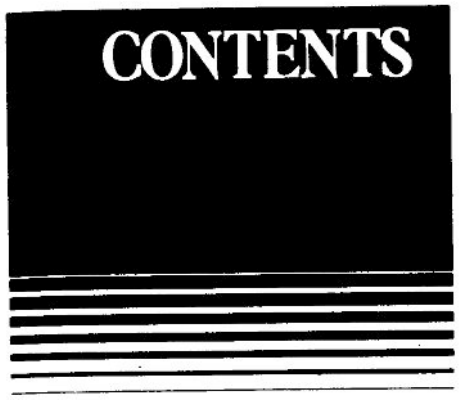
INTRODUCTION TO ADVANCED BASIC COMMANDS AND CONCEPTS

COMMODORE

INTRODUCTION TO ADVANCED BASIC COMMANDS AND CONCEPTS



This course is designed to cover advanced BASIC commands and concepts for the Commodore 128, Plus/4 and C16 computers. This third book in the Programming Course consists of two parts:

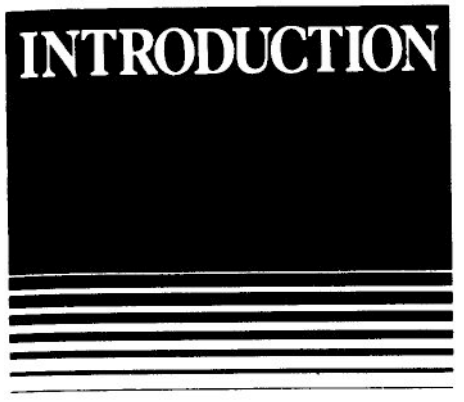
1. A self-study text of 11 units, each covering commands not covered in the first two books that focus on the C64.

2. One diskette, containing programs designed to run on all three computers, or specifically on the C128 or Plus/4 and C16.

CONTENTS LIST

Progroms Included on Disk, By Unit

	Program Name
Unit 1	COIORCHANGE
	QUIZI
	QUIZ1/4
	BOXING
	CALORIE COUNTER
Unit 2	BANK STATEMENT
	MAILING LIST
	MAILING L1ST/4
	QUIZ2
	QUIZ2/4
Unit 3	REAL ESTATE
	DEBUG EXERCISE
	TRACEEX
Unit4	EDITQUIZ
	EDITQUIZ/4
	WINDOW DEMO
	WINDOW DEMO 2
	GASPUMP
	HOROSCOPE
	QUIZ4
	QUIZ4/4
Unit 5	QUIZ5
	QUIZ5/4
Unit6	TRIVIA
	TRIVIA/4
	TRIVIA/16
Unit7	COLORPLUS
	COLORPLUS/4
Unit 8	RUG
	TRIANGLES
	OCTAGON
	PAINTDEMO
	HOUSE DRAWING
	HOUSE DRAWING/4
Unit9	SWIRLS
	SCALES
	SCALES/4
	QUIZ9
	QUIZ9/4
Unit 10	MAKESSOUND
	DEMOSOUND
	SWANSONG
Unit 11	FISHTANK
	TROPICAL



This book reviews the advanced BASIC languages that go beyond the scope of Commodare 64 BASIC 2.0. In the first two books, you learned about programming with the commands available on the Commodore 64. But updated, more advanced versions of the BASIC language that include the old BASIC commands plus additional new keywords are being used in the newer Commodore computers. *Introduction to Advanced BASIC Commands and Concepts* goes beyond the scope of Commodore BASIC 2.0 by presenting the keywords used in the newer Commodore computers. Many of the new commands in BASIC 3.5 (for the Plus/4 and C16) and BASIC 7.0 (for the C128) add versatility to your programs-saving you time and increasing what you can do with your computer.

This volume will:

- teach you the new, advanced BASIC commands
- demonstrate the power of these commands on your computer
- help you design programs which incorporate these commands

Some of the topics explored related to the new commands are:

- structured programming
- formatting output
- the use of windows
- advanced data input
- screen editing
- graphic design
- sprites
- sound and music
- other advanced techniques

There are also experiments in each chapter, designed to help you apply what you learned about the new commands and concepts to programs or exercises. These hands-on experiments will increase your understanding as you use the new information in actual programming situations and problem-solving. Of course, if you need a little help, the answers are in the appendix.

Some discussions in the book may apply only to a certain version of BASIC or a certain computer. The section on Sprites, for example, discusses BASIC 7.0 commands available only on the C128. Other topics, such as music, are tailored to your specific Commodore computer. Most concepts and commands, however, are relevant to all three computers, the C128, the Plus/4 and the C16. A machine-specific topic or command is designated by a stripe on the upper corner of the page. Look for this stripe to see if the concept being presented relates to a specific computer: a black stripe for BASIC 7.0 and the C128; a gray stripe for BASIC 3.5 and the C16 and Plus/4. If there's no stripe, the material is relevant for all three computers.

For your convenience, this book includes a separate disk containing quiz and demonstration programs and also many of the longer programs listed in the text. Because of the differences between different versions of BASIC, you'll find certain programs don't run on your machine. Certain programs have two versions, one for the C128 and one for the Plus/4 and C16. The latter programs appear with a "/4" tacked on the end of the program name. Programs without the extra" /4" run either on the C128 alone, or all three computers.

Whatever computer you have, this book will help you understand some of the new commands and concepts of Commodore BASIC.

UNIT 1:

Structured Programming

This unit introduces commands to help you improve the structure of your programs. These allow you to better organize your programs sequentially, keeping different subroutines in the order they are used in your program. Structured programs are easier to follow (particularly when using flow charts) and mistakes are easier to track down. Commands reviewed in this unit that improve structure are DO/LOOP, the **ELSE** clause and BEGIN/BEND.

DELAYS

Type in this short program, then RUN it to see what happens. Press the RUN/STOP key when you've seen enough.

```
10 X=X+1:PRINT X
20 PRINT"[HOME ][SHIFT ] and S"
30 PRINT"[HOME ][SHIFT ] and Q"
40 GOTO 10
```

At the speed this program is being executed, it's hard to tell exactly what's going on. One of the uses for loops is to create a delay, to slow down the speed of execution of certain parts of a program. Usually, an "empty" FOR/NEXT loop is used to slow execution. This is a FOR/NEXT loop that does nothing but count a variable to slow down the execution of a program. You can see what this delay loop does by adding this line to that program:

```
25 FOR N = 1 TO 100:NEXT N
```

BASIC 7.0 contains a command designed specifically for the purpose of slowing program execution, the SLEEP command. In one simple statement, yau can create a delay just by instructing the computer to SLEEP followed by a value corresponding to the time in seconds for the delay. You saw what a difference the empty FOR/NEXT loop made. Using the SLEEP command, you can create a comparable delay of one second by replacing line 25 with:

```
25 SLEEP 1
```

You can see how important delays can be by comparing the program with and without line 25. Without the delay, the heart barely flickers and the counter numbers fly by. Delays using the FOR/NEXT loop and the SLEEP command give you control over the speed of what happens on the screen. You'll notice delay loops turning up in a lot of programs in this section (and the rest of the book).

LOAD the program COLORCHANGE* from your disk. This program features the COLOR command. COLOR is used to change the background, border, character or secondary colors. COLOR is followed by two numbers, the first specifying what is to be changed (4 for border, 0 for background, 5 or 1 for character*), the second corresponding to the new color (1 for black, 2 for white, 3 for red, etc., according to the color keys). The COLOR command for the Plus/4 and C16 can use a third number to change the luminance, or brightness, of the color. This number can be between 0 (dark) and 7 (light). The program first cycles through the 16 different colors for the screen border, then the 16 colors of the background, and finishes by printing characters in all the available colors. The only thing wrong is that you'd never know it, because the program doesn't have any delays to slow things down. When you RUN the program, the color changes whiz right by.

(a) Add an empty FOR/NEXT LOOP or a SLEEP statement to slow down the changing border colors.

(b) Add a second delay to slow down the changing background colors.

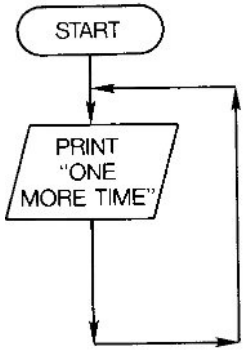
(c) Add a third delay to slow down the changing character colors.

* Experiment 1.1 Completed *

*Note: Change line 110 to read: 110 COLOR1,X,4 to RUN this program on a Plus/4 or C16.

DO/LOOP

DO and LOOP statements can only be used in a program and must always be used in conjunction with one another. The statements following the DO statement are carried out; upon encountering the statement featuring LOOP, control is transferred back to the DO statement for the sequence to be run again. The Simplest form of DO/LOOP:




```
10 DO
20 PRINT "ONE MORE TIME"
30 LOOP
```

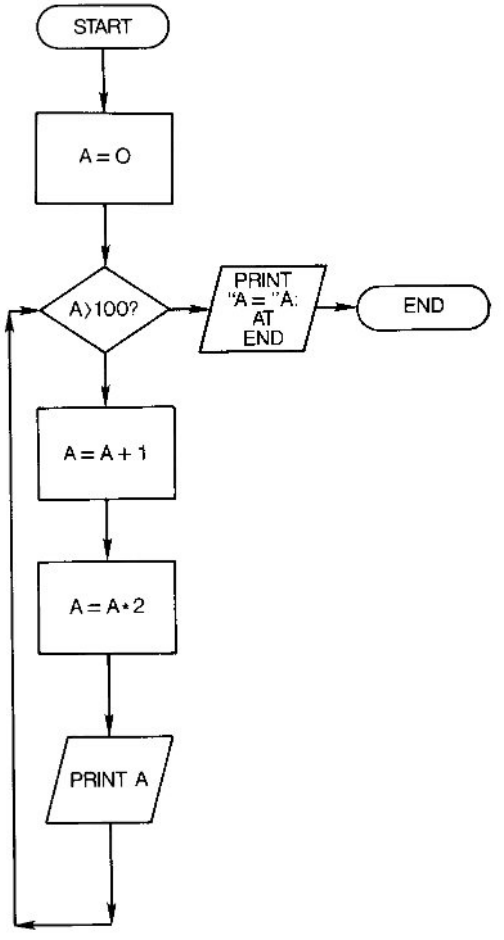
This loop continues indefinitely (until you press RUN/STOP), and is not all that useful. But combined with an additional clause, UNTIL or WHILE, the DO/LOOP becomes a powerful and versatile programming tool. The UNTIL clause specifies a condition that must be met so that execution of a program can go beyond the loop. The steps within the loop are carried out over and over UNTIL the condition in the clause is met. DO UNTIL X = 10 is an example of one such condition. Change line 10 to include the UNTIL clause:

```
10 DO UNTIL X=10:X=X+1
```

The WHILE clause is quite similar, where the loop is repeated only WHILE that condition is true.

```
10 DO WHILE X<10:X = X + 1
```

Trace what happens in the following program with each loop, calculate the loop variable, and predict how many loops will be completed before the program satisfies the UNTIL condition. Then RUN the program to see if you were right.




```
10 DO UNTIL A>l00
20 A=A+l
30A=A*2
40 PRINT A
50 LOOP
60 PRINT "A = ";A;"AT END"
```

The WHILE statement works Similarly, executing the statements within the loop only while the condition is true. Which WHILE statement would give you the same result?

```
10 DO WHILE A>l00
10 DO WHILE A<l00
10 DO WHILE A<>l00
```

Try each to see how the results relate to the UNTIL clause.

THE ELSE CLAUSE

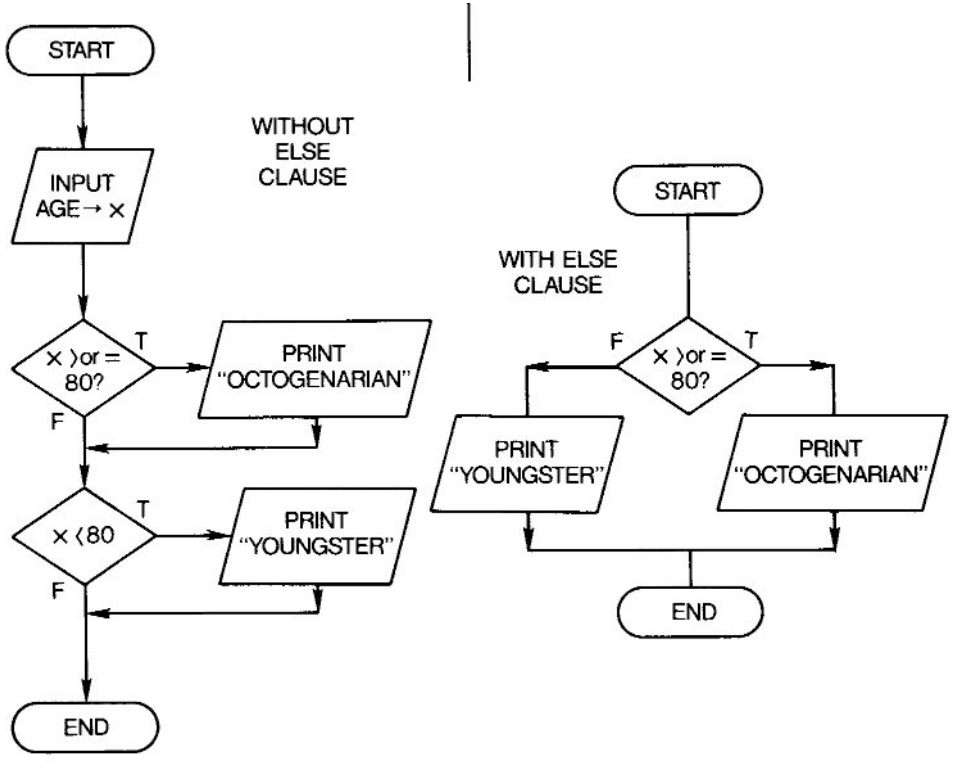
The IF/THEN statement was introduced in Introduction to Basic, Part 2 of this set. To expand on what you can do with this statement, advanced BASIC language contains a related clause, ELSE.When the condition in the IF statement is not met, rather than control passing to the next line in the program, the ELSEclause establishes an action to be carried out (as the THEN clause would be executed with a true IF statement).

```
 5 INPUT "AGE";X
10 IF X> =80 THEN PRINT
   "OCTOGENARIAN"
20 IF X<80 THEN PRINT
   "YOUNGSTER"
```

The ELSEclause can direct execution to another part of the program (to a subroutine or another line), assign a value to a variable, etc. In a line featuring an IF/THEN statement, the ELSEclause is preceded by a colon (:), as in

```
10 IF X> =80 THEN PRINT
   "OCTOGENARIAN":ELSE PRINT
   "YOUNGSTER"
```

The ELSEclause used properly can both Simplify and tighten the organization of a program. The following program utilizes ELSEclauses in a boxing simulation that scores point totals for rounds for two fighters. Here are the details:



- There are two boxers, a challenger and champion. There may be a handicap, giving one boxer an advantage over the other in scoring.
- The scoring for the fight is based on a ten-point total round scoring system. There are ten points total awarded to both fighters in each round, according to performance. If a fighter dominates a round, he gets more points (7 or 8) to his opponents lesser total (3 or 2). An even round is scored 5 points for each fighter.
- A knockout can be scored one of two ways: when a fighter gets all the points in a round or when one boxer's cumulative total *far* exceeds that of the other combatant.

Let's look at some of the individual tasks and what steps are specifically involved to set up the program.

ESTABLISH FIGHTERS

SETUP HANDICAP

SETUP SCORING SYSTEM *

EVALUATE SCORE * IF/THEN:ELSE clauses *

DISPLAY OUTCOME *

This program features seven possible outcomes:

1- The fight is a draw

2-Champion wins on points

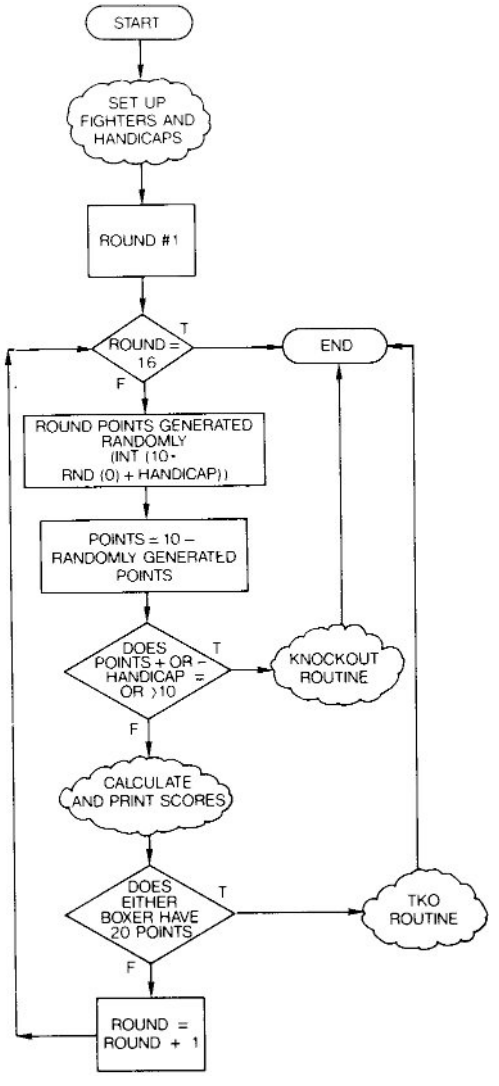
3-Challenger wins on points

4-Champion wins by knockout

5-Challenger wins by knockout

6--Champion wins by technical knockout (TKO)

7-Challenger wins by TKO



It could get pretty messy in specifying who won the fight and how, particularly with the "handicap" variable that gives one fighter the scoring advantage over the other. There are three possible responses for the handicap variable, each of which affects the scoring.

An even fight (with no handicap) does not necessitate any scoring adjustment. A knockout is scored when one fighter wins *a* round by getting all the points, which is determined by the random number and won't happen too often. But if a fighter is handicapped -1, there is no possibil ity of a

knockout unless an adjustment is made, since even if he wins all 10 points, 1 point is automatically subtracted from his score. Also, the + 1 handicap boxer has an even greater chance of a knockout since one point is always added to his score. To even things out as far as knockouts without adding too many confusing statements, we can use the ELSE statement. This program, entitled "BOXING", can be found on your program disk.

```
 5 REM****SET VARIABLES FOR
   BOXERS NAMES, HANDICAP,
   SCORING AND ROUND
10 INPUT"NAME OF CHAMPION";A$
20 INPUT"NAME OF CHALLENGER";
   B$
30 PRINT"HANDICAP (+ 1 FOR "A$"
   ADVANTAGE, -1 FOR "B$", 0 FOR
   EVEN)"
40 INPUT HA
50 PRINT"SCORING IS ON A 10-
   POINT SYSTEM"
60 PRINT,A$,B$
70 FOR HH=l TO 15
80 PRINT"ROUND ";HH
88 REM******ROUND SCORING******
90 TS= INT(10*RND(0)) + HA
100 TP = 10-TS
105 REM***JUDGE FOR KNOCKOUT***
110 IF TS-HA> = 10 OR TP + HA> = 10
    THEN 235
120 IF TS>= 10 OR TP> = 10 THEN 235
130 PRINT" "TS,TP
135 REM********TOTAL SCORE********
140 F = F + TS
150 G=G+TP
160 PRINT" CHAMPION
    CURRENTLY= ";F-G
165 REM******JUDGE FOR TKO******
170 IF F-G>20 OR G-F>20 THEN 225
175 GETKEYA$:
180 NEXT HH
185 REM******FINAL SCORING******
190 PRINT "FINAL POINT TOTALS
    ";A$; F,B$; G
195 REM*********DECISION*********
200 IF F=G THEN
    PRINT"DRAW":GOT0240
210 IF F>G THEN PRINT"THE WINNER
    IS ";A$:ELSE PRINT"THE WINNER
    IS ";B$
220 GOT0240
225 IF F>G THEN PRINT A$" WINS
    ON A TKO":ELSE PRINT B$"
    WINS ON A TKO"
230 GOTO 240
235 IF TS>TP THEN PRINT A$"
    KNOCKS OUT "B$:ELSE
    PRINTB1$" KNOCKS OUT "A1$
240 END
```

Lines 110 and 120 are checking for a possible knockout. When the IF clause in either line is true, the program jumps to a later subroutine to evaluate for the knockout. Line 170 checks for a technical knockout, comparing the totals established in lines 140 and 150. If neither outcome occurs, the fight lasts 15 rounds. After 15 rounds, the point totals are compared and a winner is declared. Lines 210, 225 and 235 use the ELSE clause to evaluate the fight and declare the winner.

Glossary
A1$-Champion's name
B1$-Challenger's name
HA-Handicap
HH-Round number
TS-Champion's score in a round
TP-Challenger's score in a round
F-Champion's cumulative total
G-Challenger's cumulative total

C128 ONLY

Extending the IF/THEN:ELSE-BEGIN/BEND

BEGIN/BEND allows you to extend THEN and ELSE clauses. Often a single program line is not enough to do all you want to do in a THEN or ELSE clause. Without resorting to directing execution elsewhere, BEGIN/BEND gives you the ability to lengthen these clauses. When you add on these statements, you can expand the THEN or ELSE-type clauses to include several program lines. BEGIN/BEND can be added to the THEN clause to let you add an unlimited number of statements. The additional statements are carried out when the IF clause is true, until the BEND statement is encountered. BEGIN/BEND can also be added to the ELSE clause for the same effect of extending the statements performed when the IF statement is false. Here's an example that counts pennies, featuring THEN BEGIN clauses in lines 40-50,60-70 and 110-140 and ELSE BEGIN in lines 140-150.

```
10 PRINT "[SHIFT] and [HOME]":
   PRINT" PENNY SAVER"
15 PRINT "":PRINT ""
20 PRINT "HOW MANY DAYS OF THE
   WEEK"
30 INPUT " DO YOU PLAN TO
   SAVE";D
40 IF 0>7 THEN BEGIN:PRINT "THERE
   ARE ONLY 7 DAYS IN A WEEK"
45 IF D = 8 THEN PRINT "DESPITE
   WHAT THE BEATLES SAY"
50 PRINT "TRY AGAIN":GOTO
   20:BEND
60 IF D<1 THEN BEGIN
65 PRINT "YOU SHOULD PUT
   SOMETHING AWAY FOR A RAINY
   DAY"
70 GOTO 5O:BEND
80 PRINT "HOW MANY PENNIES"
90 INPUT" DO YOU PLAN TO
   SAVE";M
100 PRINT ""
110 IF M>O THEN BEGIN: M1 = (M*D)
120 M2=M1*52
130 PRINT "down cursor four times
    YOUR YEARLY SAVINGS SHOULD
    BE"; M2;"PENNIES"
140 BEND: ELSEBEGIN: PRINT "AT
    THIS RATE YOU WON'T SAVE ANY
    PENNIES"
150 PRINT "AND A PENNY SAVED IS
    A PENNY EARNED":BEND
160 INPUT "down cursor four times
    DO YOU WANT TO START AGAIN
    (Y/N)";A$
170 IF A$ = "Y" THEN GOT010:ELSE
    END
```

You can see in lines 110-150, for example, how much more BEGIN/BEND lets you add to IF/THEN ELSE statements. The BEGIN/BEND statements can make certain subroutines unnecessary and make the organization more straight-forwcrd, as well as increase what you can do in a THEN or ELSE clause.

Returning to the BOXING program, we can use BEGIN/BEND to extend the IF/THEN/ELSE clauses that are used to declare the winner to make a more elaborate ring result. The following three sections use BEGIN and BEND in both THEN and ELSE clauses to give the ring announcer a little more to say.

```
210 IF F>G THEN BEGIN
212 PRINT "AFTER 15 ROUNDS, ON A
    JUDGES DECISION"
214 PRINT "THE WINNER AND STILL
    CHAMPION-";A1$:BEND
216 ELSEBEGIN:PRINT"AFTER 15
    ROUNDS, ON A JUDGES
    DECISION"
218 PRINT "THE WINNER AND NEW
    CHAMPION-";B1$:BEND

225 IF F>G THEN BEGIN
228 PRINT "ON A TECHNICAL
    KNOCKOUT IN ROUND ";HH
230 PRINT "THE WINNER AND STILL
    CHAMPION-";A 1$:BEND
232 ELSEBEGIN:PRINT"ON A
    TECHNICAL KNOCKOUT IN
    ROUND ";HH
234 PRINT "THE WINNER AND NEW
    CHAMPION-";B1$:BEND

235 IF TS>TP THEN BEGIN
236 PRINT "THE WINNER BY A
    KNOCKOUT IN ROUND ";HH
```

C128 ONLY

```
237 PRINT "STILL CHAMPION-
    ";AI$:BEND
238 ELSE BEGIN:PRINT"THE    WINNER
    BY A KNOCKOUT   IN ROUND
    ";HH
239 PRINT "THE NEW CHAMPION-
    ";BI$:BEND
```

LOAD the program QUIZI from disk. RUN it and answer the questions about delays and DO/LOOP.

* **Experiment 1.2 Completed** *

C128 ONLY

LOAD and RUN the CALORIE COUNTER program. LIST the program to see how the BEGIN/BEND commands are used to extend the IF/THEN statements. Follow the same spacing in the rest of the program and add another option, BREAD SPREADS, with the following data:

BUTTER	1 TBSP.	100
JAM	1 TBSP.	55
MARGARINE	1 TBSP.	100
PEANUT BUTTER	1 TBSP_	95

Use BEGIN/BEND to fit the new lines in the same IF/THEN clause. Finish the entry as the other food selections are finished, by telling the user to press RETURN to continue and go back to the main menu. Make sure you include the new option for BREAD SPREADS in the main menu. (Hint: The BREAD SPREAD option will be seventh on the main menu, so the IF/THEN clause would begin IF X = 7 THEN BEGIN and present the data in PRINT statements. You'll have to make a change in line 200 so X< >8 instead of 7 to compensate for adding another option to the menu.)

* Experiment 1.3 Completed *

UNIT 2:

Formatting Output

Sometimes you'll want the computer's output to appear in a certain way on the screen. For instance, when dealing with money, you might want any arithmetic calculated to be displayed in dollars and cents. If numbers are being divided, you may wish to eliminate extraneous decimal places past the first one or two. Your data might be better understood if displayed in scientific notation. You may need to limit inputs to numbers containing fewer than five digits. You might want to have words or letters appear in a particular column format, centered on the screen. You can do all this and more with the PRINT USING command.

There is a related command, PUDEF,which lets you define characters being utilized in the PRINT USING statement, for example, inserting a special symbol which is displayed in the output instead of a comma. PUDEF,when used in conjunction with PRINT USING, gives you the capability to select the characters used in the data format in addition to setting the parameters of the format itself, substituting different symbols for frequently-used characters such as commas, periods, dollar signs and blank spaces.

THE PRINT USING COMMAND

PRINT USING is always followed by a statement specifying the format for the output, which is enclosed in quote marks. This statement is a series of hash marks (#), with or without additional punctuation which further defines the format. The hash marks set the number of spaces for letters or numbers; the format is followed by a semi-colon (;) and a variable(s) or data. A statement reading

```
10 PRINT USING "########"; A
```

would set up a data format which accepts numbers of up to eight places. The number contained in variable A would be displayed or printed in the specified format. The display is right-justified format, meaning the last digit is printed in the eighth space, and any blank digits appear on the left. If the number displayed is only three digits, the first five digits are blank. If the number is nine digits, the computer returns a line of eight asterisks (as many as there are hash marks in the format statement) because the number does not conform to the format. This is called an OVERFLOW. Any decimal places in the number would be dropped, since the format does not specify decimal input.

Type in line 10 (the PRINT USING statement shown above) and add line 5 (5 INPUT A), which lets you INPUT different values for the variable A, to be displayed in the format specified in line 10. This gives you a two-line program that lets you try different numbers to see how they appear in the PRINT USING format in line 10. Here are some values to try:

FORMAT: VALUE	######## RESULT	DIAGRAMS	EXPLANATION
3.59	4	3.59	—rounds up
–.1	–0	–.1	—rounds to zero, keeps minus sign
12.354	12	1 2.354	—rounds down
1233	1233		—right justifies
13234.0	13234	1 3 2 3 4.0	—drops decimal place
123456789	********	1 2 3 4 5 6 7 8 9	—overflow

What if you want decimals? Just change line 10 in your program to include a decimal point as part of your format, like this:

```
10 PRINT USING "######.##"; A
```

FORMAT: VALUE	######.## RESULT	DIAGRAMS	EXPLANATION
3.59	3.59	3 . 5 9	—prints number with decimals
–.1	–0.10	– . 1	—adds leading zero, one decimal place
12.356	12.36	1 2 3 5 6	—rounds up last place
1233	1233.00	1 2 3 3	—adds two decimal places
13234.0	13234.00	1 3 2 3 4 . 0	—adds one decimal place
123456789	********	1 2 3 4 5 6 7 8 9	—overflow

Commas can be included in your displayed data if you include a comma in the format line, changing line 10 to

```
10 PRINT USING "###,###";A
```

Try the same values you've been using and see what comes out.

With values of three places or less, the comma is not displayed, but with four digits or more, it appears in front of the third number from the right.

There are other items you can include in the format. You can have a dollar sign ($) appear at the left-most column (as in $####.##) or floating (appearing next to the left-most number displayed), in a format such as this: #$####.## The numbers displayed below show the difference between the left-column dollar sign and the floating dollar format.

Left-column dollar	$ 1.25
Floating dollar	$1.25

You can also include plus or minus signs in your data formats, either before or after the hash marks, like so:

```
+####
-####
####+
####-
```

With a plus sign in the format, positive numbers include a "+" while negative numbers are displayed with a "-". If the sign placed is after the format, it appears following the number when displayed. If the sign is placed before the format, the display features a sign in front of the number. The same principle holds for negative numbers and a minus sign in the data format. Negative numbers get a "-", while positive numbers do not receive a sign. These are appropriate for uses like ledgers or checkbook balancing programs.

You can also arrange your data in scientific notation format with PRINT USING. This involves setting a format line and following it with four carets (up arrows). The number is printed in scientific notation according to your format line. For example, the number 1234 would be displayed according to the format specified:

```
10 PRINT USING "SCIENTIFIC
   NOTATION #.###    ";1234
20 PRINT USING "SCIENTIFIC
   NOTATION ####     ";1234
30 PRINT USING "SCIENTIFIC
   NOTATION ##.####  ";1234
```

Your format line can dictate a completely different scientific notation display for the same value. Notice that you can include regular printing inside the quote marks along with the format with no ill effects on the print using line.

(a) Write a program featuring the PRINT USING command to perform a loan calculation that figures interest payments on a loan of between $100 and $1000. The program should calculate
- Interest on the amount of the loan
- Total money owed
- Monthly payments

The user should supply the loan amounts (A) and percentage interest rates (P) through INPUT statements, and the program should use these values to calculate interest on the amount of the loan (A*(P/100) = I), total money owed (A + I) and monthly payments ((A + 1)112).

These values should be placed in dollar and cents format, with floating dollar signs. The PRINT USING statement should reflect the limits on the amount of the loan ($100-$1000).

C128 ONLY

(b) LOAD and RUN the program BANK STATEMENT. After you RUN it, LIST it and take special note of the print format statements and the use of BEGIN/BEND. Change the PRINT USING statements to change the size of acceptable transactions-into the millions and billions, or restrict it to less than $10 transactions.

* **Experiment 2.1 Completed** *

PRINT USING WITH STRINGS

You can use this command to format text strings as well. It is useful for accepting data of a specific length or establishing a column format. You can center text strings or right-margin justify words (have the last letter of each aligned in the same column). The command works with text strings and string variables in the same fashion as it does with numbers. The string is formatted or positioned according to the format specified in the PRINT USING line.

For example, with three strings-CAR, TRUCK and MOTORCYCLE-we can see some of the different ways the format affects text strings. The hash mark (#) is again used to set the number of spaces in the format line. Only two other symbols may be used in the format line for text strings, the equal sign (=) and the greater than sign (». These symbols also count as a space for a letter in the format line like the # sign, but they do a bit more. The text is typically left-justified, unless one of these two symbols appears in the format line. With the equal sign (=), text is centered in the field, and with the greater than sign (», text strings are right-justified.

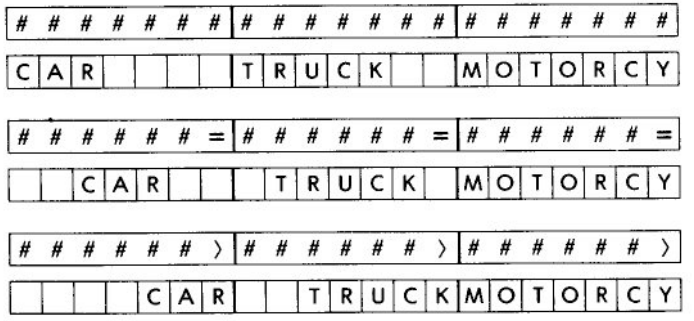
If the string takes up more space than is available in the format, it is shortened (or truncated), with the overflow letters being dropped from the right side. For example:

```
10 PRINT USING"#####";"CAR",
   "TRUCK", "MOTORCYCLE"
```

appears as

```
CAR TRUCKMOTOR
```

Use of centering or right justification affects how strings are shortened or positioned. If you use a format line with seven hash marks (#######) and two other formats replacing a # with a > or = sign at the end of the format, you'll get different positioning. The diagram shows why:



(a) Set up a brief program that will allow you to input different PRINT USING formats in which to display data. Try to predict how the strings CAR, TRUCK and MOTORCYCLE will appear in each of the following formats, and then type in the statements to see if you were right.

```
#
############ =
#####>
###,###
THE ##########
```

(b) LOAD and RUN the program MAILING LISTor MAILING LIST/4.

1-Add PRINT USING statements so each address is centered rather than left-justified.

2-Use a PRINT USING statement to format the display of zip codes to be five numbers.

* Experiment 2.2 Completed *

CHANGING PRINT USING CHARACTERS

PUDEFlets you redefine characters in the PRINT USING statement, such as blank spaces, commas, decimal points and dollar signs. With PUDEF,you can change these characters to appear on the screen or printed out as anything from the keyboard: letters, punctuation, graphic symbols, cursor movements, even color changes.

The command PUDEFis always used in conjunction with the PRINT USING statement. PUDEFis followed by double quote marks enclosing up to four spaces. Each space corresponds to a character, as follows:

```
10 PUDEF"  ,. $"
           1234
```

To redefine any of these characters, include the command with the symbol you want to use in the place (space 1-4) corresponding to the character you want redefined. So if you wanted commas to appear as slashes, you would use this line:

```
100 PUDEF" /"
```

An example where PUDEFis most appropriate is a program converting American money into English pounds. The following short program converts American money into British currency by multiplying the American value by the exchange rate. The only thing missing is changing the dollar sign to a pound sign, and that's where PUDEFcomes in.

```
10 INPUT "AMOUNT IN
   AMERICAN MONEY";A
20 INPUT "EXCHANGE RATE
   (ONE DOLLAR EQUALS X
   POUNDS)";E
30B=A*E
40 PRINT USING"THAT AMOUNT IN
   BRITISH MONEY IS #$#####.##";B
```

Only a single line is needed to make this program correct:

```
35 PUDEF " ,.£"
```

Note: You can only redefine the dollar sign ($) with PUDEFwhen it is a floating dollar sign, that is, only when it appears with a hash mark in front of it. Otherwise, it will appear as a normal dollar sign.

LOAD and RUN the program QUIZ2, and answer the questions based on what you learned in this section.

* Experiment 2.3 Completed *

UNIT 3:

Finding Program Errors

You may have had the experience of carefully typing a new program, and when you run it, the computer responds with an error message. You may have forgotten the dollar sign on a string variable, or perhaps you typed a GOTO statement with no reference line number. These mistakes are common, but relatively easy to locate in short programs. In Unit 8 of Introduction to BASIC Part 1 you learned to trace short programs by hand to evaluate problems. Complicated programs require more sophisticated problem-solving skills. Finding and fixing program errors is called debugging. This chapter describes some of the debugging techniques available to you: the HELPfunction, computer tracing with TRON and TROFF, and TRAP/RESUMEsubroutines.

HELP

Your computer displays error messages when a command can't be executed. By now you are probably adept at pinpointing common problems like syntax errors. Yet even syntax errors can sometimes be difficult to spot. When you get an error message, you can use the HELPfunction to locate incorrect program information. When you press the HELPkey or type HELPand press RETURN, your computer displays the program line which caused the error, with the incorrect section highlighted (flashing on the Plus/4 and C16).

To see how the HELPcommand works, type and RUN this one-line program:

```
10 PRINT 1;2:3;4  (Notice the: is a
                   deliberate error.)
```

The program will fail after displaying 1 and 2. Press the HELPkey or type HELPand press RETURN. You will see:

```
10 ?1 ;2:[3;4]The  3;4 will be
                   highlighted, to show the
                   error is somewhere near
                   the 3.
```

The HELPfacility isn't always so clear. Try typing

```
10?1+2+3**4
```

RUN it, then use the HELPcommand. As you can see, the syntax error was so obvious that HELPwasn't very helpful. In many cases, however, the error will not be so obvious and HELPcan be used to point them out.

Here's another example. It contains several errors, but in a complicated command a missing quote or dollar sign can be hard to find. Look over the program and see if you notice the errors. Then type the program exactly as is:

```
10 REM CONTAINS ERRORS
20 A$ = "BET ":B$ = "YOU ";C$ =
   "CAN'T ";D$ "SPOT ":E$ "THE
   :F$ = "ERRORS."
30 ?A$;B$;C$;D$;E$;F$
```

Now use HEIP. You'll get a series of syntax and other errors. Correct them one at a time, running the program each time, then use the HELPfunction to locate the next error. Now you should have a better idea of how HELPcan be used.

TRACING: WHEN HELP DOESN'T HELP

Used as a first step in the debugging process, HELPis an excellent tool. If the HELP function doesn't supply enough information for you to debug a program, however, other debugging options are available. You can trace the program the way you did in UNIT 8 of Introduction to BASIC Part I, and evaluate each line number to determine if the program does what you intended. Or, you can let the computer trace the program automatically.

When you trace a program by hand, you look at each command and pretend to be the computer. After you simulate the command's execution, you switch on your human intelligence and ask "Is this what I expected?" If the answer is yes, continue the trace. If the answer is no, you have a better idea of how to correct the program.

When the computer does the tracing for you, it is still necessary to evaluate the program step by step, but the computer calculates any results. This leaves you free to concentrate on the structure of a program, to ensure it meets its purpose.

Your computer has a set of commands used to trace programs. Using TRON (TRace ON) and TROFF (TRace OFF) the computer will execute the program and display any output, but at the same time will display a list of line numbers in the order they are obeyed.

Type the following program.

```
10 INPUT "GIVE A NUMBER TO BE
   MANIPULATED";N
20 PRINT USING "THE SQUARE ROOT
   IS ####.##";SQR(N)
30 INPUT 'WANT TO GIVE ANOTHER
   NUMBER";A$
40 IF A$ = "Y" THEN INPUT "NEW
   NUMBER";N:GOT020
50 IF A$< > "N" THEN PRINT "NO
   FAIR, YOU HAVE TO ANSWER YES
   (Y) OR NO (N)":GOTO 30
60 PRINT "OKAY, NO MORE MATH"
70 END
```

RUN the program a couple of times to familiarize yourself with it.

