for 1=8 to 4 5 P r i n t cursor<8,I> 5 r P t $ < 3 0," ■“ > \* snext sprint cursor <8,5) 5 r pt$<30, ummm> ?

580 for 1=6 to 22!Print cursor(0, I)?rPt$(38,”+“);Jnextsprint cursor(3,2)5

518 LL = Ll+65 restore 112 8 s f o r 1 = 17 2 to 183!read flU stchr ft $ t o I! n ex t

528 for 1=2 to LLJscod X, 1 88 5 n ext 3 scod 0,172sscod 1,176

1. 8 scol 8, 1!s c o1 1,15 f or 1=2 to L L:s c o1 1,8!next
2. 8 Joint!joint 29 to 28,2s Joint 38 to 2 9,3!Joint 31 to 38,4

558 MX=56!MV=168!J=0!Jl=l!loc 28 to 48,16!move 28 in 8 step

1,0,15

568 for 1=32 to 168\* \oc 0 to 56,I!s9 8,1\*2,15!sleeP 1 , 1 :next ! s 9 8 , , 8

578 if sPrite<0,8><168 then goto 578

588 for 1=2 to LLsIoc I to rnd<248),rnd<112>+40!moue I step- 2, 0,3 ! n e x t

598 Print cursor<25,21)IV+iifor 1=2 to L L ! i f statusCI)<>0 th en goto 618

608 loc I to 240,rnd<112)+48!ittoue I st-eP-2,8,35SS = rr«d<58>!fo

r J = 208 + SS to 150 + SS steP-5ss9 8,J,J/205 next s s9 0,,0

618 next!K2=inkey<8>!if K2=51 then MV = MY-1:9osub 768

628 if K 2 = 5 5 then MX = MX+1!9osub 768

630 if K2=54 then MX=MX-1!9osub 768

640 if K2 = 46 then MV = MV+1!9osub 768

650 BO=coinc<0,2,LL)5if B0>=2 and BQ<=LL then 9oto 788 668 B0=coinc<i,2,LL>!if B0>=2 and B O < = L L then goto 788 678 MU=coinc<8,28,29)sif MU=28 or MU=29 then Play “sltl78o5a f M s 9 o t o 48 688 if s P r i t e < 2 8 \* 1)>224 then Soto ?88 698 Soto 598

780 for 1=0 to 18:sS 8\*150+1\*18,15:sieeP 5\*l:next:sS 8\*\*8 718 c1s:V=V+1:PP=8:erase:if V<3 then Print cursor(8,18)S"STfl RT ! " \* V + 1 :sleep 3:Soto 40 720 if P> HI then HI=P

738 Print cursor <8, ?)? “HIGH SCORE" ' HI > cursor (8? 18) \* "GAME OME R"1 cursor<7, 12) ;P \*M Points”leursor<8? 14)? " R E P L fl V < s/n > ? "

748 Z$=inke¥$:if Zt<>"y" and 2 $ < >"n " then Soto 748

758 if 2$="¥" then p=8:pp=8:L=0:V=0:Soto 48 else end

768 if J = 8 then J = 1 : J 1 = 0 else J = 0 : J 1 = 1

778 if M V < 3 8 then MV = 38

788 if MX < 8 then MX = 8

798 if M X > 2 2 4 then MX=224

808 if MV> 168 then MV=168

810 loc J to MX \* MV: 1oc J1 to 10O»-20s return

820 Print "i" s erase! stchr r P t. $ ( 1 6 » " b " ) t o 128\*4

838 for 1=0 to 4!Pr int cursor(8»I)!rPt$(30> "■") ! !next!Pr int

cursor (0\*5) S r P t $ < 38 > "\*\*\*“> $

840 for 1=6 to 22JPrint cursor(8? I)!rPt$(38> "+")»!next 858 restore 1158: for 1 = 172 to 183 tread fi$!stchr fl $ to I!next 868 scod 8,172:scod 1 \* 1 7 6 : f o r 1=2 to 4 : s c o d I\*180:next:for I = 8 to 4 : s c o 1 I\*8s next:joint:Joint 1 to 8\*3 878 UX=120s Sosub 990

880 for 1=2 to 4:1oc I to rnd<224),24:move I step 1 \* 8\* 3:next 890 for 1=2 to 4 s i f sPrite<I,1>>248 then loc I to 8,24!move I step 1\*8\*3

908 next:if status<8)=l then Soto 948

918 K3=inke¥(8):if K3=55 then UX=UX+1:Sosub 998

928 if K3=54 then UX=UX-1:Sosub 998

938 if K 3 > =9 and K 3 < = 16 then wove 8 step 8 \* - 1 , K 3 - 8

948 BBl=coinc<0\*2\*4):BB2=coinc<1,2\*4):if<BB1<2 or BBl>4)and<

BB2<2 or BB2>4)ther» Soto 968

958 K 3 = 8 s erase 0\*1: for 1=0 to 15:sS 3>5\*15-l:sleeP 10\*1:next

: sS 3 \* \* 0: S ot o 988

968 if sPrite<1 \* 8)<8 then P = P+<K3-8)\*500: for 1 = 1 to K3-8:pla ¥ " sit- 170o5af " : next JSoto 988

978 Soto 898

988 els!erase!Pr int cursor<9\*11)»“BONUS”J<K3-8)\*580:sleep 3: Soto 48

998 if U X > 2 2 4 then UX = 224 1800 if U X < 8 then UX = 8

18 10 loc 8 to UX \* 13 6: 1o c 1 to U X \* 15 2:r e t ur n

1028 data 000088081c1c3e3e,3e3e7f7f7f7f7f7f,0028203878787c7c

* 7e4e4eeeff f f f f f f

1030 data 8f8f 1 f1f3f3f7fff, 0000000000000000, fffef8e8c0c0e0e0

,0000000000000000

1040 data 181elflflflflfff,7f3f7fff80080000,000080c0c0e8e0fe , fb f f fefC80000000

1850 data 00080000ff7f3f7f,ff1f1f1f1f1f1e18 \* 80088000fefbfffe

* fce0e0c0c0S00008

I860 data 880000070f0f3f7f,3f1f7f1f01088000,78f8ffffffffffff

* f f f f f f f f f f3f0000

1078 data 8000ffffffffffff,ffffffffcf030000\*8800e0c0fcfcfefe

* fefefcfcf8e88008

1088 data 0000800008080001 \* 8187071 f1f7f7f88,8686061e1e7e7efe

* fefefefefefefe8e

1098 data 08c8c0c0f8f0 fc fc,f f f f f f f f f f f \* f f c8 \* 0080808880000800 ,0000C0C0f0f0fC00

1100 data ffffffffffff8000,0000000000000800,f8f8c0c000000000 \*0000000000000800

1110 data 3f3f0f0f03010000\*0000000000000000,ffffffffffff0000 ,0000000000080000

1120 data 83476733198f0703\*030303078cl83060\*c0e2e6cc98f0e0c0

* C0c8c0e030180C06

1138 data 8307078301878 f1b \* 3363438 78c183868 \* c0e8e0cB80e8f048 \*ccc6c2e038180c06

Illlilililsiilii:  
■■•■■■S'  
, . • ■

1150 data 01030307070 \* 6 \* 8f,1 a 1 a 1 a 1 © 1 a 1 a 1 a 1e,88c0c0©0e0i0i0f0 » 5858587858585878

1160 data 1 © 1 a 1 a 1 a 1 © 1 a 1 a 1 a , 0 1 0 i 8 f 07 070 30 3© 1,785^5858 7 8-585 850

,\*0f0\*8e0e0c0c880

;

1170 data 8000 1 4 1 c.888 a M 1 b • \* c 1 41 c9b 3 f C08088 >3f \*83\*d9\*C030880

■

r y » 4 I ^ $ i -

. 000028381051 Hd8

3

**SPACE**

**ADVENTURE**

HOW TO PLAY:  
SCENE 1  
(LAUNCHING)  
SCENE 2

(METEORITE SHOWER)

This program is an expansion of the previous one-scene game “Space Adventure.” The one-scene game is scene 2 in the expanded version, which adds scenes 1, 3 and 4. Scene 4 is “warp”, triggered by the CTRL key; we’ll describe it later.

Input the program. After you type RUN, the screen clears, its mode switches to full graphic (GII) and the picture at the top of the next page is displayed on the screen. This scene 1 is a demonstration graphic rather than part of the game. When you press [RETURN] after the display of scene 1, the mobile launching pad crosses the screen during countdown. When the launching pad stops moving, the engine ignites with a roar and the ship rises slowly toward space.

As the ship goes higher, it gradually disappears from the screen. When it’s completely gone, the screen clears and scene 2 is displayed. Now the game starts.

You control the ship’s movements in the same way as in the one-scene version of the game. The differences are as follows: a space map is displayed at bottom right; the quadrant at bottom left represents the earth. A red invader ship displayed at top right is your target. Your ship, represented by a small blue dot in the space map, aims for this target.