Type TRON, then run the program again. line numbers appear on the screen in brackets to show the order in which commands were obeyed. Output was generated by lines 10,20,30 then by 40 ,50 or 60 depending on the answer chosen. Notice some line numbers appear more than once. This is because the TRON function prints a line number each time a function is executed, even if more than one function is contained on a line. For example, in line 40, the computer executes theIF/THEN, and an INPUT statement before it follows the GOTO command. TRON makes the computer output [40] [40] [20] to show that this is the order in which commands were executed. Type TROFFto end tracing.

For an example of tracing a more complex program, load the program TRACEEXfrom your program disk. Plus/4 and C16 users should list line 10 before running the program. Change the color values to COLOR 4, 15,3: COLOR 0,4,3: COLOR 1,1. Also list line 1000 and change the SLEERcommand to: FOR J = 1 to 500: NEXT J. Run the program, then type TRON and run the program again.

Now that you have seen how TRON and TROFFwork, we can discover how tracing on the computer can help locate errors.

Sometimes, especially when your program contains loops, the screen becomes filled with rows and rows of bracketed line numbers. You will have to consider each line number, one-by-one, until you determine where the error is.

If you are sure that most of a program works, trace only a small section of the program. This can be done by putting numbered TRON and TROFFcommands into the program itself like this:

```
10-
20-

59 TRON
110-
120-

189 TROFF
190-
200-
```

When you have debugged the problem area, you can easily take the TRON and TROFFcommands out again.

Tracing is particularly helpful when the error in the program is a logic error. Your program may run, and produce output, but you notice that the results are not what you expected. It is likely that you did not accurately translate your algorithm into computer code. Perhaps your program exits from a loop too soon, or jumps to a subroutine at an inappropriate time. Tracing can be used to methodically analyze the program's execution so logic errors come to light.

Here is a program designed to report to the user the value of an average home in a selection of American cities over a period of years. The city names, dates, and values are contained in arrays. You could arrange the data like this:

	Atlanta	Denver	Los Angeles	New York	Philadelphia	Washington
1950	14000	15000	27000	24000	23000	25000
1960	17000	18000	32000	25000	26000	27000
1970	27000	30000	40000	39000	41000	43000
1980	45000	50000	56000	60000	65000	67000

The program is saved on your disk as "REAL ESTATE". LOAD and LIST the program.

```
10 DATA ATLANTA, DENVER,LOS ANGELES,NEW YORK, PHILADELPHIA,WASHINGTON
20 DATA 1950,1960,1970,1980
30 DATA 14000,15000,27000,24000, 23000,25000
40 DATA 17000,18000,32000,25000, 26000,27000
50 DATA 27000,30000,45000,39000, 41000,43000
60 DATA 45000,50000,56000,60000, 65000 ,67000
70 DIM C$(6),Y$(4),P$(6,4)
80 FOR J = 1 TO 6: READ C$(J): NEXT J
90 FOR K = 1 TO 4: READ Y$(K): NEXT K
100 FOR J = 1 TO 6
110 FOR K = 1 TO 4
120 READ P$(K,J)
130 NEXT J
140 NEXT K
150 PRINT "THIS PROGRAM OUTPUTS THE VALUE OF THE"
160 PRINT "AVERAGE FAMILY HOME IN THE"
170 PRINT "FOLLOWING AMERICAN CITIES:"
180 PRINT
190 PRINT "ATLANTA, DENVER, LOS ANGELES, NEW YORK,"
195 PRINT "PHILADELPHIA AND WASHINGTON"
200 PRINT
210 PRINT "IN THE YEARS:"
220 PRINT "1950,1960,1970,1980"
230 PRINT "WHAT CITY";P$
240 FOR J = 1 TO 6
250 IF LEFT$(P$,I) = LEFT$(C$(J), 1) THEN GOTO 270
260 NEXT J
270 INPUT"WHAT YEAR";D$
280 FOR K = 1 TO 4
290 IF D$ = Y$(K) THEN GOTO 310
300 NEXT K
310 PRINT "THE VALUE OF THE AVERAGE HOME IN ";C$(J)" IN THE YEAR ";D$" WAS";P$(J,K)
320 INPUT "ANOTHER VALUE (Y/N)";A$
330 IF A$ = "N" THEN 350
335 IF A$ = "Y" THEN GOTO 230
340 GOTO 320
350 PRINT "END OF HOME-COST COMPARISON"
360 END
```

RUN the program, and it returns a bad subscript error in 120. If you use the HELP function, the computer prints the entire line highlighted. Not very useful. So, let's use TRON and TROFF. Note: if you spotted the error, don't correct it just yet. Wait until you've seen how you might use TRON and TROFF to find the error.

Type TRON and press RETURN and then RUN the program. Your screen is swamped by bracketed numbers, and you're probably thinking you were better off muddling through and finding the error on your own. But wait until you've given TRON its best shot.

Maybe it would help to trace only a small section of the program. The error was in line 120, a line that reads data into a two-dimensional array. So type TROFF to end the trace, and add these lines to your program:

```
95 TRON
145 TROFF
```

This is the section of the program involved with setting up the two-dimensional array.

RUN the program now. While quite a few line numbers appear before an error stops the program's execution, the trace will be much more manageable than before. This is what appears on your screen:

```
[100][110][120][130][110][120][130]
[110][120][130][110][120][130][110]
[120]
```

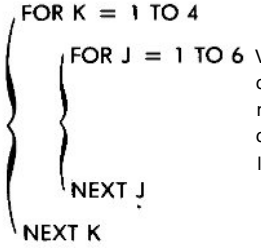
The computer executes line 100, setting up a FOR/NEXT loop for the variable J. Then it sets up, in line 110, a FOR/NEXT loop for the variable K. It reads the first value of P (line 120) from DATA line 40 and sets P(I, 1) to 14000.

The next line executed is line 130 which reads NEXT J. This command loops back to line 110, and the value of P(2,1) is read from DATA line 40 as 15000.

Think about this for a minute. You see from the table that P(2, 1) should be the value of a home in Atlanta in 1960. But we have just read P(2, 1) from our DATA line as 15000-the value of a home in Denver in 1950. Somehow the DATA is reading down the table instead of across. This should be a clue to the problem.

Look at the trace once again. Notice that the loop for K is executed four times, then J begins to increment. But this process is opposite from what should happen (according to the way we have arranged our data) and it becomes apparent that the FOR/NEXT loops in lines 100-140 are nested improperly. We must rearrange the loops so that the READ statements read the data lines in a manner that sets up a two-dimensional array similar to the table above.

This is the way the loops must be nested:



Values of P(J,K) are read from the data with increasing values of J while K remains constant at 1. Then K is incremented and new values of P(J,K) are read with K = 2. This process continues until K reaches 4 , then the program drops through to line 150

In addition, the READ statement must be: READ P$(J,K) in order to set up the data correctly.

TraCing, as you have seen, has its limitations as a debugging tool. The important thing is that you be patient and work methodically. If you try to anticipate errors or assume a section of a program works, you'll run into trouble. There are times, however, when you could trace for hours and still not find a solution. Suppose you have misunderstood some of the rules of BASIC. Your program could be constructed logically as you see it, but if it violates BASIC structure it won't work. And tracing won't help, because you'll never see the logic error.

Don't be discouraged if you have trouble locating program errors at first, even with the TRON and TROFF mechanism.

EXPERIMENT 3·1

Modify the program above so it stores the values of a coin or stamp collection at 10-year intervals. The table of data might look like this:

	Penny	Nickel	Dime	Quarter	Half	Dollar
1940	1	5	10	25	50	100
1950	2	6	20	27	53	105
1960	1	6	30	25	60	106
1970	3	7	35	31	65	107
1980	2	9	32	31	66	110

Add a section of code that calculates and outputs the total value of the collection in any given year.

Use TRON and TROFF to trace the program if you run into trouble.

* Experiment 3.1 Completed *

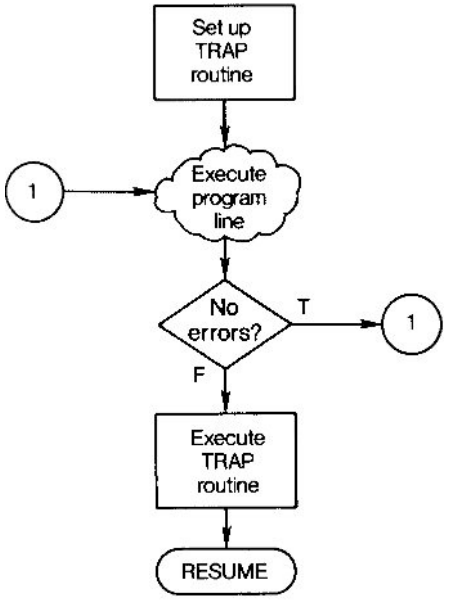
TRAP AND RESUME SUBROUTINES

TRON and TROFF are useful commands when you know an error has occurred. You can trace the program step by step until you locate the problem. Sometimes, however, it is helpful to anticipate errors before they occur, so the program can be set up to deal with errors and perhaps continue to execute.

Your computer has a built-in capability to check for errors. It works by jumping to a subroutine when an error occurs. The subroutine can handle the error in several ways. It might request more appropriate input, or print an error message. Then another statement tells the computer to continue execution of the program.

The command that sets up the error-checking subroutine is TRAP, and RESUME is the command used to switch control back to the main part of the program after the subroutine has served its purpose.

Here is a rough flow chart of a program with a TRAP subroutine:



The TRAP command should be on a line at the top of a program. The TRAP command references a line number to skip to, much like a GOTO statment. The reference line number is the first line of a subroutine that deals with the error.

TRAP routines can access several built-in variables to keep track of errors that occur and the corresponding error messages.

- ER stores the error number. Each type of error has a numerical label
- ERR$ is the error message corresponding to a particular ER value
- EL keeps track of the program line number containing the error

ERR$ only has meaning in conjuntion with the corresponding ER value. To see what type of error occurred, place statements such as PRINT EL, ERR$(ER) within the TRAP subroutine. The computer will print the line number the error occurred in and the appropriate error message.

Here is a TRAP subroutine built into a program that records and displays errors as they occur. The RESUME statement tells the computer to begin execution again. Obviously, if a serious error has occurred, continuation of the program may not be possible.

```
10 TRAP 500
20 PRINT "DEBUGGING PROGRAMS"
30 FOR X = = 1 TO 4 STEP 2
40 PRINT "HELP!"
50 PRINT
60 NEXT X
70 PRINT "OH NO!"
80 NEXT Y
500 REM TRAP SUBROUTINE
510 PRINT "ERROR IN LINE ";EL
520 PRINT "ERROR TYPE: ";
    ERR$(ER)
530 RESUME NEXT
540 END
```

RUN this program. Several errors occur, but the program can continue to execute. The error messages are listed, and you should be able to locate the syntax error in line 30, and the NEXT without FOR in line 80. But RUN the program again after you fix these errors, and the output is unusual indeed, mentioning an error in line 65535! The line number 65535 then generates an illegal quantity error in line 520.

Why does the computer execute the TRAP subroutine if no error is found? Look at the program again. Even though line 500 labels the subroutine, the program reaches

lines 500-540 in its normal execution. Add 80 END to the program (you have probably already eliminated the old line 80) and the program should run smoothly.

USING TRAP TO FILTER SPECIFIC ERRORS

While TRAP routines such as this are helpful, TRAP is more commonly used to handle specific errors. For example, perhaps you would like to write a program that cannot be stopped by the RUN/STOP key. Note: RUN/STOP causes a break in the program, and is considered an error.

Type NEW, then enter this program:

```
10 TRAP 1000
20 PRINT "AGAIN"
30 GOTO 20
1000 TRAP 1000:RESUME
```

If you run this program, you will notice that it will print "AGAIN" ad infinitum, and RUN/STOP will not break the program. The only way to stop the program is to hit RUN/STOP and RESTORE or reset the computer. Why does this happen? Let's examine the program.

When an error (RUN/STOP is pressed) occurs, the TRAP routine is put into effect. While the TRAP routine is being executed, the TRAP function is disabled. That is, when program control is shifted to line 1000, the program cannot trap more errors. If a user hit RUN/STOP while the computer was executing the TRAP routine in line 1000, the program would break.

In our program, however, the TRAP routine itself puts the TRAP function into effect, so the TRAP function is never disabled. Line 1000 not only keeps the TRAP routine in operation, but resumes execution of the program where it left off when the error occurred.

Actually, in our simple example the RESUME itself was sufficient to put the TRAP routine back into operation. Change line 1000 to:

```
1000 TRAP 1000:?"YOU CAN'T STOP
     ME":RESUME
```

Now the TRAP 1000 clause is necessary. While the computer was printing "YOU CAN'T STOP ME", RUN/STOP would ordinarily be effective. But our program cycles the TRAP routine to itself, so the computer is continuously looking for errors. The user could never stop the program with RUN/STOP.

This should give you an idea of the many ways TRAP routines can be used. You can set up TRAP routines to find specific errors (error messages are listed in your computer's user manual) and react in a way appropriate for that error. TRAP is especially useful when programming graphics. This will be discussed further in the graphic section.

C128 ONLY

Load the program "DEBUG EXERCISE" from your program disk. It contains a number of programs for you to debug.

Correct the programs on paper, then compare your listings to the answer provided in the exercise.

* **Experiment** 3.2 **Completed** *

UNIT 4:

ScreeD Editing

Creating useful and entertaining programs can be exciting, but getting the programs from paper into the computer's memory can involve a lot of effort. Despite the INST/DEL key, you will surely spend much of your time at the keyboard correcting errors. Even the most careful programmer will need to change a variable or correct a syntax error, so it would help to know the shortcuts for getting around the screen. These shortcuts are called screen editing functions. This unit describes and illustrates screen editing functions, highlighting the use of screen windows.

Screen windows allow you to define a portion of the screen as your workspace. In this area you can create a program or view program output, while the rest of the screen serves a different purpose. All the screen editing functions work within windows as well as on the full screen. Screen windows can be used for decorative purposes or to aid in debugging by allowing you to see a program and its output at the same time.

SCREEN EDITING FUNCTIONS

Your computer has a series of functions designed to help change the text of your program or whatever else is on screen at any given moment. All of these functions are activitated by pressing and releasing th ESCape key, then pressing another key. The ESCape key functions are listed below.

Key	Function on C128	Function on Plus/4 and C16
ESC C	Cancel quote and insert mode	Cancel automatic insert mode
ESC Q	Erase to end of current line	Erase everything to end of current line
ESC P	Erase to start of current line	Erase everything to cursor on current line
ESC @	Clear to end of screen	
ESC J	Move to start of current line	Move to beginning of current line
ESC K	Move to end of current line	Move to end of current line
ESC A	Enable auto-insert mode	Automatic insert mode
ESC O		Cancel quote, reverse and flash modes
ESC 0	Delete current line	Delete current line
ESC I	Insert line	Insert a line
ESC Y	Set default tab stop (8 spaces)	
ESC Z	Clear all tab stops	
ESC L	Enable scrolling	Turn on scrolling
ESC M	Disable scrolling	Turn off scrolling
ESC V	Scroll up	Scroll screen up
ESC W	Scroll down	Scroll screen down
ESC G	Enable bell (by control G)	
ESC H	Disable bell	
ESC E	Set cursor to non-flashing mode	
ESC F	Set cursor to flashing mode	
ESC B	Set bottom of screen window	Set bottom right corner of screen window
ESC T	Set top of screen window	Set top left corner of screen window
ESC X	Swap 40/80 columns display	Cancel the escape function
ESC U*	Change to underlined cursor	
ESC S*	Change to block cursor	
ESC R*	Set screen to reverse video	Reduce screen display
ESC N*	Return screen to non-reverse state	Return to normal screen display size.

*Only in 80-column mode

Load the program EDITQUIZ or EDITQUIZ/4 from your program disk. Run the program so yOUare familiar with the way it works. STOP the program by pressing "N" at the final prompt, or use RUN/STOP RESTORE.Then LISTthe first part of the program (lines 10-170).

Suppose that in reviewing the program yOUdecide there should be a better introduction to the quiz:

- Use CLR/HOME to home the cursor. Then move the cursor to the first character of line 20.
- Press ESC(escape) and then K. Do not hold down ESCwhile you press the K. The cursor moves to the end of line 20.
- Move the cursor back to the quote, then press ESCand A. This places the screen in automatic insert mode.
- Add an explanatory sentence about the quiz, such as "The topic is science." The words you add do not eliminate characters already in the line.
- When your sentence is complete, press ESCC to turn off automatic insert mode.
- Press RETURNto "set" the insert.
- Move the cursor to line 170. We'll change the message in this line as well. Move the cursor to the space between BYEand THEN. Press ESCQ to erase everything from the cursor to the end of the line. Finish the message with: FOR NOW." Press RETURNto enter the new line.

C128 ONLY

Let's see if there are any ways this program can be streamlined. Look at lines 80 and 90. There should be a way to use IF/THEN/ELSE with BEGIN/BEND to handle this condition. We'll add some lines of space so we don't write on top of existing commands:

- Move the cursor to the beginning of line 80 and press ESC I. A blank line appears above line 80.
- Move the cursor to the first character in line 100 and press ESC I.

Now the lines you want to work with are surrounded by a little space. You could, of course, work at the bottom on the program but it is sometimes easier to type new program lines in the position you want them to appear.

Type:

```
75 IF A$ = B$ THEN
    BEGIN:J = J + 1:GOSUB 1000:
    BEND:ELSE GOSUB 2000
```

Delete lines 80 and 90 in the usual manner:

- Move to the bottom of the program and type 80 followed by RETURN, 90 then RETURN. You may notice from the table of ESCape codes that ESC D deletes the current line, but this function does not erase the program line from memory.

Now would be a good time to run the program again to be sure it still works. You can practice using ESC functions by making other changes. Perhaps you want to add more questions (how will this affect the rest of the program?) or change the way the computer reacts to right and wrong answers. There are many ways the program can be improved in appearance and structure. See if you can discover how the other ESC functions listed work. Not all functions work in both 40 and 80 columns on the C128, and some functions work differently on the Plus/4 and C16. It won't take you long to appreciate how helpful and time saving the ESC functions are.

EXPERIMENT 4·1

C128 ONLY

Modify the right answer subroutine (starting at 1000) from the quiz program in the previous section to include a bell reward.

(Hint: the CHR$ values of ESC G and ESC H can be used, in conjunction with Control-G, to create a bell-like sound.)

* **Experiment 4.1 Completed** *

SCREEN WINDOWS

Up to this point in the course we have made use of the entire screen as we programmed. Sometimes it can be to your advantage to limit the part of the screen you can use. Perhaps you would like to look at the program and the output at the same time-to help in debugging. Or you have created a program that outputs sections of data you would like to compare, but they scroll off the screen before you can get a good look. Screen windows can handle these problems.

For a demonstration of screen windows, (on the C128 only) LOAD and RUN the program WINDOW DEMO from your program disk. Notice that several windows can exist at once. Windows can overlap or occur one within the other, but only one window can be active at any time.

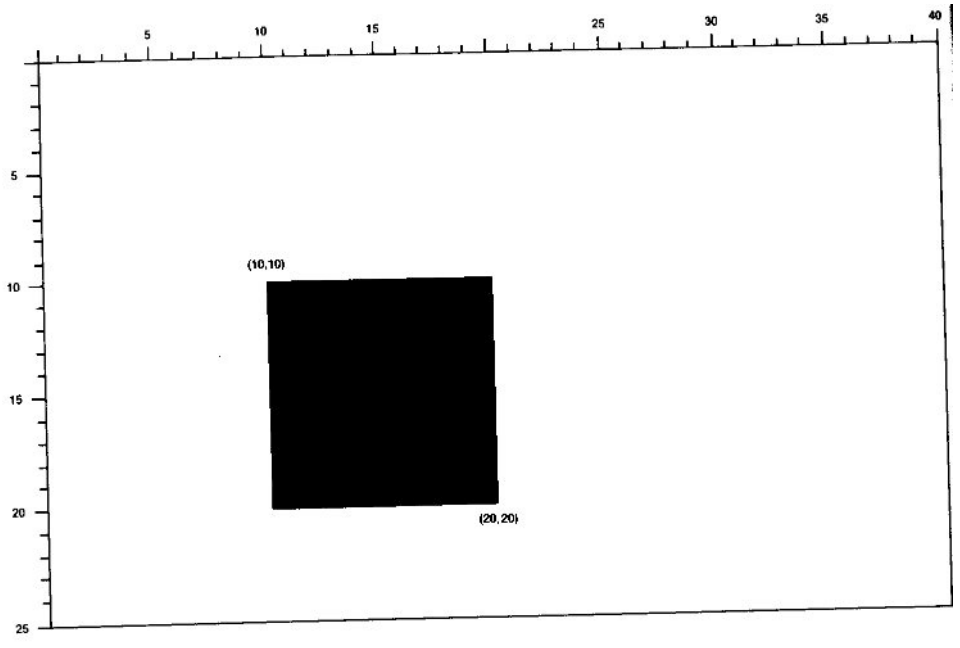
If you create a window in direct mode, everything you type, (commands, listings of programs, etc.) as well as program output, from that point on appears within the window's boundaries. The screen outside the window area is not affected. This is useful in comparing program listings with output. You run a program, then when it is finished you create a screen window in an area of screen not taken up by program output, and use the window to view the program listing.

Using windows within programs makes it easier to organize your output into a structured format. Windows can be used to increase the clarity of program output, add creativity ta quiz or game programs, or simply add to the aesthetic appeal of the program.

CREATING SCREEN WINDOWS

There are two ways to create windows on the Commodore 128, but only one way on the C16 and Plus/4. Both methods require setting the coordinates of two opposite corners of the window. Once you describe two opposing corners, the other dimensions of the window are also defined. Both methods are described below.

You must first decide where you want the window to appear. Think of the screen as a grid of 25 horizontal rows and 40 vertical columns. You may find it helpful to use graph paper. Choose a location where a row and column intersect as the upper left corner of the window, and another intersection as the lower right corner.



When the two corners are chosen,you can use the WINDOW command or ESCape commands to create the window.

The ESCape commands are the only way a window can be created on the Plus/4 or C16. Move the cursor to the point chosen as the upper left corner. Press ESC (Escape key) and then T. Think of this as setting the "top" corner.

Now move the cursor to the lower right corner. Press ESC and then B to set the "bottom" corner. The window is now set.

List the WINDOW DEMO program (or any other program), which should still be in your computer's memory. The output will scroll in the window only. Notice that when the listing reaches the edge of the window, it continues immediately on the next line. This effect is called "wrapping."

You can work within the window to create programs, using the same screen editing functions available with the full screen.

THE WINDOW COMMAND

The WINDOW command allows you to type the coordinates of two corners of the window.

Here is an example of a typical WINDOW command:

```
WINDOW10, 10,20,20, 1
```

WINDOW has five parameters:

First: top left row coordinate (in the range 1-25)

Second: top left column coordinate (in the range 1-40)

Third: bottom right row coordinate (in the range 1-25)

Fourth: bottom right column coordinate (in the range 1-40)

Fifth: clear option; 0 = off, 1 = on

If the clear option is not chosen, any text that was in the area of the window will still be there when the window is created.

Type the command noted above. A window is now set in direct mode. List the WINDOW DEMO program. It should behave exactly as it did when created with ESC commands.

To cancel a window hit CLR/HOME twice. Hitting CLR/HOME only once moves the cursor to the upper left corner of the current window.

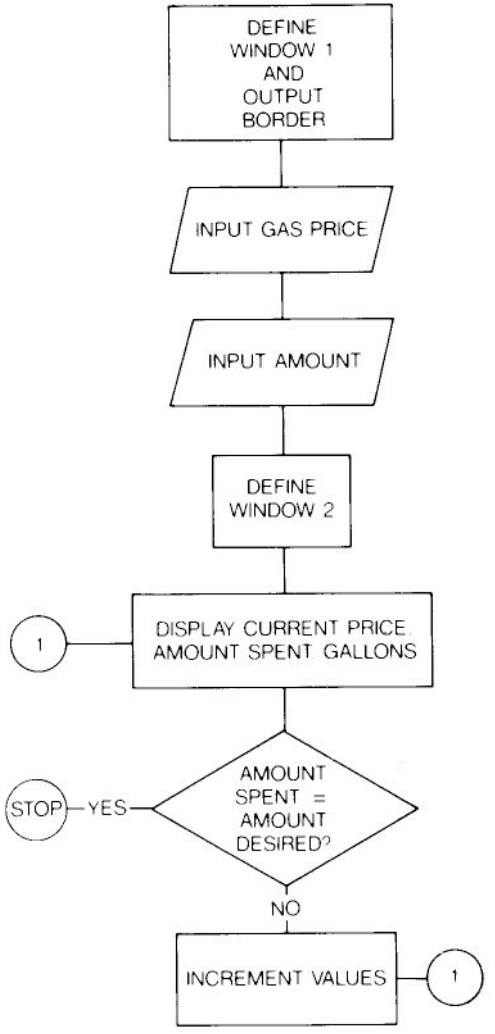
The WINDOW command can be used within programs. NEW the memory, then load the program WINDOW DEMO 2 from your disk. RUN the program, which creates two windows and directs output to each window alternately. List the program, and you'll see that lines 30 to 130 outline the two windows with' symbols, to demonstrate the windows' dimensions. Actual output is created by lines 180 to 220 in the first window and by lines 240 to 280 in the second window.

Notice that the windows must be defined each time they are used. Even though line 30 sets up a window, once line 80 sets up a second window the first window is no longer available. So line 170 recreates the window line 30 created, then line 230 recreates the window originally created by line 80.

C128 ONLY

Let's try to create a program that uses two windows. One example would be for a user to input the current price of gas, and the amount he would like to spend in one window, and have a simulated gas pump in another window show the amount and the current cost.

First, a flow chart:



You see from the flow chart that the pricing window can be fairly straightforward. For appearance, let's outline the window in a band of reversed spaces. A subroutine can be used to print the border, because a window such as this might come in handy later. Once the border is created, you should redefine the window so that the colored

C128 ONLY

border is excluded. Otherwise, the print statements for inside the window will need to include cursor movements to compensate for the border.

Inside the window, input statements establish the variables P (price per gallon) and M (amount the user wants to spend), then the pricing window is no longer necessary. For appearances sake, let's allow the user to input the first letter of the type of gas he wants (Economy, Regular, Super) instead of the price per gallon. You should now be able to construct this code. LOAD the program GASPUMP from the disk. list lines 10-100, and lines 1000-1060:

- line 10 defines border, screen and cursor color. You could choose any other colors you prefer.
- line 20 defines the window, and line 30 sends the computer to the subroutine to draw the border.
- lines 1000-1060 (the subroutine) draw the border by creating a series of reversed spaces. It is very important that the border falls exactly on the edge of the window, so when you redefine the window in line 50 you don't eliminate the border by mistake.
- Line 40 cancels the window, and line 50 creates a new window, smaller by one space in each direction.
- lines 60--90 set up the window to accept user input. Notice how reversed characters are used to highlight the first letter of each type of gasoline.
- li nes 91-95 use IF/THEN statements to evaluate the input in line 90 and determine what type of gasoline is wanted.
- Finally, line 100 asks the user to input the amount he wants to spend.

Now you have all the input needed to use another window for output. The second window, to simulate a gas pump, should be more decorative. Use graphic characters to outline the window instead of reversed spaces. Lines 110 to 330 form the second window and calculate the program's output. Look at this section of the program a few lines at a time.

You can change the color of the gas pump if you wish. Just insert different control characters in the print statements. The window can be as decorative or as plain as you wish.

The necessary calculations used to update the window continually as the gas is pumped are combined with the PRINT statements used to create the window. Note that you can't use a subroutine to draw the gas pump because the output would scroll incorrectly.

You want the number of gallons pumped and the amount spent to increment on the screen as they do in a real gas pump; a FOR/NEXT loop should be right for this. You want the gas pump to update the current cost and gallons until the predetermined maximum cost is reached.

VARIABLES:

- Gallons (updated as the gas is "pumped"): stored in G
- Amount spent so far (updated in conjunction with gallons): stored in A

Let's think about the FOR/NEXT loop. You want the gallons and cost to be updated continually until the amount the user wants to spend (stored in M) has been reached. So the statement to declare the loop is:

FORA=l TO INT(M*l00)

At first glance, this command probably seems more complicated than is necessary. Why multiply the amount desired by 100? You'll only have to divide other variables 100 to keep the results valid. The answer has to do with the floating point decimal within the computer. In rare cases, because of the way the computer calculates certain decimal values, it is necessary to use integer values and sometimes manipulate the values slightly to obtain the result you want.

Look at the rest of the program, and be sure you understand the structure. Now run the program. Notice that the gas pump looks as if only the numbers are changing, when actually the entire pump is being drawn each time through the loop.

You may think the action is too slow to be realistic. That's because gas is being pumped in $0.01 increments. If you change the value used to multiply M, A and P in lines 170,250, and 310from 100to 20, the gas is updated in 5-cent increments and the program is much more realistic.

To see another interesting use for windoWS, LOAD and RUN the program from your disk called HOROSCOPE. This program lets the user input a birthdate for an astrological reading. Notice how separate windoWSare used for elements such as sign of the zodiac, ruling planet, and personality. Note: The HOROSCOPE program does not run on the C16 due to memory limitations.

C128 ONLY

EXPERIMENT 4·2

(a) Create a program that displays a series of overlapping windows. Each window should be drawn by reversed spaces of a different color.

(b) Create a slot machine that scrolls random-color blocks in three separate windows, and prints the payoff in a fourth window.

(c) LOAD and RUN the program QUIZ4. Answer the questions about screen editing and windows.

* **Experiment 4.2 Completed** *

UNIT 5:

Advanced Data Input Using the Keyboard

You are familiar with several methods of data input from the keyboard, including **INPUT** and GET. In this unit we will discuss other methods of data input available in BASIC 3.5 and BASIC 7.0 that may help you create more sophisticated programs, or simplify the process of obtaining input.

GETKEY

You learned to use the GET command in Introduction To BASIC Port I. To refresh your memory, GET is used to obtain one character of input. You will remember the GET command can accept the null string as a character, and therefore sequences such as the following were common.

```
10 PRINT "PRESS ANY KEY"
20 GET A$
30 IF A$="" THEN 10
40 PRINT "THE KEY YOU PRESSED
   WAS ";A$
```

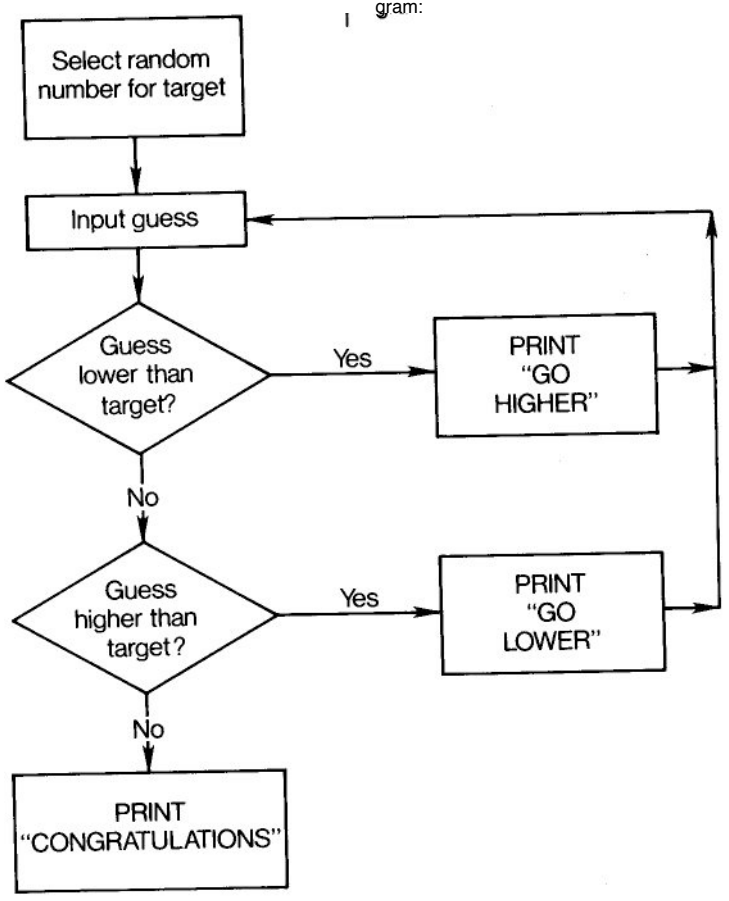
The IF/THEN statement sow to it that the computer executed lines 20 and 30 as a loop until the user hod time to press a key.

Your computer has another command, called GETKEY, that automatically waits for the user to press a key. Try the following program. Notice there is no need to check for the null string.

```
10 PRINT "PRESS ANY KEY"
20 GETKEY A$
30 PRINT "THE KEY YOU PRESSED
   WAS";A$
```

The main advantage of GETKEY is in shortening programs. GETKEY waits for the user to press a key, and assigns only that character to the string specified. For example, let's write a program that lets the user pick a number from 1 to 9, trying to match a number randomly chosen by the computer. We wont the computer to respond accordingly if the number input by the user is higher or lower than the target number, and give a congratulatory message when the user chooses the target number.

Here's the flow chart for such a program:



GETKEY is the obvious command for user input. Use the RND function to choose a random number from 1 to 9. The rest of the code is simple enough; messages that tell the user to guess higher or lower can be contained in IF/THEN statements that compare the user's guess with the randomly chosen target.

You may recognize one possible snag: GETKEY recognizes a string, and you wont to compare numerical values. luckily, the VAL function can be used to extract the numerical value from a number inputted as a string. Refer to Introduction to BASIC Port 1 if you are unsure how the RND and VAL functions work.

Here is the program:

```
10 PRINT"[ SHIFT] and [HOME]"
20 T = INT(9*RND(1)) + 1
30 PRINT"PICK A NUMBER FROM 1
   TO 9!"
40 GETKEY N$:N = VAL(N$)
50 IF N> T THEN PRINT "GO LOWER
   THAN"N:GOTO 40
60 IF N<T THEN PRINT "GO HIGHER
   THAN"N:GOTO 40
70 PRINT"[ SHIFT] and "[HOME!"
80 PRINT:PRINT"CONGRATULATIONS,
   YOU GOT IT!":REM N =T
90 PRINT"TRY AGAIN (YIN)?"
100 GETKEY A$
110 IF A$ = "Y" THEN GOTO 10
120 PRINT
130 END
```

Notice that the user's answer to the question in line 90 is handled by GETKEY. You could use on INPUT statement, but GETKEY works just as well. This program can be modified to include scoring, or more complex messages.

GETKEY is useful for programs requiring user input of one stroke only. It is possible, however, to use a series of GETKEY statements to create a longer string. One occasion where the GETKEY string can be useful is in a program requiring a password. You are probably familiar with databases that require users to enter a password before they can view information. A simplified example of this type of programming is not hard to replicate.

You can use a GETKEY statement to store a series of characters.

```
GETKEY A$;B$;C$;D$
```

is a valid command.

Once the various strings are stored, they can be concatenated into a complete password with an assignment statement:

```
E$= A$ + B$ + C$ + 0$
```

Then E$ can be tested against a pre-assigned password to determine if the user should be allowed access to the program.

Another way to concatenate the strings is to start with the null string and use a loop to add one character at a time. Type NEW, then type:

```
10 P$ = "CODE"
20 PRINT" "
30 PRINT "ENTER THE CORRECT
   PASSWORD"
400$=""
50 FOR X = 1 TO 4
60 GETKEY B$
700$=0$ + B$
80 NEXT X
90 IF 0$< >P$ THEN PRINT 'WRONG
   CODE-ACCESS DENIED":GOTO
   110
100 PRINT"YOU HAVE GAINED
   ACCESS"
110 END
```

The Password program is useful, but a clever user trying to gain access would quickly discover the password by listing the program. Modify the program so RUN/STOP is disabled and the user cannot list the program once it is running.

* Experiment 5.1 Completed *

DEFINING FUNCTION KEYS

GETKEY is a useful command when a program requires input of one stroke at a time. You also sow that it is possible to concatenate strings obtained with GETKEY, but this is somewhat clumsy. Some programs require very long strings to be typed repetitively, and for this your computer has definable function keys.

You have used your computer's built-in function keys for LIST, DSAVE, DIRECTORY, etc. You have also probably purchased software that programmed the function keys to serve a number of purposes. In this section we will learn how to redefine function keys in direct and program mode, and discover some of the ways these redefined function keys can be used.

Here is the list of function keys on the C128. You can use these function keys at any time they haven't been given another definition by software.

Key	Function on C128
Fl	GRAPHIC
F2	DIOAD
F3	DIRECTORY
F4	SCNCIR
F5	DSAVE
F6	RUN
F7	LIST
F8	MONITOR (HEIP on the PIUS/4 and C16)

To use any of the functions, just press the key of the function you desire. The command appears on the screen, and is executed.

Suppose you are using a program that requires repetitive typing of long sequences of keystrokes. An example of this might be a telecommunications program that uses complex commands to operate the program, or communicate with another computer. You could program any common command onto a function key and save yourself considerable time and typing errors.

The method for redefining a function is very simple:

```
KEY number,"new definition"
```

There are eight function keys you can redefine. Remember that if you redefine a function key you will have to reset the computer to get the original functions bock.

You may wont to include the character code for RETURN-CHR$(13)-as port of the new key definition, so you don't have to press RETURNafter you press the function key. There may be times, however, that you want to program only strings onto the keys, not actual functions, and in these cases you will not need the CHR$(13).

Function keys can also be defined within programs. let's write a program that redefines function keys 1 through 4 for use in telecommunications. We'll program a telephone number, user number, password and a common command:

```
10 KEY 1,"8005552121"+CHR$(13)
20 KEY 2,"22345321" + CHR$(13)
30 KEY 3,"SECRET" + CHR$(13)
40 KEY 4,"GO CBM" + CHR$(13)
```

You must place quotes around the definition.

RUN the program, and notice that the keys now have the new definitions. Each time you press a redefined function key the computer responds "syntax error," but this is only because you are using the function keys out of context. If you were running a terminal software program for a modem, the redefined function keys would be effective and helpful.

You have to reset the computer to restore the original values. Other programs you could write might redefine the function keys for use throughout the program, then restore the original functions when the program is done executing. This process avoids the need to reset the computer.

Try this example. It shows how to restore a function key to its original function.

```
10 PRINT"THIS PROGRAM DEFINES
   THE Fl KEY"
20 PRINT"TO BE YOUR NAME"
30 INPUT"YOUR NAME";N$
40 KEY 1,N$+CHR$(13)
50 PRINT"WHEN I ASK A QUESTION,
   YOU JUST HIT THE Fl KEY"
60 INPUT"WHO'S THE FAIREST";A$
70 INPUT"WHO'S THE SMARTEST";A$
80 INPUT"WHO'S THE GREATEST";A$
90 INPUT"WHO'S THE NARCISSIST";A$
100 KEY 1,"GRAPHIC"
110 END
```

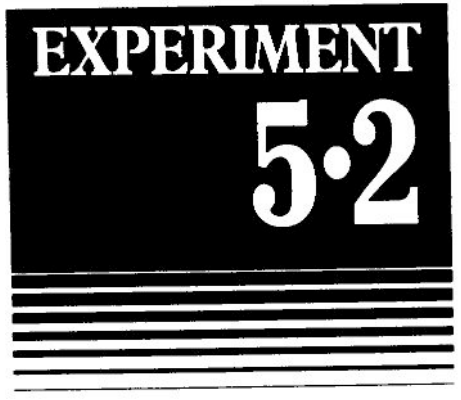
RUN the program. After it stops, press Fl again. It has been restored to its usual function.