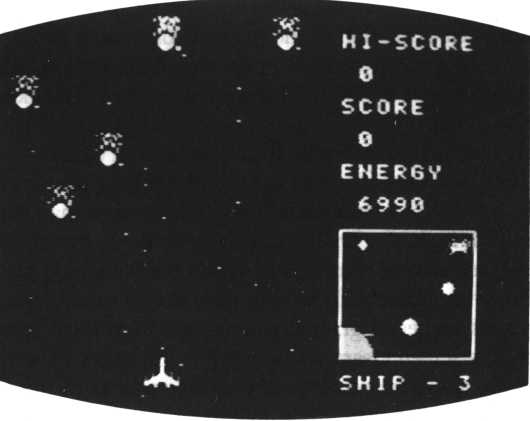
(The shades of blue and green used may vary depending on your TV set. Adjust the set for the best results.)

The key to successful play in scene 2 is to avoid the meteorites as much as you can. The more you fire at them to explode them, the more energy you use up and the less you have left to reach your target. Attack the meteorites only when you cannot avoid them. In an emergency, warp.

At the scene change from 2 to 3, you are halfway across the space map. You are out of the meteorite shower, but the invaders have appeared.



GAME STARTS



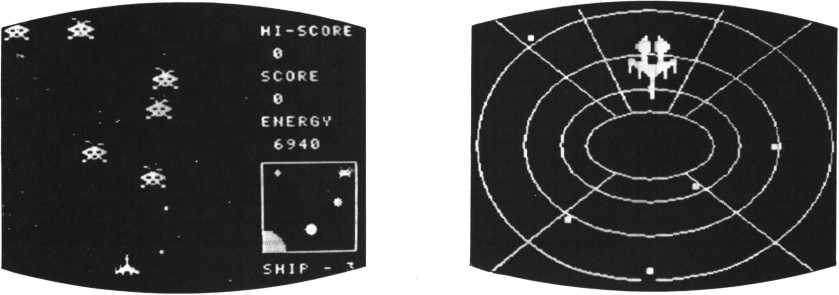
ATTACKS BY METEORITES

HOW TO PLAY:  
SCENE 3  
(INVADERS)  
SCENE 4  
(THE WARP)

Before you faced meteorites; now struggle against the invaders. The invaders’ speed is slower than that of the meteorites; but they are firing at you—and their aim is excellent. You must dodge the bullets; you can’t intercept them with a missile. The speed of the bullets keeps going up with every stage.

The way to win is to hit the invaders before they hit you. Yet you must conserve energy carefully. Shoot only when you have an invader in the line of fire, and use the minimum movement necessary to dodge their bullets.

For a temporary escape from this difficult situation, you can warp (scene 4). The screen mode switches to full graphic (GII). The ship enters a warp indicated by a set of ellipses and lights. But warp can last only so long—then you’re back in scene 2 or 3 (depending on where you warped from). Use warp skilfully— energy consumption during warp differs from the norm. You can warp only three times in any one stage of a scene. Also, as the stages proceed the ship moves higher and higher, making dodging either meteorites or bullets more difficult.



METEORITE CLUSTER WARP

PROGRAM TECHNIQUE

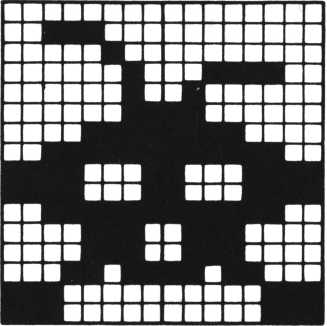
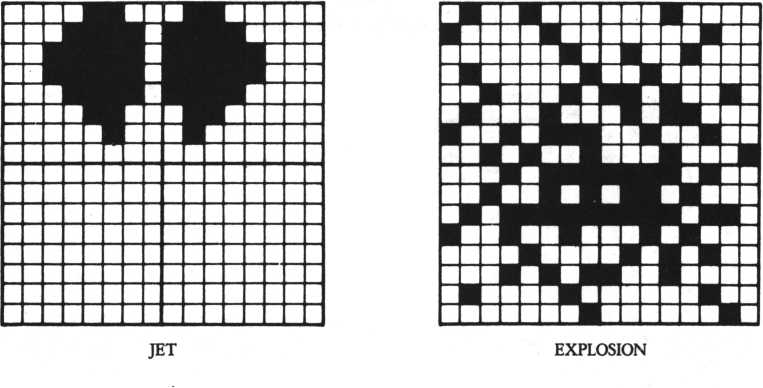
For the program technique of scene 2, see the appropriate sectin in Part II. This section mainly describes the full graphic mode (GII). In GII, a color can be specified for each dot but no more than two colors can be used for an image. As you can see in scene 1, if you attempt to specify three colors for an image, lines are displayed instead of dots. Since both front and back screens cannot be in GII mode at the same time, there are many control codes to handle this problem. In GII mode, we can set the area selected by the VIEW statement to full graphic. This is how the space map is made.

Up to six interrupt processes are possible using the M5. This program uses three of them.

The first is the ON ERROR GOSUB statement. When an error occurs during processing, this statement transfers the process to the line number shown by the GOSUB statement. This type of statement has the highest priority of the six interrupt processes.

The second is the ON EVENT GOSUB statement. This is used to interrupt at the interval specified by the EVENT statement. In this program, ON EVENT GOSUB is used to delete sprites.

The third is the ON COINC GOSUB statement. Each time there is a sprite collision, the COINC function checks their numbers.



INVADER

List of sprite assignments

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sprite | Character generater | Sprite number | Color | Post number | |
| Ship l | 0—3 | 1 | 15 | \ | ■ 0 |
| Ship 2 | 4—7 | 2 | 6 | > |  |
| Explosion | 12—15 | 0 | 8 |  | X |
| Meteorites | 16—19 | 21—27 | 7 |  | 1—7 |
| Missile(for ship) | 20—23 | 28-29 | 14 |  | 10—11 |
| Invader(for map) | 24—27 | 30 | 13 |  | X |
| Warp(lights) | 28—31 | 3-10 | 11 |  | 3—10 |
| Invader | 123—126 | 11 — 17 | 4 |  | 1-7 |
| Bullets(for invaders) | 127—130 | 19-20 | 2 |  | 8—9 |

INPUTTING THE GAME

Those of you who have input the program of the one-scene version of Space Adventure should compare that program listing with the one given here. You’ll see that the line numbers match up. The four-scene game is an expanded version of the one-scene game; you need only input the additions to the one-scene program.

Input it carefully, checking line by line, in text mode. When you first power on the M5, the screen mode is GI. To switch to Text mode, press [CTRL] and [T] together.

Instead of inputting parts of the program and then running it piece by piece, it’s better to input the whole program, make sure that it runs properly, and then set the parameters of the CONSOLE statement to 0,0. The CONSOLE statement inhibits the function keys and other leading input for the purposes of the gartie, which would make debugging difficult.

10rem SCENE-1

**26** clear > **&3FFD;** Poke & 8 **F F** E , 8 ,

38 randomize P e e k < & 7 8 4 9 > s o n -7000s U = 3 s T = 0 sS = 8 sU = 0

40 Print "IHiillSMT : ma9 2\*9 modes girt itsrestore sread fl$s stchr t o I »0s next

0!console 8, 8

error 9osub 270sH=Peekw<&8FFE>sE

220s for 1-8 to i1

50 31 68 70 88 83 s 90 100 9 move 110 120 138 148 150

scod 1 , 0s scol 8s sco1 31,18

fcol 15s for 1\*8 fcol 18 s b ar 0

1 » 15s scod 2? 4s scol 2, 6s Joint 2 to hi:scod

to 50s Plot rr»d<25 5>,rnd<151>snext  
188,255, 191 s fcol Usbar 8,184,255,187

fcol 7s 9wove 108,183s draw 112,176sdraw 143,176sdraw 147,1 Paint

128,180 fcol 14 s b ar 136,158,138, 9 m o u e 0,152sfcol 2:for

168 Paint 9 moue 9 mo ye 9 move f co 1

183s for 1=0

1 1

175s bar 138, 163, 135, 164 1=8 to 12:read sread X,V:draw

X,Y:dr aw X,Vs next! X , V s n e x t

t o

8, 183spaint 72,183spaint 280,183 32,39s fcol 9scircle 16sPaint 3 2,39 68,15sfcol 5s circle 9:P a i n t 68,15 224,55sfcol 7:circle SsPaint 224,55

14 s c i r c1e 28,8, ,-140, 170,70!circle 16,6,,-140, 170,7

0s c i r c 1 e 160 loc

18,7,,-140,170,78 to 120,161s1oc 2 " 6RME STfiRT 5 cursor <18,16) 178 if i n k e y < 0 > < > 8

-161sprint#l cursor<18,14) Return K e y M 5

120,

“Hit

**then goto 178**

**188** P **r** i n **t # 1 cursor<18,14)Stab<12)Scursor<18,16>Stab<14)SSfco 1 1 :bar 138,16 3, 135, 164s.joint 31 to 2,3**

198\_ f or 1 = 137 to 14ls fcol 14sbar I, 158, X+2, 175sfco1 8:bar I

8 , , 8 s s 1 e e P 3 8 ' 1 : n

t o

t, 158,1-1 , 175 5 s9 8,38 0, 15!sieeP 10, 1 s s9

next

208 loc 31 to 119\* 176s for 1=0 to 10ss9 0,0,15-lss9 3\*6\*15-1’.

sleep 18 >1 \* n e x t \* s 9 8, , 8 s s 9 3 , , 6 s m o v e 1 in 8 step 0,-1

218 if status<0>=l then 9oto 218 else 9oto 1880

228 data 8181818181818181,83871 f3ffCC08088\* 0000088800008000,

88c8f8f87e868008

238 data 0800800280000018 \* 18988080800C0C00,0008008880000018 \* 18128202006106000

240 data 8c1e3e3e1e8c0480,0000000800080000,68f8f8f8f0684888 \* 0000000008000000

250 data 4 \* 159 \* 12 \* 167, 18 \* 179 \* 28 \* 167 \* 24 \* 152 \* 28,16?»36, 183 \* 48, 162,52\*171,57,171,64,175,76\*144\*95,183

260 data 176,152,183,168,188,168,192,175,200,160,210,179,214 ,172,223,172,234,152,248,144,252,175,255,171

270 if<err=20>\*<<err1=2078)+<err1=2088)>then resume else Pri nt "MM?err,errl:stoP 1 080ren SCENE-2

1018 s9 3,6,©sprint " UHNMSSIM" s m a 9 2 s C = 8 s M\*8 s T = T +1 s N = 28 s R = 1 \* 9 o sub 2080son coinc 9osub 1328

1020 if T>1 then Print cursor<9,6)5“BOUNUS “ 5EsS»S + E/185E\* E+1008

1038 if T = 5 then 9oto 4888 else Print cursor(18,11>"ST86E -- “ 8T,cursor<l1\*16)5“Ready !‘“\*sleep 4 s Print "I"

1848 restore 1480sfor 1=12 to 27sread fi$5stchr 8$ to I,8snex t

1050 scod 1,8\*scol l,15!scod 2,6sscol 2,6s Joint 2 to 1, 1\*sco d 0,12sscol 8,8sscod 3 8,2 4 s s c o 1 38,13

I860 for 1=21 to 27:scod I,16 s sco1 1,7 s n ext s f or 1=28 to 29ss cod I,20sscol I, 14s next

18 70 Print cursor(23,1)? “HI-SCOREM 5 cursor <23,5> ?“SCORE"5 curs or <23,9) i “ENERGV” ? cur sor <23,22) 5 “SHIP

1088 locate 23,3sif H = 8 then Print H else Print H;n<-0"

1098 stchr “8000001008000000“ to 127,7s for 1=8 to 30sprint c ursor <rnd<21>,rnd<22)>5“«“5snext

1100 view 23,13,38,20s9 in itsbox 8,8,63,63 $fco1 12sgmove 8,63 scircle 16 \*,,278,368spaint 1,62

1110 fcol 9 s 9 m o v e 32,47scircle 4spaint 32,47s9move 18,7scirc le 2spaint 18,7s9move 58,28scircle 3spaint 58,28sview 1128 coinc on\*X«88sV=184-T\*16s1oc 1 to XYVMoc 2 to X,Vsloc 38 to 232, 104s s9 3,6,6 1138 if E<1 then 9oto 4880

1148 Print cursor<23,11> 5 E 5 slocate 23,7sif s = 8 then Print S else Print S s “ +-8 " 5

1150 if<S>=588)\*<U=0)then U = U+1 sprint cursor<29,22)»U»tU=l 1168 9 move 184, 167sfcol 0Jcircle B/58+17,,,315,315sfco1 4sci rcle 0 58+17,,,315,315sB=C

1178 view 0,0,21,23sprint "W"sprint cursorCrnd<21),0)5”9“S5C \*C + l8 v i ew:if C mod 10 = 8 then E = E-10 1180 if C< 2508 then 9oto 1218

1198 for 1=1 to 7sif status<I)=l then 9oto 1258 1280 nextseuent off s 9oto 1888

1210 if 01308 then if C<1325 then 9oto 1250 else N=10sR\*4se vent on

1228 Q«Post<1,7)sif<Q=-1>+ <C mod<R\*2 +1)<>0>then 9oto 1250 1238 M \* < r n d < 19 > +1) \* 8 s i f C mod<R\*7-l)=0 then M=sPrite<l,l) 1248 loc Q + N to M,0s move Q + N in Q step 0,1, R

1258 R\*inkey<8)\*I=inkey<l)\*if 1=4 then 9osub 1388 else if<I= l)\*<W<3)then 9osub 3888

1268 if 8 = 54 then X=X~<T+1>/2\*4s 9oto 1280

1270 if 8=55 then X=X+<T+1)/2\*4

1280 if X>168 then X=168 else if X<8 then X=8

1298 moue 1 in 0 to X,VS9oto 1138

1308 P = Post< 10,11>s if P = -l then return

1318 loc P+18 to X,V-16S moue P+18 in P step 8,-1,1sE = E-20sre turn

1328 if coinc(l)=-l then 9oto 1368

1338 event offsloc 0 to sPrite<l,1),VsU=U-lsuiewsprint curso

r < 29 > 22>5 U ?

1340 for 1=0 to 14: s 9 3,6, 14-1: sleep 1 5 , 1 : next: sleep Uerase

13 50 if U = 0 then 9 o t o 4000 else return 1120

1360 D=coinc<28,N+l , M + 7 > : i f DO-l then erase 28: 9osub 1390:r eturn

1370 D=coinc<29,N+l,N+7>:if DO—1 then erase 29:9osub 1390 1380 return

13 90 loc 0 to sl>r ite<D? 1 ) » sPr i t e(D> 0) : erase D:s9 3 » 6 , 1 5 s s 1 e e p 3 , 1:s 9 3> >6:erase 0:S = S+1 :return

1400 data 4884024122449729,17255f9208154201,10049820d258a411 ,f468f62854900402

1410 data 0922048402820808,2003870f0f0f8783,1048203088484010 ,1890c8e0e0e0c08c

1428 data 0000000088800000,8088000001010101,0800800800800800 ,0000000080000000

1430 data 0080880008242f2b,1f0f081080008080,800000002048e8a8 , f 0 e 0 2 0 1 0 0 0 0 0 0 0 0 8 2880r em SECNE-3

20 18 event 3 8, 12 8:o n event 9 o syb 2 8 4 8

28 28 restore 2890: for 1 = 124 to 131:read A $:stchr ft $ to I,8:n ext

2830 for 1=11 to 17:scod I,124:scol I scod I, 128:sco1 I ? 2:next:return

5 s n e x t : for 1 = 19 to 20:

28 4 8 i or 1=8 to 9 : i f s t a t- u s < I ) = 8 then erase I + 1 1 2850 next

28 6 0 2 = P o s t <8,9 > : i f Z = -1 then return

28 7 0 6 = r n d < 6 > + 1: i f s t a t u s(6)= 8 then return else O =

: sP ri t e < 6 + 1

8,8): i f 0 > = V-6 4 then return

20 8 0 loc 2+11 to s P r i t e < 6 + 1 8, 1) ,0:mo v e Z +11 in Z to X,V + 8,2- <T-l>/2:return

2098 data 8838040282030f3f,f1f13f8e8elbe8e0,8008083c48c0f8fc , 8 f 8 ffc7078d88707

2108 data 8103818080800000,#000888800008008,8 8 c 8 8 8 8 8 8 8 8 8 8 88 8 ,0000000080000088 3800r effi SCENE-4

38 10 event of f:coinc off:s9 3, ,8:e r a s e:v i e w:c1s:C = C + 2 8 8:E = E- 80sU=W+lsma9 3

30 2 8 9 init{restore 3 8 9 8: f o r 1=28 to 31:r e a d fl$:stchr ft $ to I ,0:next:for 1 = 3 to 10:scod I ,2 8 s s c o1 I,11:next 3830 fcol 7:K = 32:L=16:9move 132,95:for 1 = 1 to 5:K=I\*8 + K:L=I \*

8+L:circle K,L:next

3840 for 1 = 1 to 6:read K 1 , K 2 , L 1 , L 2 : d r a u K1,K2,L1,L2:next: fco 1 8:P a i n t 132,95

3058 scod 31,8 \* sco 1 31,6Moc 1 to 116,128Uoc 2 to 116,128:1 oc 31 to 116,159

3860 for ft = 0 to 28:s9 3,6,15-fl/2

3878 restore 3110:for 1=0 to 7:read K1,K2,L1,L2:1oc 1+3 to K

1, K2: move 1 + 3 step LI , l.2» 1 : si eeP 3, 1 : next: s9 3, , 8: next 3088 Print "MMiSlir: mag 2:9osub 2838:return 1850 3090 data 0000000000000001,0180000088000008,0000000080000080 ,8000000000000000

3100 data 109,75,24,0,189,116,24,191,154,75,240,0,154,116,24 0,191,120,119,87,191,144,119,176,191

3118 data 132,79,5,8,100,95,-4,4,116,63,8,-4,132,95,4,4,100, 79,-5,8,132,63,4,-4,116,95,0,4,180,63,-4,-4 4 0 0 0 r e m END ROUTINE 4810 event of f:coinc off

4820 Print “iftMMm1' 5 cur sor < 8,9) ? " == GAME OUER = = “

4830 locate 7,3s if<U = 8) + <E = 0>then goto 4040 else E = E~1080:pr int H C 0 N 6 R ft T U R ft T I 0 N ! ! I2I3KMCB 0 U N U S - - " 5 E ? : 9 o t o 4050 4840 if U = 0 then Print “SHIP ftLL DESTROVED M" else Print ME NERGV OUT !!"