It is possible to program more than one function onto a single function key. In fact, the only limit to how many functions can be programmed onto a function key is the program line length-only 160 characters will be accepted as part of one line. For example, create a function key that will load a program from a directory, list it, then run the program. Type NEW, then type

```
10 PRINT" "
20 PRINT"PIEASE INSERT YOUR DISK"
30 PRINT "HIT ANY KEY WHEN
   READY":GETKEY A$
40 DIRECTORY
50 PRINT"PIEASE ENTER THE NAME
   OF THE"
60 INPUT"PROGRAM YOU WISH TO
   LOAD";N$
70 KEY1,"DIOAD" + CHR$(34) + N$ +
   CHR$(13) + "LIST" + CHR$(13) +
   "SLEEP7 + CHR$(13) + "RUN" +
   CHR$(13)
80 PRINT"PRESS Fl WHEN READY"
```

Plus/4 and C16 users should delete the SLEEP 7 and CHR$(13) that follows it in line 70 for this program to RUN.

Notice in line 30 we have used GETKEY A$ more or less as "filler". The GETKEY statement delays the program until the user has put a disk in the drive and is ready to continue. CHR$(34) in line 70 stands for the quote symbol, needed in combination with the DIOAD command.



lOAD and RUN the quiz on your disk for this unit, QUIZ5 (or QUIZ5/4).

* **Experiment 5.2 Completed** *

UNIT 6: Case Study

TRIVIA PROGRAM

So far in this book, you've seen different applications or examples demonstrating the new BASIC commands, such as COLOR, DO and lOOP, BEGIN and BEND, PRINT USING, SLEEP,KEY and GETKEY. Now it's time for a program that features all these commands together.

Our case study program is a game of question-and-answer trivia. The main tasks in writing a trivia program are to:

I-ask questions and give choices for answers

2-accept answers

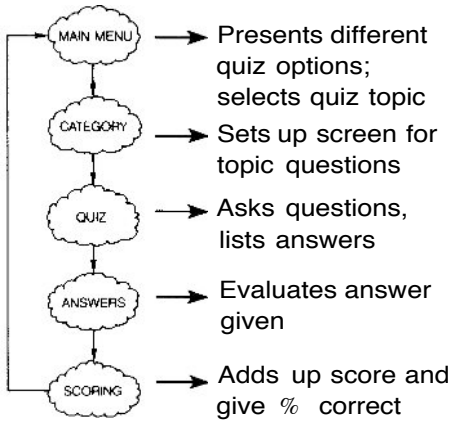
3-evaluate answers

4-keep track of correct answers

Different topics within the game make the quiz more elaborate.

There are many ways to write a trivia program. A good program might feature a menu of different topics and multiple choice questions for each topic. A scorekeeper also odds to the attractiveness of the program.

The trivia program for this case study is saved on your program disk under the nome TRIVIA for the C128, TRIVIA/4 for the Plus/4 and TRIVIA/16 for the C16. The C16 version does not contain all five trivia categories due to memory limitations. The program is broken down into sections for which flow charts are provided. A complete listing appears at the end of the program. You can follow along by lOADing the program and liSTing the line numbers specified for each different routine. The algorithm for TRIVIA is straight-forward:



The important subroutines that have to be worked out are:

Selecting options from Main Menu

Setting up individual topic quizzes

Presenting quiz questions and answers

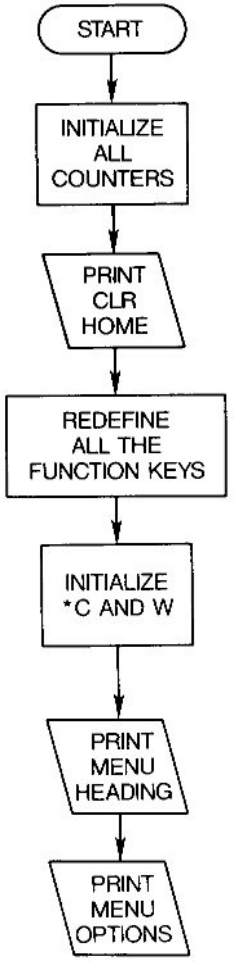
Accepting and evaluating answers given

Counting answers and calculating the percentage correct

THE MAIN MENU

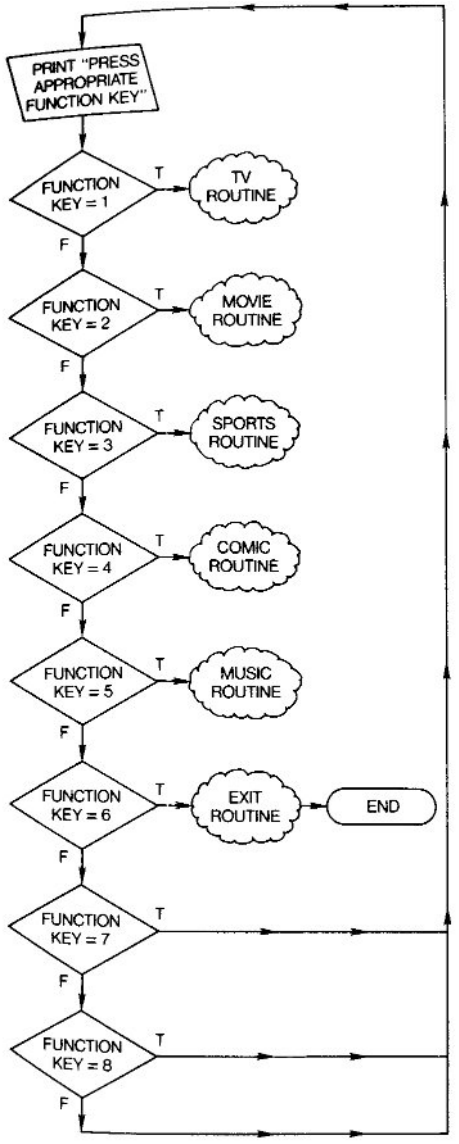
The main menu offers the user a choice of quiz topics. Since trivia, by definition, covers all subjects, we'll restrict our choices to five popular topics: television, movies, sports, comics and music. The design of menu is determined by the choices it off Our menu consists of the five quiz topics on EXIToption, so the user can quit the program easily.

For the user to make a selection, we can use one of the new commands we'v reviewed: KEY, which can assign selecti~on of each topic to a function key. The routi for setting up the main menu looks like t



*C and Ware counters for correct and wrong answers.

The specifiC routine.for th.e pressing the function key looks like thiS:



It's easy to assign topic selection to a function key with KEY. Use the GOTO statement and assign the line number of each particular category's subroutine when redefining each key. In this fashion, function keys 1 through 5 send the user to the subroutine for television, movies, sports, comics, and rnusic, respectively. Function key 6 lets the user exit from the program into BASIC. Function keys 7 and 8 are defined to return the program to the main menu (GOTO 7).

PROGRAM LINES: LIST-1I0

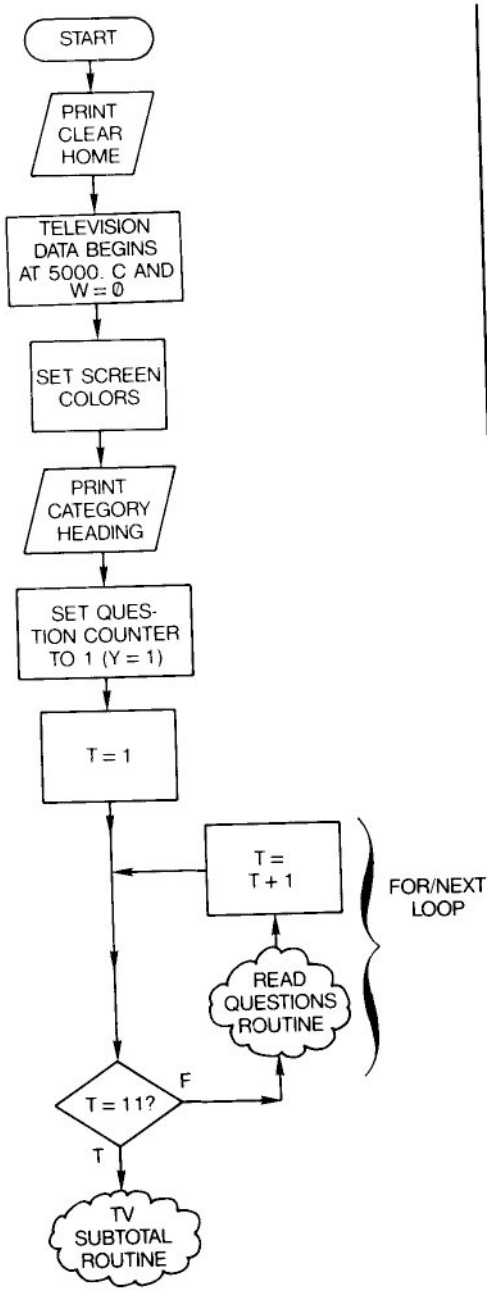
Note: Remember after you have exited the TRIVIA program, redefined function keys retain the new definition until the computer is reset or turned off.

CATEGORY ROUTINE

Each time a function key is pressed and a category is selected, a new subroutine comes into ploy. The routine is essentially the some for all topics. We'll just look at the routine for television.

Our task is to set up a specific category screen and prepare to ask ten questions. This entails designing the screen, directing the program to the data for the questions, and setting a counter to keep track of the number of questions.

Notice that the COLOR command sets the screen background, foreground and border colors. New colors are set in this fashion in each category. The category subroutine flow chart below begins with the press of the appropriate function key.



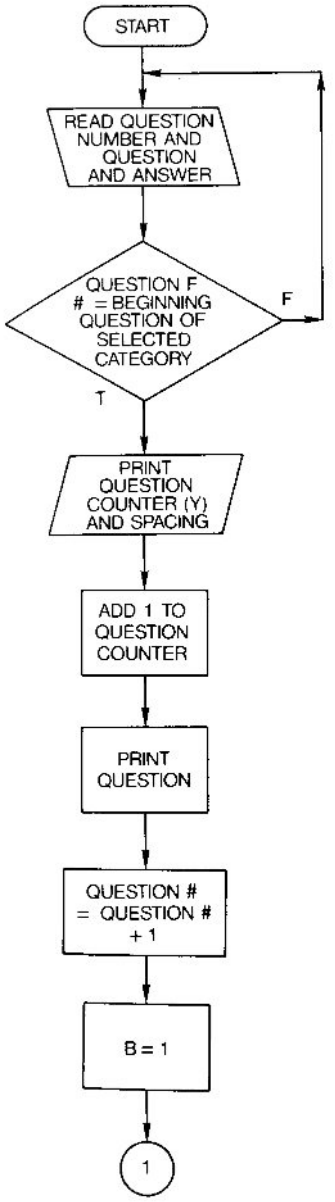
PROGRAM LINES: LIST 140-250

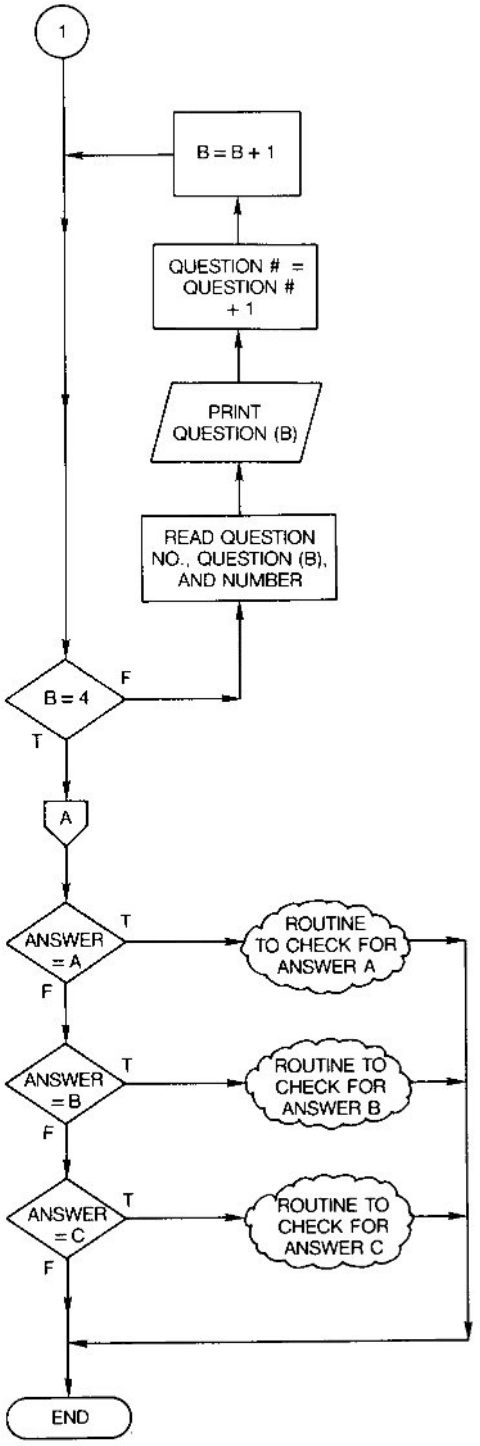
line 200 sets variable IN to 5000, the line number where the DATA statements with questions and answer choices for this category are located. Variables C and W, set to 0 in line 200, are used later to count answers: C for correct, W for wrong responses. Variable Y in line 230 holds a counter number displayed on the screen as the question number. lines 240-250 set up a FOR/NEXT loop containing a GOSUB statement. This GOSUB transfers execution to another routine at line 4000, the quiz routine.

THE QUIZ ROUTINE

The heart of the program is the question and answer routine. This routine must establish the procedure for asking the question, presenting the choices, accepting the answer and evaluating the answer chosen.

The quiz routine can be broken down into two routines: read question and evaluate answer. The question routine follows this flow chart:





PROGRAM LINES: LIST 3900-4120

Several commands from earl ier chapters are included in this subroutine, such as BEGIN/BEND, PRINT USING and EISE. The question and possible answers are READ from DATA statements later in the program, as directed by line 4000 (READ Q,Q$,A). PRINT USING creates a format for the question number (variable Y) to be displayed, for questions 1 through 10 in each category. BEGIN/BEND and ELSE are used with on IF/THEN clouse that checks that the right question is being read (variable Q must match IN, which stands for line number of the correct question). If Q doesn't match IN, the EISE clouse returns execution to line 4000 to start the question again.

Since the actual data for questions, answer choices and correct answer are contained in DATA statements at the end of the program (from line 5000 on), the READ statements in lines 4000 and 4080 each read three variables: the number of the question (to make sure that the right question for the topic is being read), the string variable with either a question or answer choice in it (Q$), and a third number. This third number determines the correct answer for the question. The DATA statements for each question are grouped in fours, for example:

```
5000 DATA5ooo,WHAT ROIE DID
     RALPH WAITE PLAY ?, 1500
5001 DATA5001 ,A) JOHN WALTON
     ON THE WALTONS, 1
5002 DATA5002,B) GEORGE APPLE
     ON APPLE'S WAY,2
5003 DATA5003,C) IARRY TAIT ON
     BEWITCHED,3
```

The first number each statement contains matches the line number. It is read into variable Q in line 4000 and compared to variable IN, which is a counter that makes sure the right question is being read.

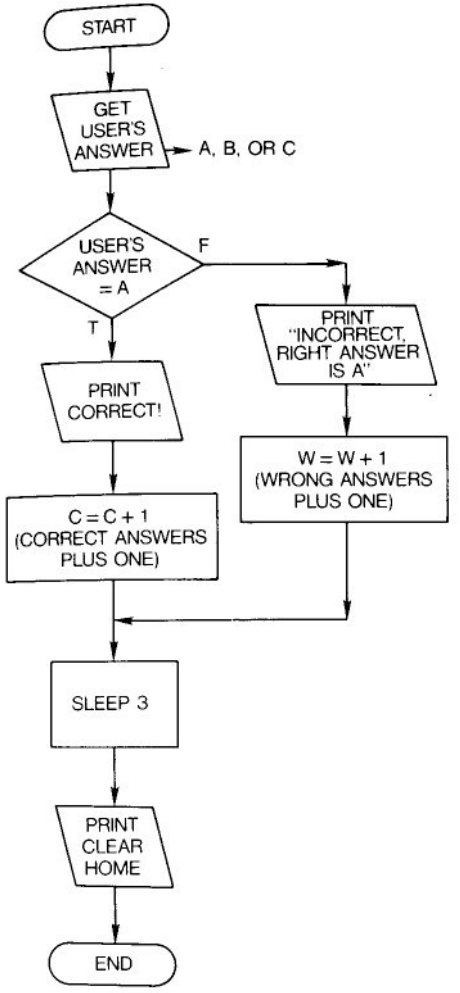
The first statement then has a text string that asks the question, read into Q$. The other three statements feature strings that give possible answers A, B and C.

There is also a number after each string. For the statement which asks the question (5000 in this example), a number is read into variable A as a direction to a subroutine. lines 4115-4117 use this variable to transfer execution to the subroutine for correct answer. This value is either 1500 if answer A is correct, 2000 if answer B is right, or 3000 if C is the correct choice. The DATA statements with answer choices also have a number following the string, but this number is a "dummy" variable read into variable according to line 4080. This keeps the variables that are READ even, since th~e READ statement in line 4000 is reading three variables. If the wrong line were read as a question, the program would crash.

Once the correct question and answer choices appear on the screen, execution is transferred to one of three answer routines.

THE ANSWER ROUTINE

There are three answer subroutines, each identical except for the answer that is correct. The answer subroutine for the correct answer A (line 1500) begins by using GETKEY to get a single letter for the answer The letter assigned to the GETKEY variable should be A for a correct answer if the answer is A. Any other key press is considere a wrong answer. The subroutines for corre answers B (line 2000) and C (line 3000) are identical except for the value for the GETKE. variable..

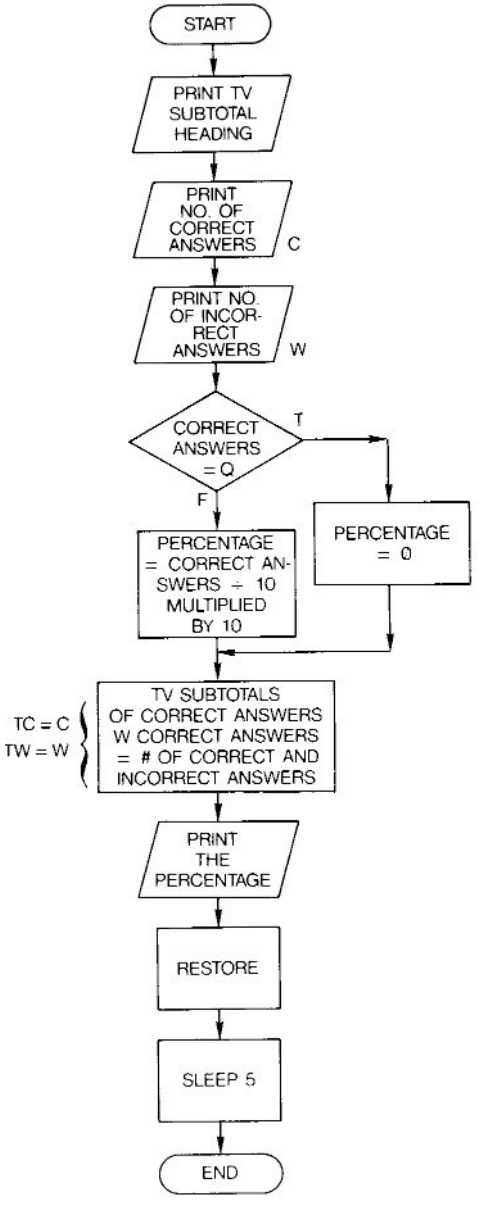


PROGRAM LINES: LIST 1490-1620

The variable W in line 1530 is a counter for wrong answers, while the variable C in line 1570 is a counter for correct answers. BEGIN/BEND statements make handling the situation much easier. The SLEEP statement in line 1590 is a delay to give the user time to review the correct answer after it is revealed in line 1520 or 1560. The RETURN in line 1620 takes it back to the quiz question subroutine, which is then RETURNed to the original category routine, to either ask onother question (if less than ten have been asked) or to go to the routine for scoring total for that category.

SCORING ROUTINES

After the tenth question for a category, the scoring routine "announces" the user's score, based on number of correct answers (variable C) and wrong answers (variable W), as well as the percentage correct. There is a separate scoring routine for each of the five categories, as well as a final total scoring routine.



PROGRAM LINES: LIST 376-435

lines 390, 400 and 420 feature a PRINT USING statements to format the numbers displayed on the screen. The counter variables C and W are totaled in line 415 in variable TC for use in the final score routine for answer counts for the television category. Other scoring routines features variables MC, SC, CC and UC for movies, sports, comics and music category totals, respectively. line 410 figures the percentage of correct answers by taking the total of correct answers for the category (C) and dividing it by the number of questions (10), and then multiplying it by 100 to get the percentage. line 425 creates a five-second delay so the user can fully digest the scoring figures. line 435 sends the program bock to the main menu.

The exit scoring routine (1370-1400) does the some thing as the category scoring routines, but uses the total score variables (TC, TW, MC, MW, etc.) to find total score and percentage. The last line of the subroutine, line 1400, ends the program.

Since the function keys were redefined with the KEYcommand in this program, you'll have to turn your computer off and on again to return the function keys to their old definitions.

Following the logic of the TRIVIA program, odd another trivia category and questions. Choose whatever category you like. Change the screen colors to be different from the other category screens. Remember to redefine another function key and reproduce the other routines as they are handled in the program.

* **Experiment 6.1 Completed** *

UNIT 7:

Graphic Commands and Drawing

THE GRAPHIC COMMAND

To use any of the graphic commands to draw shapes and lines, you must first create a graphic area. This is done by issuing the GRAPHIC command, either within a program or as a direct command.

GRAPHIC can be followed by three parameters:

First: Mode
- O-normal 40-column text
- I-high-resolution graphics
- 2-high-resolution with split screen
- 3-multicolor graphics
- 4-multicolor with split screen
- 5-80-column text (C128 only)

Second: Graphic Clear Option
- • O-OO not clear screen
- I-Clear graphic screen

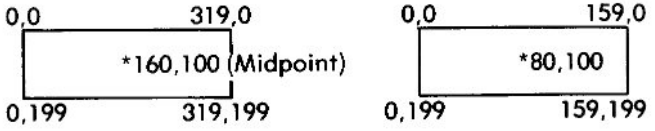
Third: Starting line for Split Screen
Can be used in mode 2 or 4
Default is 19

Split screen provides lines at the bottom of the graphic screen for text.

The difference between high-resolution and multicolor modes is based on the precision with which you can draw and the number of colors available at one time. GraphiC screens are divided into dots, or pixels, that you use as screen coordinates when plotting lines and shapes with graphic commands. The high-resolution screen is broken down into more pixels than multicolor and lets you draw with finer control of the pixels on the screen. Although multicolor has lower resolution (fewer pixels), it allows you to use two additional colors. This short program should illustrate some of the differences between the graphic modes.

```
10 FOR X = 1 TO 4
20 GRAPHIC X,1
30 COIOR 1, X + 2
40 PRINT "SPLIT SCREEN"
50 BOX 1,100,100,200,200" 1
60 FOR T = 1 TO 100:NEXT T
70 NEXT X
80 GRAPHIC CLR
```

The lost line, GRAPHIC CIR, puts you bock in normal40-column text mode. Notice how the bottom of the screen is cut off in the split-screen modes, and the printing appears in that area. Also, you can see how the BOX command, even though the values are exactly the some in all the modes, draws a box in different places on the screen in high-resolution and multicolor modes. This is because the screen coordinates are different in the two graphic modes. While the screens are the some size, the resolutions are different. BOX will be reviewed in-depth in Unit 8. Here are the screen coordinates for high-resolution and multicolor modes:



When in a graphic mode, most or all of the screen is used to show graphics. To return to the normal40-column screen that displays what you type, you'll need to type GRAPHIC 0 or GRAPHIC CIR. If you don't, it will seem like the computer isn't responding at all when you try to type, when in actuality you just can't see what you're doing because the computer is still displaying the graphic mode. It might be a good ideo to program a function key to execute the command to toke you out of graphic mode and bock to text mode, using the KEY command:

```
KEY 7,"GRAPHIC 0" + CHR$(13)
```

You should be aware of two things when using the GRAPHIC command. The high-resolution bit-mop mode uses 10,000 bytes for the internal representation of the picture. This area is reserved when you give the

```
GRAPHIC 1,1
```

command, and that port of memory is no longer available for other uses such as storing programs or variables. Since the Plus/4 has 65,535 bytes and the C128 has 131,070, the loss of 10,000 bytes is not such a serious matter.

Secondly, the

```
GRAPHIC 1,1
```

command destroys any string variables which happen to exist when GRAPHIC is issued. Placing the GRAPHIC command at or near the beginning of the program, before any strings have been used, ovoids this problem.

THE COLOR COMMAND

The trade-off for greater resolution is the capacity to use more colors in multicolor mode. COLOR is the BASIC command used to change the color of any screen element (foreground, background, border and assigning additional colors for multicolor mode). A typical example of the COLOR command is

```
COIOR 1,5
```

The first parameter specifies color source (the screen element involved), the second is for the color. There can be a third value, as mentioned earlier, for luminance for the Plus/4 and C16, to give the command this form:

```
COIOR 1,5,5
```

First: Color Source
- O-background
- I-foreground (character on Plus/4)
- 2-multicolor 1
- 3-multicolor 2
- 4-border
- 5-character (C128 only)
- 6-80-column background color (C128 only)

Second: Color (C128 values)

I-black	9-orange
2-white	10-brown
3-red	11-1 ight red
4-cyan	12-dark gray
.5-purple	13-medium gray
6-green	14-light green
.7-blue	15-light blue
8-yellow	16-light gray

These values correspond to the color keys. They are slightly different for the Plus/4 and C16 computers. Values for colors 1 through 10 are the same as on the C128, but values for 11 to 16 are as follows:

II-yellow	green
12-pink	
13-blue	green
14-light	blue
IS-dark	blue
16-light	green

Use the COLOR command in direct mode to take a look at how the different shades look in different areas of the screen. Try typing these commands:

```
COIOR 1,11        COIOR 0,4
COIOR 0,13        COLOR 5,5
```

Experiment with the color values, and if you are doing this with a Plus/4 or C16, a third value for luminance if you are doing this with a Plus/4 or C16, like this:

```
COLOR 0,5,7 (for the lightest shade
             of purple for the
             background)
```

This program uses COLOR to let you try every different screen color combination. By pressing number keys (1 and 2 for background color, 4 and 5 for foreground and 7 and 8 for border), you can cycle forward and backward through the possible combinations of colors. When you find the most attractive screen color combination, press the Y key, and your screen is color-coordinated. This program, entitled COIORPLUS can be found on your program disk. A Plus/4 version of this program with additional control over luminance is on your disk, entitled COIORPIUS/4 .

```
10 PRINT"[ SHIFT ] and [ HOME ]
   SCREEN COIOR SELECTION"
20 COIORO,I :COIOR4,2:COLOR5,2:
   REM STARTING COIOR VALUES
25 XI =1: X2=2: X3=2
30 PRINT"PRESS 1 OR 2 TO CHANGE
   CHARACTER COIOR"
40 PRINT" AND 4 OR 5 TO CHANGE
   BACKGROUND COLOR"
50 PRINT" AND 7 OR 8 TO CHANGE
   BORDER COIOR"
60 PRINT"PRESS Y TO STOP"
70 PRINT" "
80 PRINT"THIS IS HOW THIS
   CHARACTER COIOR"
90 PRINT"IOOKS ON THIS
   BACKGROUND"
```

```
100 GETKEY Q$
110 IF Q$ = "1" THEN X2 = X2 + 1
120 IF Q$ = "2" THEN X2 = X2-1
130 IF X2<1 THEN X2= 16
140 IF X2>16 THEN X2 = 1
150 IF Q$ = "4" THEN XI = XI + 1
160 IF Q$="5" THEN XI =XI-I
170 IF XI <1 THEN XI = 16
180 IF XI >16 THEN XI = 1
190 IF Q$ ="7" THEN X3 = X3 + 1
200 IF Q$ = "8" THEN X3 = X3-1
210 IF X3<1 THEN X3=16
220 IF X3>16 THEN X3 = 1
230 IF XI = X2 AND Q$ = "1" THEN
    X2=X2+1
235 IF X2>16 THEN X2=1
240 IF XI = X2 AND Q$ = "2" THEN
    X2=X2-1
245 IF X2<1 THEN X2 = 16
250 IF XI = X2 AND Q$ = "4" THEN
    XI =XI + 1
255 IF XI >16 THEN XI = 1
260 IF XI = X2 AND Q$ = "5" THEN
    XI =XI-I
265 IF X1<I THEN XI = 16
270 IF Q$ = "Y" THEN 300
280 COIOR O,XI :COIOR5,X2:
    COIOR4,X3
290 GOT070
300 PRINT"[SHIFT][ HOMEI":END
```

Glossary:

XI: Background color value

'X2: Character color value

X3: Border color value

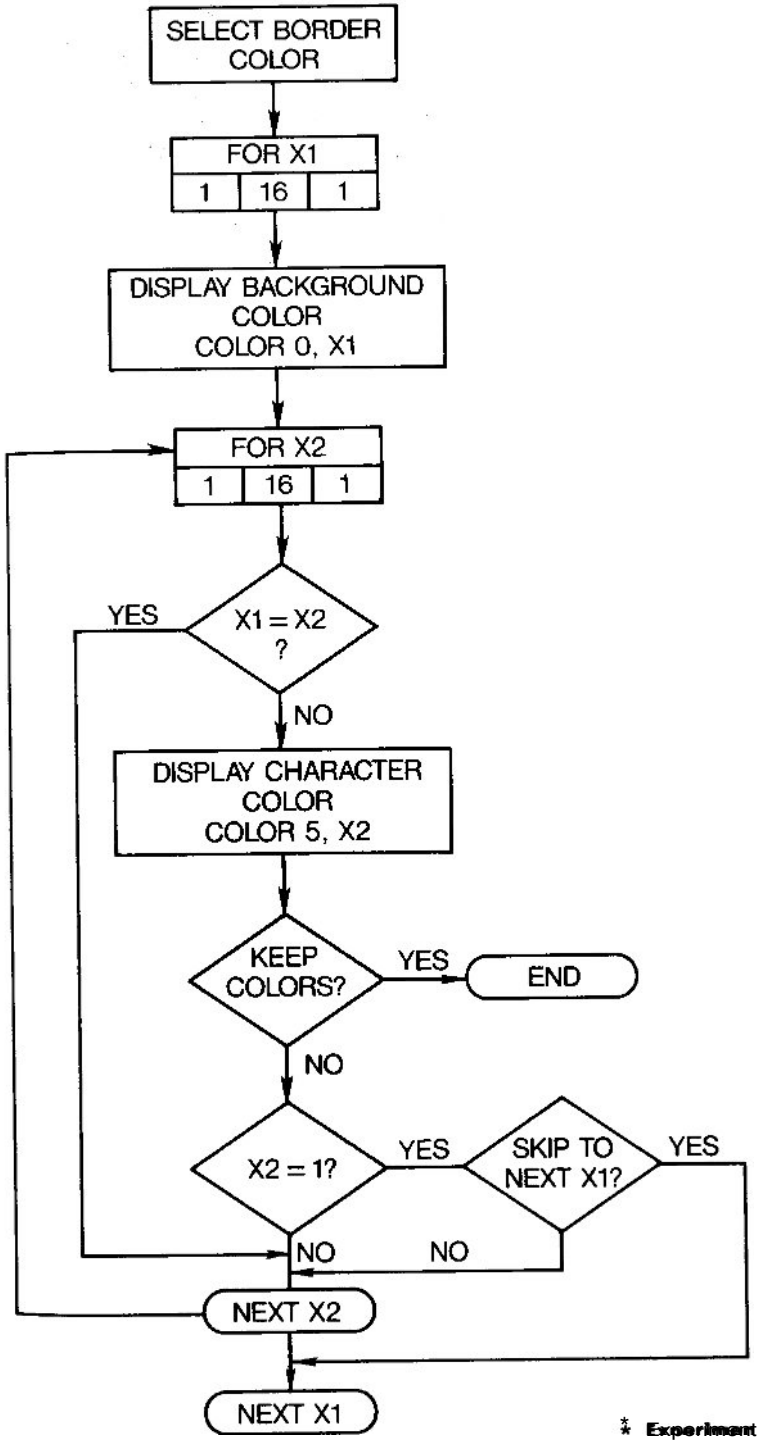
Q$: String variable for key to assign value to change screen color

Note: To RUN this program on a Plus/4 or C16, references to COLOR5 in lines 20 and 260 are changed to COLOR 1. COIORPIUS/4 also includes changing luminance.

We'll concentrate on high-resolution mode in this section and look at multicolor graphics later.

EXPERIMENT 7·1

Write another screen color selection program using FOR/NEXT loops to select screen background and character colors. Show each character color on each different background color. let the user go through the different combinations until they find one to their liking. To speed up the program, allow the user to choose the screen border color at the outset, and give them the option of skipping to the next background color without going through all 16 character colors each time. Follow this flow chart to design the program.



* Experiment 7.1 Completed *

DRAWING ON THE SCREEN

The simplest computer pictures consist of collections of "points" or foreground pixels. To draw a point in the foreground color, use the command

```
DRAW 1,X,Y
```

where X and Y are the screen coordinates of the point. This is illustrated in the following program, which draws a mathematical curve by plotting a series of points:

```
10 COlOR 0,8
20 GRAPHIC 1,1
30 COlOR 1,1
40 FOR X =0 TO 315 STEP5
50 Z = X-160
60 Y = 0.0001 *(Z 3 + 100*Z 2-
   10000*Z) +50
70 DRAW 1,X,Y
80 NEXT X
90 END
```

Type in this program and RUN it. It shows a curve of black points on a yellow background. Each point is one pixel. The program has several interesting features:

The first three commands create a yellow display area and select block as the foreground color. line 40 sets X, which is used as the horizontal screen coordinate, to move across the display area in steps of 5.

The main loop of the program runs from line 40 to line 80. lines 50 and 60 calculate a value for Y in terms of X. The formula is selected to produce a wavy line (the details of the calculation are not important for our purposes). line 70 plots a Single point at screen coordinates X, Y, and line 80 makes the machine repeat the loop for the next value of X.

Notice part of the curve is outside the display area. For example, when X = 300 the value of Y is about 478, which is outside the screen. The program shows you what happens if you try to plot a point outside the limits of the display area-nothing.

The formula implies that the values of Yare often fractional, such as 1.64326. When given such a screen coordinate, the computer takes the nearest lower whole number, so 1.64326 is interpreted as 1.

After examining the curve on the screen, you may want to modify or store the program. Remember, since you are still in high-resolution graphic mode, that you won't be able to see what you type until you return to text mode.

A quick way to return to text mode on a C16 or C128 is to hit the Fl key (which prints the word GRAPHIC), then press zero and RETURN. As soon as you do so, the ordinary screen with the character listing comes bock.

Another way to get bock to the main screen is to put the command GRAPHIC 0 at the end of your program. This has the drawback that your picture disappears as soon as it is complete. This can be avoided with a sequence like this at the end of a program:

```
85 GETKEY A$
88 GRAPHIC 0
```

Now the picture won't disappear until you hit a key.

If the program contains any errors, it will automatically return to the character screen. This gives you yet a third way of switching from the picture to the characters: type any line with a deliberate error, such as GRAPHIC by itself, followed by RETURN.

When pixels are close together, they merge to form a continuous line. Change line 40 in the program so that X goes up in steps of 1 (rather than 5) and run it again to see the effect.

Use the "dot-plotting" technique to draw some interesting curves. Here are some suggestions for curves. Make these routines into graphic programs. Remember to number each line and to start the program by issuing a GRAPHIC command.

(a) Parabola:

```
FORX=OTO 189
Y = 199-(X-80) 2/60
DRAW 1,X,Y
NEXTX
```

(b) Circle:

```
FORJ = 0 TO 359
X= 160 + 80*SIN(J*π/180)
Y = 100 + 6O*COS(J*π/180)
DRAW 1,X,Y
NEXTJ
```

(c) Sine curve:

```
FORX = OTO 319
Y= 100 + 90*SIN(XI25)
DRAW 1,X,Y
NEXTX
```

* Experiment 7.2 Completed *

LINE DRAWINGS

One method of drawing a line is to plot a row of pixels next to each other, like this:

```
10 COlOR 0,6
20 GRAPHIC 1,1
30 COlOR 1,3
40 FOR X = 30 TO 270
50 DRAW I,X,75
60 NEXT X
70 GETKEY D$:GRAPHIC 0
```

This program draws a straight horizontal line between the points with coordinates

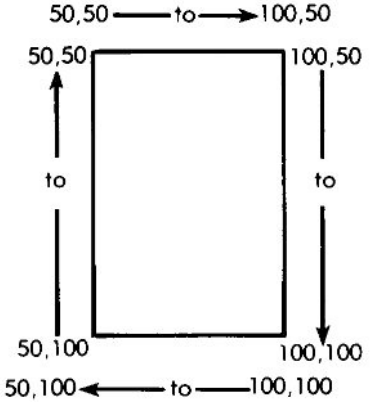
(30,75) and (270,75)

This method is time-consuming. Fortunately, the computer provides a more convenient way of drawing lines. You can extend DRAW by attaching the word TO and the screen coordinates of an "end point." This produces a straight line between the two points defined in the command. look at these two examples:

```
DRAW 1,30,75 :          Plots a pixel at
                        (30,75)
DRAW 1,30,75 TO 270,75 : Plots a line
                        between (30,75)
                        and (270,75)
```

The machine is not limited to drawing horizontallines as in this example, but can draw them in any direction. Most useful drawings consists of several lines, joined at the ends. For example, you could draw a square with the commands:

```
DRAW 1,50,50 TO 100,50
DRAW 1,100,50 TO 100,100
DRAW 1,100,100 TO 50,100
DRAW 1,50,100 TO 50,50
```

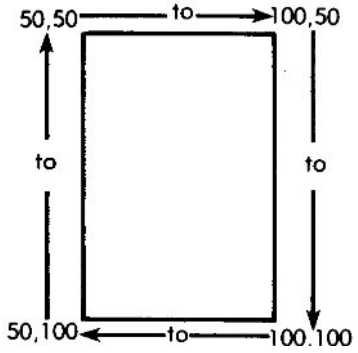


If you examine this sequence, you will notice that every point is mentioned twice. For example, the point (100,50) appears as the end point in the first command and also as the starting point in the second command. To avoid this repetition, the system uses the concept of a "current position." In most cases, the current position is simply the place where the lost pixel was drawn.