4850 Print cursor<9,13)5"SCORE :if 5=8 then Print S else

Print S » M +-0 "

48 6 0 if H< = S then Print cursor(6,17)1HI - SCORE :Poke &8FFE» S mod 256,S/256

4878 Print cursor <9,21)5"One More RePlay << y / n >>"5 4888 A=inkey<0):if A=22 then run 38 4890 if A=38 then end else goto 4880

VIII

ADD

DRAMATIC

MUSIC

MUSIC PATTERNS:  
HOW TO MAKE MUSIC

Music is a way of expressing emotion. In movies, appropriate music goes with scenes that arouse emotions. Games on the M5, being dramatic situations, need music as well. This section gives some tips on how to program the M5 to make music.

Music can be classified into two emotional types: major-chord and minor-chord music. Major chords express happy or positive feelings and minor chords express sad feelings. Let’s see the difference, using the M5. Key in the following:

PLAY “C”, “E”, “G”

This produces a chord in C major: do-mi-so. It is a cheerful chord. Now key in the following:

PLAY “C’\ “E-”, “G”

This produces a chord in C minor. Compared with the first chord, this one sounds gloomy. This is because the sound of mi (“e” in C major) is lowered by a half-tone to “e-”.

Another important element of music is rhythm. Marches are good for victories in games; waltzes create a graceful atmosphere; slow, monotonous rhythms are sad. To write and play music with the M5, you use a PLAY command. For example, to play “do, do, re, re, mi, mi, re,” key in the following:

PLAY “CCDDEED”

You can write in this way for a brief melody, but if the music is longer or includes harmony, you should use a DATA statement which groups the notes into bar units.

Look at the program on this page. Lines 120 and 130 are used for the first bar’s melody and accompaniment; lines 150 and 160 are used for the second. Program the music in this style for easier debugging.

50 next I

60 Play "t80M

T0 d"im fi\*<2>,B\*<2> 28 restore 120 30 for I»1 to 2 40 read



70 PlaV '\* o5sll4tt >'\* o4s!18M

80 for J\*1 to 2



90 Play J),B$<J)

100 next J

110! 1 measure

120 data ccdd

130 data c9e9o3bo49d9

140! 2 measure

150 data eedr

160 data c9e9o3bo49d9

170 end

MUSIC FOR SADNESS  
OR DANGER

SAD MUSIC

This music is a melody in two chords, with an accompaniment. This is in D minor using D as do. Three “ # ”s are normally written at the beginning, but here the key of C is used instead because of its relation with the program data.

A tremolo is also included.

Bl$ and Cl$ represent two chords of the melody. Al$ represents the accompaniment. Each line of a DATA statement is used for a bar of each part. Lines 60 and 70 are very important: they are used for initial specifications set by PLAY. Line 60 specifies the tempo; line 70 we’ll need to describe in detail.

04 and 05 specify the musical intervals. SI and S4 specify the envelopes, which vary the tone. SI, a piano-like sound, is specified for the accompaniment (All); and S4, an accordion-like sound, is specified for the two chords making up he melody (Bll and Cl I). The “H8” specifies the length of each sound; 8 is the longest possible, giving a slur effect.

SAD MUSIC

J>= 116

COMPOSED BY KAZUAKI UCHIYAMA

|  |  |  |  |
| --- | --- | --- | --- |
| Ui; » r | = \*  1 |  |  |
| ll ] | =p | „ l Tft i - i | 1  j h , si | [ . 1 V •' |
| ® » 1. ¥ | » | V 9  0^^ ' | i di  mmi |

10 dim fll$<8> , Bl\*<8> ,Cl\*<8>

20 restore 130

30 for 1=1 to 8

40 read ft 1 $<I> ,B1 \*< 1 > >C1 \*<I)

50 next I 60 Play M t 1 16"

7 0 Play "s1o 4">"s 4o5 "> ”s 4 o 5 h8"

80 for 1=1 to 8

90 Play A1\*<I>,B1$<I)\*C1$'CI>

100 next I

110 goto 88

120! 1 measure

130 data dfad fa

14 0 data f 2.d 2.

150 data a 2 . f 2 .

160! 2 measure

1. 0 data c + e 9 c + e g
2. 0 data e 2 d c + 2.
3. 0 data 9 2 f 4 e 2.
4. 0 01 3 measure

210 data dfadfa 220 data d2.o4a2o5d 230 data f2.d2f 240! 4 measure

250 data c+e9c+e9 268 data c+2o4bo5c+2r 270 data e2d4e2r4 280! 5 measure

290 data dfadfa 308 data f2.d2.

310 d a t a a 2. f 2 .

1. 2 0! 6 ru easure
2. 0 data d 9 a + d 9 a +
3. 0 data 9 2 f 4 d 2 r

358 data a+2a492r 360! 7 measure

370 data dfac+eg

1. 0 data d 2 f c + 2 e
2. 0 data f 2 a e 2 9

400! 8 measure

410 data dfadfa

1. 2 0 data o 4 a 2 a a 2 r o 5 4 3 0 data d 2 c + d 2 r 440 end

MUSIC EXPRESSING  
DANGER

This music includes two melodies and an accompaniment. The melody in four measures is repeated twice; melody 2 is changed and repeated twice. The program returns to the beginning to play the first melody again.

MUSIC EXPRESSING DANGER

COMPOSED BY KAZUAKI UCHIYAMA



10 dim <3>,B#<3>,Ct<3>

28 restore 220

30 lor 1\*0 to 3

1. read «\*<!>,B\*<I>>C\*<I>

50 next 1 60 Play “t120“

70 Play " s 1 o 31 8 "» “ s 1 o 5116“>" s 4 o 6 '\*

80 lor K\* 1 to 2

90 lor J\*0 to 3

108 Play A\*<J>,B\*CJ>>C$<J>

110 next J 120 next K 130 lor M\*1 to 2 140 restore 380 158 lor L\*8 to 3 168 read D\*

170 Play C$<L>

180 next L

190 next M

280 9 ot o 80

210! 1 measure

228 data d dedd 9 I e

230 data deI 9a8de19a8deI 9

248 data a214aS18

250! 2 measure

260 data ddedd9le

278 data deI 9a8de19a8deI 9

288 data d4.e8!2

298! 3 measure

306 data ccdccled

318 data cde198cde198cdeI

320 data 92e498e8

330! 4 measure

348 data ccdccled

350 data cdeI 98cdeI 98cde I

368 data c4.d8e2

370! 5 measure

388 data a Ida Ida Ida Ida Id8

390! 6 measure

480 data deI 9a9Iede19a8r8

418! 7 measure

428 data 9ec9ec9ec9ec9ec8

430! 8 measure

448 data cde19 IedcdeI 98r8

450 end

MUSIC FOR VICTORY  
MELODY

This first peice of music is mainly in the scale of 05-the third from lowest octave.

As everybody knows, rousing victory music is march music. Marches are generally written in two-quarter time; you should do the same. The tempo should be brisk; even two-quarter time can sound gloomy if played at a slow tempo. Use as quick a tempo as possible—Allegro or Presto will do; in M5 terms, 134 or more for half-notes. The tempo of half-notes in the music here is 116. See your manual for how to specify tempos.

Next comes rhythm. A march rhythm has a certain bounce. It’s effective to begin with the rhythm alone (“tan taka taka taka tan taka taka taka tan taka tan tan tatta tatta.” It also helps to change the rhythm halfway through; start with a quick one and then slow down a bit. If you use triplets a few times, the music will sound more sophisticated.

|  |  |
| --- | --- |
| VICTORY MUSIC, NO. 1 yASU | |
| 4r nnh | J J |J J^j |
| Ar-j | J = | -it =T~ . >" o (\*==&-; |
|  | ^ Jd J-lf 1 |
|  | \* “ ~ " \* |
|  | |

Finally, interval. Don’t use intervals that are too high or too low. Too high intervals are merry rather than rousing; too low intervals are overly heavy and dull. For accompaniment, however, low sounds are effective.

10 r e m UICTQRY MUSIC i 20 Play "h 7ll6r 1 6H6I 16"

30 Play “ f 16c16\*16916"

48 Play “ a 1 6 r 1 6 a 1 6 a 1 6 "

50 Play " a 16c16a16b-i6"

60 Play " o 6 c 16 r 1 6 c 1 6 c 1 6 "

70 Play " c 1 6 r1 6 h 3 c 8 “

80 Play “c8o5b-8a898h?M 90 Play M2\* 8. \*1698. a 16\* 2c"

|  |  |  |
| --- | --- | --- |
| 100 | Play | “ \*8.a!6b-a9 \*92.r - |
| 110 | p 1 ay | "94.r16916“ |
| 120 | p 1 ay | "d16 r 8 d16916r89 16 |
| 130 | p 1 ay | " b - 2 9 2 e e \* 9 a 2 . r " |
| 140 | p 1 ay | "d 4. .el6\*o6c" |
| 150 | p l ay | "o5a2cb~" |
| 160 | p 1 ay | “ a 2 9 2 " |
| 170  180 | p 1 ay end | "\* f16r16\*16\*16“ |

VICTORY MUSIC  
(WITH HARMONY)

This music, being a duet, occupies more lines of code. The main scales are 04 and 03, which are low octaves. Musical notes and the actual scale are different by an octave.

Here’s how to use chords. A chord of and so is the basis for this

piece. When so is sounded in the melody, do and mi provide accompaniment. There are also chords of do, fa and la and of ti, re and so. If the note in the melody is la, you use fa in the harmony, if it’s re in melody, you use ti in harmony. This is the basis for the accompaniment. When the melody note is fa, sometimes do, sometimes re is used; when the melody note is re, do is used to produce a brief discord.

These rules aren’t foolproof—making good music is a lot more mysterious than that! But they give you a basis to start experimenting. One more note: the accompaniment does not have to have the same sound length as the melody; in the program listing given here they are often different.

VICTORY MUSIC. NO. 2 HASHMOTOY YASU



10 r e m victory music 2

28 Play Mc»4c8. ci6M, 9 8 a 9 f 2 r “ » ” o 4 c o 3 a c»4 c o 3 a o 4 c o 3 a o 4 c o 3 a \*'

30 Play M ab-ay \* 9 2r " \* “ o4co39fc o4co39 ©4co39 o4co39 "

40 Play “c8.c 1694.9 89 ab-242","o4co39o4co39o4co39 o4co39”

50 Play ,'cc8,b~16a9f2r”>“o4co3yo4co39c«4co3^o4co3^"

60 Play "o4fd2.98. f 16 c 2 \* '\* > \*\*03a8r8a8r8a8r8a8r8a8r8f8r8f8r8f 8"

70 Play Mo3ab-ab-%Herfr"

80 Play wo4d8.cl6o3a2.rtt,”\*8r8<8r8f2rH

90 Play “ 9894a8b-2a8ab-84‘ , "erer Ir f r?r9ro4clc2. H

100 Play “o4c2o3b8bo4c8d "

110 Play "o3bo4h8ele2.h 7 “

128 Play Mo4c8.cl6M.98a9f2r","o4co3ao4co3ao4co3ao4co3a" 130 Play Mab-a9 f 92r M > mo4co39o4co3^o4c£»~39g4c©39 “ —

148 Play " c8 . c 169 4.9 89 ab-29 2 " > M c«4co39 o4co39 o4co39 o4co39 " 150 Play Mcc8,b-16a9f2rH,”o4co39Q4co39o4co3fo4co3f"

160 end

CHEERFUL, HAPPY  
MUSIC

This music consists of one melodic note and two accompanying chords. A PLAY command consists of character-type array variables, manipulating DATA statements in terms of bars. The first and second lines are used for the accompaniment: the third line is used for the melody.

This music returns to the beginning (da capo) and ends at a marked point (fine). That is, bars 1 through 8 are repeated. In this type of situation, a PLAY statement handling variables is the easiest way to program.

HAPPY MUSIC

COMPOSED BY GO ASANO ARRANGED BY KAZUAKI J> = 170 UCHIYAMA



10 dim fi\*<8> >B\*<8) >C\*<8>

20 dim fll$(4> > B 1 $ < 4 >>C1$(4) 30 Play 1112 0"

40 Play "si o4 1 8" , " s io4 1 8'% H si o5 1 8

50 for 1 = 1 to 8

60 read < I > ,B\* < I > ,C$( I >

70 next I

80 for J = 1 to 4

90 read P1 \*<J> ,81 %<J) ,C1S<J)

10 0 next. J

110 for 1=1 to 8

120 Play I) >B$<I),C$< I>

130 next I

140 for J = 1 to 4

150 Play Rl\*<J>'81\*<J>>C1\*<J>

168 next J

170 for 1=1 to 8

180 Play »B\*<1> »C\*Oi

190 next I

280 end

218! 1 measure

220 data c999c9o39o49 230 data reeerere 240 data 99a9o6c4o5ba 250! 2 measure

260 data c999c999 278 data reeereee 280 data 949994r4 290! 3 measure

1. 0 0 data o 3 bo4 9 9 9 o3b o 4 9 9 9 310 data rdddrddd

320 data dddef4ed 330! 4 measure

340 data c9o39o49c999 350 data rerereee 360 data c4ccc4r\*4 370! 5 measure

380 data c999c9o39o49 390 data reeerere

1. 8 0 data 9 9 a 9 o 6 c 4 o 5 b a

410! 6 measure

420 data c999c999

430 data reeereee

440 data 9 4 9 9 9 4 r 4

45 0! 7 ft i e a su r e

460 data o3bo4999o?\*bo4999

470 data rdddrddd

480 data dddef4ed

498! 8 measure

500 data c9o39o49c99r

510 data rerereer

520 data c 4 c c o 6 c 4 o 5 r 4

530 !

540! 9 measure

550 data o39bo4dfbfdo3b

560 data rrrrrrrr

570 data ff4ed4ef

580! 10 measure

590 data 9o4ce9o5co49ec

600 data rrrrrrrr

610 data 999fe4r4

620! 11 measure

630 data o3fo4dfao5do4afd

640 data rrrrrrrr

1. 5 8 data aa 4 9 f 4 9 a

668! 12 measure

670 data o39bo4dfbfdo3bo4

680 data rrrrrrrr

69 8 data b b b b b 4 r 4

IX

THE WORLD OF GRAPHICS SIMULATION

PATTERN GRAPHICS  
DRAWING  
GEOMETRICAL  
PATTERNS

Let’s draw geometrical patterns on the full graphic screen (GII mode). List 1 draws an array of curves to form a pattern. The SIN and COS functions are effective for drawing patterns like this. Use just one color per line, though; because of certain limits in GII mode, multicolored fine lines will be blurry.

However, if the blurred lines themselves are part of the pattern, the result is attractive (List 2). Lines in two colors can be drawn if the program is written as shown in List 3. Lists 4 and 5 draw circles and ellipses with different semidiameters: because of the fineness of the dots, the whole screen can be turned into a vast pattern. (If you make a few changes in List 5, you have List 6, giving you the same pattern in blurred colors.)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 ! | 1 \*###\* | | GI 1-1 | ##### |  |
| 20 | Print |  | M(IIS»M : 9 | i n i t s b co | t $:01 |
| 30 | f col | &83:for | | X=-28 to | 120 step 5 |
| 40 | X 1 «s i | t n | <160,X> | s X 2 \* c o s < | 120,X) |
| 50 | for ] | I = | -55 to | 285 step | 65 |
| 60 | draw | X | l + I .5.X2 + I.190 | |  |
| 70 | next | I | , X t P r i n | t MW“ |  |
| 80 | 9 o t 0 | 80 | |  |  |

10! ##### 61 I - 2 (Color) #####

20 Print "WISM" s 4 i n i t.

30 for X“-20 to 120 steP 5 40 Xl»sin<160.X):X2=cos<120,X)

50 for I = -55 to 205 step 65:fcol 806

60 if I =- 55 then f c o1 & 0 6 ? g o to 10 0

70 if 1 = 10 then fcol & 8 C:9o t o 160 80 if 1=75 then fcoI & 8 ft ?9 o t o 180 98 if 1 = 14 8 then f co1 & 8 4 8 9 o t o 188

188 draw X 1 -PI , 5 ? X2 + I > 19 8 110 next I , X • P r int "W“

128 9oto 120

18! ##### 6 I I-3 #####

28 Print "IU;1H 1 s g in it 5 bcol &8 1

38 for X = -6 8 t o 6 8 step 4

48 X1= s i n <168,X) :X2 = 0

1. 8 for I =- 38 t o 2 9 5 s t e P 6 5

68 fcol &88 s dr aw X 1 + I , 180,X2+I, 198

1. 0 fcol &8fts dr aw X2 +I, 1,XI + I,99

80 next I \* X ? P r i n t " W\*

90 goto 98 18 ! ##### 43 J. 1-4 #####

28 Print “ WWM ? g i n i t ? b c o 1 & 01

38 groove 125,100?RX=08RV=0

40 rePeat?C2=rnd<11>+2?if C1=C2 then goto 48

58 RX=RX+8?RV=RV+2? fcol C2?Cl-C2?circle R X \* R V ? r a n d o ro i z e ? u n 11 1 RV>100?Print "W"

68 goto 68

10! ##### GI 1-5 #####

28 Pr int NMMH i9initibco1 &8 1

38 groove 1 25 >108? RX = 8 48 repeat

50 R X = R X + 2 ? fcol &07?Cl=C2?circle RX? randowize 68 until RX>168 78 Print HWM 88 goto 88

10! ##### 5 I I -6 ####4

45 C 2 = r n d <11> + 2 ? if C1=C2 then goto 45

58 RX = R X + 1 ? fcol C 2 ? C1= C 2 ? c i r c1e RX:ran doro i z e

DRAWING MOSAICS

Lists 1 to 3 are programs that draw small rectangles with a BAR command. Colors are changed to form a pattern. Although it’s time consuming, the end result is a beautiful mosaic pattern. If blanks are left and the natural black of the screen shows through, illusions of three dimensions can be created. List 4 draws a mosaic pattern in multicolor mode, which is excellent for this kind of mosaic work. This listing gives you an image showing a flower garden near a beach with the sea and sky beyond.