When you write the DRAW command, you can leave out the starting point, and the machine will simply assume that the line is to start from the current position. Once the line has been drown, the current position is reset to the end point.

Using this ideo, you could rewrite the commands to draw the square as:

```
DRAW 1,50,50 TO 100,50
DRAW 1 TO 100,100
DRAW 1 TO 50,100
DRAW 1 TO 50,50
```



The starting point still has to be specified in the first command. To make all the DRAW commands the some, you can use the lOCATE command, which does nothing except set up a new current position. The keyword is followed by a pair of coordinates, like this:

```
LOCATE 104,82
```

Using lOCATE, another way of drawing the square is to write

```
lOCATE 50,50
DRAW 1 TO 100,50
DRAW 1 TO 100,100
DRAW 1 TO 50,100
DRAW 1 TO 50,50
```

This is especially useful if the DRAW command is inside a loop. For example, this program will draw any shope you wont, according to the data points you enter:

```
10 COIOR 0,1
20 GRAPHIC 1,1
30 COIOR 1,2
40 READ K
50 READ X,Y
60 lOCATE X,Y
70 FOR J = 1 TO K
80 READ P,Q
90 DRAW 1 TO P,Q
100 NEXT J
110 GETKEY A$
120 GRAPHIC 0
130 END
140 DATA 5: REM NUMBER OF POINTS
150 DATA 240,100: REM STARTING POINT
160 DATA 95,147: REM POINTS TO BE VISITED
170 DATA 185,23
180 DATA 185,176
190 DATA 95,52
200 DATA 240,100
```

Press any key to return to the text screen. DRAW is so flexible that the options are still not exhausted. The first parameter after the keyword controls the color of the dot or line produced. In this position, a "1" selects the current foreground color. "0", however, implies "background color", so that the command can be used to delete lines already drawn. Here, for example, is a program which draws a flashing cross:

```
10 COIOR 0,2
20 GRAPHIC 1,1
30 COIOR 1,1
40A=1
50 DRAW A,140,70 TO 160,70
60 DRAW A,150,60 TO 150,80
70 FOR I = 1 TO 300: NEXT I
80A=1-A
90 GOTO 50
```

In this program line 70 acts as a delay to stop the cross flashing too fast. line 80 switches the value of A between 1 and 0 on alternate passes through the loop.

Finally, DRAW can produce several connected lines at the same time. You can keep on adding "TO" and another endpoint, and in each case the command will draw another straight line linked to the end of the previous segment. For example, the square discussed earlier could have been drawn with

```
DRAW 1,50,50 TO 100,50 TO 100,100
     TO 50,100 TO 50,50
```

The limit to the number of segments is set by the maximum command line length of 80 characters.

EXPERIMENT 7:3

Using graph paper, draw a picture consisting of a few straight lines. Either a geometrical figure or an artistic drawing will do. Then carefully work out the coordinates of the ends of each line, and write a program to reproduce your picture on the screen. Put the coordinates in DATA statements, so that you can easily adapt the program to different pictures.

* Experiment 7.3 Completed *

UNIT 8:

Higher Level Graphic Commands

Commands for choosing colors and plotting points are often called graphics "primitives," because they provide the framework on which everything else is built. With these commands you can, in principle, draw any picture; without them you can do nothing. But, in practice, the creation of pictures is difficult and tedious if you are restricted to the primitive commands. The computer therefore provides a selection of "high-level" commands which do common but complicated tasks automatically.

Three high-level commands we'll review are: BOX, CIRCLE and PAINT. Before going into details, it is worth dwelling on one feature which is common to all three commands: they each toke numerous parameters. For example, BOX can be followed by no fewer than seven parameters, all of which may be numbers, variables or expressions. A possible BOX command would be:

```
BOX 1,30,50,80,100,15,1
```

The parameters are, as usual, separated by commas.

Quite a few of the parameters in these commands are optional, and if they are not supplied the machine takes "default" values. For example, the fourth and fifth parameters in the BOX command default to the "current position" on the scr.een.

If you wont to leave out a parameter, you must still put in the comma which separates it from the following parameter. If there are no more parameters you can leave the commas out. For example, in

```
BOX 1,10,60
```

parameters four, five, six and seven are all token by default. In the command

```
BOX 1,3,12"" 1
```

the missing parameters are the fourth, fifth and sixth.

DRAWING RECTANGLES WITH BOX

The BOX command lets you draw a rectangle anywhere on the screen. The rectangle can be of any size and point in any direction.

The seven parameters of box are as follows:

First: Color
1 for foreground color, 0 for background.

Second and Third: The screen coordinates of one corner of the rectangle.

Fourth and Fifth: The screen coordinates of the opposite corner of the rectangle. If these coordinates are not supplied the computer tokes the "current position" as default.

Sixth: The angle of rotation, in degrees.
If this angle is zero, then the rectangle is drown with its corners at the positions given by parameters two to five. Otherwise it is turned about its center before being plotted.

Note that:

0) The angle of rotation is in degrees, not radians as used for SIN and COS.
b) If the rectangle is rotated, it doesn't pass through the selected corner points at all.

Seventh: This parameter controls the color of the inside of the rectangle. If the parameter is 0, the rectangle is drown in outline only. If the value is 1, it is painted as a solid block of color.

After you have drawn a box, it may be a good idea to use the lOCATE command to place the cursor where you wont it.

Here are some programs that illustrate different aspects of drawing boxes. Each program uses the BOX command to create a different effect. Type each in one at a time and RUN them. Ploy around with the parameter values to get a feel for how the BOX command can be used.

The first program draws boxes of identical size on the screen in a consistent pattern.

```
10 COLOR 0,2
20 GRAPHIC 1,1
30 COLOR 1,1
40 REM******BOXES IN DIFFERENT
   PLACES
50 FOR J = 12 TO 282 STEP 30
60 FOR K = 3 TO 183 STEP 15
70 BOX 1,J,K,J + 25,K + 12
80 NEXT K,J
100 GETKEY A$:GRAPHIC 0
```

Things you can try changing:

Size of the boxes (lines 50, 60 and 70)

EXAMPLE: 70 BOX 1,J,K,J + 30,K + 6

Placement of the boxes (lines 50, 60 and 70)

```
EXAMPLE: 50 FOR J = 5 TO 285 STEP 50
         60 FOR K = 10 TO 200 STEP 40
```

The next program draws solid boxes of different colors, arranged in two "staggered" columns.

```
10 COLOR 0,2
20 GRAPHIC 1,1
30 COLOR 1,1
40 REM*******COLORED BOXES
50 FOR J = 1 TO 5
60 COLOR 1,J + 2
70 BOX 1,30*J,36*J-30,30*J+50,
   36*J - 2" 1
80 COLOR 1,J + 7
90 BOX 1,30* J + 100,36* J - 30,30*
   J + 150,36* J - 2" 1
100 FOR L = 1 TO 500: NEXT L
110 NEXT J
120 GETKEY A$:GRAPHIC 0
```

Things you can try changing:

Colors of boxes (lines 60 and 80)

EXAMPLE: 60 COIOR 1,J + 3

Number of boxes (variable J in line 50)

EXAMPLE: 50 FOR J = 1 TO 3

Size and relative placement of boxes (lines 70 and 90)

```
EXAMPLE:
70 BOX 1,50,30* J, 100,30* J + 20" 1
90 BOX 1,200,30* J,250,30* J + 20" 1
```

The third program creates a circular design by drawing a series of boxes by incrementing the angle of rotation.

```
10 COLOR 0,2
20 GRAPHIC 1,1
30 COLOR 1,1
40 REM********ROTATING BOXES
50 FOR J = 1 TO 11
60 BOX 1,100,80,240,120)*360/11
70 NEXT J
80 GETKEY A$: GRAPHIC 0
```

Things you can try changing:

Colors of boxes (line 30)

EXAMPLE: 30 COIOR 1, 5

Number of boxes (variable J in line 50)

EXAMPLE: 50 FOR J = 1 TO 25

Note: This change may not seem evident, since the program may be redrawing lines that are already plotted.

Placement or size of boxes (line 60)

```
EXAMPLE:
60 BOX 1,50,50,150, loo,J*36O/11
```

Degree of box rotation (line 60)

```
EXAMPLE:
60 BOX 1,100,80,240, 120,J*36O/6
```

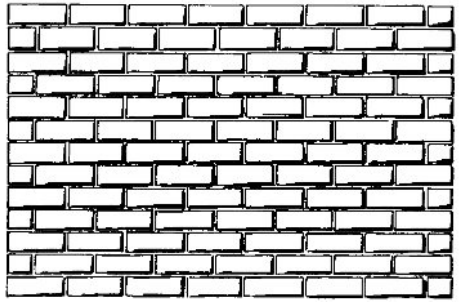
The dimension of the BOX command controlled by each parameter is sometimes most obvious when you change more than one parameter to see how the new dimensions combine to change the shape. In the

lost program, this is best demonstrated by changing lines 50 and 60 together, for example:

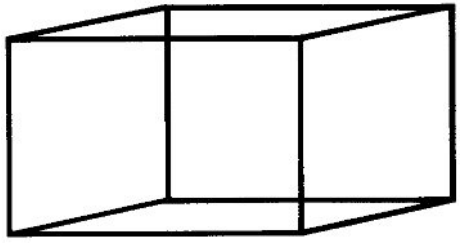
```
50 FOR J = 1 TO 200
60 BOX 1,100,80,240,  120,J*36O/2oo
```

EXPERIMENT 8·1

(a) Write a program to fill the screen with a "brick wall" like the one shown below. Use the BOX command to draw the wall, with each row of bricks staggered. Use the COLOR command to change the screen background color to orange or red, and the character color to black or gray for mortar.



(b) Write a program using the BOX command and DRAW commands to make a cube.



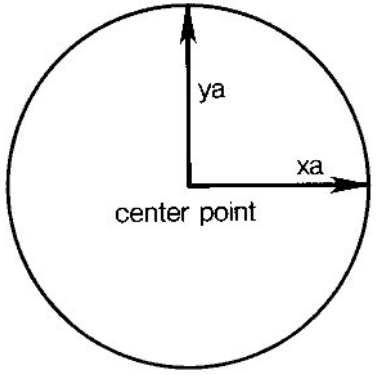
(Hint: Make two boxes, and use DRAW commands to connect the upper left corner of one box to the upper left corner of the other, upper right to upper right, and so on.)

* **Experiment 8.1 Completed** *

CIRCLE

The CIRCIE command is one of the most versatile in BASIC. We shall explain CIRCLE by giving many examples to show how it can be used to plot triangles, half-ellipses, incomplete polygons, and so on.

The basic geometric form underlying all the other shapes generated by the CIRCLE command is the ellipse. This is essentially an oval with a center, a horizontal radius (xa) and a vertical radius (yo), as shown in this drawing:



CIRCLE tokes nine parameters, as described in this list:

Parameter	Purpose	Default
First:	I-Use foreground color O-Use background color	Foreground color
Second and Third:	The screen coordinates of the center of the ellipse.	Current position
Fourth:	Horizontal radius of the ellipse	Horizontal radius
Fifth:	Vertical radius of the ellipse	0
Sixth:	Starting angle (degrees)	360
Seventh:	Ending angle (degrees)	0
Eighth:	Clockwise rotation	2
Ninth:	Step size (degrees)	

For now, we'll concern ourselves with the first five parameters, and use the default values for the lost four.

Here are some simple examples of CIRCLE commands:

```
CIRCLE 1,90,120,50,10
```

plots an ellipse in foreground color, centered at (90,120), with a horizontal radius of 50 and a vertical radius of 10.

```
CIRCLE 1",5,60
```

draws an ellipse centered at the current cursor position, and with radii of 5 and 60 units.

When the fifth parameter is left out, it defaults to the some value as the fourth parameter. The horizontal and vertical radii are now equal, and a command such as

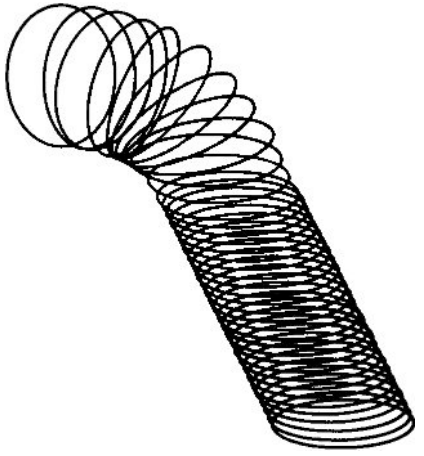
```
CIRCLE 1,150,200,40
```

actually draws a shope with horizontal and vertical radii of 40 units, centered at (150,200). This is not a perfect circle, but on ellipse. This is due to the pixels being slightly wider than toll. To draw a perfect circle, a slight adjustment must be mode. The ratio of width to height of pixels is the aspect ratio. The aspect ratio varies, depending on your monitor, but is often approximately 5 to 4. On many monitors, a radius of 65 and a vertical radius of 50 may form a perfect circle.

When you have drawn on ellipse or circle, the "current position" is left somewhere on the circumference. It is advisable to reset the cursor position with lOCATE.

EXPERIMENT 8·2

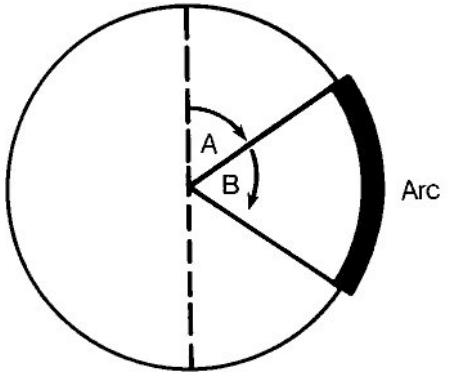
(0) Write a program to draw a picture of touching circles and ellipses, like this:



(b) lOAD and RUN the program RUG. It creates rings of ellipses that resemble a throw rug. Change the dimensions of the CIRCLEcommand to round the ellipses into evenly-rounded circles.

*** Experiment 8.2 Completed ***

The sixth and seventh parameters of CIRCLE let you draw ports or arcs of circles and ellipses.



Suppose you wont to draw the arc of the circle shown by the heavy line. Once you know the center and radius of the circles, the arc is defined by the two angles A and B. If the arc is to be drown clockwise, A is the "starting angle" and B the "ending angle." The angles are measured in degrees, like compass bearings.

To get a part-circle, all you need to do is to supply the angles A and B to the CIRCLE command as parameters six and seven. You could draw the arc pictured above by putting:

```
CIRCLE 1,100,100,50,40,45,150
```

(This assumes that angles A and Bare 45 and 150.degrees, respectively.)

The some system can be used to draw parts of ellipses. Here, for example, is a short program that draws on egg, made up of a half-circle and a half-ellipse.

```
10 GRAPHIC 1,1
20 CIRCLE 1,150,100,65,80,270,90
30 CIRCLE 1,150,100,65,50,90,270
40 GETKEY A$:GRAPHIC 0
```

If the starting and ending angles are not provided, the system tokes the default values of 0 and 360, and the entire figure is drawn.

Parameter number eight allows the whole figure (circle, ellipse or arc) to be rotated about the center. This is unnecessary for circles (since they're round, rotating has no effect), but useful for drawing ellipses with sloping diameters. The parameter specifies the clockwise rotation in degrees. For example, this command draws an ellipse with its major axis sloping at 30 degrees:

```
CIRCLE 1,150,80,100,20",30
```

A short program to illustrate rotation of an ellipse:

```
10 GRAPHIC 1.1
20 FOR X = 0 TO 180 STEP 30
30 CIRCLE 1,150,100,40,80,,,X
40 NEXT X
50 GETKEY A$:GRAPHIC 0
```

The loop in lines 20-40 resets the variable X to rotate the ellipse 30 degrees each time it is drawn.

If the eighth parameter is absent, it defaults to zero (no rotation).

The ninth and final parameter for CIRCLEcan be used to draw polygons. Normally, when the system plots a circle or ellipse, it takes positions 2 degrees apart on the circumference and joins them with short straight lines. These segments are so small that it appears to be a smooth curve.

This angle is the "step angle," and can be varied from its two-degree default. The larger the step angle, the fewer points are plotted and the more defined are the lines joining the points. A shope is formed with the number of "sides" based on the formula 360 degrees divided by the step angle. For example, a step angle of 120 forms a triangle (360/120 = 3), one of 90 draws a square (360/90 = 4), a step angle of 72 creates a pentagon (360/72 = 5), and so on.

The "step size" is supplied to CIRCLEas the ninth parameter. The command to draw a pentagon would be

```
CIRCLE 1,100,100,50"",72
```

lOAD and RUN the program TRIANGLES to see an example of how a value for the step angle of the CIRCLEcommand can be used to create other shapes, such as triangles.

The "step size" can be used with the other options to produce a large variety of shapes. For example, if the base curve is on ellipse instead of a circle, the polygon will appear flattened. If starting and finishing angles are supplied, the polygon will be open. To get the right figure, you should ensure that the total arc angle (the differ-

ence between the starting and ending angles) is an exact multiple of the step size.

A command to produce a flattened open pentagon is

```
CIRCIE l, 15, 100, l00,20,36,324,,72
```

Note that the total arc angle equals 288 (324-36), a multiple of the step angle of 72.

EXPERIMENT 8·3

(a) lOAD and RUN the program OCTAGON. This program draws two offset spiraling octagons in different colors. After seeing it, LISTthe program and change the CIRCLEcommands to create the some effect with:

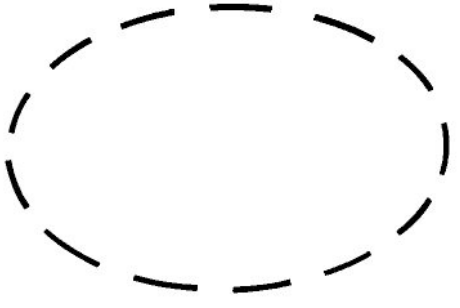
Triangles

Squares

Triskadecahedrons (13 sides)

(Hint: Change the lost parameter in all the CIRCLEstatements from 45 to Z, and odd a line defining variable Z as 360 divided by the number of sides you wont.)

(b) Write a program to draw a dashed ellipse, like this



(Hint: Draw the ellipse first, then make dashes by using the parameters to draw arcs, selecting the arc angles to produce intermittent ports of the ellipse.)

* Experiment 8.3 Completed *

PAINT

The PAINT command is used to fill complete areas of the screen with color. Unlike most other commands, PAINT is not restricted to simple geometrical figures. It can handle any shope. All that is necessary is that the area to be pointed should initially be blank (background color), and completely bordered by pixels of foreground color. For this purpose, the edge of the screen counts as a "border". It is also possible to erose the color from areas that have already been colored.

The PAINT command normally takes three parameters, like this:

```
PAINT l, 150, 100
```

First: Color Source

0-to point area with foreground color
l-to use background (default)
2-Multicolor l
3-Multicolor 2

Second and Third: the screen coordinates of any point inside the area to be painted.

To see the PAINT command in action, enter and run the following simple program:

```
10 COLOR 0,2
20 GRAPHIC 1,1
30 COLOR 1,1
40 CIRCLE 1,100, 100,50
50 PAINT 1,100,100
60 CIRCLE 0,100, loo,40
70 PAINT 0,100,100
80 GETKEY A$:GRAPHICO
```

In this program, command 40 draws a circle, and command 50 fills it in with the foreground color. line 60 now draws a circle in background color, and command 70 paints it over, leaving a ring of foreground color only.

The PAINT command is easy to use, but there are two special conditions of which you should be aware.

First, it is essential that the area being pointed be COMPLETEIY surrounded. A gop of even one pixel will let the paint leak out and flood the picture. To illustrate this, change the program in your machine by adding

```
45 DRAW 0,50,100
```

This command puts one white pixel into the circumference of the circle. When you run the program the damage will be obvious.

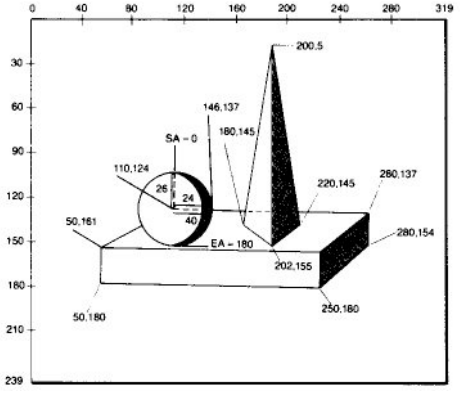
The other problem arises when two colored areas touch. The screen is divided in "cells" of 64 pixels each, and only one foreground color is allowed in each cell. When the PAINT command puts the current foreground color into any pixel, it changes the color byte and this affects all the pixels in the cell. To see this effect, RUN the following program:

```
10 COLOR 0,2
20 GRAPHIC 1,1 :COLOR 1,5
30 CIRCLE 1,150,100,70
40 CIRCLE 1, 150, 100,50
50 PAINT 1,150,60, l
60 COLOR 1,8
70 PAINT l, 150,35, 1
80 GETKEY A$:GRAPHICO
```

When PAINT is coloring complicated areas, its action is remarkable to watch. When the flood of color has to divide into two branches, the machine postpones filling one of them but comes bock when it has completed the other.

To see this effect in action, lOAD and RUN the program PAINTDEMO. This program first draws a maze in black and white and then fills ports of it in different colors. RUN the program a few times to observe the filling action of the PAINT command. Type RUN and press RETURN(even though you won't be able to see what you type) to rerun the program. When you're through with it, reset the computer or type GRAPHIC 0 and press RETURN.

This final program utilizes nearly all the graphic commands reviewed in the last two chapters. This program creates a three-dimensional design, which happened to be the symbol of the 1939 New York World's Fair, a sphere and a pyramidal tower. Sketching the design first and plotting the relevant points makes it much easier to draw in a program using graphic commands. Here is a diagram of the design, featuring the important screen coordinates:



The program to create this image, for C128, Plus/4 and C16 computers:

```
 10 GRAPHIC 1,1
 15 REM***DRAW SPHERE
 20 CIRCLE 1,110,124,40,28
 30 CIRCLE 1,11 0, 124,24,28,0, 180
 35 REM***DRAW PYRAMID
 40 LOCATE 180,145
 50 DRAW TO 200,5 to 220,145
 60 lOCATE 2oo,5:DRAW to 202, 155
 70 LOCATE 180, 145:DRAW TO
    202, 155 TO 220,145
 75 REM***DRAW BASE
 80 BOX 1,50,161,250,180
 90 DRAW 1,250,161 TO 280, 137 TO
    280,154 TO 250,180
100 DRAW I,280, 137 TO 146,137
110 DRAW 1,50,161 TO 78,140
120 DRAW 0,183,137, TO 201,137
125 REM***FILL IN SHADING
130 PAINT 1,137, 124:PAINT 1,210, 145
140 PAINT 1,270, 147:PAINT 1,202,80
1000 GETKEY A$:GRAPHIC 0
```

This lets you see how all the commands can be integrated to create one cohesive deSign, and how plotting the points first makes writing graphic programs easier. You can change the colors in line 5 for the combination that looks best on your computer.

EXPERIMENT 8·4

(0) Add a single line to the first BOX program (that drew a grid pattern of boxes) to PAINT the area outside of the boxes block.

(b) Write a program that draws a checkerboard pattern on the screen. Then use the PAINT command to fill in every other block with red or black, like on a checkerboard.

(c) lOAD and RUN the program HOUSE DRAWING or HOUSEDRAWING/4. Notice that the grass area is blank; PAINT the area green.

* Experiment 8.4 Completed *

UNIT 9: Graphic Topics

We have reviewed the elementary commands for creating computer graphics, as well as some of the more advanced time-saving commands. There are additional commands and concepts which you can use to place text in pictures and designs, change the screen coordinates, save and recall what is pictured on certain areas of the screen, use additional colors, and more.

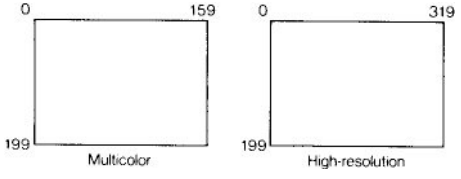
MULTICOLOR MODE

Multicolor mode is an alternative graphic mode to high-resolution. With' multicolor mode, it is much easier to draw pictures in many colors, but you give up the high-resolution sharpness of detail. You can enter multicolor mode by issuing the GRAPHIC command with the first parameter of 3 or 4. For instance

GRAPHIC 4,1

puts you in multicolor mode with a split screen, and clears the graphic screen.

A major difference between multicolor and high-resolution modes on all three computers is there are 160 pixels in each row of a multicolor, as opposed to 320 in high-resolution mode. A multicolor pixel is twice as wide as a high-resolution pixel. There are still 200 rows of pixels in a full screen display, or 160 rows if a split screen is in use.



With COLOR, you recall, the first parameter specifies color source. Values of 2 and 3 for color source refer to registers that hold a particular color for use in multicolor mode. Using COLOR in multicolor mode is like selecting colors in high-resolution, except that you can use colors assigned to source 2 and 3 in your picture. For example,

COLOR 3,9

fills the 'color3' register (multicolor l) with orange.

You are already familiar with COLOR 0, COLOR l and COLOR 4 in other contexts. You will remember that colors in the registers can be changed at any time. In DRAW, BOX, CIRCLE and PAINT, the first parameter indicates the color source to be used. This allows any of the color registers (0, l, 2 or 3) except the border to be selected for the command. So the statement

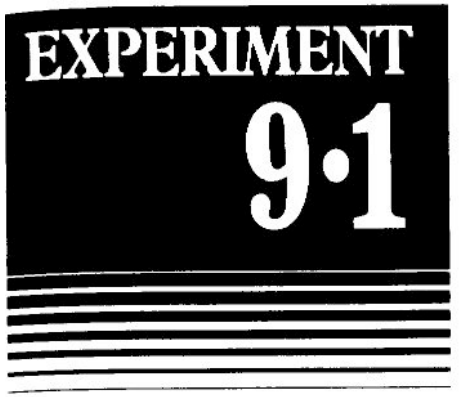
BOX 2,50,50, 100, loo" l

draws and fills in a box using the color assigned to color register 2.

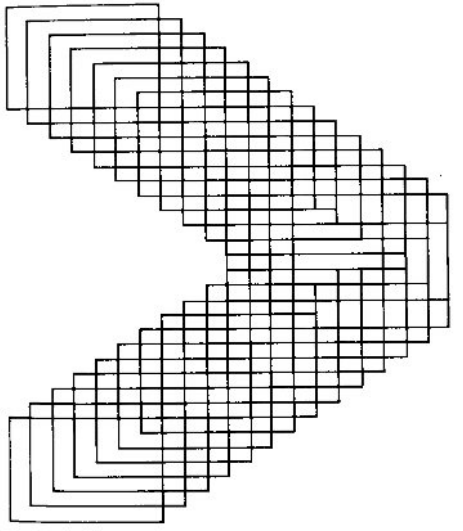
lOAD and RUN the program SWIRLS for an example of how you can use the extra colors that multicolor mode affords you.

SWIRLS creates spiral designs composed of triangles. Each side of the each triangle is a separate random color assigned to a different color source (foreground, multicolor land 2). A colorful nautilus shell pattern emerges from repetition of these triangles rotated around a moving axis, repeating the some colored sides in a spiral series of enlarging triangles. Notice the TRAP statement in line 5 (TRAP 900) that sets up a trap routine such as the ones described in Unit 3. If there is on error, the program leaves multicolor graphic mode and enters normal text mode. This routine could also include a RESUME statement transfering execution to another port of the program when on error occurs.

There is one important difference in the color registers between the C128 and the Plus/4 and C16. On the latter two computers, the two background colors, 0 and 3, are constant over the whole picture. If you draw a blue patch using color 3, and then change color 3 to point a yellow patch, the blue patch changes to yellow as well. On the other hand, the two foreground colors can vary over the screen, as long as there are not more than two different foregrounds in each color cell. To put this a different way, the two foreground colors are stored separately for each of the looo color cells, but the two background colors are only stored once for the whole picture. Multicolor mode on the C128 does not treat background colors in this fashion.



(a) Write a program that uses the BOX command and three color sources of boxes (with colors generated randomly) to form a colorful "sideways V" design across the screen, like this:



* **Experiment 9.1 Completed** *

USING TEXT WITH GRAPHICS: THE CHAR COMMAND

Many graphic screens need to feature some writing as well as pictures. This is especially important for graphs anddiagrams, but may also be necessary for artistic pictures such as backgrounds to games.

The command to put characters into a graphic screen is CHAR. The keyword is followed by four parameters:

EXAMPLE: CHAR 1,20, 10,"EXAMPIE",0

First: Color of the characters
l-current foreground color
0-background color

Second and Third: the position of the left-hand end of the string of characters to be displayed, by character row and column numbers.
These are not normal screen coordinates: CHAR assumes that the screen is divided into 40 columns and 25 rows.

Fourth: The string to be displayed.
Only character set l (capitols and graphics) can be used.

Fifth: Reverse print (background color printed on foreground)
0-normal print
l-reverse print on

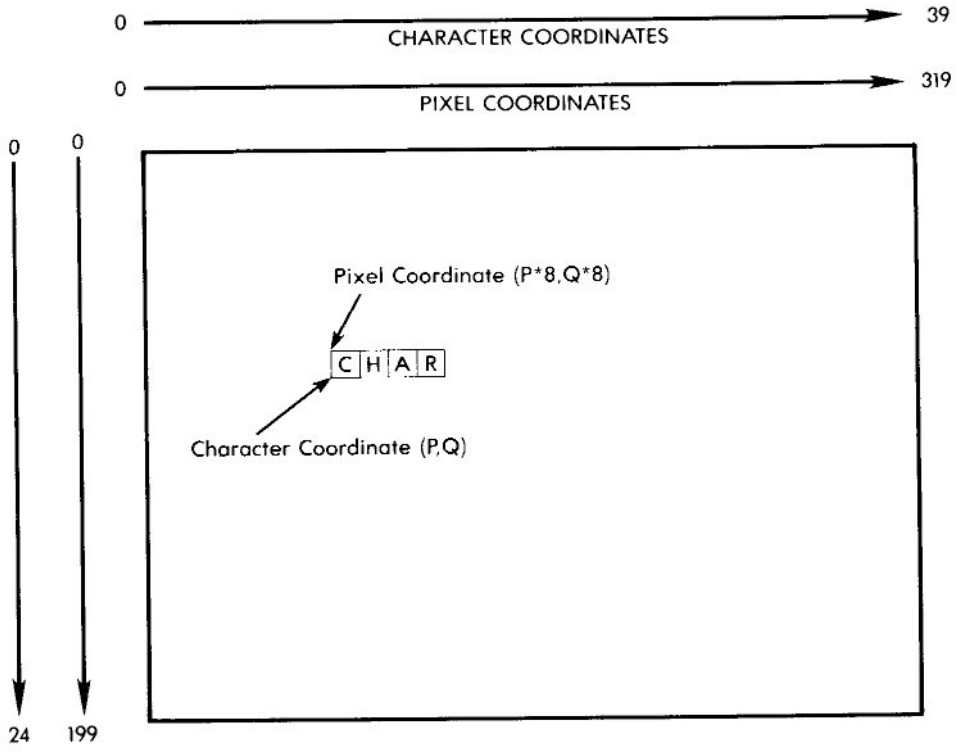
When you use CHAR, be aware of its restrictions. First, the special coordinate system limits where you can put your string of characters on the graphics screen. You are reSTricted to the some positions as are available for normal character display, unless you arrange to move the characters afterwards.

Secondly, the characters you may use are restricted; you are limited to capital letters and graphic symbols. There is therefore no way of using lower case letters, or characters you define yourself. The system uses the ASCII codes in the string to get the character definition contained in set l. Special codes like CHR$(14), which would normally switch the display to set 2, are not correctly interpreted.

The actual use of CHAR is not difficult. Here, for example, is a short program which displays a title, centered and boxed:

```
10 COLOR 0,l
20 GRAPHIC 1,l
30 COLOR 1,2
40 CHAR 1,13, 10,"INTRODUCTION"
50 CHAR 1,18,12,"TO"
60 CHAR l, 16, 14,"BASIC"
70 BOX 1,103,75,201,125
80 BOX 1,98,70,206,130
90 PAINT 1,99,71
100 GETKEY A$:GRAPHIC 0
```

Note that the first "I" of the word INTRODUCTION is in character cell (13, 10). This means that its top left-hand corner is at PIXEL(104,80). In general, the character cell coordinates must be multiplied by 8 to get pixel coordinates.



When the CHAR option is used on a multicolor screen, each character is only four pixels wide. This damages the shope of the character, but text is still easily readable.

SSHAPE AND GSHAPE

These two commands are used for "holding" complex shapes and reproducing them on the screen in various positions.

The keyword SSHAPE stands for 'Save SHAPE'. The parameters which follow define a rectangular area of the screen. The details of this area are preserved, in coded form, in a named string variable.

GSHAPE stands for 'Get Shape'. The command specifies a string and on area of store. The system assumes that the string contains a coded shope and reproduces the shape in the given position on the screen.

The parameters for SSHAPE are:

EXAMPLE: SSHAPE A$,20,30,50,50

First: The name of a string variable. This variable is used to store the shape in coded form.

Second and Third: The screen coordinates of the top left-hand corner of the area to be saved.

Fourth and Fifth: The screen coordinates of the bottom right-hand corner of the area to be saved.

Similarly, the parameters for GSHAPE are:

EXAMPLE: GSHAPE A$, l 00, 100,0

First: The name of the string which contains the coded shope.

Second and Third: The coordinates of the top left hand corner of the area where the shape is to be reproduced.

Fourth: The 'mode' of reproduction. 0 is the default, the straight-forward copying of the shape. We'll cover the other values for this parameter shortly.

SSHAPE and GSHAPE are always used together. SSHAPE makes a box around on area of the screen and holds the information for producing whatever shape is in that box. GSHAPE tokes what's in the SSHAPE box and redraws it anywhere on the screen. Here is a simple example:

```
10 COLOR 0,2
20 GRAPHIC 1,l
30 COLOR 1,1
40 FOR J = 0 TO 5: REM DRAW
   SHAPE
50 BOX 1,100,100,130,110,j*30
60 NEXT J
65 REM SAVE SHAPE
70 SSHAPE B$,loo,90,130,120
80 SCNCIR
85 REM PLACE SHAPE ON SCREEN
90 GSHAPE B$, 10, 10
100 GSHAPE B$,50, 150
110 GETKEY A$:GRAPHICO
```

In this program, lines 40 to 60 draw a pattern on the screen. The pattern is entirely contained within the area whose corners are at (100,90) and (130, 120).

Once the shope has been drawn, it is 'captured' by the SSHAPE command in line 70. The contents of the screen in the given rectangle are converted into a string of bytes and put into variable B$. Although this variable is still a string, it can't be treated like an ordinary string. For example, if you try to PRINT this variable after going bock to the character mode, you get a jumble of apparently random characters.

Once the shape has been preserved, it can be drown anywhere on the screen. Commands 90 and 100 give two possible positions.

You can draw and redraw this shope wherever and as often as you like on the screen. If you change the end of the program to read

```
90 FOR X = 10 TO 300 STEP 30
loo FOR Y = 10 TO 150 STEP 45
110 GSHAPE B$,X,Y
120 NEXT Y,X
130 GETKEY A$:GRAPHICO
```

you'll find the entire screen nearly filled with the shope.

When you use GSHAPE and SSHAPE, be aware of the following. First, all strings in CBM BASIC are restricted to a maximum of 255 bytes. When a string is used to preserve a shape, each byte holds 8 pixels, as well as certain other overheads as well. This limits the size of shope you can store to about 2000 pixels. For example, a string could just hold on area of 50*40 points.

Secondly, the SSHAPE command does not preserve any color information. You can reproduce the shape in any color you like: it depends on the foreground color selected when GSHAPE is obeyed. In the multicolor mode, the SSHAPE and GSHAPE commands work similarly. The computer stores only the color source assigned to draw each part of the shape, but not the actual color that was in the color register. When the area is reproduced with GSHAPE, the colors used to recreate the shapes are the ones currently in the color registers when GSHAPE is issued. They can be entirely different from the ones when the shape was stored.

Thirdly, there is no simple way to put in a shope as a 'predefined constant' or as a set of data values. Every shope must actually be drawn on the screen and then stored.

The fourth parameter in GSHAPE (mode) takes any of the five values 0 to 4.

0 means that the shape is to be drown into a cleared area; any port of the picture already there is destroyed.

l is the same as 0, except that the shope is drown with background and foreground interchanged-that is, like a photographic negative.

2 causes the shope to be 'OR'ed with the existing picture. Every foreground pixel in the shope is copied to the screen (overwriting what's there), but the other pixels already on the screen are left as they are.

3 makes the machine 'AND' the shape with the existing picture. Every 'background' pixel in the shape is copied (as a background pixel) but the other pixels are left unchanged.

4 specifies on 'EXCLUSIVE OR' operation between the pixels of the shape and of the existing picture. The effect changes the existing screen pixel (from foreground to background or vice versa) wherever the shape has a 'foreground' pixe1.

Option 4 is one of the most useful. If you use it to place a shope on to a blank screen, the shape appears in its normal colors. However, if you put the shope into on area of foreground, it comes up inverted. In either case, if you put the shope into the some position a second time, again using option 4, it will disappear. This works correctly no matter how complex the background, so you can use option 4 to move a shope across an existing picture without leaving a trail of destruction.

Enter and RUN this program to see how option 4 works on different color fields:

```
10 GRAPHIC 4,1
20 COLOR 0,1 :COLOR 1,7:COLOR
   2,2:COLOR 3,3
30 FOR A = 1 TO 45: REM DRAW
   TRICOlORED RING
40 BOX 1,30,30,45,45,A
50 BOX 2,30,30,45,45, (22.5 + A)
60 BOX 3,30,30,45,45,(45 + A)
70 NEXT A
80 SSHAPE A$,20,20,50,50:REM SAVE
   RING SHAPE
85 REM DRAW COLORED BOXES
90 BOX 1,50,20,80,50,,1
100 BOX 2,80,20,110,50" l
110 BOX 3,110,20,140,50" 1
115 REM REDRAW SHAPE ON
   COLORED AREAS
120 GSHAPE A$,62,20,4
130 GSHAPE A$,92,20,4
```

Notice how the ports of each ring that are the some color as the pointed area they are located on change color to the background color of block, and the other colors are reversed from the original as well.

Add the following lines and watch what happens:

```
140 FOR DE = 1T02oo:NEXT DE:REM
    DELAY LOOP
150 GSHAPE A$,62,20,4
160 GSHAPE A$,92,20,4
```

Do you understand why that happened? The EXCLUSIVEOR first put the "opposite" colors on every background the first time you use GSHAPE in that place. The second GSHAPE causes the opposite colors from the shope to be placed in that location, which is like recoloring with the original colors. You can experiment with GSHAPE by changing the last parameter value to l, 2 and 3, and placing the shape on the different colored areas.

EXPERIMENT 9·2

(0) Without changing the lost program, odd one line so the tricolored ring is redrawn in purple, green and yellow.