18! \*\*\*\*\* Part-1 28 Print "ISMT S9init5bcot & 81 38 for V\*5 to 185 stop 5 48 for X = 5 to 250 step 5 58 00\*00+1

68 if 00 mod 3\*8 then fcol & 0 8

78 if 00 mod 3\*1 then fcol &82

88 if 00 mod 3\*2 then fcol &86

98 bar X,V,X+5,V+5 100 next X,V5Print "M”

110 9 o t o 110

10! ##### Part-2 ##\*##

20 Print "WW" : 9 i n i t J b co 1 S81

38 for V\*8 to 188 step 2 48 00\*00+2

58 for X\*0 to 248 step 4 68 00\*00+1

78 if 00 mod 3\*8 then fcol 8c01

88 if 00 mod 3\*1 then fcol & 8 4

98 if 00 mod 3\*2 then fcol ?<8B

108 bar X,V,X+1»V+1 118 next X > V 128 Print "W”

130 9oto 130

10 ! \*##\*« Part-3 #####

28 Print "ISHT

38 9 i n i t s b co1 &81

48 for V\*0 to 188 step 3

58 00\*00+l+rnd<4)srandomize

60 for X\*0 to 248 step 4

78 00=00+1

88 if 00 mod 4\*8 then fcol &8G 98 if 00 mod 4\*1 then fcol &83 180 if 00 mod 4\*2 then fcol &87 118 if 00 mod 4\*3 then fcol £85 128 bar X> V,X + 4,V + 3 138 next X > V 148 Print “(«”

150 9 ot o 158

10!##### Multi-1 #####

20 Print "mm" i 9 in i t

30 for V=0 to 48:for X = 8 to 64

40 C0=rnd<2>+1:randomize

50 if V< 6 then C1= 8.07sC2=£0E : C3\*&0F

60 if V = > 6 and V<18 then C 1 =8.07 : C2 = 8.04 : C3 = &05

78 if V = > 18 and V<27 then C 1 = &03 : C2 = 8.6P: C3 = S.0B

80 if V = > 2 7 and V<30 then C 1 = &06: C2=&0R: C3=&09

90 if V = > 30 and V<33 then 0 1 =8.08: C2=&09 : C3=&03

108 if V = > 33 and V<42 then C 1 =&0R : C2»&0C : C3 = 8.82

110 if V = > 4 2 then C 1 = 8.8fl : C2=&02 : C3 = S.0B

120 on CO Soto 130\*148,150

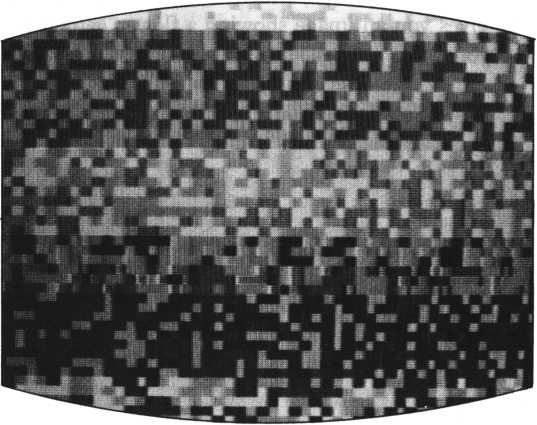
130 fcol Cl:goto 160

140 fcol C 2:Soto 160

150 fcol C3:Soto 160

160 Plot X,V:next X,V:print "W"

170 Soto 170



MOSAIC

DYNAMIC GRAPHICS  
USING SPRITES

The pattern graphics we’ve created so far are beautiful patterns, but they are rather static. We can introduce movement into these patterns by using sprites (Lists 1 to 3).

ie! ##### Sfrite-l #####

20 Print 'M2HT 5dinit!bcol &81 30 dmoue 127,103\*RX\*2\*RY\*8 40 repeat

50 RX\*RX+l€»sRV\*RY+12\* fcol rnd<13>+2 60 randomizeicircle R X , R V 78 until R X > 2 0 0

80 for 00=0 to 11:9osub 9SPSET:next 98 for C 0«0 to 11

10 0 if ?tat u s<C 0 > = 0 then d os vb 9 S P SET

110 n e x t s 9o t o 9 8

1289SPSET

130 mad 8: scod C 0 , & E 1 1 s c o 1 CO,CO+2

148 loc. CO to r n d U 8 > •+ 1 2 8 , r n d < 20) +90

1 5 8 9 S P H 0 0 E s X = r r, d < 7> +2: V = rnd <5> + 2

160 on r fid (3) + 1 doto 170,188,198,280

170 X«<-l-> \*X« Y\*<-1) \*Va 9©to 280

188 X\*<-t>\*V!doto 288

198 Y«<-1)\*Ytdoto 288

20 8 mo me CO in CO step X ,V,r n d <2 > + 1

218 randomizes return

10! ##### SPrite-2 #####

28 Print “ 1311" : d i n i t : b c o 1 Si 8 1

38 for C O~0 to 11!d o s u b 9 S P S E T 5 n e x t-

40 for C0=8 to 11

58 if status < C 0 > « 8 then dosub 9SPSET else dosub 9SPM0UE

68 n ext:d ot o 48

789SPSET

88 mad 8:scod CQ,fcE2\*sc©t C0,C0+2 98 loc CO to rnd<38>+80,rnd<38>+88 1009SPMOUE! X«rnd(5)+HV\*rnd(5)+2 110 on rnd<3>+! doto 120,130,148,158

12 8 x \* < - I > \* X i V a < - 1 > \* V ? 9 o t o 15 8

138 X«(~l>\*X\*90t0 150

140 V\*<-1>♦VS 4 ot o 150

150 move CO in CO step X»Y»rnd<5>+5

168 randomizesreturn

10! ##### SPrit\*-3 #####

28 Print "MR" s9iftitsbco 1 &81

1. 8 9os u b $ 0 C I R C L E
2. 8 4 o r C 0 \* 8 to 1 1 : 9 o s u b ISPSETi next.

58 for CO\*0 to 11

60 if st at us < C0> =8 thon 9 osub $SPSET

1. 8 next 9 o t o 50
2. 0 $ S P S E.T

98 scod C0?&9fl:scot C0?C0+2 108 loc CO to rnd<185>+28'198

c W

1 1 8 % S P M 0 U E : m o u e CO in CO to r n d < 8 > + 1 2 8 ? r n d < 2 > + 9 3 ? r r. d <

128 randomize:return

138S0CIRCLE

148 9mcue 126? 96s R X \* 3 : R V \* 2 150 repeat 168 X«RV/15+1

17 8 F-RV mod 15: if F = 1 4 or F = 8 then X = 8

180 on X 9oto 190? 28 0? 218? 220? 2 3 8? 24 0? 2 50? 26 0

198 C0 = &04:9 ot o 278

280 COas&05:9oto 278

210 C 0 \* & 8 C s9oto 270

228 C0\*&828 90to 270

238 C G = & 8 3 :9oto 270

248 C0=&0P:9oto 270

258 CO = ?<0B89oto 270

260 if R V > 5 8 then CO\*&09 else CO\*&06

270 R X = R X + 8 sRV-RV+2

288 fcol CO:circle R X ? R V

298 until RV>108

308 fcol &0D8Paint 1 26 ? 96 ? ?<04 : r et ur n

USING CHARACTERS

We express movement by displaying numerous characters, using PRINT # 1 statements, and erasing them or changing their colors. This is all done in in GII mode (Lists 1 to 3). The characters are easily painted different colors by the FCOL command because the mode is GII.

10!##### CHR-1 #####

20 on alarm 9osub SFIRST

1. 0 on event 9 osub STWA IT
2. 0 t imeS="80:8©:00H:alarmS = H©0:82"

50 event 5 8 > 1 5: P r i n t. " ISM" : 9 i n i t

60 for V = 0 to 2 2 step 2 70 for X\*0 to 31: fcol rnd<!3>+2

80 P r i n t # 1 cursor (X»V) $chrS<&El> ’ cursor (XjV+1) JchrS(&E4> ;

90 next V:alarm on!event on

100 X = r nd < 31) sV = rnd < i1)\*2iC = rnd <13> + 2:9osub SPANS:9ot o 100 118STWAIT:sleep 1? rnd(9) + 1s randomize rnd <108)+1:return 128SPANM:s9 2>18>0:s9 3,7,©:for A = 8 to 40:s9 3,\* 8-A/5:next:r e t u r n

1 3 0 $ FIR S T:a 1 a r m off:event off

1. 0 for 1=0 to 1 0 : f o r V = 8 to 22 step 2 : X » r n d < 31) s fcol r n d < I 3 )+2:randomize!9osub SPANG:next V » I : P r i n t " ffl"
2. 0 9 o t o 4 0

16©SPANS:fcol C:Pr i nt #1 cursor <X,V> 5 chr S<&E©> ? :9osub SPANN 17 0 fcol 0 : P r i n t # 1 cursor <K»V)! chrSU20> ?cursor<X» V+l) ?chrS| &20) 5

180 randomize:return

19 0 fcol 0 : P r i n t # 1 cursor <X» V) Jchr$<S20) lcursor<X»V+l) lchrt:C & 2 0 > ?:return

10 » ##### CHR-2 #####

20 Print WMinit

30 for 1 = 1 to 7 6 8: fcol r nd <13>+2: Pr int#i chrS(rnd(3>HEC> Hn ext

40 for F C = 8 to 2:for 1=0 to 100 50 for V=0 to 23

60 if FC=0 then C1=4:C2=6:C3=3:C4=5:C5=2:C6=7:C7=12

70 if F C=1 then C1=7:C2=12:C3 = 9:C4 = 3:C5=10:C6 = 4:C7 = 6

80 if FC = 2 then C1=9:C2=11:03=10:C4 = 4:C5 = 6:06 = 2:C7 = 5

90 X»rnd(31):R=X^2+V^2:if R<35 then C=C1 else if R<120 then

C\*C2 else 9oto 110

100 9 o t o 120

110 if R< 290 then C = C3 else if R<480 then C\*C4 else if R<745 then C = C5 else if R<1000 then C = C6 else C = C7 120 fcol C:Print#l cursor(X»V)!chrS(rnd(3)+&EC)5:randomize:n ext V»lS9osub S50UND:next FC:9oto 40

130$SOUNDs I ■ 3 s 9 osub $ SOUNDR 8 I \* 1 8 9 osub $SOUNDR s I = 2 : 9 osub $SOU H0R5 return

148$S0UNDR2 for J = 1 to 2s for FS = 8 to 300 step 2sS9 I,,15-FS/2 0s next FS> Js return

10! •\*##« CHR-3 #####

28 Print "IW s 9 in it

30 for 1 = 1 to ?688fcol r n d < 1 3) + 2 s p r i n t # 1 chr$(&3F) Hnext 40 RX=rnd<20)+5sRV=0sBX=rnd<20)+5sBV=23sfcol &088Print#l cur sor < RX > RV) s chr \* <£99) 5 s fcol **&0,5s** Pr in t #1 **cur** sor < BX, BV) S **ch** r $ <&8 6) 8

**50** RXX = RXs RVV = RV s 9 osub $60RspoP RX,RVsfco1 8e08sprint#l curso r(RX,flV),Schr$<flCH> ?s BXX = BX s BVV = BV

68 9osub **$608:** P oP BX,BV8fco1 &85sprint#l cursor<BX,BV)5chr$< BCH)5 s i f RX = BX and RV = BV then 9osub $CLRSH

70 fcol rnd<13)+2sprint#l cursor<RXX,RVV>5chr$<&E2>?8print#l cursor<BXX,BVV>Schr$<&82> 5 srandomizessleep 1,**58** 9oto 58 8 0 $ 6 0 R sX = R X sV = RVs 90to 188 90$GOBs X = BXs V = BV 100 randomizes FF=rnd<3> + 1s on 110 X = X-1 8 RCH = **8i988** BCH **= &898** i **f** to 158

**FF 9 ot o 1 10,128, 130,140**

**X<8 then X=X+l89oto 188 else 9o**

120 X\*X+18 RCH«$9B s BCH = &8R s i f X>31 then X=X-lS9oto 180 else 9 oto 158

130 V\*V-18RCH=fc9flsBCH=&852if **to 150.**

**V<8 then V=V+iS9oto 160 else 9o V> 23 then V=V-l89oto 100 5 s s9 1,108, 15s sleep l,5ss9 1,,**

140 V = V + 18 ft C H = & 9 9 s B C H = & 8 6 s i f 150 Push V,Xsrandomizes return 168$CLRSHs s9 1,208, 15s sieeP 1 0s return

USING MULTICOLOR  
MODE

You can’t draw fine lines on the screen in multicolor mode (GI) because it’s a semi-graphic screen. However, you can still get patterned effects in GII mode by specifying a color for each pixel and then letting them blur.

List 1 is a program which draws a moire pattern. List 2 is a color test program using a GMODE statement. Modes are changed at line 170; modes 1 and 6 are the most interesting parts of this program. The errors produced by the PAINT command are processed by the subroutine ($AGAIN), but outside modes 1 and 6 the subroutine may not perform properly.

If you run this program for too long, the display gradually distends and loses shape with continued executions of PRINT” G”.

10! \*\*\*\*\* Mu11 i-1 \*\*\*\*\*

20 Print “WM" s 9 i n i t: b co 1 S01

30 lor V\*0 to 101

40 C = V/10 + HC = C rood 2 + 1

50 if C=1 then 0\*0 else C = 3

60 Ci\*C08CO«rnd<2>+l8randomize

70 if C0\*C1 then 9oto 60

80 if V mod 18\*0 then FV«389oto 140

90 if FV\*3 then F V = 8 J Soto 158

100 on CG+C 9oto 118,128,130,160,170,180

110 fcol &04:9oto 198

128 fcol &85«9oto 198

138 fcol &0789oto 198

148 fcol &08\* 9ot o 190

150 fcol & 0 9 • 9 ot o 198

160 fcol &02 \* 9 ot o 198

178 fcol &0389OtO 198

188 fcol & 0 C s 9ot o 190

190 for X»-300 to 480 step 18

208 if X>-18 then SX\*sin<75,X>

218 if X<=-10 then SX\*sin<218,X>

228 5V\*s i n < 58,V)

230 Plot X/10+SV,SX/17+V/2 248 next X, Vs Print "M"

250 9oto 258

10! ##### Multi-2 #####

20 on error 9osub $ R G G I N 30 Print " HUM" s 9 i n i t 50 for 6=1 to 16 60bcol 6 : 9 ro o d e 0 80 C 1 = - 1 ! C2 = 1 6 5 KX = l : VV = 1 90 for V=2 to 42 steP 5 100 for X = 0 to 60 step 4

110 if V = 2 then Cl\*Cl+lifcol Cl!9oto 140

120 if V = 4 2 then C2 = C2-1: fco1 C2\*9oto 140

130 fcol rnd( 14)+1s randomize

140 bar X,V,X+2»V+3

150 next X,V

1 6 8 $ N U R U

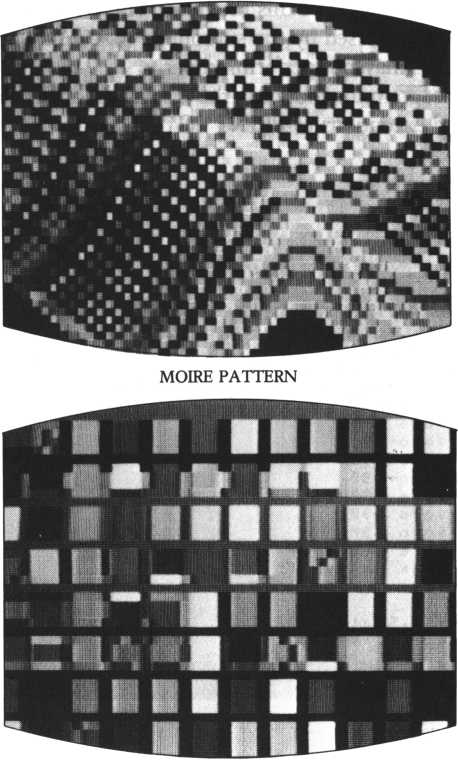
170 9 mode 6

180 fcol 6~ispaint XX,VV,6-1 190 sleep 2> 60s next 6 200 Print "W"

210 9 ot o 40 220\*fiGAIN

230 Print " sVV = VV + 20

240 resume \*NURU



COLOR TEST

AFTERWORD

RECORDING THE  
ANIMATION:  
PHOTOGRAPHING  
SCREEN GRAPHICS

One way to record your animation work is to take a photo of the display using a still camera. The other way is to record the screen images and sounds on videotape using a home VTR.