(b) Using CHAR, GSHAPE and SSHAPE, work out a method of putting text at any pixel position on the screen. Build your system into a subroutine which takes the following parameters:

Al$: The string to be displayed
Xl,Yl: The display position (top left-hand) corner of the first character)

Devise and run a simple test program for your subroutine.
(Hint: Use CHAR to put the string at a fixed position. Then capture it with SSHAPE and use GSHAPE to put a copy where you want it. Remember to remove the string as written by CHAR.)

Assume the string is not more than 20 characters long.

* Experiment 9.2 Completed *

SCALE

The SCALE command changes the way the machine interprets screen coordinates. If you turn on scaling by the command

SCALE I

then both the X and Y dimensions of the screen are token to run from 0 to 1023. The scaled coordinates of a point in the middle of the screen are (511,511).

For the C128, you can add parameters to change the default scaling of 1023. A second parameter specifies the maximum value for X and a third defines the scaled coordinate for Y. This number can be as high as 32676 for both X and Y coordinates, so a possible command might be

SCALE 1,255,200

In practice, a scaled picture is still drown as a collection of pixels, with the number per row or column depending on the GRAPHIC mode. Since there are always more scaled positions than pixels, several scale points 'mop' to the some screen position. For example, the scaled point (511,511) is indistinguishable from (51O,510).

It's easy to see the effect SCALE has on the screen coordinates. The following program draws a triangle inside a rectangle with normal (unscaled) screen coordinate values, and then it repeats the somes commands with scaled coordinates. The difference is apparent.

```
10 GRAPHIC 1,l
20 FOR A =0 TO 1
30 SCALE A
40 BOX 1,10,10,310,190
50 CIRCLE 1,16O,100,60"",120
60 NEXT A
70 GETKEY A$:GRAPHIC 0
```

If you change line 30 to read

```
30 SCALE A, 32000,32000
```

you'll see an even more striking difference.

Scaling is useful for some mathematical applications. It can be turned off by writing

SCALE 0

You can see how SCALE can be used to create an interesting effect with changing sizes in a program by looking at SWIRLS again.

The Plus/4 version of the SCALE command is similar to the C128, except you can only use the first parameter to turn scaling on or off (lor 0). The scaling coordinates automatically default to values of O-l 023 for both X and Y coordinates.

EXPERIMENT 9·3

(a) lOAD and RUN the program SCALESor SCAIES/4. Change the second and third parameters for each SCALE command to see how that affects each row of rings.

Note that because of the SCALE command form on the PIUS/4 and C16, you cannot change the parameters in this experiment.

(b) lOAD and RUN the program QUIZ9 or QUIZ9/4, which contains questions about graphics based on the material in Units 7,8 and 9.

* Experiment 9.3 Completed *

UNIT 10:
Sound and Music

This unit describes the BASIC commands that let you program music on the Plus/4, C16, and C128. Although these computers use similar commands, the parameters are different because they have different sound chips. This unit is broken down into two parts: Sound and Music on the Plus/4 and C16, and Sound and Music on the C128.

SOUND AND MUSIC ON THE PLUS/4 AND C16

The Plus/4 and C16 have one chip responsible for sound and video production. load the program DEMO SOUND, turn up the volume on your TV or monitor and play through the selection of sound effects. This gives you some idea of what the TEDchip does in the way of sound production. Your computer also creates actual melodies.

Sound production on the Plus/4 and C16 is controlled by two commands, VOL and SOUND. Combinations of these two commands can create a full range of musical notes and an infinite variety of sound effects.

VOL

VOL command has one parameter, which ranges from 0 to 8.

The command is, in effect, a volume control. The sounds are at their loudest when you have declared:

```
VOL 8
```

and reduced to silence with:

```
VOL 0
```

Intermediate degrees of volume are selected by other VOL values.

When you run a program containing sound, you may also have to turn up the volume control on your TV or monitor.

SOUND

The command used to produce an actual tone is SOUND. Each SOUND command must be followed by three parameters:

First: Voice

- 1-musical tones
- 2-musical tones
- 3-sound effects (white noise)

Note: Voice 3 is not truly an independent voice, but rather Voice 2's ability to create noise as well as tones. In addition, noise is white, that is, not unpleasant, at certain frequencies (600-900).

Second: Sound register (pitch), which determines how high or low the sound will be

- This value must be in the range 0-1023. Certain values will generate pitches not usually audible.

The higher the value, the higher the pitch. The following short program plays all the possible pitches:

```
10 VOL 7
20 FOR J =0 to 1023
30 SOUND1 ,J, 1
40 NEXT J
```

Third: duration, how long the sound

- This value must be in the range 1 65535. Duration is measured in jiffies, or l/60ths of a second.

For example, a note of 60 jiffies will last one second, and a note of 30 jiffies lasts half a second.

Try these SOUND examples; remember to NEW the computer's memory after each trial.

1) Machine Gun

```
10 VOL 7
20 FOR L= 1 TO 30
30 SOUND3,920,4
40 FOR C = 1 TO 5O:NEXT C
50 NEXT L
60 END
```

Notice Voice 3 in a high register simulates the rapid fire of a machine gun. line 40 is an empty loop to take up space between bullets.

2) Steam Engine

```
10 VOL 7
20 FOR 0 = 20 TO 6 STEP -1
30 FOR P= 1 TO 4
40 SOUND3,850,D
50 SOUND1, 1021,0*2
60 NEXT P
70 NEXT 0
80 FOR X = 1 TO 60
90 SOUND3,850,D
100 SOUND1, 1021,0*2
110 NEXT X
120 END
```

A steam engine is simulated by steadily decreasing the duration of a Voice 3 sound and repeating the sound in rhythmic pulses. In line 50, the sound register 1021 in Voice 1 is not audible separates the sounds. By doubling the duration for the separation command, making the inaudible sound twice as much long as the audible sound, we accent the pulsing of the wheels.

3) Foghorn

```
10 VOL 7
20 FOR X = 1 TO 3
30 SOUND1 ,5,90
40 SOUND3,1012,90
50 FOR J = 1 TO 3000:NEXT
60 NEXT X
```

The foghorn uses a combination of two sounds: a low tone in Voice 1 and a buzzing sound in Voice 3. Many combinations are possible.

CREATING SOUND EFFECTS

Developing sound effects on the computer involves a combination of skill, good luck. There is no formula for reproducing a sound such as a gunshot or a birdcall. There are, however, guidelines you can follow so that your sound effect experiments are more than just random button-pushing.

The task can be broken down into several parts:

1) What voice should be used?

Is the sound a combination of musical tones? Or does it have a buzzing or brushy sound best suited to Voice 3? In some cases a combination of voices will enable you to create recognizable sound effects. Hint: a surprising variety of sounds are created from musical tones.

2) Is the sound steady, or does it fall or rise?

If the sound changes drastically in pitch, you may need a series of SOUND commands. Consider using a FOR-NEXT loop to control the changing pitch if the change is regular.

3) Does the sound have a regular or random pattern?

Once you have determined the qualities of the sound, begin to experiment. Set up a simple program that creates a sound which rises, or falls, or follows the pattern you have determined. Plug in different values for voice, sound register and duration until you arrive at the sound you want.

An interesting way to hear the variety of sounds the Plus/4 and C16 can create is to use random values for frequency and duration. The RND command can be used to define the sound register and duration.

Try this program:

```
10 VOL7
20 FOR J = 1 TO 10
30 S = ((RND(0)*1023) + 1)
40 0 = ((RND(0)*6O) + 1)
50 SOUND1 ,S,D
60 NEXT J
```

This program generates a series of varied tones. Notice that line 50 indicates all the sounds should be produced by Voice 1. If we add another RND statement to allow the computer to use any voice, we should a greater range of sound. Add:

```
45 V = ((RND(0)*3) + 1)
50 SOUND V,S,D
```

When you run the program with the added line, notice several types of sound. In some cases the sounds overlap. This should give you more ideas for the variety of effects possible.

EXPERIMENT 10·1

a) Write a program that simulates the sound of a rocket blasting off.

- The engines warm up while a beep counts down ten seconds. Then all the engines fire and the rocket takes off. The sound of the engine dies off as it moves further and further from the earth.

b) Write a program to respond audibly to quiz questions. One portion of the program will create a bright beeping if a correct answer is given. The rest of the program responds with a brash noise to an incorrect answer.

- Experiment with different sounds for the responses.
- Incorporate the program as a subroutine within the TRIVIA program from Unit6.

*** Experiment 10.1 Completed ***

CREATING MUSIC

Use the SOUND command to make your computer play tunes. Each note of a melody can be played by its own SOUND command. Try the following program, which plays the first line of a familiar tune:

```
10 VOL7
20 FOR X = 1 TO 3
30 SOUND1 ,739,45:S0UND1,  1023,2
40 NEXT X
50 SOUND1,770,15
60 SOUND1,798,45
```

This method of creating music would rapidly get out of hand; a song of average length might take hundreds of SOUND commands to reproduce. luckily, there are other ways to supply the computer with the appropriate values for the SOUND commands.

READ/DATA statements work very well for programming music. We enter the values for each note, both sound register and duration; we set up a loop to read the data for one note; play that note; and loop back to read the values for the next note.

To separate each note, place a tone in an inaudible sound register between each note in the song. A duration of two jiffies is usually sufficient to provide a gap between the notes. This is especially useful in songs which have notes that repeat, such as the example above. Place these two commands at the end of that program:

```
70 SOUND 1,400,30
80 SOUND 1,400,30
```

Run the program. Instead of two short notes, the computer plays one long one. That's because the computer finishes one note and immediately plays the second. The two commands produce exactly the same result as:

```
SOUND 1,400,60
```

because the two notes run together without any gap. Add :SOUND1, 1023,2 to line 70 and the problem is solved.

load and run the program "Swansong" from your program disk. When you list the program, you'll see a six-line program, then line after line of data scroll on the screen. The program's structure is simple:

```
10 VOL 7
20 READ X,Y
25 IF X=O  THEN END
30 SOUND1 ,X,Y
35 SOUND1, 1020,2
40 GOTO 20
100-210  DATA
```

- line 20 reads the data for each note
- line 25 checks for a terminator
- lines 30 and 35 play the note and an inaudible separator note
- line 40 loops back to the READ statement.

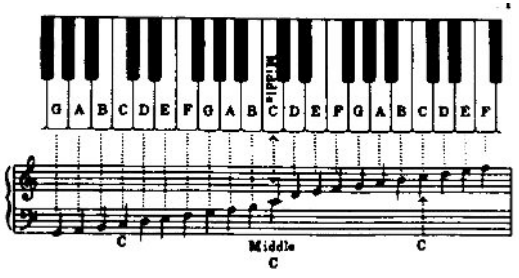
Almost any melody can be created with exactly the same program, just different DATA lines. But unless you know what values to use for the various notes in the scale, programming music is almost impossible. The chart below contains the sound registers for the notes of a chromatic scale over a range of several octaves.

Note	Sound Register	Note	Sound Register
Octave 1:		Octave 3:	
C	169	C	810
C#/Db	217	C#/Db	822
D	262	D	834
D#/Eb	305	D#/Eb	844
E	345	E	854
F	383	F	864
F#/Gb	419	F#/Gb	873
G	453	G	881
G#/Ab	495	G#/Ab	890
A	516	A	897
AH/Bb	544	AH/Bb	904
B	571	B	911
Octave 2:		Octave 4:	
C	596	C	917
C#/Db	621	C#/Db	923
D	643	D	929
D#/Eb	665	D#/Eb	934
E	685	E	939
F	704	F	944
F#/Gb	722	F#/Gb	948
G	739	G	953
G#/Ab	755	G#/Ab	957
A	770	A	960
A#/Bb	784	A#/Bb	964
B	798	B	967

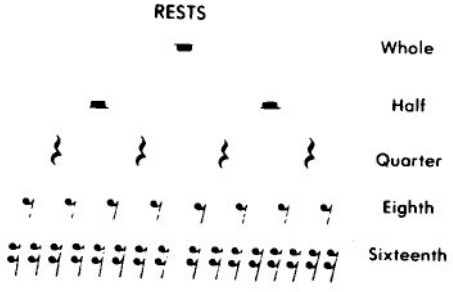
Sound Registers on the Plus/4 and C16

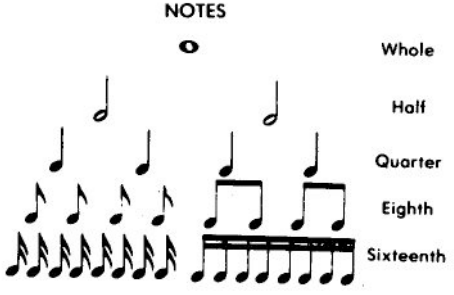
MUSICAL THEORY

Before you can begin to program music on your own, you need to have some idea of the relation of the letter names of the notes to written music. Here is a diagram showing three octaves of notes, with their corresponding letter names.



In music, the whole note is the basic measure, whether it is subdivided into half, quarter, eighth or sixteenth notes. For practical purposes, however, the quarter note is generally used as the standard for the tempo, that is the "speed",of the music. Four beats equal one whole note. Here are some important musical symbols and their time values. On the right side of the chart are the symbols for notes, the moments when sounds are made. On the left are rests, or breaks between musical sounds.





To translate a melody from written music to code your computer can understand, break the task into several steps:

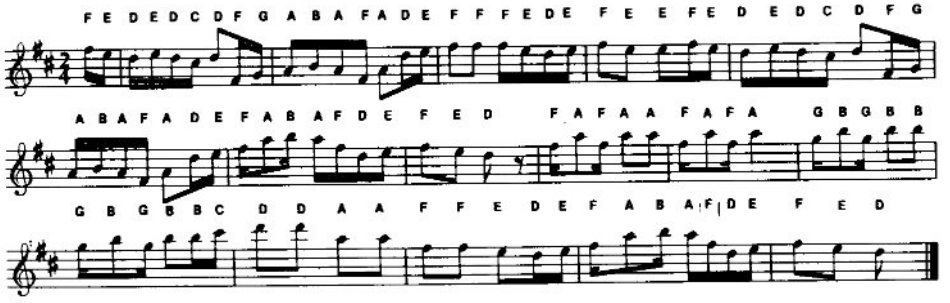
1) Write down the sound register value for each note. It may be helpful to copy the music onto a piece of paper, replacing the note symbols with their corresponding letters. Refer to the list of sound registers to locate the value for each note.
2) Determine the duration of the quarter note. In slow music try 60 or 30 jiffies as the quarter note value. If there are sixteenth notes, try 64 or 16 as the quarter note (making it easier to subdivide into sixteenth notes). Write down the duration next to each note.
3) Copy the musical program (with the READ-DATA statements). Replace DATA lines with your own data:

   Start with the first note in the song. Type its sound register, and then its duration. Then type the values for the second note, and so on. Remember- separate each piece of data with a comma.
4) Continue until the entire song is coded in DATA statements. It is helpful to place each measure of the song in one DATA line followed by a REM statement such as :REM MEASURE ONE. Perhaps, divide the DATA into musical phrases. It isn't necessary to divide the DATA; you are limited only by program line length (160 characters).

EXPERIMENT 10·2

Create a program to play the following piece of music:

Turkey In the Straw



- All F's and C's are sharp-thisis signified by the # symbols at the left of each staff, called the key signature.
- Use 10 as the eighth note length.

* Experiment 10.2 Completed *

SOUND AND MUSIC ON THE C128

The Commodore 128 is capable of producing sophisticated sound effects or complex musical compositions. It can produce three independent voices (sounds) simultaneously, giving you a great deal of creative freedom in sound production.

There are two ways to produce sound on the C128, each with a separate set of commands. Either method can be used to produce sound effects or music, but one is more suited to simpler sound effects and the other method lends itself to musical composition.

SOUND EFFECTS

Production of sound effects can be accomplished with only two commands: SOUND and VOl.

VOL

VOL is a volume control. Unless you set the volume level, the sounds you create may not be audible. VOL has only one parameter, volume level, which must be in the range 0 to 8. VOL 0 turns off the sound and VOL 8 is the loudest level. You may need to turn up the volume on your monitor or TV set in order to hear the sound effects you are creating.

SOUND

A typical SOUND command for the C128 might look like this:

```
SOUND1 ,2000,60,1,300,20,1
```

The SOUND command has eight parameters, the last five of which are optional.

```
SOUND vc,freq,dur,dir,min,svwf,pw
```

First: VoiCe-1, 2 or 3

Second: FREQuency-in the range 0-65535

Third: DURation-in 60ths of a second, called jiffies

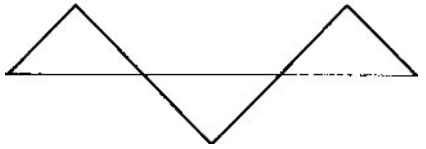
Fourth: DIRection-set the DIRection, or "sweep", in which the sound in incremented/decremented

0-increment the frequency upwards
1-decrement the frequency downward
2-oscillate the frequency up and down

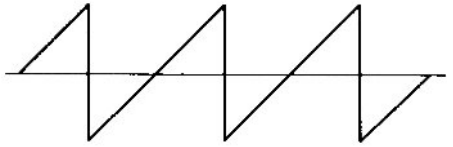
Fifth: MINimum frequency (in the range 0-65535) if the sweep (DIR) is selected

Sixth: Step Value for the sweep (in the range 0-32767)

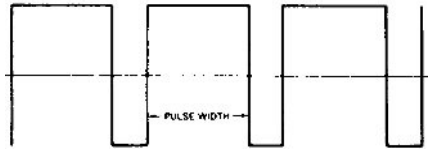
Seventh: Wave Form (0-3)
0-Triangle
1-Sawtooth
2-Variable Pulse
3-White Noise



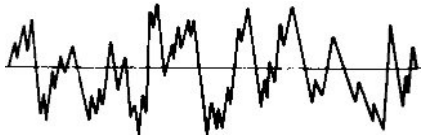
TRIANGLE



SAWTOOTH



VARIABLE PULSE



NOISE

It is only necessary to understand that triangle, sawtooth and variable pulse waveforms produce similar but distinguishable sounds, while the white noise waveform produces non-musical noise.

Eighth: Pulse Width, the width of the variable pulse wave form, in the range 0-4095

Only the first three parameters-Voice, Frequency, and Duration-are necessary in order to create a sound. Try this program:

```
10 VOL 5
20 SOUND1, 10000,60
```

When you run the program, the 128 plays a pitch in voice 1 at frequency 10000 for 60 jiffies (one second).

Change the frequency value in line 20 to 40000. When you run the program again, the computer plays a much higher pitch. In general, the greater the frequency number, the higher the pitch.

With the three necessary parameters, only combinations of musical tones can be created. But as you experiment with sound on the C128, see how many sound effects are accomplished by skillful manipulation of musical tones. Try the following:

```
10 VOL 5
20 FOR X = 1500 TO 300 STEP -20
30 SOUND1 ,X, 1
40 NEXT X
```

The program creates a groaning sound. line 10 sets the volume level. lines 20 to 40 form a FOR/NEXT loop that plays a pitch for a fraction of a second, then decreases the frequency slightly and plays another pitch. Because the change in frequency between notes is minimal, each individual note is not distinguishable.

The C128 creates a similar effect without the FOR/NEXT loop if you use all the parameters in the SOUND command. Add this line to your program:

```
50 SOUND1 ,2000,60,1,300.20,1
```

The program plays two similar sounds, but each is created in a different way. line 50 tells the computer to playa pitch invoice 1, starting at frequency 2000. The "sweep" or direction of the sound is downward (DIR = 1) and the sweep continues downward by increments of 20 until a frequency of 300 is reached. The triangle waveform is chosen.

This should give you some ideas about how powerful the SOUND statement is. You can eliminate loops to alter the pitch of a sound, because the DIR parameter handles that. You can alter the waveform (see MUSIC on the C128) to create different sounds. Try changing the 1 at the end of line 50 (for sawtooth waveform) to a 3. Depending on the frequency, waveform 3 creates a brushy or rumbling noise. Run the program with waveform 3. It sounds something like an airplane taking off.

MAKESOUND, on your program disk, lets you input values for all the possible SOUND parameters. Use values in the range for that parameter. Experiment with different waveforms and try other frequency values. The program runs until you hit RUN/STOP or reset the computer.

Notice that when declaring an upward sweep, the selected frequency is the high point of the sweep. The starting point of the sweep is contained in the MIN parameter. With a downward sweep, a negative step value is unnecessary, because you declare the sweep is to go downwards.

C128 ONLY

EXPERIMENT 10·3

Write a program that generates random sounds, using the VOL and SOUND commands.

- Be careful in constructing the random statements to multiply the seed value for the random number by the appropriate value for each parameter. Notice that 1 should be added to some, but not all, of the generated numbers in order to include the proper range of values.
- The screen should be outlined in asterisks, then use a window to scroll the values of the various parameters, which should appear in random colors.
- Allow each sound to continue until the user presses a key.

* Experiment 10.3 Completed *

PROGRAMMING MUSIC

You have learned how to create sound with VOl and SOUND. BASIC 7.0 has a set of commands that makes programming music easier than ever, and gives you more precise control over the sound you create. Tell the computer to PLAYa note (calling the note by its letter name) and the SID chip creates the sound. Anyone who reads music at all can program the C128 to produce recognizable tunes.

Sound is made up of vibrating air waves. These waves oscillate (move) at a certain rate (frequency). The faster the frequency, the higher the pitch. Two notes of the same frequency, played on different instruments, will not sound the same. This is because each sound has a number of different characteristics. A unique combination of these various characteristics creates the differing nature of sounds, called timbre. For example, a trumpet and a clarinet can play "middle C" but because of their different timbres, the sound will not be exactly alike.

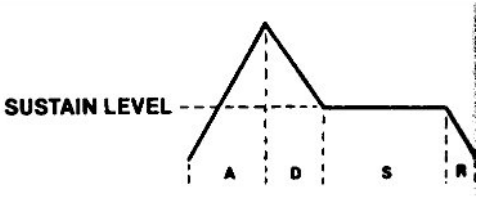
The C128 has a command, ENVELOPE, which lets you simulate sounds with different timbres. Once the ENVELOPEis set, PLAY activates the SID chip and sound is produced. You can also easily set the tempo of the music you are programming.

ENVELOPE

A note changes in volume several times during its duration, and these changes in volume are called attack, decay, sustain, and release (ADSR):

Attack	the rate at which a note reaches its peak volume
Decay	the rate at which a note decreases from its peak to its midranged (sustain) level
Sustain	the mid-ranged volume level
Release	the rate at which a note decreases from its sustain level to its zero level

Here is a diagram of ADSR.



C128 ONLY

The C128 can alter each of the four parameters of the ADSR, giving you control over the properties of the sound. In addition, the C128 has 10 preprogrammed ADSR settings, contained in the ENVELOPE,that allow you to easily recreate the sound of particular instruments.

A typical form of ENVElOPE looks like this:

```
ENVELOPE 1,5,3,6,6,1,200
```

The ENVELOPEcommand has seven parameters. Use a comma for default parameters or specify new values.

```
ENVELOPE e,a,d,s,r,wf,pw
```

First: Envelope number, in the range 0 to 9

Envelope Number	Instrument	Attack	Decay	Sustain	Release	Waveform	Width
0	Piano	0	9	0	0	2	1536
1	Accordion	12	0	12	0	1	
2	Calliope	0	0	25	0	0	
3	Drum	0	5	5	0	3	
4	Flute	9	4	4	0	0	
5	Guitar	0	9	2	1	1	
6	Harpsichord	0	9	0	0	2	512
7	Organ	0	9	9	0	2	2048
8	Trumpet	8	9	4	1	2	512
9	Xylophone	0	9	0	0	0	

Second through fifth: ADSR settings

- attack rate, in the range 0 to 15
- decay rate, in the range 0 to 15
- sustain level, in the range 0 to 15
- release rate, in the range 0 to 15

Sixth: Waveform, the type of wave created by the variety of harmonics of a note.

0-triangle
1-sawtooth
2-pulse (square)
3-noise
4-ring modulation

Seventh: Pulse width (for the pulse, or square wave form), in the range 0 to 4095

The pulse widths that create the most solid sounds fall in the middle of the range. At both the high and low extremes of the range the notes sound thin and hollow.

ENVELOPE is the command which controls the qualities of the sound being created. When you begin to program music on the C128, use the predefined envelopes. But as you experiment, take advantage of the power of ENVELOPE to create exactly the sound you want.

TEMPO

Once you define the nature of the sound, TEMPO controls the speed of the tune. TEMPO has only one parameter, which falls in the range 0 to 255, where 255 is the fastest tempo. The default value is 8.

PLAY

Without PLAY, ENVELOPE and TEMPO are useless. PLAY lets you describe what notes to play. The PLAY command has a number of parameters, but only one is essential: the musical note.

Try this command:

```
PLAY"CDEFGAB
```

Don't forget the quote mark. The keyword PLAY is always followed by a string of synthesizer control characters and musical notes. This command causes the computer to playa portion of a musical scale. Default values for all the other parameters were used.

Now add another C to the end of the string. If you are familiar with music, you might expect the command to play the second C one octave higher than the first, to complete the scale. But since you haven't told the computer which octave to use, all the notes are in the default octave. In order to gai n more control over the PLAY statement use all the parameters.

PLAY has two types of parameters: parameters relating to the notes themselves, and synthesizer control parameters. Each parameter value is preceded by a letter code; the order of the parameters has no effect.

NOTE PARAMETERS

Duration-you can determine the relative length of each note to the others.

Place one of the following characters before the letter name of the note:

W	whole note (default)
H	half note
Q	quarter note
I	eighth note
S	sixteen note

For an introduction to written music and musical symbols, see the section of this unit entitled Musical Theory (in the Plus/4 and C16 portion of the unit).

Playa rest (beat of silence) with R instead of a note, and place the appropriate duration letter before the R. For example, change the PLAY command above to:

```
PLAY "Q CDEFGAB WR Q CDEFGAB
```

This command plays quarter notes up the scale, rests for the duration of a whole note, then plays the quarter note scale again. Notice that you define the duration once, and it remains until you change it. In the example above, the quarter note duration is used for the first scale, but is then redefined as whole note. To make the second scale sound like the first, place a Q before the first note in the second scale.

Spaces in your PLAY command do not affect the sound; plenty of spaces add readability.

Create dotted notes, notes of one and one-half times the usual duration, by placing a period (.) before the letter name of the note. If you want to switch from quarter notes to a dotted half note (equal to three quarter notes) try:

```
PLAY "QC .HD QEFGAB
```

Without discussing any other parameters, you can create simple tunes with the PLAY statement and duration parameter:

```
10 TEMPO 15
20 PLAY "Q CDEC QCDEC
30 PLAY "Q EF HG Q EF HG
40 PLAY "IGAGF QEC IGAGF QEC
50 PlAY"Q CC HC Q CC HC
```

line 50 of this little program doesn't sound exactly the way you expected it to. This is because we are still working within the default octave. To sound the way we expect that musical phrase to sound, we must change the octave.

Create sharp or flat notes by placing # for sharp or b for flat before the letter name of the note. These "accidentals", as they are called in standard music notation, only take effect for the note immediately following the # or b.

SYNTHESIZER PARAMETERS

There are five sound synthesizer parameters available for the PLAY statement. The characters to type and the possible values for each parameter are listed below.

SOUND SYNTHESIZER PARAMETERS

Control Code	Parameter Range	Description	Default Setting	
V	1-3	Voice	1	You are already familiar with the C128's three independent voices.
O	0-6	Octave	4	The C128 has a range of 7 octaves, about the same as a piano
T	0-9	Envelope	0	(It might help you to think of this as tone, or timbre). If you want to use any of the predefined envelopes, place Tn (where n is the number of the envelope you desire) in your PLAY statement. If you refine the sound with an ENVELOPE statement, however, be sure to use the "new" envelope's number in the T parameter.
U	0-15	Volume	9	The volume parameter controls the volume note by note, if desired, rather than playing an entire song at one volume level.
X	0-on 1-off	Filter	0	Filters are used to further refine the sound created by PLAY statements. There are several types of filter, to be discussed later. For now, we'll leave the filter control off in all our PLAY statements.

These synthesizer control characters can appear in any order, but for the best results place the characters in PLAY statements in the order shown above. Knowing all this about the synthesizer controls, modify line 50 in the program that played "Frere Jacques"

```
50 PLAY "QC 03G 04HC QC 03G
    04HC
```

Notice you have to revert back to octave 4 ofter changing to octave 3 for the low note.

Place the spaces in the PLAY strings wherever it seems most logical to you. Without spaces these strings are difficult to interpret.

USING SYNTHESIZER CONTROLS

Much of music programming on the C128 is a matter of personal preference. The only way to hear the variety of sound the C128 produces is to experiment. Begin by adding an envelope (Tn) parameter to each of the PLAY statements in Frere Jacques. You have already heard what the tune sounds like in the default (piano) envelope. Place T4 in line 20:

```
20 PIAY"'T 4 Q CDEC CDEC"'
```

When you run the program, it plays the song in the flute envelope, which sounds very different from the piano. Change the T4to T8, and run the program again. This is the trumpet envelope. Try all the pre-set envelopes (see Figure 10-9). You may have to reset the SID chip by pressing RUN/STOP and RESTORE between each experiment. Notice that with some envelopes each note is distinct, while other envelopes run the sounds together.

So far you have used only the default envelopes. By placing an ENVELOPE command in your program you can change the envelopes slightly or drastically to create a different effect.

Add this line to your program:

```
15 ENVELOPE8,3, 15,6, 1,2,512
```

You have redefined the trumpet envelope with a shorter attack, longer decay and sustain, and kept the same release, waveform and pulse width. Change the T4 in line 20 to T8 and try the program now. Then change line 15to:

```
15 ENVELOPE8,3, 15,6, 1,0
```

This time you have changed the square waveform to a triangle waveform. When you run the program the sound is very smooth compared to previous trials.

You can adjust the volume within your sound programs. Add U2 immediately after the quote in line 20, and U15 after the quote in line 40:

```
20 PLAY"' U2 T8      .
40 PIAY"U 15         .
```

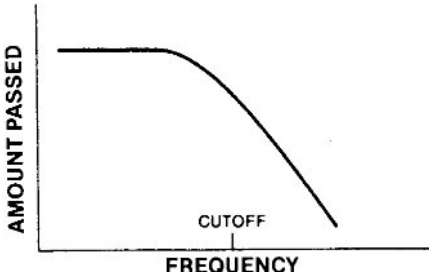
To hear the effect of the volume control, change the waveform in the ENVELOPE command back to 2 (square). Continue to experiment with various ENVElOPEs and other synthesizer controls. Remember, RUN/STOP and RESTORE resets the SID chip each time you change a parameter.

FILTERING

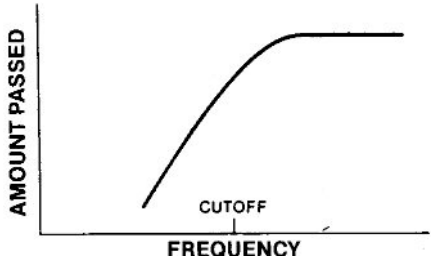
You have experimented with all the synthesizer and note parameters except one, Xn, the filter command. Set the filter with a separate command, then turn it on within a PLAY statement.

The SID chip is capable of three kinds of filtering:

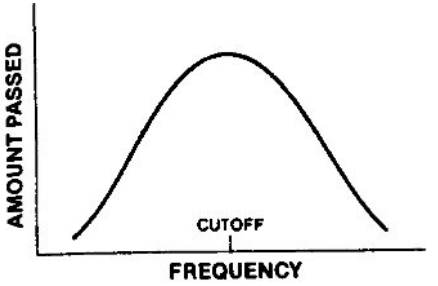
low-pass filters filter out sounds above a cutoff frequency you specify. The sounds are not eliminated entirely, but their volume level is attenuated (drastically reduced). A low-pass filter creates solid sounds.



High-pass fi lters filter out sounds below the cutoff frequency. This filter creates tinny, buzzing sounds.



A bandpass filter passes a range of sounds above and below the cutoff frequency and filters out all other sounds.



A typical command that defines a filter is:

```
FILTER 1005,0,0,1,9
```

Here is the syntax of FILTER:

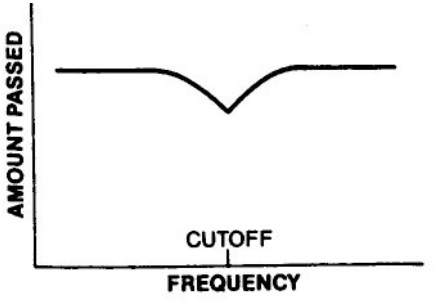
```
FILTER cf,lp,bp,hp,res
```

First:	Cutoff frequency (in the range 0 to 2047)
Second:	Low-pass filter 0 = off, 1 = on
Third:	Band-pass filter 0=off, 1 =on
Fourth:	High-pass filter 0= off, 1 = on
Fifth:	Resonance (in the range 0 to 15)

The cutoff frequency values are in the same ratio as note frequencies. Note frequencies (see the SOUND command) range from 0 to 65535. Cutoff frequencies for the filter range from 0 to 2047. The high and low values are roughly equivalent. For a value in the mid-range, choose a cutoff frequency of about 1000.

Resonance affects the sharpness and clarity of a sound. A higher resonance gives sounds that reach their peak frequency more rapidly.

It is possible to activate more than one filter at once end create special effects. For example, using both low-pass and high-pass filters at once allows all but a narrow range of frequencies to pass through the SID chip. This is called a "notch-reject" filter



C128 ONLY

C128 ONLY

Add a FilTERcommand to your program:

```
11 FILTER750,1,0.0,14
```

This command sets up a low-pass filter, with cutoff frequency of 750 and a resonance of 14. The best way to understand how the cutoff frequencies relate to the notes themselves is to experiment.

Activate the filter by placing X1 (Filter/on) in line 20, following the T8. Run the program now and notice the difference between the sound with and without the filter. Try changing the filter command so it sets up a band-pass, or a high-pass filter, or a combination.

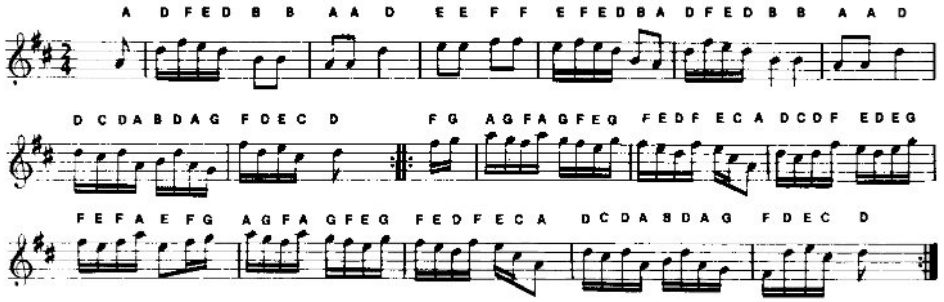
You have learned all the BASIC sound commands on the C128--but you have only begun to experiment. The best way to learn the effects of sound commands, particularly the filters, is through trial and error. Be aware of the limitations of the SID chip. Your computer creates a vast number of sound effects and musical compositions, but it cannot accurately reproduce every sound in nature.

EXPERIMENT 10·4

Write a program to play the following song. Use as many of the synthesizer control characters as you wish. Notice the vertical bars with two dots in the middle and the end of the music. These are "repeat" symbols. The first time you see the symbol, it means go back to the beginning of the song and repeat it. After the second time through, the last half of the song is played. When the end of the song (and the next repeat symbol) is reached, go back to the repeat symbol in the middle of the song and repeat from there to the end.

The answer to this program in Appendix B gives only the notes, as the other PLAY parameters are a matter of personal preference.

Arkansas Traveler



- All F's and C's are sharp. This is signified in the key signature at the left of each staff.

* Experiment 10.4 Completed *

C128 ONLY

SYNCHRONIZING VOICES

To recreate written music it is necessary to use more than one voice, so harmonies are possible. There are certain guidelines to remember in writing computer code to generate music with more than one voice. You will discover all the details as you experiment, but here is a list of some points you may find helpful.

- Use different envelopes for each voice, at least while you are starting, so you can differentiate the melody and harmony.
- Create an intermediary code: for each measure of music determine the letter names, and the durations and write them on a piece of paper. Determine if the octave changes within the measure, and if any other note or synthesizer parameters will be necessary.
- Coding two voices together is not the same as coding two separately. In order for the voices to be synchronized correctly, every note in Voice 1 must be matched by a note in Voice 2. If one voice has a long, sustained note, it should be placed in the PLAY statement first. Then code as many notes in the other voice as it takes to equal the duration of the long note.
- Play the tune at a very slow tempo first, to be sure the synchronization is correct. Then increase the tempo to an appropriate speed for the song.

EXPERIMENT 10·5

Create a program to play the following piece of music. Experiment with filters and envelopes once the synchronization is correct.

To get you started, here is what the first PLAY statement might look like:

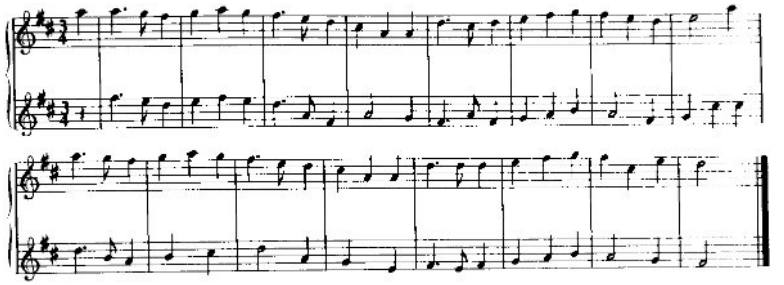
```
PLAY "V1TO X1 04 QA.QA V2T8 X1
     .Q#F V1 TO IG V2T8 IE V1TO
     Q#F V2T8 0"
```

Notice that two different voices are used. Voice 1 is in Envelope 0 (TO)and Voice 2 is in Envelope 8 (T8). The octave (4) is declared once, and remains in effect for the entire statement.

Voice 1's first notes are a quarter-note A, and a dotted quarter-note A. The second voice doesn't begin until the first quarter note in voice 1 has finished. That note is the pickup note.

Notice how it is necessary to switch back and forth between voices after each note (or notes of equivalent duration).

Streets of Laredo



- Check the key signature for sharps before you begin.

* Experiment 10.5 Completed *

UNIT 11:

C128 Sprite Commands

There are three ways to draw sprites on the C128. One is the C64 method using pokes, described in Unit 26 of *Introduction to BASIC Part 2*. There are two easier ways to create sprites, both available only in 40-column C128 mode. The second way makes use of the graphic commands to create a shape, save it (using SSHAPE) and store it as a sprite with SPRSAV. The third method is with the C128's built-in sprite editor, accessed with the SPRDEFcommand. These last two methods feature timesaving commands to make sprite creation much easier. Both ways are reviewed in this section, and we'll use them in longer programs that feature sprite movement.

DESIGNING SPRITES

To review, a sprite is a movable programmable character. They can be standard sprites, composed of foreground and background color only, or multicolor sprites, with two additional colors available. In size, standard sprites can be up to 24 X 21 pixels, while the pixels for multicolor sprites are twice as wide horizontally, so they can be only 12 X 21 pixels.

The two advanced C128 ways to design a sprite:

1) Creating a sprite within a program

2) Creating a sprite with the sprite editor:

The first method lets you use graphic commands like DRAW, BOX and PAINT to create sprites, while the sprite editor lets you draw elaborate, intricately designed sprites pixel by pixel.