To take a photo of a video screen image:

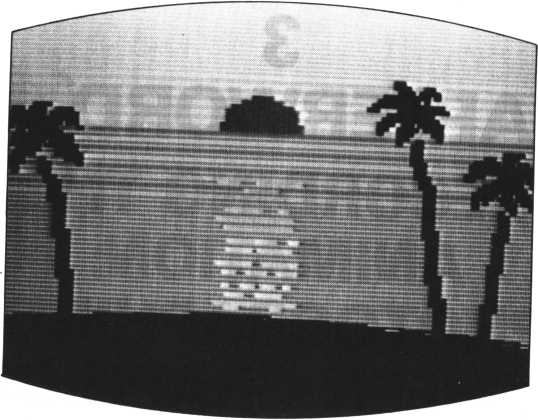
Prepare a 35mm single-lens reflex camera, macro or 100m lens, a tripod and film. If you use 150/100 film, drop down to F4 and a shutter speed of l/30th second-or F5.6 and 1/15th second.

Unless the shutter speed is l/30th of a second or less, the scanning lines of the video display will appear on the photo. Brightness depends on your particular video, so take photos using different exposures and find the most suitable one for your display. If you use color film, get film for daylight and use a filter to compensate for the strong blueness of the TV screen. Modify the program to “freeze” at a given point, so that your slow shutter speed is not thrown off by any movement.

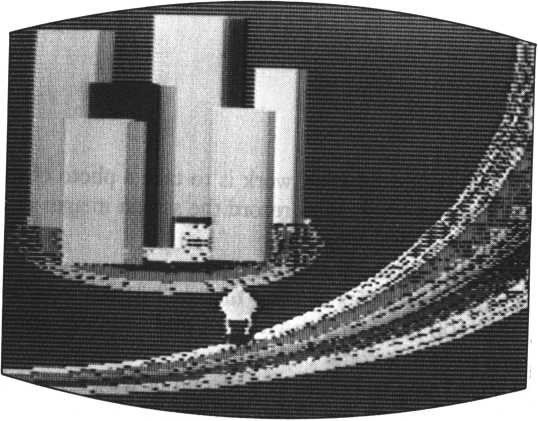
To record animation using a home VTR:

Connect the video output terminal of the personal computer and the video input terminal of the VTR. Connect the SOUND output terminal of the personal computer and the sound input terminal of the VTR.

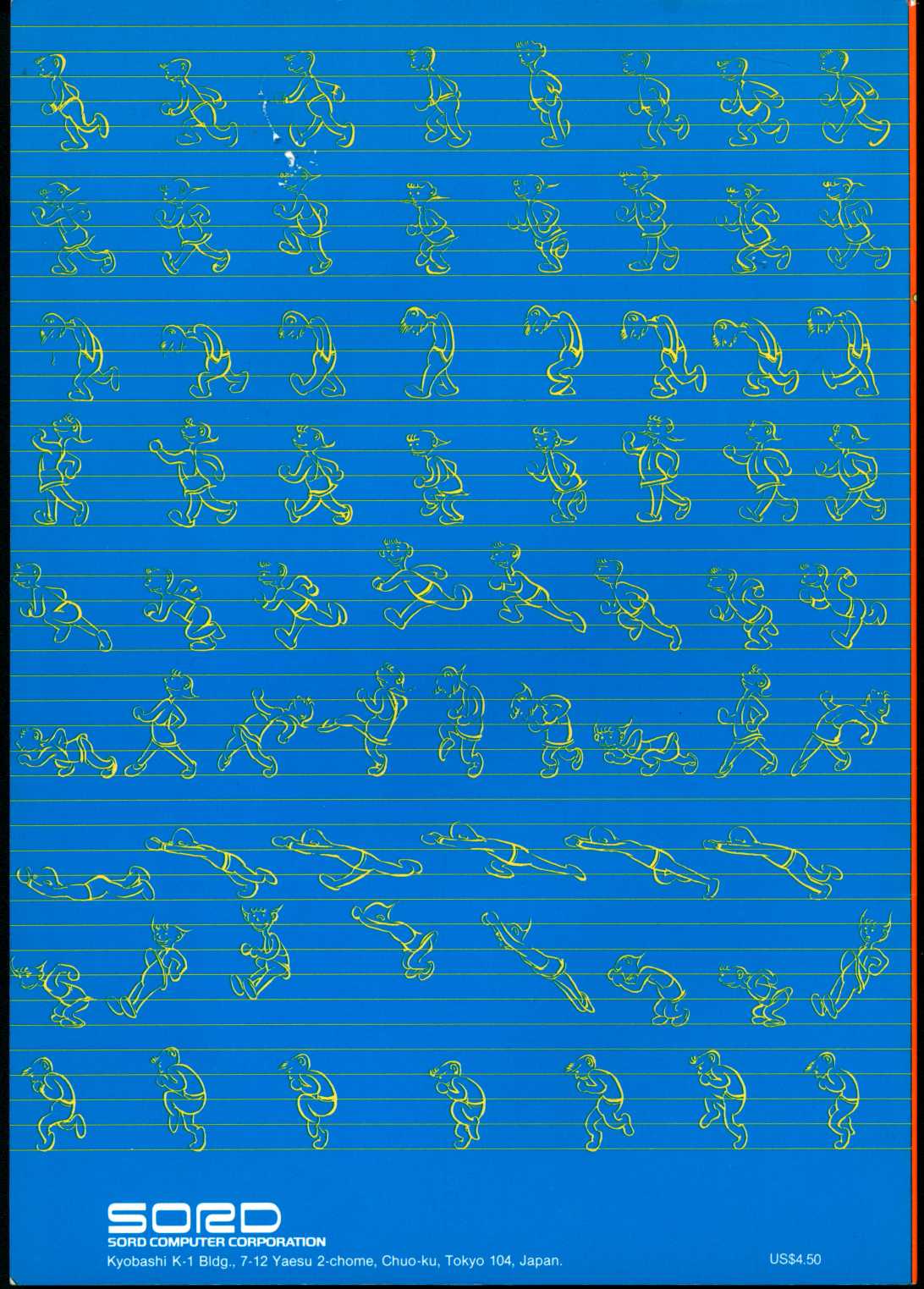
Set the video tape and prepare to record. Now run the program. You’ll have a complete record of your animation work on videotape.



TWILIGHT SCENE



CITY OF THE FUTURE



1 0r em SCENE-1

20 cl ear,&8FFD!Poke & 8 F F E ? 8 > 8 scon sole 8-8

30 randomize Peek < & 7 8 4 9 > son error 9osub 2 7 8:H = P e e kw< & 8 F F E >:E = 7000! U = 3! T = 8 s S = 8;U = 8

40 Print 1 11MMHM" ! m a 9 2! 9 mode! 9 in it: res tore 2 2 8 ? f o r 1=8 to ?! read fit!stchr A $ to I? 8s next!9oto 18 88 60 9oto I860

220 data 818 1818181010 18 1? 8 38 7 1 f 3 ff c c 8 8 8 8 8 ^ 0 0 8 88 8 8 0 0 8 00 8 0 0 8? 88c0f8f87e068880

238 data 8088880200800818,18908880088c8c08,8888888888088810, 1012020280606080 278 Print "II"; err > err 1 s stop 1 080r em SCENE-2

1018 s 9 3> 0! Pr int " WIMSWMT 5 m a 9 2! C = 8 ! W = 8 i T = T+1 i N = 28 ! R= 1 ! on

coinc 9os u b 1328

18 20 if T>1 then Print cursor <9? 6)>"BOUNUS " JE!S = S + E'18!E = E+ 1080

1038 if T = 5 then 9oto 4888 else Print cursor <10\* ID "STAGE - - "5 TS cur sor <11 \* 16>!"Ready H"s sieeP 4 i Pr i nt

1048 restore 1480!for 1=12 to 27:read :stchr to I,8!nex

t

1. Print "ISBM" [↑](#footnote-ref-1)
2. 20 v i e w 1,8,30,23 3 0 r em sprite

   40 stchr •'0103020303030303“ to 252 50 stchr " 0303030303000000 \*' to 253

   1. 0 stchr 11 8 0 c 0 4 0 c 0 c 0 c 0 c © c 0 " to 254
   2. 0 stchr "C0C0C0C0C0000800 " t o 255
   3. 0 stchr “0 0 00 0 0 0 0 0 4 0 4 0c1c" to 248

   90 stchr M7cf8\*0800003070\*“ to 249 100 stchr ,,0000000020203038M to 250 110 stchr "3eH070100c0e0f8" to 251 [↑](#footnote-ref-2)
3. 0 m a 9 2 [↑](#footnote-ref-3)
4. 138 scod 0 ? 2 5 2 [↑](#footnote-ref-4)
5. 0 s c o1 0 , 7

   150 scod 1,248 160 scol 1,2 178 Joint 1 to 0,1 1 8 0 r e m m o u e . . to 198 moue on

   208 move 8 in 8 on 110,175 210 mome 8 in 1 to 118,8,5 220 end [↑](#footnote-ref-5)