DRAWING A SPRITE WITHIN A PROGRAM

There are three steps in this process:

1) Draw a shape on the screen in a program using graphic commands.

2) Save the shape with SSHAPE.

3) Make the shape a sprite with SPRSAV.

The first step is to write a program to draw the shape. Remember, since a sprite can be no larger than 24 X 21 pixels, size is a constraint in drawing a design that will be used as a sprite. Here is a program that draws a small fish.

```
10 GRAPHIC 1,1
20 COLOR 1,5
30 CIRCLE 1,19,21 ,7,3:REM FISH
   BODY
40 DRAW1,26,19 TO 32,17 TO 29,20
   TO 32,23 TO 26,21 :REM TAIL
50 REM FISH FINS
60 DRAW1, 18,24 TO 24,30 TO 22,24
70 DRAW1, 16, 18 TO 22,13 TO 21,18
80 DRAW1,14,24 TO 18,27 TO 16,24
90 REM FILL IN BODY AND EYES
100 PAINT 1,19,21 :CIRCLEO,15.20, 1,1 :
   PAINTO,14,20
```

Once the design is complete, it is then saved into a text string with the SSHAPE command. The fish is located with the upper left corner at screen coordinates (12,14). To capture the entire shape, the SSHAPE command should read

```
110 SSHAPE A$,12,14,33,38
```

Once the data for the shape has been stored in the SSHAPE string variable, it can be transferred into a sprite.

SPRSAV takes the data in the SSHAPE string and puts it into the sprite data area. The command

```
120 SPRSAV A$,1
```

transfers the fish data in A$ into sprite number 1. After you RUN the program, the design for sprite number 1 is complete and ready to use in a program, which must activate it with the SPRITE command. You can add another line to return to text mode at the end of the program, or you can press RUN/STOP and RESTORE.

A word of caution: make sure the coordinates for SSHAPE capture the entire shape. If they are slightly off, the data may be for only part of a sprite or even a blank section of screen.

USING THE SPRITE EDITOR

The left side of the sprite editor screen is a 24 X 21 "block" rectangle in which you can design the sprite. Each block represents a single pixel, and the cursor on this screen appears as a "+". You can design your sprite in this work area, and the right half of the screen displays how the sprite will look. If you are designing a multicolor sprite, the size of the area remains the same, but the cursor becomes twice as wide (i.e. it becomes "++").

When you turn on the sprite editor (by issuing the SPRDEF command), you are prompted:

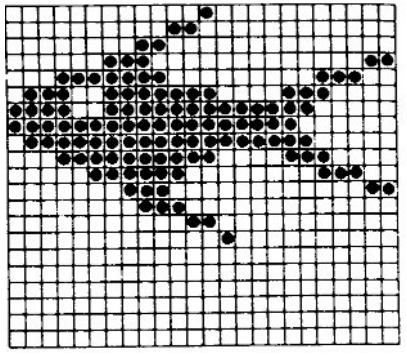
```
SPRITE NUMBER?
```

The sprite editor can hold 8 different sprites, numbered 1 through 8. If you have designed a sprite in a program or recalled stored sprite data, the sprite appears in the work area on the editor. Since we've already drawn a sprite in a program and assigned it to sprite number 1, it appears in the sprite editor in the work area for sprite number 1. If there's no sprite assigned to a number, a random pattern appears. To clear the pattern, press the SHIFT and CIR keys at the same time.

Here are the keys to use in designing sprites:

	Erases the work area
	Fills in background color
2	Fills in foreground color
3	Fills in multicolor 1
4	Fills in multicolor 2
CRSR keys	Move pixel (+) within work area
A	Turns Automatic cursor movement OFF/ON
M	Turns Multicolor ON/OFF
CONTROL /1-8	Sets sprite color
C=/1-8	
RETURN	Moves cursor to start of next line
HOME	Moves cursor to top left corner
X	Expands sprite horizontally
Y	Expands sprite vertically
C	Copies one sprite to another sprite number
SHIFT RETURN	Saves sprite from work area and returns to SPRITE NUMBER? prompt
STOP	Returns to SPRITE NUMBER? prompt without saving the displayed sprite
RETURN	(at the SPRITE NUMBER? prompt) Exits SPRDEF mode

Sprite design is not a mechanical activity. There is no formula to follow, there is no right way or wrong way. Play around with designs in the work area, and you can see what you are creating on the right side. Remember drawing sprites in *Introduction to BASIC Part II,* using POKES? We drew the sprite design on a grid first; that can still be helpful. Here is the sprite we designed at that time:



It's much easier to duplicate using the sprite editor. Start by answering the SPRITE NUMBER? prompt with a 2. Then copy the picture of the grid above in the work area, moving the cursor to the filled spaces on the grid and pressing the 2 key to fill in those spaces on the editor. When you have the shape copied in the sprite editor, you can change its color with CONTROL or the C= key along with a number key from 1 to 8.

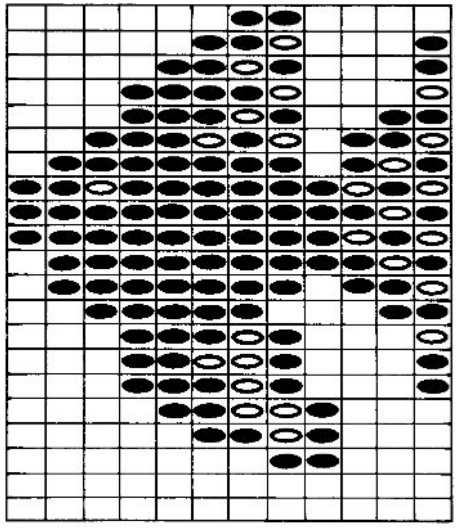
In addition to standard single color sprites, the editor can create multicolor sprites. Press the M key, and whatever is on the screen changes to multicolor. Multicolor lets you design the sprite using two colors in addition to the foreground and background colors. These additional colors can be added by pressing the 3 and 4 keys.

When you have completed the design in the work area, press SHIFT along with RETURN. This causes the sprite editor to retain the sprite under that sprite number. It will stay there as long as the computer remains on. To exit the sprite editor, press RUN/STOP to get the SPRITE NUMBER? prompt. At the prompt, press RETURN without entering a number. This puts you back in BASIC text mode.

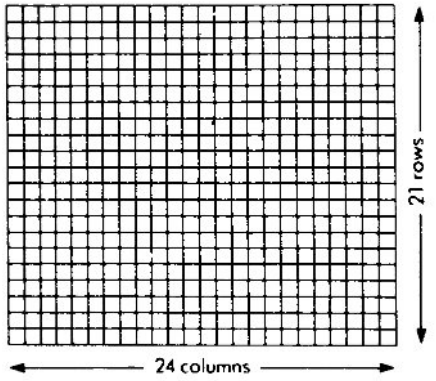
C128 ONLY

EXPERIMENT 11·1

(a) Duplicate this shape as sprite number 3, a multicolor fish sprite. Remember to press M before drawing the design. The second color can be selected with CONTROL or Cl: and a number key from 1 to 8. Press SHIFT and RETURN when you are done.



(b) Draw another shape as sprite number 4. Draw it on a 24 X 21 block grid first, then draw it on the work area. Press SHIFT and RETURN to store it in the editor.



* Experiment 11.1 Completed *

C128 ONLY

SAVING DATA FROM THE SPRITE EDITOR

Saving sprite data is relatively easy. You can save a block of data for eight sprites from the editor at a time. Issue the BSAVE command in direct mode, giving the set of sprite data a name. A command that saves sprite data looks like:

```
BSAVE "FISH SPRITES", BO, P3584 TO
     P4096
```

In addition to the name and memory bank number (BO), two additional numbers, P3584 TO P4096, correspond to the area in the C128's memory where sprite data is stored.

After storing the sprite data, get a DIRECTORY of what's on the disk. The sprite data name is listed as a program. To recall the sprite data, use BLOAD to load the binary file. Always load the sprite data BEFORE going to the editor. An acceptable form of the command to issue is

```
BLOAD "FISH SPRITES"
```

although you may need to include optional parameters in the command specifying drive, device or memory bank number:

```
BLOAD "FISH SPRITES", DO, U8, B1,
     P3584
```

After the sprite data has been recalled, issue SPRDEF. You'll find the sprites are already contained in the work areas for their respective numbers.

TURNING ON A SPRITE-THE SPRITE COMMAND

Once a shape has been drawn and stored using the sprite editor or in a string variable by SSHAPE and SPRSAV, it must be turned on with the SPRITE command so it can appear on the screen. SPRITE features seven different parameters, for example:

```
SPRITE 1,1,5,0,0,0,0
```

The parameters are as follows:

First:	Sprite Number (1 through 8), used to identify the sprite.
Second:	Turns sprite ON (1) or OFF (0).
Third:	Color (1 through 16, corresponding to the color numbers).
Fourth:	Priority. meaning whether the sprite goes in front of or behind objects on the screen. 0 is in front of, 1 for behind.
Fifth:	Normal horizontal size (0) or expands sprite horizontally (1).
Sixth:	Normal vertical size (0) or expands sprite vertically (1).
Seventh:	Multicolor sprite (1) or standard single-colored sprite (0).

When issuing the SPRITE command, set the last three parameters to make the sprite look how you intended when creating it. Pay attention to the multicolor setting, since a sprite presented in the wrong mode can be unrecognizable.

When you give the second parameter a value of 1, the sprite appears on the screen. It remains on the screen until you either reissue the SPRITE command for that sprite with a value of 0 or press RUN/STOP and RESTORE simultaneously.

Once the sprite is turned on, colors selected, etc., you are ready to move it around the screen.

To activate the sprites created in this section, issue the following SPRITE commands:

SPRITE 1,1,5.0,0,0,0
activates the fish sprite drawn with graphic commands in purple

SPRITE 2,1,9,1,0,0,0
activates an orange fish sprite

SPRITE 3,1,7,1,0,0,1
sets the multicolor fish sprite with the secondary color of blue

MOVING SPRITES

C128 BASIC 7.0 features a new command that makes moving sprites around the screen easy. MOVSPR can place a sprite at a given location on the screen, move the sprite a specific distance and angle from the pixel cursor or move it continuously at different speeds. There are four forms of MOVSPR:

1) Position the sprite on the screen

MOVSPR 1,20,100 Places sprite number 1 at column 20, row 100

First parameter: Sprite number (1-8)
Second parameter: X-coordinate
Third parameter: Y-coordinate

2) Move sprite a distance defined by relative screen coordinates (see Appendix A)

MOVSPR 3, - 30, + 40 Moves sprite 3 to the left 30 pixels (- 30) and 40 pixels down (+40), relative to the current pixel cursor position

First parameter: Sprite number (1-8)
Second parameter: Relative distance for X-coordinate
Third parameter: Relative distance for Y-coordinate

3) Move sprite a given distance and angle from pixel cursor

MOVSPR 2,50;90 Moves sprite 2 a distance of 50 pixels from the cursor at a 9O-degree angle

First parameter: Sprite number (1-8)
Second parameter: Distance
Third parameter: Angle (0-360)

4) Move sprite continuously at a certain angle and speed

MOVSPR 6,45#15 Moves sprite 6 at the highest speed (15) at a continuous 45-degree angle

First parameter: Sprite number (1-8)
Second parameter: Angle (0-360)
Third parameter: Speed (0-15)

To move a sprite in a program, first turn the sprite on with the SPRITE command. Then you can use MOVSPR.

Assuming you have sprites in the sprite editor (right now, unless you've turned the computer off or reset it, you should have sprite numbers 1 through 3 already designed), here is a program that activates two sprites and places them on the screen. Notice you don't need to enter a graphic mode to use sprites. Type in this program and let it stay in memory; we'll be adding to it through the rest of this section.

```
10 COIOR 0,1 :COLOR 4,1
15 GRAPHIC 1,1
20 SPRITE 1.1,5,1,0,0,0
30 SPRITE 3,1,7,0.0,0,1
40 MOVSPR 1,200,150
50 MOVSPR 3.100,100
```


This positions the two sprites on the screen. Notice how the white and blue fish appears in front of the program lines; this is because its priority (defined by the fourth parameter of SPRITE in line 30) is O. The purple fish swims behind the letters, because it has a priority of 1. Make sure that the sprite numbers in the MOVSPR command are the same sprite numbers used in the SPRITE commands. For a little more action, add these lines:

```
60 MOVSPR 1,270#4
70 MOVSPR 3,200#3
```

See the fish continuously move from their starting positions (set in lines 40 and 50) and wrap around the screen. To stop them and clear the screen, press RUN/STOP and RESTORE at the same time. The program and sprite data remain in memory.

Not everything moves in a consistently straight line. Use MOVSPR and a little experimentation to create wavering movement, or a circular pattern, or change speeds or whatever. For example, adding a loop can make a sprite change direction or speed slightly or noticeably. Add these lines to the last program to see a little variable movement:

```
54 A= 100
55 A=-A
75 SLEEP3
80 MOVSPR 1,270#10
90 MOVSPR 3,250 + A#2
100 SLEEP3
110 GOTO 55
```

One fish will be changing speeds noticeably, while the other will be bouncing all over the screen in all directions. With some experimentation, you can capture the exact movement you want. After you've finished experimenting with movement patterns, use NEW to remove the program from memory.

THE COLLISION COMMAND

When working with sprites, their paths sometimes cross. When this happens, you can let them continue unaffected, or do something as a result of the collision. COLLISION detects a sprite coming into contact with another object and transfers execution to a subroutine designed to "make something happen" as a result of or response to the collision. A typical form of this command is

COLLISION 1,1000

First parameter:
Defines type of collision
1-sprite-to-sprite
2-sprite to display data
3-light pen (40 columns only)

Second parameter:
Line number for subroutine

COLLISION with no line number following turns off collision detection.

Assume the program sets two sprites on a collision course, as in the following:

```
5 GRAPHIC 1,1
10 COLOR 0,1:COLOR 4,1
20 SPRITE 1,1,5,1,0,0,0
30 SPRITE 2,1,7,0,0,0,1
40 MOVSPR 1,SO, 100
50 MOVSPR 2,200, 1oo:MOVSPR
   2,270#2
```

The COILISION command can be added in the following routine:

```
60 COLLISION 1,100
70 GOTO 60
100 COLLISION 1
110 SPRITE 1,0,5,1,0,0.0
120 RETURN
```

When the sprites touch, execution jumps to the collision subroutine in line 100, which first turns off collision detection, then sprite number 1 is removed. Then execution is returned to the line where the collision was detected (line 60).

Another type of collision, sprite with display data (anything drawn on the screen, such as a BOX), works Similarly. Change COLLISION 1 in lines 60 and 100 to 2 and add another line to the program:

```
15 BOX 1,0,0,10,190
```

Change the subroutine beginning at line 100:

```
100 COLLISION 2:C =C + 1:IF C>16
    THEN C= 1
110 COLOR 1,C:BOX1,O,O, 10,190
120 RETURN
```

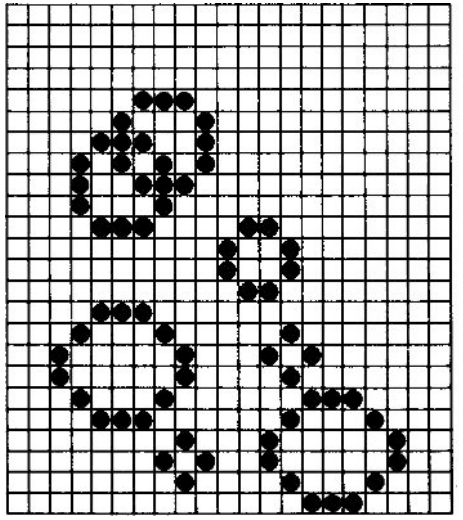
C128 ONLY

The sprites touch with nothing happening, since there is nothing set up to detect that collision. But when the sprite hits the object on the screen (the box drawn in line 15), the collision routine goes into effect. As a result of the collision, the box is redrawn in changing colors.

Working with collision routines is sometimes frustrating, because it may be difficult to get the subroutine to do exactly what you want, and collision detection and reaction isn't as precise as you need.

EXPERIMENT 11·2

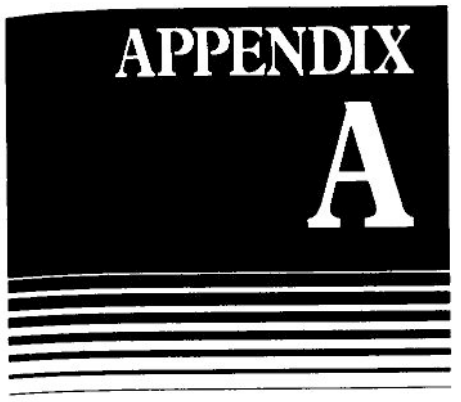
(a) Design two multicolor fish sprites (one facing left, one facing right) using the sprite editor and assign them to sprite numbers 5 and 6. Using whichever method you prefer, create a sprite featuring water bubbles and assign it to sprite number 7. You should have sprites numbered 1, 2, 3, 4, 5, 6 and 7.



(b) Copy the water bubble sprite (number 7) to sprite number 8, using the C command with the sprite editor. Make slight changes (add bubbles or change placement) and make the sprite a different color from sprite 7. Store all the sprites using the BSAVE command.

(c) lOAD the program FISHTANK. This provides a backdrop for an aquarium. Use your fish sprites to populate the aquarium. Write a program to move the fish sprites around the aquarium at changing speeds and slightly varying directions. There are already movement routines in the program for sprites numbers 1 and 7 in lines 700 on. Notice how the bubbles float straight upward. With all the elements in place, the graphic background picture and your moving sprites, you should have a thriving aquarium.

Note: Additional sprite data may be found on the binary file on your program disk called TROPICAl. If you load this file, issue the SPRDEFcommand to see the sprites in the editor.



APPLYING RELATIVE COORDINATES

Most times when we issued commands, we used absolute screen coordinate values. However, there are other ways to specify screen locations using relative coordinates.

Many commands, including lOCATE, DRAW, CIRCLE, BOX and PAINT need a starting position. Two familiar ways of defining this position are:

a) By omission, in which case it defaults to the 'current position'.

b) By stating its screen coordinates.

There is a third way of defining the starting position by giving the coordinates relative to the current cursor position. This is done by either of two methods:

Rectangular displacement: Specify how many pixels to move right, and how many down. The numbers preceded by + or - indicate the displacement is relative. Compare these two sequences:

```
CIRCLE 1,30,40,20
```

(Draws circle centered at (30,40)

```
lOCATE 100,100
CIRCLE 1,+30,+40,20
```

(drows circle centered at 130,140)

POLAR DISPLACEMENT:

Here you specify how many pixels to move, and in which direction. The direction is given in degrees clockwise from up-like a compass bearing. The distance comes first and is separated from the direction by a semicolon (;). For example,

```
LOCATE 100,100
CIRCLE 1,50;90,20
```

TURTLE GRAPHICS

Here's a special appl ication that lets you see a way to use relative coordinates in drawing graphics. "Turtle Graphics" is the name used for an alternate way of drawing computer pictures. All the examples in these books (prior to now) used absolute screen coordinates. The "pencil" which drew the picture was directed to a given set of coordinates in the display area and made to draw a line to another given position. Turtle graphics makes use of the relative screen coordinates in fashioning pictures.

In Turtle Graphics, the navigator, or "Turtle", moves based on the current location. It doesn't consider screen coordinates or fixed directions like left, right, up or down. All it understands are four basic commands:

```
Raise or lower your pencil
Move so many steps forward,
drawing a line of the pencil is down
Turn so many degrees clockwise
Turn so many degrees counter-
   clockwise
```

Turtle instructions to draw a square wide sides of 100 units are:

```
lower pencil
Go 100 steps forward
Turn 90 degrees right
Go 100 steps forward
Turn 90 degrees right
Go 100 steps forward
Turn 90 degrees right
Go 100 steps forward
```

Of course, the position of this square depends on where the turtle starts and the direction the turtle is facing at the beginning.

Many kinds of drawings are easier to make with turtle graphics than with fixed navigational systems. For example, you might have a set of turtle instructions to draw some complicated shape like a tree or a symbol in a circuit diagram. Simply by changing the starting conditions, you can draw the object many times over, in various pOSitions and orientations. A simple modifi-

cation will change the scale of the picture to anything you need.

The C16, Plus/4 and C128 have no built-in support for turtle graphics, so your program must translate the turtle's instructions into ones suitable for fixed navigation.

First, let's define some variables.

XI and Y1 are the current screen coordinate position of the turtle (somewhere in the display area).

The current direction, D1, is expressed as a compass bearing between 0 and 360 degrees. 0 degrees is North or Up, 90 degrees is East or Right, and so on.

The value of P1 will indicate the state of the pencil, as follows:

Pencil up : PI is 0 (move cursor without drawing)

Pencil down: PI is 1 (draw on screen)

We can now express the turtle instructions in terms of these commands:

- Raise Pencil: PI = 0
- lower Pencil: P1 = 1
- Turn N degrees right: D1 = 01 + N: IF D1 >= 360 THEN 0 = 0-360
- Turn N degrees left: 01 = D1 - N: IF DI <0 THEN D1 = D1 + 360
- Move FI steps forward:

We assume that a step is the same size as a pixel. This lets us calculate the turtle's new position (XX, YY) by using the following formulas:

```
DO + 01 * 1("/180 REM CONVERTS 0 TO
                  RADIANS
XX = XI + F1 * SIN(DD)
YY = Y1 + F1 * COS(DD)
```

Next, we draw a line from the old position to the new one, using P1 to select foreground or background.

Finally, we copy the new position into X1 and Y1.

The move is the only step sufficiently complicated to need a subroutine, so here it is:

```
1000 REM ORGANIZE TURTLE'S MOVE
     FORWARD
1010 REM MOVE F1 PIXELS IN
     DIRECTION 01, STARTING AT
     X1,Y1
1020 DO = 01 * /180
1030 XX = XI + F1 * SIN (DO)
1040 YY = Y1-F1 * COS (DO)
1050 IF F1<0 OR X1 <0 OR Y1 <0 OR
     XX<O OR YY<O THEN 1080
1060 IF X1 >319 OR XX>319 or Y1 >199
     OR YY>199 THEN 1080
1070 DRAW P1,X1,Y1 TO XX,YY
1080 X1 = XX:Y1 = YY
1090 RETURN
```

Once this subroutine has been provided, turtle graphics are easy to translate into BASIC. For instance, you can draw a pentagon by writing

```
10 COLOR 0,1
20 GRAPHIC 1,1
30 COLOR 1,2
40 XI = 200: Y1 = 160
50 01 = 18:P1 = 1:REM START AT
   (200,160) FACING NORTHEAST
   WITH PEN DOWN
60 FOR J = 1 TO 5
70 F1 = 50: GOSUB 1000: REM
   FORWARD 50 UNITS
8001 = 01-72: IF 01<0 THEN
   01 =01 +360
90 NEXT J
100 GETKEY A$
110 GRAPHIC 0
120 END
```

A few modifications can produce major differences. For instance, if you increase the number of steps in line 60 to 8, and change the angle in line 80 to 45 (D1 = D1-45), you will get an octagon. If you now change the 45 to 135, the figure becomes an 8-pointed star.

To get a more complex figure, go back to the original pentagon, change the number of steps in line 60 to 100 and the angle in line 80 to 73. You will get a rotated ri ng of pseudo-pentagons, all of the same size.

To make the shape decrease in size as it rotates, try

```
60 FOR J = 100 TO 10 STEP -0.5
```

and

```
70 FI =J: GOSUB 1000
```

In general, turtle graphics give interesting and unexpected results. This sequence, for example, leads to a spiral disappearing into the distance:

```
40 XI = 50: YI = 169
50 01 =O:PI =1
60 FOR J=I TO 200
70 FI = 0.0003*(XI + YI) 2:GOSUB
   1000
80 01 = 01-30: IF 01<0 THEN
   01 = D1 +360
90 NEXT J
```

Keeping the same subroutine (lines 1000-1070), you can design a program that lets you directly control the drawing of each step. You can write a program that lets you input values for P1 (pencil up or down), D1 (direction in degrees) and 11 (length). When you RUN this program, be careful that you don't use unacceptable values for screen coordinates. An error trap comes in handy here. Here is the program (not including the movement subrouti ne):

```
 5 TRAP 2000
10 COLOR 0,7:COLOR 1,2
20 GRAPHIC 2,1
30 XI = 160: YI = 100:PRINT
   "STARTING POINT IS (160,100)"
40 DRAW I,XI,YI
45 REM SET VALUES
50 PRINT "PRESS Q TO QUIT,
   ANYTHING ELSE TO CONTINUE"
60 GETKEY A$:IF A$ = "Q" THEN
   120
65 PRINT "CURSOR DOWN CD
   CD CD CD"
70 INPUT "PENCIL UP (0) OR
   DOWN (l)";PI
80 INPUT "DIRECTION";D1
90 INPUT "LENGTH";F1
100 GOSUB 1000:REM GO TO THE
    DRAWING SUBROUTINE
110 GOTO 50
120 END
2000 PRINT "TRY AGAIN": GOTO 70
```

Note: CDI in line 65 refers to the DOWN CURSOR, so press CURSOR DOWN 5 times.

This is just one application using relative coordinates instead of absolute values to plot graphics. Neither form is better than the other. Absolute may be easer to visualize because it's more in line with the way we view the screen; relative coordinates offer different advantages, such as continuity in plotting lines and drawing. Which form to use depends on which you are more comfortable with and which is more suited to the specific programming task.

APPENDIX B

ANSWERS TO EXPERIMENTS

These program listings are not the only solutions, nor are they necessarily the best. But they do solve the problems presented in the experiments using the relevant commands. Your own answers may differ and still be correct.

The following program listings are printed as they appear on your screen. Spaces in strings should be typed. Here is a list of control characters you will see in these listings, and the keystrokes which created them.

Symbol	Keystrokes
S	HOME
♥	SHIFT and CLR/HOME
Q	Cursor down
]	Cursor right
O	Cursor up
[	Cursor left
R	CTRL and RVS ON
■	CTRL and RVS OFF

Several other symbols, such as ||, ||, and ▪ represent color changes; these are activated by pressing and holding the CTRl key and pressing a number key (not from the numeric keypad on the 128). You can substitute colors at any time for an arrangement more pleasing to you.

UNIT:1

Experiment 1.1

```
10 X=1
20 DOWHILEX<16
25 PRINT""
30 COLOR4,X
40 X=X+1
45 SLEEP1
50 LOOP
60 DOUNTILX=1
70 COLOR0,X
80 X=X-1
```

```
85 SLEEP1
90 LOOP
100 DO WHILE X<16:PRINT"        CHANGING COLORS"
110 COLOR5,X
120 X=X+1
125 SLEEP1
130 LOOP
140 END

READY.
```

Experiment 1.1/4

```
10 X=1
20 DOWHILEX<16
25 PRINT""
30 COLOR4,X
40 X=X+1
45 FOR N=1 TO 500:NEXT N
50 LOOP
60 DOUNTILX=1
70 COLOR0,X
80 X=X-1
85 FOR N=1 TO 500:NEXT N
90 LOOP
100 DO WHILE X<16:PRINT"        CHANGING COLORS"
110 COLOR1,X,4
120 X=X+1
125 FOR N=1 TO 500:NEXT N
130 LOOP
140 END

READY.
```

Experiment 1.3

```
1 CLR:AA$="■■■■■■■■■■■■■■■■■■■■"
10 X=1
20 REM***** SET UP MENU SCREEN *****
30 PRINT"□"
40 PRINT"          CALORIE COUNTER"
50 PRINT"          ------- -------"
60 PRINT""
70 PRINT"            FOOD TYPE"
80 PRINT"            - - - - -"
90 PRINT"           [ ]BEVERAGES"
95 PRINT""
100 PRINT"           [ ]BREAKFAST FOODS"
105 PRINT""
110 PRINT"           [ ]LUNCH FOODS"
115 PRINT""
120 PRINT"           [ ]DINNER FOODS"
125 PRINT""
130 PRINT"           [ ]DESSERTS"
135 PRINT""
140 PRINT"           [ ]SNACKS AND CANDY"
145 PRINT""
150 PRINT"           [ ]SPREADS"
155 PRINT""
156 PRINT"           [ ]EXIT
157 PRINT""
158 REM***** MOVE CURSOR, DETERMINE CURSOR POSITION AND CATEGORY *****
160 PRINT"    MOVE THE CURSOR TO YOUR SELECTION,"
165 PRINT"          THEN HIT RETURN"
167 COLOR5,1
180 DOUNTILA1$=CHR$(13)
190 GETA1$:IFA1$="" THEN 220
200 IFA1$="■" AND X<>8 THEN BEGIN AA$=AA$+"■■"
202 :X=X+1
204 :GOTO220
206 :BEND
210 IFA1$="□" AND X<>1 THEN BEGIN AA$=LEFT$(AA$,(LEN(AA$)-2))
215 :X=X-1
218 :BEND
220 PRINT AA$"■ ":FOR I=1TO100:NEXT:PRINT AA$" "
230 LOOP
240 REM***** PRINT OUT CATEGORY 1 INFORMATION *****
250 IFX=1 THEN BEGIN:
260 PRINT"□"
270 PRINT"             BEVERAGES"
280 PRINT"             ---------"
290 PRINT"   BEVERAGE          SERVING    CALORIES"
300 PRINT"   - - - - -         - - - -    - - - -"
310 PRINT""
320 PRINT"    ALE              12 OUNCES      150"
330 PRINT"    BEER             12 OUNCES      150"
340 PRINT"    BEER,LITE        12 OUNCES      105"
350 PRINT"    BRANDY           1.5 OUNCES     73"
360 PRINT"    COCOA            8 OUNCES       245"
370 PRINT"    CHAMPAGNE        3.5 OUNCES     85"
400 PRINT"    COFFEE,TEA       8 OUNCES       2"
430 PRINT"    LEMONADE         8 OUNCES       105"
440 PRINT"    MARTINI,DRY      2.5 OUNCES     140"
450 PRINT"    MILK,SKIM        8 OUNCES       120"
460 PRINT"    MILK,WHOLE       8 OUNCES       150"
470 PRINT"    ORANGE JUICE     8 OUNCES       120"
500 PRINT"    SODA,COLA        12 OUNCES      145"
510 PRINT"    SODA DIET        12 OUNCES      1"
520 PRINT"    WHISKEY          1.5 OUNCES     105"
530 PRINT"    WINE,DRY         3.5 OUNCES     85"
535 PRINT""
540 PRINT"PRESS RETURN WHEN YOU ARE READY TO"
545 INPUT" CONTINUE"; G$:BEND
550 REM***** PRINT OUT CATEGORY 2 INFORMATION *****
560 IF X=1 THEN GOTO 1
565 REM***** PRINT OUT CATEGORY 2 INFORMATION *****
570 IF X=2 THEN BEGIN:
575 PRINT"□"
580 PRINT"            BREAKFAST FOODS"
590 PRINT"            --------- -----"
600 PRINT""
610 PRINT"   FOOD TYPE     SERVING SIZE    CALORIES"
620 PRINT"   - - - - -     - - - - - - -   - - - - - -"
630 PRINT"   BACON            2 STRIPS        85"
640 PRINT"   CEREAL,DRY       6 OUNCES        70"
650 PRINT"   EGG MCMUFFIN        1            352"
660 PRINT"   ENGLISH MUFFIN      1            186"
670 PRINT"   (WITH BUTTER)"
680 PRINT"   FRIED EGG           1            115"
690 PRINT"   OATMEAL          2/3 CUP         87"
700 PRINT"   OMLET,CHEESE     6 OUNCES        340"
710 PRINT"   PANCAKES            3            180"
720 PRINT"   ROLL,BUTTERED       1            260"
740 PRINT"   SAUSAGE          2 LINKS         120"
750 PRINT"   WAFFLES             1            210"
760 PRINT""
770 PRINT"PRESS RETURN WHEN YOU ARE READY TO "
780 INPUT"CONTINUE ";G$
790 BEND
800 IF X=2 THEN GOTO 1
805 REM***** PRINT OUT CATEGORY 3 INFORMATION *****
810 IF X=3 THEN BEGIN:
820 PRINT"□"
830 PRINT"             LUNCH FOODS"
840 PRINT"             ----- -----"
850 PRINT""
860 PRINT"   FOOD ITEM      SERVING SIZE    CALORIES"
870 PRINT"   - - - - -      - - - - - -    - - - - -"
880 PRINT"   BOLOGNA          1 SANDWICH      313"
890 PRINT"   BURGER,ROLL      4 OUNCES        518"
```

```
900 PRINT"   AND CHEESE      "
910 PRINT"   CHILI WITH      6 OUNCES         259"
920 PRINT"   BEANS"
930 PRINT"   CLUB SANDWICH   10 OUNCES        670"
940 PRINT"   GRILLED CHEESE  1 SANDWICH       350"
950 PRINT"   BURGER,ROLL     4 OUNCES         418"
960 PRINT"   HAM SANDWICH    1 SANDWICH       350"
970 PRINT"   HAM & CHEESE    1 SANDWICH       458"
980 PRINT"   HOT DOG,ROLL       1             291"
990 PRINT"   PIZZA            1 SLICE         145"
1000 PRINT"   ROAST BEEF      1 SANDWICH       429"
1010 PRINT""
1020 PRINT"PRESS RETURN WHEN YOU ARE READY TO"
1030 INPUT" CONTINUE";G$
1040 BEND
1050 IF X=3 THEN GOTO 1
1055 REM****** PRINT OUT CATEGORY 4 INFORMATION *****
1060 IF X=4 THEN BEGIN
1070 PRINT"□"
1080 PRINT""
1090 PRINT"              DINNER FOODS"
1095 PRINT"              ------------"
1100 PRINT"   FOOD TYPE      SERVING SIZE  CALORIES"
1110 PRINT"   - - - - -      - - - - - - -  - - - - - -"
1120 PRINT"   BEEF POT PIE     8 OUNCES       560"
1130 PRINT"   BEEF STEAK       3 OUNCES       330"
1140 PRINT"   BEEF STROGANOFF  8 OUNCES       500"
1150 PRINT"   BLUEFISH         3 OUNCES       135"
1160 PRINT"   BURRITO,BEEF     6.5 OUNCES     468"
1170 PRINT"   CHICKEN POT PIE  8 OUNCES       535"
1180 PRINT"   FRIED CHICKEN    1 WING         151"
1190 PRINT"   FISH & CHIPS     7 OUNCES       370"
1200 PRINT"   FISH STICKS      4              180"
1210 PRINT"   FRIED SHRIMP     3 OUNCES       190"
1220 PRINT"   LAMB CHOP        3.1 OUNCES     360"
1230 PRINT"   MACARONI &       8 OUNCES       430"
1240 PRINT"   CHEESE"
1250 PRINT"   PORK CHOP        2.7 OUNCES     305"
1260 PRINT"   POT ROAST        2.5 OUNCES     140"
1270 PRINT"   SPAGHETTI WITH   8 OUNCES       260"
1280 PRINT"   SAUCE"
1290 PRINT""
1300 PRINT"PRESS RETURN WHEN YOU ARE READY TO "
1310 INPUT"CONTINUE";G$
1320 BEND
1330 IFX=4 THEN GOTO 1
1335 REM****** PRINT OUT CATEGORY 5 INFORMATION *****
1340 IF X=5 THEN BEGIN
1350 PRINT"□"
1360 PRINT"              DESSERTS"
1370 PRINT"              --------"
1380 PRINT"   DESSERT       SERVING SIZE  CALORIES"
1390 PRINT"   - - - - -      - - - - - -  - - - - - -"
1400 PRINT"   BANANA SPLIT      1              540"
1410 PRINT"   BROWNIE           1              95"
1430 PRINT"   CHEESECAKE        5.7 OUNCES     400"
1440 PRINT"   CHOCOLATE CAKE    1 SLICE        250"
1460 PRINT"   COFFEE CAKE       1 SLICE        230"
1470 PRINT"   COOKIES           4              200"
1480 PRINT"   DANISH PASTRY     1              275"
1490 PRINT"   DONUT,FILLED      1              225"
1500 PRINT"   DONUT,PLAIN       1              100"
1520 PRINT"   ICE CREAM         8 OUNCES       270"
1530 PRINT"   JELLO             8 OUNCES       140"
1540 PRINT"   PIE,FRUIT         1 SLICE        345"
1550 PRINT"   PIE,PECAN         1 SLICE        495"
1560 PRINT"   PIE,PUMPKIN       1 SLICE        275"
1570 PRINT"   PUDDING,          8 OUNCES       385"
1580 PRINT"   CHOCOLATE"
1590 PRINT"   SHERBERT          8 OUNCES       270"
1600 PRINT""
1610 PRINT"PRESS RETURN WHEN YOU ARE READY TO "
1620 INPUT"CONTINUE";G$
```



```
1630 BEND
1640 IF X=5 THEN GOTO 1
1645 REM****** PRINT OUT CATEGORY 6 INFORMATION *****
1650 IF X=6 THEN BEGIN
1660 PRINT"□"
1670 PRINT"              SNACKS AND CANDY"
1680 PRINT"              ----------------"
1690 PRINT"   SNACK           SERVING      CALORIES"
1700 PRINT"   - - -           - - - -      - - - - - -"
1710 PRINT"   CANDY,HARD       1 OUNCE        110"
1720 PRINT"   CARAMEL          1 OUNCE        115"
1730 PRINT"   CHOCOLATE,BAR    1 OUNCE        145"
1740 PRINT"   FRENCH FRIES     10             135"
1750 PRINT"   FUDGE            1 OUNCE        115"
1760 PRINT"   MARSHMALLOWS     1              25"
1770 PRINT"   NUTS,DRY         1 OUNCE        160"
1780 PRINT"   ROASTED"
1790 PRINT"   NUTS,MIXED       1 OUNCE        180"
1800 PRINT"   ONION RINGS      1 PORTION      300"
1810 PRINT"   POPCORN,         8 OUNCES       85"
1820 PRINT"   BUTTERED"
1830 PRINT"   POPCORN,         8 OUNCES       25"
1840 PRINT"   UNBUTTERED"
1850 PRINT"   POTATO CHIPS     1 OUNCE        150"
1860 PRINT"   PRETZELS         1 OUNCE        110"
1870 PRINT"   SUGAR            1 TBSP.        45"
1880 PRINT""
1890 PRINT"PRESS RETURN WHEN YOU ARE READY TO "
1900 INPUT"CONTINUE";G$
2000 BEND
2010 IF X=6 THEN GOTO 1
2015 REM****** PRINT OUT CATEGORY 7 INFORMATION *****
2020 IF X=7 THEN BEGIN
2030 PRINT"□"
2040 PRINT"              SPREADS"
2050 PRINT"              -------"
2060 PRINT"   TYPE          SERVING SIZE   CALORIES"
2070 PRINT"   - - -         - - - - - -    - - - - - -"
2075 PRINT""
2080 PRINT"   BUTTER           1 TBSP.        100"
2085 PRINT""
2090 PRINT"   JAM              1 TBSP.        55"
2095 PRINT""
2100 PRINT"   MARGARINE        1 TBSP.        100"
2105 PRINT""
2110 PRINT"   PEANUT BUTTER    1 TBSP.        95"
2115 PRINT"":PRINT"":PRINT""
2120 PRINT"PRESS RETURN WHEN YOU ARE READY TO "
2130 INPUT"CONTINUE";G$
2140 BEND
2150 IF X=7 THEN GO TO 1
2160 IF X>7 THEN PRINT"□":END
```

UNIT:2

Experiment 2.1a

```
10 INPUT "LOAN AMOUNT";A
20 IFA<1000RA>1000THENGOSUB100:GOTO10
30 INPUT "PERCENTAGE INTEREST RATE";P
35 PRINT""
40 I=A*(P/100)
50 PRINTUSING"INTEREST FOR ONE YEAR EQUALS$######.##";I
60 PRINTUSING"TOTAL PAYMENT EQUALS $######.##";A+I
70 M=(A+I)/12
```

```
80 PRINTUSING"EACH MONTHLY PAYMENT EQUALS $$####.##";M
90 END
100 PRINT "LOANS AVAILABLE BETWEEN $100 AND $1000":RETURN
```

Experiment 2.1b

```
5 PRINT"□"
10 X=1
20 PRINT"              BANK STATEMENT"
30 PRINT"              ---- ---------"
40 PRINT""
50 INPUT"ENTER BEGINNING BALANCE";BB
55 WW=BB
60 PRINT""
70 INPUT"IS THIS TRANSACTION A WITHDRAWAL(W) OR A DEPOSIT(D)";T$
75 INPUT"ARE YOU SURE";K$:IF K$="Y"THEN GOTO80
77 GOTO70
80 INPUT"ENTER TRANSACTION AMOUNT";TA(X)
90 PRINT""
100 IFT$="W" THEN TA(X)=-1*TA(X):GOTO120
110 IFT$<>"D" THENINPUT"ENTER D OR W";T$:GOTO100
120 BB=BB+TA(X)
130 PRINT""
140 INPUT"DO YOU WISH TO ENTER ANOTHER TRANSACTION (Y/N)";A$
150 IF A$="Y" THEN BEGIN
160 PRINT"□"
170 PRINT"              BANK STATEMENT"
180 PRINT"              ---- ---------"
190 X=X+1
200 BEND
210 IF A$="Y" THEN GO TO 70
220 IF A$<>"N"THEN GO TO 140
230 PRINT"□"
240 PRINT"               BANK STATEMENT"
250 PRINT"               ---- ---------"
260 PRINT""
265 PRINT USING " BEGINNING BALANCE : $$##,###,###,###.##";WW
266 PRINT" --------- --------"
270 PRINT"       DEPOSITS          WITHDRAWALS"
280 PRINT"       - - - - -         - - - - - -"
290 FOR Y=1 TO X
300 IF TA(Y)<0 THEN BEGIN
310 PRINT USING"                          $$###,###,###.##-";TA(Y)
320 BEND
330 IF TA(Y)>0 THEN BEGIN
340 PRINT USING "      $$###,###,###.##";TA(Y)
350 BEND
360 NEXT Y
370 PRINT""
380 PRINT USING "    ENDING BALANCE : $$##,###,###,###.##";BB
390 PRINT"    ------- --------"
```

Change lines 310 and 340 to change transaction amount format.

Experiment 2.2a

```
10 A$="CAR":B$="TRUCK":C$="MOTORCYCLE"
20 INPUT"FORMAT:";F$
30 PRINT USING F$;A$,B$,C$
```


Experiment 2.2b

```
5 DIM N$(400):DIMCN$(500):DIM SA$(700):DIM A2$(700):DIM C$(250):DIM S$(100)
6 DIM Z$(100)
20 FOR E=1 TO 20
40 PRINT"□"
60 PRINT"          ADDRESS FILE"
80 PRINT"          ------- ----"
100 PRINT"ENTRY'S NAME"
120 INPUT N$(E):PRINT""
125 PRINT"IS THERE A COMPANY NAME (Y/N) ?"
126 GETKEY WC$
127 IF WC$="Y" THEN BEGIN
130 PRINT""
133 PRINT "ENTER COMPANY NAME"
134 INPUT CN$(E)
135 BEND CN$="□"
136 PRINT""
137 PRINT"ENTER STREET ADDRESS"
140 INPUT SA$(E)
160 PRINT""
180 PRINT"DO YOU NEED ANOTHER LINE FOR THE STREET ADDRESS (Y/N) ?"
200 GETKEY AL$
220 IF AL$="Y" THEN BEGIN
240 PRINT""
260 INPUT"ENTER STREET ADDRESS";A2$(E)
280 BEND A2$="□"
300 PRINT""
320 INPUT"ENTER THE CITY";CT$(E)
340 PRINT"."
360 INPUT"ENTER THE STATE";S$(E)
380 INPUT"ENTER THE ZIP CODE"; Z$(E)
400 PRINT"DO YOU WANT TO CONTINUE (Y/N) ?"
420 GETKEY C$
440 IF C$="N" THEN GOTO 500
460 IF C$<>"Y" THEN GOTO 400
480 NEXT E
500 PRINT"□"
510 Y=1
520 FOR X=1 TO E STEP2
540 PRINT""
560 IF X+1>E THEN PRINT USING "RECORD NO. ##     ";X:GOTO580
570 PRINT USING "RECORD NO. ##     ";X,X+1
580 PRINT""
620 PRINT USING"##################= ";N$(X),N$(X+1)
640 PRINT USING"##################= ";CN$(X),CN$(X+1)
650 PRINT USING"##################= ";SA$(X),SA$(X+1)
670 PRINT USING"##################= ";A2$(X),A2$(X+1)
680 PRINT USING"##################= ";CT$(X),CT$(X+1)
681 PRINT USING"##################= ";S$(X),S$(X+1)
682 PRINT USING"#####";Z$(X),Z$(X+1)
690 IF Y=2 THEN BEGIN
700 PRINT"HIT ANY KEY WHEN YOU ARE READY TO"
701 PRINT"CONTINUE"
702 GETKEY F$
703 PRINT"□"
710 Y=0
715 BEND
720 Y=Y+1
740 NEXT X
```

Answer (1)

Answer (2)

UNIT:3

Experiment 3.1

```
10 DATA PENNY,NICKEL,DIME,QUARTER,HALF DOLLAR, SILVER DOLLAR
20 DATA 1940,1950,1960,1970,1980
30 DATA 1,5,10,25,50,100
40 DATA 2,6,20,27,53,105
50 DATA 1,6,30,25,60,106
60 DATA 3,7,35,31,65,107
70 DATA 2,9,32,31,66,110
80 DIM C$(6),Y(5),P(6,5)
90 FOR J=1 TO 6:READC$(J):NEXTJ
100 FOR K=1 TO 5:READ Y(K):NEXT K
110 FOR K=1 TO 5
120 FOR J=1 TO 6
130 READ P(J,K)
140 NEXT J
150 NEXT K
160 PRINT"THIS PROGRAM STORES THE VALUES OF A SET OF 1940 COINS:"
170 PRINT"PENNY, NICKEL, DIME, QUARTER,"
180 PRINT" HALF DOLLAR AND SILVER DOLLAR"
190 PRINT:PRINT"IN THE YEARS:"
200 PRINT"1940,1950,1960,1970,1980"
210 PRINT:PRINT"IT CAN ALSO CALCULATE THE TOTAL WORTH OF THE COLLECTION IN A
IVEN YEAR"
220 PRINT:PRINT"PARTICULAR COIN, OR"
230 INPUT"TOTAL VALUE (P/T)";W$
240 IF W$ = "P" THEN GOTO 270
250 IF W$="T" THEN GOTO 410
260 PRINT:PRINT"INCORRECT ENTRY, CHOOSE AGAIN":GOTO 220
270 PRINT:INPUT"WHAT COIN";M$
280 FOR J=1 TO 6
290 IF M$=C$(J) THEN X=J:GOTO 310
300 NEXT J
310 INPUT"WHAT YEAR";D
320 FOR K=1 TO 5
330 IF D=Y(K) THEN Z=K:GOTO 350
340 NEXT K
350 PRINT:PRINT"THE VALUE OF A 1940 ";M$" IN  ";D"WAS";P(X,Z)" CENTS"
360 PRINT:INPUT"ANOTHER VALUE Y/N";A$
370 IF A$ <>"Y" THEN 390
380 GOTO 220
390 PRINT"END OF COIN VALUE COMPARISION"
400 END
410 PRINT:INPUT"WHAT YEAR";H
420 FOR K=1 TO 5
430 IF Y(K) = H THEN GOTO 450
440 NEXT K
450 A=0
460 FOR J=1 TO 6
470 A=A+P(J,K)
480 NEXT J
490 PRINT"THE TOTAL VALUE IN ";H"WAS ";A
500 GOTO 360
```


UNIT:4

Experiment 4.1

```
10 COLOR0,4:COLOR5,1
20 PRINT"THIS IS A QUIZ.THERE ARE 10 QUESTIONS.":SLEEP 3
30 J=0
40 COLOR0,4
50 READ Q$
60 IF Q$="END" THEN GOTO 120
70 PRINT"□";Q$:INPUTA$
80 READ B$
90 IF A$=B$ THEN J=J+1:GOSUB 1000
100 IF A$<> B$ THEN GOSUB 2000
110 GOTO 40
120 PRINT"□YOU GOT ";J"OUT OF 10 CORRECT"
130 PRINT"THAT'S ";J/10*100"PERCENT"
140 INPUT"CARE TO TRY AGAIN";C$
150 RESTORE
160 IF C$="Y" THEN GOTO 40
170 PRINT"OKAY, BYE THEN"
180 END
1000 REM RIGHT ANSWER
1010 PRINT CHR$(27)"G"
1020 FOR X=1 TO 3
1030 PRINT "■":FOR N=1 TO 200:NEXT N
1040 NEXT X
1500 PRINT CHR$(27)"H":RETURN
2000 REM WRONG ANSWER
2010 COLOR0,1:SLEEP 1
2020 COLOR0,4 :RETURN
3000 DATA WHAT IS A GROUP OF GEESE CALLED, GAGGLE
3010 DATA WHAT IS 64 IN ROMAN NUMERALS,LXIV
3020 DATA WHAT PRIMARY COLOR HAS THE SHORTEST NAME,RED
3030 DATA WHAT ANIMAL CAN RUN FASTEST,CHEETAH
3040 DATA WHAT DO CARCINOGENS CAUSE,CANCER
3050 DATA HOW MANY FEET IN A MILE, 5280
3060 DATA WHAT DO DATES GROW ON,PALM TREES
3070 DATA WHAT'S A LOVE APPLE,TOMATO
3080 DATA WHAT'S THE LARGEST PLANET, JUPITER
3090 DATA WHAT VITAMIN COMPLEX INCLUDES NIACIN AND RIBOFLAVIN,B
3100 DATA END
```

Experiment 4.2a

```
10 PRINT"■■□"
20 WINDOW 5,5,25,15,1
30 FOR X=1 TO 10
40 PRINT"■■                          ■"
50 NEXT X:SLEEP2
60 WINDOW10,10,20,20,1
70 FOR X =1 TO 10
80 PRINT"■■          ■
90 NEXT X:SLEEP2
100 WINDOW 16,16,24,24,1
110 FOR X=1 TO 9
120 PRINT"■■     ■";
130 NEXT X:SLEEP2
140 WINDOW 20,20,30,24,1
150 FOR X=1TO8
160 PRINT"■■          ■";
170 NEXT X
180 SLEEP2
190 GOTO 10
```

Experiment 4.2b

```
5 REM***** CLEAR THE SCREEN AND DEFINE COLORS *****
10 PRINT"□"
20 COLOR4,1:COLOR0,1:COLOR5,3
25 REM***** SET UP COLOR WINDOWS *****
30 WINDOW1,2,13,14
40 GOSUB500
50 PRINT"■■■"
60 WINDOW14,2,26,14
70 GOSUB500
80 PRINT"■■■"
90 WINDOW27,2,39,14
100 GOSUB500
110 PRINT"■■■"
115 REM***** SET UP PAYOFF WINDOW *****
120 WINDOW1,15,39,23:COLOR5,7
130 PRINT"■                                          ■"
140 FOR X=1 TO 6
150 PRINT"■ ■                                        ■ ■"
160 NEXT X
170 PRINT"■                                          ■"
175 PRINT"■■■":COLOR5,2:PRINT"     LUCKY JACKPOT !! TRY YOUR LUCK !!
180 PRINT"■■■■■■■■■■■■■■":PRINT"   PRESS THE S KEY TO STOP THE WHEEL"
190 WINDOW3,4,11,12
191 REM***** ROLL SLOT MACHINE WINDOWS *****
195 GOSUB1000
196 C1=C
200 WINDOW16,4,24,12
210 GOSUB1000
215 C2=C
220 WINDOW29,4,37,12
230 GOSUB1000
235 C3=C
240 WINDOW2,16,37,22
245 COLOR5,7
246 REM***** DETERMINE IF USER HAS MATCHING COLORS AND IF PAYOFF IS NECESSA
*
250 J=J-.25
255 IF C1=C2 AND C2<>C3 THEN GOSUB 1300
260 IF C2=C3 AND C3<>C1 THEN GOSUB 1300
265 IF C1=C3 AND C3<>C2 THEN GOSUB 1300
270 IF C1=C2 AND C2=C3 THEN GOSUB 1200
275 PRINT USING" YOUR TOTAL IS $###,###.##";J
280 REMIF J=0 THEN PRINT "YOU BROKE EVEN"
285 REMIF J>0 THEN PRINT USING" YOU HAVE GAINED $###,###.##";J
295 PRINT"DO YOU WISH TO CONTINUE (Y/N)"
300 GETKEY K$
310 IF K$="N" THEN PRINT"■■□ THE END":END
320 IF K$<>"Y" THEN PRINT"□□":GOTO295
330 GOTO 10
499 REM***** SLOT MACHINE COLOR WINDOW DIMENSIONS *****
500 PRINT"■                  ■"
510 FORX=1TO10
520 PRINT"■ ■                ■ ■"
530 NEXT X
540 PRINT"■                  ■"
550 RETURN
999 REM***** ROLL WINDOWS UNTIL USER PRESSES THE S KEY *****
1000 GET S$
1010 DOUNTILS$="S"
1015 C=INT(RND(1)*14+2)
1020 COLOR5,C
1030 FOR X=1 TO 8
1040 PRINT"■                ■"
1050 NEXT X
1070 GET S$
1080 LOOP
1090 PRINT"■■■"
1100 RETURN
1110 REM***** PAYOFF SUBROUTINE *****
1200 PRINT"THREE MATCHING COLORS :"
1210 COLOR5,2
1215 PRINT"        J A C K P O T  $100.00 ! ! !"
1220 J=J+100
1225 RETURN
1300 COLOR5,7
1310 PRINT"TWO MATCHING COLORS
1320 COLOR5,2
1325 J=J+25
1330 PRINT"             PAYOFF OF $ 25.00
1340 RETURN
```

UNIT:5

Experiment 5.1

```
2 TRAP 200
5 P$="CODE"
10 PRINT"□"
15 PRINT"ENTER THE CORRECT PASSWORD"
20 D$=""
30 FORX=1TO4
40 GETKEY B$
50 D$=D$+B$
60 NEXT X
70 IF D$<>P$ THEN PRINT"WRONG CODE ACCESS DENIED":GOTO90
80 IF D$=P$ THEN PRINT"YOU HAVE GAINED ACCESS"
90 NEW
200 TRAP200:RESUME
```

UNIT:6

Experiment 6.1

```
1 REM************************************
2 REM*        INITIALIZE                *
3 REM************************************
4 TC=0:TW=0:MC=0:MW=0:SC=0:SW=0:CC=0:CW=0:UC=0:UW=0
5 GC=0:GW=0
7 REM************************************
8 REM*        MAIN MENU                 *
9 REM************************************
10 PRINT"□"
11 KEY 1,"GOTO170"+CHR$(13)
12 KEY 2,"GOTO444"+CHR$(13)
```

```
13 KEY 3,"GOTO659"+CHR$(13)
14 KEY 4,"GOTO899"+CHR$(13)
15 KEY 5,"GOTO1139"+CHR$(13)
16 KEY 6,"GOTO4203"+CHR$(13)
17 KEY 7,"GOTO1374"+CHR$(13)
18 KEY 8,"GOTO7"+CHR$(13)
19 C=0:W=0
20 PRINT"■"
25 PRINT"              TRIVIA QUIZ"
30 PRINT"              ==========="
35 PRINT""
40 PRINT"              CATEGORIES"
45 PRINT""
50 PRINT"           F1: TELEVISION"
60 PRINT"           F2: MOVIES"
70 PRINT"           F3: SPORTS"
80 PRINT"           F4: COMICS"
85 PRINT"           F5: MUSIC "
86 PRINT"           F6: GEOGRAPHY
87 PRINT"           F7: EXIT"
90 PRINT""
100 PRINT"■■■■■■"
110 PRINT"PRESS THE APPROPRIATE FUNCTION KEY":END
140 REM*************************************
150 REM***        TV ROUTINE             ***
160 REM*************************************
170 PRINT"■"
200 LN=5000:C=0:W=0
201 COLOR4,3:COLOR5,3:COLOR0,2
210 PRINT"              TELEVISION"
220 PRINT""
230 Y=1
240 FOR T=1 TO 10
241 GOSUB 4000
250 NEXT T
376 REM*************************************
377 REM*           TV SUBTOTAL             *
378 REM*************************************
380 PRINT"         TELEVISION"
385 PRINT""
390 PRINT USING"IN THIS SECTION YOU HAVE ANSWERED ##";C
395 PRINT""
400 PRINT USING "CORRECTLY AND ## INCORRECTLY";W
405 IF C=0 THEN P=0:GOTO 415
410 P=(C/10)*100
415 TC=C:TW=W
416 PRINT"■■■■■■■■■■"
420 PRINT USING "YOUR PERCENTAGE IS ###.##%";P
424 RESTORE
425 SLEEP5
435 GO TO 20
440 REM*************************************
442 REM***       MOVIES ROUTINE          *
443 REM*************************************
444 PRINT"■"
445 LN=5040:C=0:W=0
446 COLOR4,7:COLOR5,7:COLOR0,15
448 PRINT"                 MOVIES
450 PRINT""
455 Y=1
460 FOR T=1 TO 10
465 GOSUB 4000
475 NEXT T
600 REM*************************************
601 REM*          MOVIES SUBTOTAL          *
602 REM*************************************
605 PRINT"                   MOVIES"
610 PRINT""
615 PRINT USING "IN THIS SECTION YOU HAVE ANSWERED ##";C
618 PRINT""
620 PRINT USING "CORRECTLY AND ## INCORRECTLY ";W
624 PRINT""
625 PRINT"■■■■■■■■■■"
```



```
626 IF C=0THEN P=0:GOTO 635
630 P=(C/10)*100
635 MC=C:MW=W
640 PRINT USING "YOUR PERCENTAGE IS ###.##%";P
642 RESTORE
645 SLEEP5
650 GO TO 20
653 REM*************************************
656 REM*        SPORTS ROUTINE             *
658 REM*************************************
659 PRINT"■"
660 LN=5080:C=0:W=0
661 COLOR4,1:COLOR5,2:COLOR0,12
665 PRINT"                 SPORTS"
670 PRINT""
675 Y=1
680 FOR T= 1 TO 10
685 GOSUB 4000
690 NEXT T
823 REM*************************************
824 REM*          SPORTS SUBTOTAL          *
825 REM*************************************
826 PRINT"           SPORTS"
830 PRINT""
835 PRINT USING"IN THIS SECTION YOU HAVEANSWERED ##";C
840 PRINT""
845 PRINT USING "CORRECTLY AND ## INCORRECTLY";W
850 PRINT""
853 IF C=0 THEN P=0:GOTO 860
855 P=(C/10)*100
860 SC=C:SW=W
862 PRINT"■■■■■■■■■■"
865 PRINT USING "YOUR PERCENTAGE IS ###.##%";P
867 RESTORE
870 SLEEP5
880 GO TO 20
885 REM*************************************
890 REM***        COMIC ROUTINE          **
895 REM*************************************
899 PRINT"■"
900 LN=5120:C=0:W=0
901 COLOR4,5:COLOR5,3:COLOR0,11
905 PRINT"              COMICS"
910 PRINT""
915 Y=1
920 FOR T =1 TO 10
925 GOSUB 4000
935 NEXT T
1071 REM*************************************
1072 REM*          COMIC SUBTOTAL           *
1073 REM*************************************
1075 PRINT"                COMICS"
1080 PRINT""
1085 PRINT USING "IN THIS SECTION YOU HAVE ANSWERED ##";C
1090 PRINT""
1095 PRINT USING "CORRECTLY AND ## INCORRECTLY ";W
1100 PRINT"■■■■■■■■■"
1105 IF C=0 THEN P=0:GOTO 1115
1110 P=(C/10)*100
1115 CC=C:CW=W
1116 PRINT USING "YOUR PERCENTAGE IS ###.##%";P
1127 RESTORE
1130 SLEEP5
1135 GO TO 20
1136 REM*************************************
1137 REM*           MUSIC ROUTINE         *
1138 REM*************************************
1139 PRINT"■"
1140 LN=5160:C=0:W=0
1141 COLOR4,6:COLOR5,6:COLOR0,1
1145 PRINT"                 MUSIC "
1150 PRINT""
1155 Y=1
```

```
1160 FOR T = 1 TO 10
1165 GOSUB 4000
1180 NEXT T
1306 REM*********************************
1307 REM*          MUSIC SUBROUTINE     *
1308 REM*********************************
1310 PRINT"          MUSIC "
1315 PRINT""
1320 PRINT USING"IN THIS SECTION YOU HAVE ANSWERED ##";C
1325 PRINT""
1330 PRINT USING "CORRECTLY AND ## INCORRECTLY";W
1335 PRINT""
1337 IF C=0 THEN P=0:GOTO 1345
1340 P=(C/10)*100
1345 UC=C:UW=W
1346 PRINT"■■■■■■■■■■"
1350 PRINT USING "YOUR PERCENTAGE IS ###.##%";P
1353 RESTORE
1355 SLEEP5
1365 GO TO 20
1370 REM*********************************
1371 REM***   EXIT AND FINAL TOTALS   *
1372 REM*********************************
1374 PRINT"□"
1375 PRINT"         YOUR FINAL SCORE"
1376 GT=MC+TC+SC+CC+UC+GC
1377 GW=MW+TW+SW+CW+UW+GW
1378 PRINT"         ---- ----- -----"
1379 PRINT""
1380 PRINT"QUESTIONS ANSWERED "
1381 PRINT""
1382 PRINT USING "           CORRECTLY    :####";GT
1384 PRINT USING "           INCORRECTLY  :####";GW
1385 IF GT=0 THEN GP=0:GOTO 1390
1386 GP=(GT/(GW+GT))*100
1390 PRINT"■■■■■■■■■■"
1392 PRINT USING "YOU ANSWERED ###.##% CORRECTLY ";GP
1394 SLEEP5
1400 PRINT"□" : END
1490 REM*********************************
1492 REM*       SUBROUTINE TO CHECK      *
1494 REM*          FOR ANSWER "A"        *
1496 REM*********************************
1500 GET KEY A1$
1510 IF A1$<>"A" THEN BEGIN
1520 PRINT"WRONG ANSWER, THE CORRECT ANSWER IS : A"
1530 W=W+1
1540 BEND
1550 IF A1$="A" THEN BEGIN
1560 PRINT"CORRECT !"
1570 C=C+1
1580 BEND
1590 SLEEP3
1600 PRINT""
1610 PRINT"□"
1620 RETURN
1900 REM*********************************
1992 REM*     SUBROUTINE TO CHECK        *
1994 REM*       FOR ANSWER "B"           *
1996 REM*********************************
2000 GETKEY A2$
2010 IF A2$<>"B" THEN BEGIN
2020 PRINT"WRONG ANSWER, THE CORRECT ANSWER IS : B"
2030 W=W+1
2040 BEND
2050 IF A2$="B" THEN BEGIN
2060 PRINT"CORRECT !"
2070 C=C+1
2080 BEND
2090 SLEEP3
2100 PRINT"□"
2200 RETURN
```



```
2990 REM*********************************
2992 REM*     SUBROUTINE TO CHECK        *
2994 REM*       FOR ANSWER "C"           *
2996 REM*********************************
3000 GETKEY A3$
3010 IF A3$<>"C" THEN BEGIN
3020 PRINT"WRONG ANSWER, THE CORRECT ANSWER IS : C"
3030 W=W+1
3040 BEND
3050 IF A3$="C" THEN BEGIN
3060 PRINT"CORRECT !"
3070 C=C+1
3080 BEND
3090 SLEEP3
3100 PRINT"□"
3200 RETURN
3900 REM*********************************
3992 REM*  SUBROUTINE TO READ AND LIST *
3994 REM*      QUESTIONS AND ANSWERS    *
3996 REM*********************************
4000 READ Q,Q$,A
4010 IF Q=LN THEN BEGIN
4020 PRINT USING "QUESTION NO. ##";Y
4025 PRINT"":PRINT""
4030 Y=Y+1
4040 PRINT Q$
4050 PRINT""
4060 BEND:ELSE GO TO 4000
4065 LN=LN+1
4070 FOR B=1 TO 3
4080 READ Q,Q$(B),D
4090 PRINT "      ";Q$(B)
4095 LN=LN+1
4100 PRINT""
4110 NEXT B
4115 IF A=1500 THEN GOSUB 1500
4116 IF A=2000 THEN GOSUB 2000
4117 IF A=3000 THEN GOSUB 3000
4120 RETURN
4140 REM*********************************
4201 REM*       GEOGRAPHY ROUTINE        *
4202 REM*********************************
4203 PRINT"□"
4204 LN=5200:C=0:W=0
4205 COLOR4,9:COLOR5,2:COLOR0,1
4206 PRINT"              GEOGRAPHY"
4207 PRINT""
4208 Y=1
4209 FOR T=1 TO 10
4210 GOSUB 4000
4211 NEXT T
4300 REM*********************************
4301 REM*       GEOGRAPHY SUBTOTAL       *
4302 REM*********************************4150
4303 PRINT"         GEOGRAPHY"
4304 PRINT""
4305 PRINT USING"IN THIS SECTION YOU HAVE ANSWERED ##";C
4306 PRINT""
4307 PRINT USING "CORRECTLY AND ## INCORRECTLY";W
4308 IF C=0 THEN P=0:GOTO 4310
4309 P=(C/10)*100
4310 GC=C:GW=W
4311 PRINT"■■■■■■■■■■"
4312 PRINT USING "YOUR PERCENTAGE IS ###.##%";P
4313 RESTORE
4315 SLEEP5
4316 GO TO 20
4500 REM*********************************
4510 REM* DATA/ QUESTIONS AND ANSWERS *
4520 REM*********************************
5000 DATA5000,WHAT ROLE DID RALPH WAITE PLAY ?,1500
5001 DATA5001,A) JOHN WALTON ON THE WALTONS,1
5002 DATA5002,B) GEORGE APPLE ON APPLE'S WAY,2
```

```
5003 DATA5003,C) LARRY TAIT ON BEWITCHED,3
5004 DATA5004,WHO PLAYED THE JOKER ON THE BATMAN      SERIES ?,1500
5005 DATA5005,A) CESAR ROMERO,1
5006 DATA5006,B) BURGESS MEREDITH,2
5007 DATA5007,C) FRANK GORSHIN,3
5008 DATA5008,WHAT WAS THE NAME OF THE DOG ON THE       BRADY BUNCH ?,3000
5009 DATA5009,A) PRINCE,1
5010 DATA5010,B) ALICE,2
5011 DATA5011,C) TIGER,3
5012 DATA5012,NAME THE WESTERN SERIES THAT HELPED MAKEJAMES GARNER FAMOUS ?,2000
5013 DATA5013,A) WANTED DEAD OR ALIVE,1
5014 DATA5014,B) MAVERICK,2
5015 DATA5015,C) RAWHIDE,3
5016 DATA5016,ON THE SERIES GILLIGAN'S ISLAND WHAT WAS THE SKIPPER'S FULL NAME ?
,2000
5017 DATA5017,A) ROY HINKLEY,1
5018 DATA5018,B) JONAS GRUMBY,2
5019 DATA5019,C) GINGER GRANT,3
5020 DATA5020,ON THE BEVERLY HILLBILLIES WHAT WAS     GRANNY'S FIRST NAME ?,1500
5021 DATA5021,A) DAISY,1
5022 DATA5022,B) PEARL,2
5023 DATA5023,C) ANNIE MAY,3
5024 DATA5024,WHAT IS THE ID NUMBER FOR THE FEDERATIONSTARSHIP ENTERPRISE?,2000
5025 DATA5025,A) AIR FORCE ONE,1
5026 DATA5026,B) NCC 1701,2
5027 DATA5027,C) FSS 1902,3
5028 DATA5028,WHO PLAYED THE GREEN HORNET'S VALET KATO ON  TV ?,3000
5029 DATA5029,A) ANN B DAVIS,1
5030 DATA5030,B) BURT WARD,2
5031 DATA5031,C) BRUCE LEE,3
5032 DATA5032,WHO PLAYED DENNIS (THE MENACE) MITCHELL ON THAT TV SHOW ?,1500
5033 DATA5033,A) JAY NORTH,1
5034 DATA5034,B) TONY DOW,2
5035 DATA5035,C) MASON REESE,3
5036 DATA5036,ON THE SERIES THE MUNSTERS WHAT WAS THE NAME OF THE DRAGON ?,2000
5037 DATA5037,A) PUFF,1
5038 DATA5038,B) SPOT,2
5039 DATA5039,C) CECIL,3
5040 DATA5040,WHICH LEAD ACTOR NEVER CO-STARRED       WITH A LIVE APE ?,3000
5041 DATA5041,A) CLINT EASTWOOD,1
5042 DATA5042,B) RONALD REAGAN,2
5043 DATA5043,C) CLIFF ROBERTSON,3
5044 DATA5044,WHICH OF THE FOLLOWING WOODY ALLEN      MOVIES WAS NOT FILMED IN B
LACK & WHITE ?,2000
5045 DATA5045,A) MANHATTAN,1
5046 DATA5046,B) BANANAS,2
5047 DATA5047,C) ZELIG,3
5048 DATA5048,WHICH MOVIE DID NOT TAKE PLACE IN       PHILADELPHIA ?,2000
5049 DATA5049,A) TRADING PLACES,1
5050 DATA5050,B) CHU CHU & THE PHILLY FLASH,2
5051 DATA5051,C) ROCKY,3
5052 DATA5052,WHICH OF THE FOLLOWING WAS NOT A DIRTY  HARRY MOVIE ?,3000
5053 DATA5053,A) SUDDEN IMPACT,1
5054 DATA5054,B) MAGNUM FORCE,2
5055 DATA5055,C) TIGHTROPE,3
5056 DATA5056,WHAT WAS THE FIRST JAMES BOND MOVIE ?,3000
5057 DATA5057,A) THUNDERBALL,1
5058 DATA5058,B) GOLDFINGER,2
5059 DATA5059,C) DR. NO,3
5060 DATA5060,WHICH KING ARTHUR MOVIE DID NOT FEATURE A MAGICIAN NAMED MERLIN ?,
2000
5061 DATA5061,A) EXCALIBUR,1
5062 DATA5062,B) MONTY PYTHON & THE HOLY GRAIL,2
5063 DATA5063,C) THE SWORD AND THE STONE,3
5064 DATA5064,WHAT MOTEL WAS FEATURED IN ALFRED       HITCHCOCK'S PSYCHO ?,1500
5065 DATA5065,A) BATES MOTEL,1
5066 DATA5066,B) ROACH MOTEL,2
5067 DATA5067,C) MOTEL CALIFORNIA,3
5068 DATA5068,WHICH OF THE FOLLOWING WAS NOT A CHARAC-TER PORTRAYED BY ROBERT DE
NIRO ?,3000
5069 DATA5069,A) JAKE LAMOTTA,1
5070 DATA5070,B) RUPERT PUPKIN,2
5071 DATA5071,C) SONNY CORLEONE,3
```



```
5072 DATA5072,WHAT FRATERNITY DID JOHN BELUSHI BELONG TO IN ANIMAL HOUSE ?,3000
5073 DATA5073,A) PHI OMEGA,1
5074 DATA5074,B) ZBT,2
5075 DATA5075,C) DELTA HOUSE,3
5076 DATA5076,WHO WON AN OSCAR FOR THE YEAR OF LIVING DANGEROUSLY ?,3000
5077 DATA5077,A) MEL GIBSON,1
5078 DATA5078,B) SIGOURNEY WEAVER,2
5079 DATA5079,C) LINDA HUNT,3
5080 DATA5080,WHAT CITY HAD THE 1ST PRO BASEBALL TEAM THE AMERICAN RED STOCKINGS
,2000
5081 DATA5081,A) CHICAGO,1
5082 DATA5082,B) CINCINNATI,2
5083 DATA5083,C) BOSTON,3
5084 DATA5084,WHAT DID THE Y IN Y A TITTLE STAND FOR ?,1500
5085 DATA5085,A) YELBERTON,1
5086 DATA5086,B) YORKTOWN,2
5087 DATA5087,C) YELLOW,3
5088 DATA5088,WHAT TWO TEAMS PLAYED THE FIRST INTER-  COLLEGIATE FOOTBALL GAME ?
,2000
5089 DATA5089,A) FRANKLIN AND MARSHALL,1
5090 DATA5090,B) HARVARD AND MCGILL,2
5091 DATA5091,C) HARVARD AND PRINCETON,3
5092 DATA5092,ANAHEIM STADIUM IS NOT THE HOME STADIUM FOR WHICH TEAM ?,3000
5093 DATA5093, A) CALIFORNIA ANGELS,1
5094 DATA5094, B) LA RAMS,2
5095 DATA5095, C) LA RAIDERS,3
5096 DATA5096,WHO WERE BOB AND CAROL TED AND DALLAS INSPORTS IN THE LATE 60S ?,2
000
5097 DATA5097,A) TENNIS MIXED DOUBLES,1
5098 DATA5098,B) BOSTON BRUINS DEFENSEMEN,2
5099 DATA5099,C) LEADING SHOW DOGS,3
5100 DATA5100,WHICH OF THE FOLLOWING DID NOT GO  TO  THE PHILLIES IN A TRADE FO
R DICK ALLEN ?,2000
5101 DATA5101,A) TIM MCCARVER,1
5102 DATA5102,B) CURT FLOOD,2
5103 DATA5103,C) WILLIE MONTANE2,3
5104 DATA5104,WHICH NBA FRANCHISE WAS ORIGINALLY THE  SYRACUSE NATIONALS ?,3000
5105 DATA5105,A) DETROIT PISTONS,1
5106 DATA5106,B) NEW YORK KNICKS,2
5107 DATA5107,C) PHILADELPHIA 76ERS,3
5108 DATA5108,WHO SAID NICE GUYS FINISH LAST ?,1500
5109 DATA5109,A) LEO DUROCHER,1
5110 DATA5110,B) VINCE LOMBARDI,2
5111 DATA5111,C) GUY LAFLEUR,3
5112 DATA5112,THE GREEN JACKET IS CONNECTED WITH WHAT GOLFING EVENT ?,2000
5113 DATA5113,A) BRITISH OPEN,1
5114 DATA5114,B) MASTERS TOURNAMENT,2
5115 DATA5115,C) US OPEN,3
5116 DATA5116,WHICH LOCATION WAS NOT A SITE FOR A     FRAZIER AND ALI TITLE FIGH
T ?,2000
5117 DATA5117, A) MANILA,1
5118 DATA5118, B) ZIMBABWE,2
5119 DATA5119, C) NEW YORK CITY,3
5120 DATA5120, WHAT DOES CHARLIE BROWN'S FATHER DO FOR A LIVING ?,1500
5121 DATA5121, A) CUT HAIR,1
5122 DATA5122, B) DELIVER MAIL,2
5123 DATA5123, C) DELIVER MILK,3
5124 DATA5124,WHAT CHARACTER FROM ANOTHER COMIC STRIP IS RELATED TO BEETLE BAILE
Y ?,1500
5125 DATA5125,A) LOIS OF HI & LOIS,1
5126 DATA5126,B) SAD SACK,2
5127 DATA5127,C) LOLLY,3
5128 DATA5128,WHAT COMIC STRIP FEATURED SHMOOS ?,1500
5129 DATA5129,A) LIL ABNER,1
5130 DATA5130,B) POGO,2
5131 DATA5131,C) BARNEY GOOGLE,3
5132 DATA5132, THE CHARACTER BD IN DOONESBURY WAS BASED ON WHOM ?,3000
5133 DATA5133,A) CARTOONIST GARRY TRUDEAU,1
5134 DATA5134,B) SINGER BOB DYLAN,2
5135 DATA5135,C) YALE QUARTERBACK BRIAN DOWLING,3
5136 DATA5136,IN WHAT COMIC STRIP WILL YOU FIND A DOG NAMED HOTDOG ?,1500
5137 DATA5137,A) ARCHIE,1
5138 DATA5138,B) DENNIS THE MENACE,2
```

```
5139 DATA5139,C) FRED BASSETT,3
5140 DATA5140,IN WHAT LONG RUNNING COMIC STRIP WILL   YOU FIND A LION TAMER'S CL
UB ?,1500
5141 DATA5141,A) MUTT AND JEFF,1
5142 DATA5142,B) THE FLINTSTONES,2
5143 DATA5143,C) BLONDIE,3
5144 DATA5144,WHAT COMIC STRIP LATER BECAME SNUFFY    SMITH ?,2000
5145 DATA5145,A) GASOLINE ALLEY,1
5146 DATA5146,B) BARNEY GOOGLE,2
5147 DATA5147,C) THE TOONERVILLE TROLLEY,3
5148 DATA5148,WHAT OLD CARTOON STRIP FEATURED MAGGIE & JIGGS ?,3000
5149 DATA5149,A) THE JIGG IS UP,1
5150 DATA5150,B) MY LIL MAGGIE,2
5151 DATA5151,C) BRINGING UP FATHER,3
5152 DATA5152,WHAT WAS CAPTAIN MARVEL'S NICKNAME ?,2000
5153 DATA5153,A) THE RED TORPEDO,1
5154 DATA5154,B) THE BIG RED CHEESE,2
5155 DATA5155,C) THE RED BEE,3
5156 DATA5156,WHICH SUPER HERO GOT HIS POWERS OTHER   THAN BY NUCLEAR ACCIDENT?,
3000
5157 DATA5157,A) SPIDERMAN,1
5158 DATA5158,B) THE INCREDIBLE HULK,2
5159 DATA5159,C) THE FLASH,3
5160 DATA5160,WHICH OF THE FOLLOWING IS NOT A BEATLES SONG ?,1500
5161 DATA5161,A) SUNNY AFTERNOON,1
5162 DATA5162,B) SUN KING,2
5163 DATA5163,C) I'LL FOLLOW THE SUN,3
5164 DATA5164,WHO SANG THE HIT THEME SONG FROM THE    JAMES BOND MOVIE GOLDFINGE
R ?,1500
5165 DATA5165,A) SHIRLEY BASSEY,1
5166 DATA5166,B) SHEENA EASTON,2
5167 DATA5167,C) NANCY SINATRA,3
5168 DATA5168,PETER NOONE WAS THE STAR OF WHICH       ENGLISH SINGING GROUP ?,15
00
5169 DATA5169,A) HERMAN'S HERMITS,1
5170 DATA5170,B) THE SEARCHERS,2
5171 DATA5171,C) THE MOODY BLUES,3
5172 DATA5172,WHO WROTE THE SONNY AND CHER HIT SONG    THE BEAT GOES ON ?,2000
5173 DATA5173,A) NEIL DIAMOND,1
5174 DATA5174,B) BOB DYLAN,2
5175 DATA5175,C) NEIL SEDAKA,3
5176 DATA5176,WHAT WAS ELVIS PRESLEY'S FIRST HIT ?,3000
5177 DATA5177,A) BLUE SUEDE SHOES,1
5178 DATA5178,B) HOUND DOG,2
5179 DATA5179,C) HEARTBREAK HOTEL,3
5180 DATA5180,SIMON & GARFUNKEL WERE ORIGINALLY CALLED WHAT NAME ?,1500
5181 DATA5181,A) TOM & JERRY,1
5182 DATA5182,B) THE TEENAGERS,2
5183 DATA5183,C) THE NURK TWINS,3
5184 DATA5184,ERNEST EVANS'S STAGE NAME IS ?,3000
5185 DATA5185,A) FATS DOMINO,1
5186 DATA5186,B) MEATLOAF,2
5187 DATA5187,C) CHUBBY CHECKER,3
5188 DATA5188,WHAT GROUP DID ERIC BURDON START AFTER  LEAVING THE ANIMALS ?,200
5189 DATA5189,A) CREAM,1
5190 DATA5190,B) WAR,2
5191 DATA5191,C) DELANEY & BONNIE,3
5192 DATA5192,WHO HAD A BIG 60S HIT WITH THE SONG     BERNADETTE ?,1500
5193 DATA5193,A) THE TEMPTATIONS,1
5194 DATA5194,B) THE TRAMMPS,2
5195 DATA5195,C) THE HOLLIES,3
5196 DATA 5196,WHICH SONG DOES NOT MENTION GREYHOUND   BUSES ?,1500
5197 DATA5197,A) BUS STOP BY THE HOLLIES,1
5198 DATA5198,B) RAMBLIN MAN BY THE ALLMAN BROS,2
5199 DATA5199,C) AMERICA BY SIMON & GARFUNKEL,3
5200 DATA5200,WHERE IS THE THE BERNESE OBERLAND ?,1500
5201 DATA5201,A) SWITZERLAND,1
5202 DATA5202,B) GERMANY,2
5203 DATA5203,C) KENYA,3
5204 DATA5204,WHAT IS THE CAPITAL OF LUXEMBOURG ?,1500
5205 DATA5205,A) LUXEMBOURG CITY,1
5206 DATA5206,B) VADUZ,2
5207 DATA5207,C) THE HAGUE,3
```



```
5209 DATA5209,A) ST MARTIN,1
5210 DATA5210,B) GUAM,2
5211 DATA5211,C) ARUBA,3
5212 DATA5212,THE ORIENT EXPRESS ORIGINATES IN        WHAT CITY ?,3000
5213 DATA5213,A) PEKING,1
5214 DATA5214,B) LONDON,2
5215 DATA5215,C) PARIS,3
5216 DATA5216,ICELAND BELONGS TO WHAT COUNTRY ?,3000
5217 DATA5217,A) GREENLAND,1
5218 DATA5218,B) CANADA,2
5219 DATA5219,C) DENMARK,3
5220 DATA5220, SCHIPHOL AIRPORT SERVES WHICH CITY ?,3000
5221 DATA5221,A) JOHANNESBURG,1
5222 DATA5222,B) NICE,2
5223 DATA5223,C) AMSTERDAM,3
5224 DATA5224,WHICH CANADIAN PROVINCE IS THE FARTHEST    WEST ?,3000
5225 DATA5225,A) BRITISH COLUMBIA,1
5226 DATA5226,B) ALBERTA,2
5227 DATA5227,C) YUKON,3
5228 DATA5228,WHERE IS GRANT'S TOMB ?,2000
5229 DATA5229,A) OHIO,1
5230 DATA5230,B) NEW YORK,2
5231 DATA5231,C) WASHINGTON DC,3
5232 DATA5232,WHAT IS THE SECOND LARGEST CITY IN     FRANCE ?,1500
5233 DATA5233,A) LYON,1
5234 DATA5234,B) STRASBOURG,2
5235 DATA5235,C) MARSEILLE,3
5236 DATA5236,WHICH OF THE FOLLOWING IS NOT A CAPITAL   CITY ?,3000
5237 DATA5237,A) BUDAPEST,1
5238 DATA5238,B) BELGRADE,2
5239 DATA5239,C) MUNICH,3
```

Experiment 6.1/4

```
1 REM************************************
2 REM*        INITIALIZE                *
3 REM************************************
4 TC=0:TW=0:MC=0:MW=0:SC=0:SW=0:CC=0:CW=0:UC=0:UW=0
5 GC=0:GW=0
7 REM************************************
8 REM*        MAIN MENU                 *
9 REM************************************
10 PRINT"□"
11 KEY 1,"GOTO170"+CHR$(13)
12 KEY 2,"GOTO444"+CHR$(13)
13 KEY 3,"GOTO659"+CHR$(13)
14 KEY 4,"GOTO899"+CHR$(13)
15 KEY 5,"GOTO1139"+CHR$(13)
16 KEY 6,"GOTO4203"+CHR$(13)
17 KEY 7,"GOTO1374"+CHR$(13)
18 KEY 8,"GOTO7"+CHR$(13)
19 C=0:W=0
20 PRINT"□"
25 PRINT"            TRIVIA QUIZ"
30 PRINT"            ==========="
35 PRINT""
40 PRINT"            CATEGORIES"
45 PRINT""
50 PRINT"          F1: TELEVISION"
60 PRINT"          F2: MOVIES"
70 PRINT"          F3: SPORTS"
80 PRINT"          F4: COMICS"
85 PRINT"          F5: MUSIC "
86 PRINT"          F6: GEOGRAPHY
87 PRINT"          F7: EXIT"
90 PRINT""
```

```
100 PRINT"■■■■■■"
110 PRINT"PRESS THE APPROPRIATE FUNCTION KEY":END
140 REM************************************
150 REM***        TV ROUTINE             ***
160 REM************************************
170 PRINT"□"
200 LN=5000:C=0:W=0
201 COLOR4,3:COLOR5,3:COLOR0,2
210 PRINT"                TELEVISION"
220 PRINT""
230 Y=1
240 FOR T=1 TO 10
241 GOSUB 4000
250 NEXT T
376 REM************************************
377 REM*            TV SUBTOTAL           *
378 REM************************************
380 PRINT"          TELEVISION"
385 PRINT""
390 PRINT USING"IN THIS SECTION YOU HAVE ANSWERED ##";C
395 PRINT""
400 PRINT USING "CORRECTLY AND ## INCORRECTLY";W
405 IF C=0 THEN P=0:GOTO 415
410 P=(C/10)*100
415 TC=C:TW=W
416 PRINT"■■■■■■■■■■"
420 PRINT USING "YOUR PERCENTAGE IS ###.##%";P
424 RESTORE
425 FORQQ=1TO1500:NEXTQQ
435 GO TO 20
440 REM************************************
442 REM***       MOVIES ROUTINE          *
443 REM************************************
444 PRINT"□"
445 LN=5040:C=0:W=0
446 COLOR4,7:COLOR5,7:COLOR0,15
448 PRINT"                  MOVIES
450 PRINT""
455 Y=1
460 FOR T=1 TO 10
465 GOSUB 4000
475 NEXT T
600 REM************************************
601 REM*           MOVIES SUBTOTAL        *
602 REM************************************
605 PRINT"                     MOVIES"
610 PRINT""
615 PRINT USING "IN THIS SECTION YOU HAVE ANSWERED ##";C
618 PRINT""
620 PRINT USING "CORRECTLY AND ## INCORRECTLY ";W
624 PRINT""
625 PRINT"■■■■■■■■■■"
626 IF C=0THEN P=0:GOTO 635
630 P=(C/10)*100
635 MC=C:MW=W
640 PRINT USING "YOUR PERCENTAGE IS ###.##%";P
642 RESTORE
645 FORQQ=1TO1500:NEXT QQ
650 GO TO 20
653 REM************************************
656 REM*        SPORTS ROUTINE           *
658 REM************************************
659 PRINT"□"
660 LN=5080:C=0:W=0
661 COLOR4,1:COLOR5,2:COLOR0,12
665 PRINT"                  SPORTS"
670 PRINT""
675 Y=1
680 FOR T= 1 TO 10
685 GOSUB 4000
690 NEXT T
823 REM************************************
824 REM*           SPORTS SUBTOTAL        *
825 REM************************************
```



```
826 PRINT"          SPORTS"
830 PRINT""
835 PRINT USING"IN THIS SECTION YOU HAVEANSWERED ##";C
840 PRINT""
845 PRINT USING "CORRECTLY AND ## INCORRECTLY";W
850 PRINT""
853 IF C=0 THEN P=0:GOTO 860
855 P=(C/10)*100
860 SC=C:SW=W
862 PRINT"■■■■■■■■■■"
865 PRINT USING "YOUR PERCENTAGE IS ###.##%";P
867 RESTORE
870 FORQQ=1TO1500:NEXTQQ
880 GO TO 20
885 REM************************************
890 REM***       COMIC ROUTINE          **
895 REM************************************
899 PRINT"□"
900 LN=5120:C=0:W=0
901 COLOR4,5:COLOR5,3:COLOR0,11
905 PRINT"              COMICS"
910 PRINT""
915 Y=1
920 FOR T =1 TO 10
925 GOSUB 4000
935 NEXT T
1071 REM************************************
1072 REM*          COMIC SUBTOTAL         *
1073 REM************************************
1075 PRINT"                COMICS"
1080 PRINT""
1085 PRINT USING "IN THIS SECTION YOU HAVE ANSWERED ##";C
1090 PRINT""
1095 PRINT USING "CORRECTLY AND ## INCORRECTLY ";W
1100 PRINT"■■■■■■■■■■"
1105 IF C=0 THEN P=0:GOTO 1115
1110 P=(C/10)*100
1115 CC=C:CW=W
1116 PRINT USING "YOUR PERCENTAGE IS ###.##%";P
1127 RESTORE
1130 FORQQ=1TO1500:NEXT QQ
1135 GO TO 20
1136 REM************************************
1137 REM*           MUSIC ROUTINE          *
1138 REM************************************
1139 PRINT"□"
1140 LN=5160:C=0:W=0
1141 COLOR4,6:COLOR5,6:COLOR0,1
1145 PRINT"                MUSIC "
1150 PRINT""
1155 Y=1
1160 FOR T = 1 TO 10
1165 GOSUB 4000
1180 NEXT T
1306 REM************************************
1307 REM*          MUSIC SUBROUTINE        *
1308 REM************************************
1310 PRINT"           MUSIC "
1315 PRINT""
1320 PRINT USING"IN THIS SECTION YOU HAVE ANSWERED ##";C
1325 PRINT""
1330 PRINT USING "CORRECTLY AND ## INCORRECTLY";W
1335 PRINT""
1337 IF C=0 THEN P=0:GOTO 1345
1340 P=(C/10)*100
1345 UC=C:UW=W
1346 PRINT"■■■■■■■■■■"
1350 PRINT USING "YOUR PERCENTAGE IS ###.##%";P
1353 RESTORE
1355 FORQQ=1TO1500:NEXT QQ
1365 GO TO 20
1370 REM************************************
1371 REM***   EXIT AND FINAL TOTALS    *
1372 REM************************************
```

```
1374 PRINT"□"
1375 PRINT"         YOUR FINAL SCORE"
1376 GT=MC+TC+SC+CC+UC
1377 GW=MW+TW+SW+CW+UW
1378 PRINT"         ---- ----- -----"
1379 PRINT""
1380 PRINT"QUESTIONS ANSWERED "
1381 PRINT""
1382 PRINT USING "           CORRECTLY    :####";GT
1384 PRINT USING "           INCORRECTLY  :####";GW
1385 IF GT=0 THEN GP=0:GOTO 1390
1386 GP=(GT/(GW+GT))*100
1390 PRINT"■■■■■■■■■■"
1392 PRINT USING "YOU ANSWERED ###.##% CORRECTLY ";GP
1394 FORQQ=1TO1500:NEXT QQ
1400 PRINT"□" : END
1490 REM*************************************
1492 REM*       SUBROUTINE TO CHECK       *
1494 REM*          FOR'ANSWER "A"         *
1496 REM*************************************
1500 GET KEY A1$
1510 IF A1$="A" THEN GOTO 1560
1520 PRINT"WRONG ANSWER, THE CORRECT ANSWER IS : A"
1530 W=W+1
1540 GOTO1590
1560 PRINT"CORRECT !"
1570 C=C+1
1590 FORQQ=1TO1500:NEXTQQ
1600 PRINT""
1610 PRINT"□"
1620 RETURN
1900 REM*************************************
1992 REM*     SUBROUTINE TO CHECK         *
1994 REM*       FOR ANSWER "B"            *
1996 REM*************************************
2000 GETKEY A2$
2010 IF A2$="B" THEN GOTO 2060
2020 PRINT"WRONG ANSWER, THE CORRECT ANSWER IS : B"
2030 W=W+1
2040 GOTO2090
2060 PRINT"CORRECT !"
2070 C=C+1
2090 FORQQ=1TO1500:NEXT QQ
2100 PRINT"□"
2200 RETURN
2990 REM*************************************
2992 REM*     SUBROUTINE TO CHECK         *
2994 REM*       FOR ANSWER "C"            *
2996 REM*************************************
3000 GETKEY A3$
3010 IF A3$="C" THEN GOTO 3060
3020 PRINT"WRONG ANSWER, THE CORRECT ANSWER IS : C"
3030 W=W+1
3040 GOTO3090
3060 PRINT"CORRECT !"
3070 C=C+1
3090 FORQQ=1TO1500:NEXT QQ
3100 PRINT"□"
3200 RETURN
3900 REM*************************************
3992 REM*  SUBROUTINE TO READ AND LIST *
3994 REM*      QUESTIONS AND ANSWERS     *
3996 REM*************************************
4000 READ Q,Q$,A
4010 IF Q<>LN THEN GOTO 4000
4020 PRINT USING "QUESTION NO. ##";Y
4025 PRINT"":PRINT""
4030 Y=Y+1
4040 PRINT Q$
4050 PRINT""
4065 LN=LN+1
4070 FOR B=1 TO 3
4080 READ Q,Q$(B),D
```



```
4090 PRINT "      ";Q$(B)
4095 LN=LN+1
4100 PRINT""
4110 NEXT B
4115 IF A=1500 THEN GOSUB 1500
4116 IF A=2000 THEN GOSUB 2000
4117 IF A=3000 THEN GOSUB 3000
4120 RETURN
4200 REM*************************************
4201 REM*       GEOGRAPHY ROUTINE         *
4202 REM*************************************
4203 PRINT"□"
4204 LN=5200:C=0:W=0
4205 COLOR4,9:COLOR5,2:COLOR0,1
4206 PRINT"              GEOGRAPHY"
4207 PRINT""
4208 Y=1
4209 FOR T=1 TO 10
4210 GOSUB 4000
4211 NEXT T
4300 REM*************************************
4301 REM*       GEOGRAPHY SUBTOTAL        *
4302 REM*************************************4150
4303 PRINT"        GEOGRAPHY"
4304 PRINT"
4305 PRINT USING"IN THIS SECTION YOU HAVE ANSWERED ##";C
4306 PRINT""
4307 PRINT USING "CORRECTLY AND ## INCORRECTLY";W
4308 IF C=0 THEN P=0:GOTO 4310
4309 P=(C/10)*100
4310 TC=C:TW=W
4311 PRINT"■■■■■■■■■■"
4312 PRINT USING "YOUR PERCENTAGE IS ###.##%";P
4313 RESTORE
4315 FORQQ=1TO1500:NEXT QQ
4316 GO TO 20
4500 REM*************************************
4510 REM* DATA/ QUESTIONS AND ANSWERS *
4520 REM*************************************
5000 DATA5000,WHAT ROLE DID RALPH WAITE PLAY ?,1500
5001 DATA5001,A) JOHN WALTON ON THE WALTONS,1
5002 DATA5002,B) GEORGE APPLE ON APPLE'S WAY,2
5003 DATA5003,C) LARRY TRIT ON BEWITCHED,3
5004 DATA5004,WHO PLAYED THE JOKER ON THE BATMAN      SERIES ?,1500
5005 DATA5005,A) CESAR ROMERO,1
5006 DATA5006,B) BURGESS MEREDITH,2
5007 DATA5007,C) FRANK GORSHIN,3
5008 DATA5008,WHAT WAS THE NAME OF THE DOG ON THE       BRADY BUNCH ?,3000
5009 DATA5009,A) PRINCE,1
5010 DATA5010,B) ALICE,2
5011 DATA5011,C) TIGER,3
5012 DATA5012,NAME THE WESTERN SERIES THAT HELPED MAKEJAMES GARNER FAMOUS ?,2000
5013 DATA5013,A) WANTED DEAD OR ALIVE,1
5014 DATA5014,B) MAVERICK,2
5015 DATA5015,C) RAWHIDE,3
5016 DATA5016,ON THE SERIES GILLIGAN'S ISLAND WHAT WAS THE SKIPPER'S FULL NAME
?,2000
5017 DATA5017,A) ROY HINKLEY,1
5018 DATA5018,B) JONAS GRUMBY,2
5019 DATA5019,C) GINGER GRANT,3
5020 DATA5020,ON THE BEVERLY HILLBILLIES WHAT WAS     GRANNY'S FIRST NAME ?,1500
5021 DATA5021,A) DAISY,1
5022 DATA5022,B) PEARL,2
5023 DATA5023,C) ANNIE MAY,3
5024 DATA5024,WHAT IS THE ID NUMBER FOR THE FEDERATIONSTARSHIP ENTERPRISE?,2000
5025 DATA5025,A) AIR FORCE ONE,1
5026 DATA5026,B) NCC 1701,2
5027 DATA5027,C) FSS 1902,3
5028 DATA5028,WHO PLAYED THE GREEN HORNET'S VALET KATO ON  TV ?,3000
5029 DATA5029,A) ANN B DAVIS,1
5030 DATA5030,B) BURT WARD,2
5031 DATA5031,C) BRUCE LEE,3
5032 DATA5032,WHO PLAYED DENNIS (THE MENACE) MITCHELL ON THAT TV SHOW ?,1500
```

```
5033 DATA5033,A) JAY NORTH,1
5034 DATA5034,B) TONY DOW,2
5035 DATA5035,C) MASON REESE,3
5036 DATA5036,ON THE SERIES THE MUNSTERS WHAT WAS THE NAME OF THE DRAGON ?,2000
5037 DATA5037,A) PUFF,1
5038 DATA5038,B) SPOT,2
5039 DATA5039,C) CECIL,3
5040 DATA5040,WHICH LEAD ACTOR NEVER CO-STARRED        WITH A LIVE APE ?,3000
5041 DATA5041,A) CLINT EASTWOOD,1
5042 DATA5042,B) RONALD REAGAN,2
5043 DATA5043,C) CLIFF ROBERTSON,3
5044 DATA5044,WHICH OF THE FOLLOWING WOODY ALLEN       MOVIES WAS NOT FILMED IN
LACK & WHITE ?,2000
5045 DATA5045,A) MANHATTAN,1
5046 DATA5046,B) BANANAS,2
5047 DATA5047,C) ZELIG,3
5048 DATA5048,WHICH MOVIE DID NOT TAKE PLACE IN        PHILADELPHIA ?,2000
5049 DATA5049,A) TRADING PLACES,1
5050 DATA5050,B) CHU CHU & THE PHILLY FLASH,2
5051 DATA5051,C) ROCKY,3
5052 DATA5052,WHICH OF THE FOLLOWING WAS NOT A DIRTY  HARRY MOVIE ?,3000
5053 DATA5053,A) SUDDEN IMPACT,1
5054 DATA5054,B) MAGNUM FORCE,2
5055 DATA5055,C) TIGHTROPE,3
5056 DATA5056,WHAT WAS THE FIRST JAMES BOND MOVIE ?,3000
5057 DATA5057,A) THUNDERBALL,1
5058 DATA5058,B) GOLDFINGER,2
5059 DATA5059,C) DR. NO,3
5060 DATA5060,WHICH KING ARTHUR MOVIE DID NOT FEATURE A MAGICIAN NAMED MERLIN ?
2000
5061 DATA5061,A) EXCALIBUR,1
5062 DATA5062,B) MONTY PYTHON & THE HOLY GRAIL,2
5063 DATA5063,C) THE SWORD AND THE STONE,3
5064 DATA5064,WHAT MOTEL WAS FEATURED IN ALFRED        HITCHCOCK'S PSYCHO ?,1500
5065 DATA5065,A) BATES MOTEL,1
5066 DATA5066,B) ROACH MOTEL,2
5067 DATA5067,C) MOTEL CALIFORNIA,3
5068 DATA5068,WHICH OF THE FOLLOWING WAS NOT A CHARAC-TER PORTRAYED BY ROBERT D
NIRO ?,3000
5069 DATA5069,A) JAKE LAMOTTA,1
5070 DATA5070,B) RUPERT PUPKIN,2
5071 DATA5071,C) SONNY CORLEONE,3
5072 DATA5072,WHAT FRATERNITY DID JOHN BELUSHI BELONG TO IN ANIMAL HOUSE ?,3000
5073 DATA5073,A) PHI OMEGA,1
5074 DATA5074,B) ZBT,2
5075 DATA5075,C) DELTA HOUSE,3
5076 DATA5076,WHO WON AN OSCAR FOR THE YEAR OF LIVING DANGEROUSLY ?,3000
5077 DATA5077,A) MEL GIBSON,1
5078 DATA5078,B) SIGOURNEY WEAVER,2
5079 DATA5079,C) LINDA HUNT,3
5080 DATA5080,WHAT CITY HAD THE 1ST PRO BASEBALL TEAM THE AMERICAN RED STOCKING
,2000
5081 DATA5081,A) CHICAGO,1
5082 DATA5082,B) CINCINNATI,2
5083 DATA5083,C) BOSTON,3
5084 DATA5084,WHAT DID THE Y IN Y A TITTLE STAND FOR ?,1500
5085 DATA5085,A) YELBERTON,1
5086 DATA5086,B) YORKTOWN,2
5087 DATA5087,C) YELLOW,3
5088 DATA5088,WHAT TWO TEAMS PLAYED THE FIRST INTER-  COLLEGIATE FOOTBALL GAME
,2000
5089 DATA5089,A) FRANKLIN AND MARSHALL,1
5090 DATA5090,B) HARVARD AND MCGILL,2
5091 DATA5091,C) HARVARD AND PRINCETON,3
5092 DATA5092,ANAHEIM STADIUM IS NOT THE HOME STADIUM FOR WHICH TEAM ?,3000
5093 DATA5093, A) CALIFORNIA ANGELS,1
5094 DATA5094, B) LA RAMS,2
5095 DATA5095, C) LA RAIDERS,3
5096 DATA5096,WHO WERE BOB AND CAROL TED AND DALLAS INSPORTS IN THE LATE 60S ?,
000
5097 DATA5097,A) TENNIS MIXED DOUBLES,1
5098 DATA5098,B) BOSTON BRUINS DEFENSEMEN,2
5099 DATA5099,C) LEADING SHOW DOGS,3
```



```
5100 DATA5100,WHICH OF THE FOLLOWING DID NOT GO  TO   THE PHILLIES IN A TRADE FO
R DICK ALLEN ?,2000
5101 DATA5101,A) TIM MCCARVER,1
5102 DATA5102,B) CURT FLOOD,2
5103 DATA5103,C) WILLIE MONTANEZ,3
5104 DATA5104,WHICH NBA FRANCHISE WAS ORIGINALLY THE  SYRACUSE NATIONALS ?,3000
5105 DATA5105,A) DETROIT PISTONS,1
5106 DATA5106,B) NEW YORK KNICKS,2
5107 DATA5107,C) PHILADELPHIA 76ERS,3
5108 DATA5108,WHO SAID NICE GUYS FINISH LAST ?,1500
5109 DATA5109,A) LEO DOROCHER,1
5110 DATA5110,B) VINCE LOMBARDI,2
5111 DATA5111,C) GUY LAFLEUR,3
5112 DATA5112,THE GREEN JACKET IS CONNECTED WITH WHAT GOLFING EVENT ?,2000
5113 DATA5113,A) BRITISH OPEN,1
5114 DATA5114,B) MASTERS TOURNAMENT,2
5115 DATA5115,C) US OPEN,3
5116 DATA5116,WHICH LOCATION WAS NOT A SITE FOR A     FRAZIER AND ALI TITLE FIGH
T ?,2000
5117 DATA5117, A) MANILA,1
5118 DATA5118, B) ZIMBABWE,2
5119 DATA5119, C) NEW YORK CITY,3
5120 DATA5120, WHAT DOES CHARLIE BROWN'S FATHER DO FOR A LIVING ?,1500
5121 DATA5121, A) CUT HAIR,1
5122 DATA5122, B) DELIVER MAIL,2
5123 DATA5123, C) DELIVER MILK,3
5124 DATA5124,WHAT CHARACTER FROM ANOTHER COMIC STRIP IS RELATED TO BEETLE BAILE
Y ?,1500
5125 DATA5125,A) LOIS OF HI & LOIS,1
5126 DATA5126,B) SAD SACK,2
5127 DATA5127,C) LOLLY,3
5128 DATA5128,WHAT COMIC STRIP FEATURED SHMOOS ?,1500
5129 DATA5129,A) LIL ABNER,1
5130 DATA5130,B) POGO,2
5131 DATA5131,C) BARNEY GOOGLE,3
5132 DATA5132, THE CHARACTER BD IN DOONESBURY WAS BASED ON WHOM ?,3000
5133 DATA5133,A) CARTOONIST GARRY TRUDEAU,1
5134 DATA5134,B) SINGER BOB DYLAN,2
5135 DATA5135,C) YALE QUARTERBACK BRIAN DOWLING,3
5136 DATA5136,IN WHAT COMIC STRIP WILL YOU FIND A DOG NAMED HOTDOG ?,1500
5137 DATA5137,A) ARCHIE,1
5138 DATA5138,B) DENNIS THE MENACE,2
5139 DATA5139,C) FRED BASSETT,3
5140 DATA5140,IN WHAT LONG RUNNING COMIC STRIP WILL   YOU FIND A LION TAMER'S CL
UB ?,1500
5141 DATA5141,A) MUTT AND JEFF,1
5142 DATA5142,B) THE FLINTSTONES,2
5143 DATA5143,C) BLONDIE,3
5144 DATA5144,WHAT COMIC STRIP LATER BECAME SNUFFY    SMITH ?,2000
5145 DATA5145,A) GASOLINE ALLEY,1
5146 DATA5146,B) BARNEY GOOGLE,2
5147 DATA5147,C) THE TOONERVILLE TROLLEY,3
5148 DATA5148,WHAT OLD CARTOON STRIP FEATURED MAGGIE & JIGGS ?,3000
5149 DATA5149,A) THE JIGG IS UP,1
5150 DATA5150,B) MY LIL MAGGIE,2
5151 DATA5151,C) BRINGING UP FATHER,3
5152 DATA5152,WHAT WAS CAPTAIN MARVEL'S NICKNAME ?,2000
5153 DATA5153,A) THE RED TORPEDO,1
5154 DATA5154,B) THE BIG RED CHEESE,2
5155 DATA5155,C) THE RED BEE,3
5156 DATA5156,WHICH SUPER HERO GOT HIS POWERS OTHER   THAN BY NUCLEAR ACCIDENT?,
3000
5157 DATA5157,A) SPIDERMAN,1
5158 DATA5158,B) THE INCREDIBLE HULK,2
5159 DATA5159,C) THE FLASH,3
5160 DATA5160,WHICH OF THE FOLLOWING IS NOT A BEATLES SONG ?,1500
5161 DATA5161,A) SUNNY AFTERNOON,1
5162 DATA5162,B) SUN KING,2
5163 DATA5163,C) I'LL FOLLOW THE SUN,3
5164 DATA5164,WHO SANG THE HIT THEME SONG FROM THE    JAMES BOND MOVIE GOLDFINGE
R ?,1500
5165 DATA5165,A) SHIRLEY BASSEY,1
5166 DATA5166,B) SHEENA EASTON,2
```

```
5167 DATA5167,C) NANCY SINATRA,3
5168 DATA5168,PETER NOONE WAS THE STAR OF WHICH        ENGLISH SINGING GROUP ?,15
00
5169 DATA5169,A) HERMAN'S HERMITS,1
5170 DATA5170,B) THE SEARCHERS,2
5171 DATA5171,C) THE MOODY BLUES,3
5172 DATA5172,WHO WROTE THE SONNY AND CHER HIT SONG     THE BEAT GOES ON ?,2000
5173 DATA5173,A) NEIL DIAMOND,1
5174 DATA5174,B) BOB DYLAN,2
5175 DATA5175,C) NEIL SEDAKA,3
5176 DATA5176,WHAT WAS ELVIS PRESLEY'S FIRST HIT ?,3000
5177 DATA5177,A) BLUE SUEDE SHOES,1
5178 DATA5178,B) HOUND DOG,2
5179 DATA5179,C) HEARTBREAK HOTEL,3
5180 DATA5180,SIMON & GARFUNKEL WERE ORIGINALLY CALLED WHAT NAME ?,1500
5181 DATA5181,A) TOM & JERRY,1
5182 DATA5182,B) THE TEENAGERS,2
5183 DATA5183,C) THE NURK TWINS,3
5184 DATA5184,ERNEST EVANS'S STAGE NAME IS ?,3000
5185 DATA5185,A) FATS DOMINO,1
5186 DATA5186,B) MEATLOAF,2
5187 DATA5187,C) CHUBBY CHECKER,3
5188 DATA5188,WHAT GROUP DID ERIC BURDON START AFTER  LEAVING THE ANIMALS ?,2000
5189 DATA5189,A) CREAM,1
5190 DATA5190,B) WAR,2
5191 DATA5191,C) DELANEY & BONNIE,3
5192 DATA5192,WHO HAD A BIG 60S HIT WITH THE SONG      BERNADETTE ?,1500
5193 DATA5193,A) THE TEMPTATIONS,1
5194 DATA5194,B) THE TRAMMPS,2
5195 DATA5195,C) THE HOLLIES,3
5196 DATA 5196,WHICH SONG DOES NOT MENTION GREYHOUND   BUSES ?,1500
5197 DATA5197,A) BUS STOP BY THE HOLLIES,1
5198 DATA5198,B) RAMBLIN MAN BY THE ALLMAN BROS,2
5199 DATA5199,C) AMERICA BY SIMON & GARFUNKEL,3
5200 DATA5200,WHERE IS THE THE BERNESE OBERLAND ?,1500
5201 DATA5201,A) SWITZERLAND,1
5202 DATA5202,B) GERMANY,2
5203 DATA5203,C) KENYA,3
5204 DATA5204,WHAT IS THE CAPITAL OF LUXEMBOURG ?,1500
5205 DATA5205,A) LUXEMBOURG CITY,1
5206 DATA5206,B) VADUZ,2
5207 DATA5207,C) THE HAGUE,3
5208 DATA5208,WHICH IS NOT A CARRIBEAN ISLAND ?,2000
5209 DATA5209,A) ST MARTIN,1
5210 DATA5210,B) GUAM,2
5211 DATA5211,C) ARUBA,3
5212 DATA5212,THE ORIENT EXPRESS ORIGINATES IN          WHAT CITY ?,3000
5213 DATA5213,A) PEKING,1
5214 DATA5214,B) LONDON,2
5215 DATA5215,C) PARIS,3
5216 DATA5216,ICELAND BELONGS TO WHAT COUNTRY ?,3000
5217 DATA5217,A) GREENLAND,1
5218 DATA5218,B) CANADA,2
5219 DATA5219,C) DENMARK,3
5220 DATA5220, SCHIPHOL AIRPORT SERVES WHICH CITY ?,3000
5221 DATA5221,A) JOHANNESBURG,1
5222 DATA5222,B) NICE,2
5223 DATA5223,C) AMSTERDAM,3
5224 DATA5224,WHICH CANADIAN PROVINCE IS THE FARTHEST    WEST ?,3000
5225 DATA5225,A) BRITISH COLUMBIA,1
5226 DATA5226,B) ALBERTA,2
5227 DATA5227,C) YUKON,3
5228 DATA5228,WHERE IS GRANT'S TOMB ?,2000
5229 DATA5229,A) OHIO,1
5230 DATA5230,B) NEW YORK,2
5231 DATA5231,C) WASHINGTON DC,3
5232 DATA5232,WHAT IS THE SECOND LARGEST CITY IN       FRANCE ?,1500
5233 DATA5233,A) LYON,1
5234 DATA5234,B) STRASBOURG,2
5235 DATA5235,C) MARSEILLE,3
5236 DATA5236,WHICH OF THE FOLLOWING IS NOT A CAPITAL   CITY ?,3000
5237 DATA5237,A) BUDAPEST,1
5238 DATA5238,B) BELGRADE,2
5239 DATA5239,C) MUNICH,3
```


UNIT:7

Experiment 7.1

```
5 PRINT"SCREEN COLOR SELECTION"
10 INPUT"SELECT BORDER COLOR BY NUMBER (1-16)";C
20 COLOR 4,C
30 FOR X1=1 TO 16
40 COLOR 0,X1
50 FOR X2=1 TO 16
60 IF X1=X2 THENX2=X2+1:IF X2>16 THEN X2=1
70 COLOR5,X2
80 PRINT"J"
90 PRINT"HOW DOES THIS LOOK?"
100 INPUT"DO YOU WANT TO KEEP THIS                  COLOR COMBINATION";K$
110 IF K$="Y" OR K$="YES" THEN END
120 PRINT""
130 IF X2=1 THEN BEGIN
135 IF X1 =16 THEN BEGIN:PRINT"LAST BACKGROUND COLOR AVAILABLE"
136 INPUT"DO YOU WISH TO START AGAIN";C$:BEND:GOTO145
140 INPUT "DO YOU WANT TO SKIP TO THE NEXT BACKGROUND COLOR";C$
145 BEND
150 IF C$="Y" OR C$="YES" THEN C$="":GOTO170
160 NEXT X2
170 NEXT X1
180 GOTO 10
```

Experiment 7.1/4

```
5 PRINT"SCREEN COLOR SELECTION"
10 INPUT"SELECT BORDER COLOR BY NUMBER (1-16)";C
20 COLOR 4,C
30 FOR X1=1 TO 16
40 COLOR 0,X1
50 FOR X2=1 TO 16
60 IF X1=X2 THENX2=X2+1:IF X2>16 THEN X2=1
70 COLOR1,X2
80 PRINT"J"
90 PRINT"HOW DOES THIS LOOK?"
100 INPUT"DO YOU WANT TO KEEP THIS                    COLOR COMBINATION";K$
110 IF K$="Y" OR K$="YES" THEN END
120 PRINT""
130 IF X2=1 THEN 140
135 IF X1 =16 THEN :PRINT"LAST BACKGROUND COLOR AVAILABLE"
137 GOTO 150
140 INPUT"WANT TO SKIP TO A NEW BACKGROUND COLOR";C$
150 IF C$="Y" OR C$="YES" THEN GOTO170
160 NEXT X2
170 NEXTX1
180 GOTO10
```

Experiment 7.2a

```
10 COLOR0,1
20 GRAPHIC1,1
30 COLOR1,8:COLOR4,1
40 FOR X=0 TO 319
50 Y=199-(X-160)12/160
70 DRAW1,X,Y
80 NEXT X
90 END
```

Experiment 7.2b

```
10 COLOR0,3
20 GRAPHIC1,1
30 COLOR1,2:COLOR4,3
40 FOR J=0 TO 359
50 X=160+80*SIN(J*π/180)
60 Y=100+60*COS(J*π/180)
70 DRAW1,X,Y
80 NEXT J
90 END
```

Experiment 7.2c

```
10 COLOR0,15
20 GRAPHIC1,1
30 COLOR1,7:COLOR4,15
40 FOR X=0 TO 319
50 Y=100+99*SIN(X/20)
70 DRAW1,X,Y
80 NEXT X
90 END
```

UNIT:8

Experiment 8.1a

```
15 COLOR0,3:COLOR4,3:COLOR1,1
20 GRAPHIC1,1
25 WIDTH 2
30 X=0:Y=0:F=1
40 FOR Y=0 TO 200 STEP10
41 F=-(F)
50 A=X:B=Y:C=A+20:D=B+10
60 FOR HH=1TO16
70 BOX1,A,B,C,D
90 A=A+20:C=C+20
100 NEXT HH
195 X=0
200 IF F<0 THEN X=10:ELSE 210
210 NEXT Y
220 BOX 1,0,0,319,199
230 GETKEY Z$:GRAPHIC0
```

Experiment 8.1b

```
5 GRAPHIC4,1:COLOR1,8:COLOR0,1:COLOR4,1
10 BOX1,150,150,75,75
20 BOX1,100,100,25,25
30 DRAW1,150,150TO100,100
40 DRAW1,75,75TO25,25
50 DRAW1,75,150TO25,100
60 DRAW1,150,75TO100,25
70 SLEEP10:GRAPHICCLR:END
```


Experiment 8.2a

```
10 GRAPHIC3,1:C1=12
15 COLOR0,1
20 COLOR4,1
25 COLOR1,C1
30 X=10:Y=20
35 CIRCLE1,X,Y,12,20
40 FOR A=0 TO15 STEP5
45 CIRCLE1,X,Y,12,20,,,A
50 X=X+10:Y=Y+6
60 NEXT A
65 FOR A=15 TO 60 STEP15
70 CIRCLE1,X,Y,12,20,,,A
75 X=X+5:Y=Y+5
80 NEXT A
90 FOR A=60TO90 STEP 15
95 CIRCLE1,X,Y,12,20,,,A
100 X=X+2:Y=Y+7
105 NEXT A
110 DOUNTIL Y>180
120 CIRCLE1,X,Y,12,20,,,90
130 X=X+2:Y=Y+10
140 LOOP
150 IF C1=1 THEN GRAPHIC 0:END
160 C1=1:GOTO25
```

Experiment 8.2b

```
5 Z=1
10 GRAPHIC3,1
15 S=80:T=100
20 FOR Q=1 TO 9
31 X=10
32 Y=8
45 IF Z>0 THEN W=1
46 IF Z<0 THEN W=2
50 CIRCLE W,80,100,S,T
60 PAINT W,80,100
65 S=S-Y:T=T-X
67 Z=(-Z)
70 NEXT Q
80 GETKEYA$:GRAPHIC0
```

Experiment 8.3 (a,b&c)

```
10 COLOR0,15;COLOR4,1
15 GRAPHIC4,1
16 FORX=60 TO 0 STEP -4
20 CIRCLE2,75,90, X, X,,,,Z
22 NEXT X
25 FOR X= 2 to 60 STEP 4
30 CIRCLE3,75,90, X, X,,,,Z
42 NEXT X
50 FORX=60 to 0 STEP -4
60 CIRCLE0,75,90, X, X,,,,Z
70 NEXT X
80 FOR X= 0 TO 60 STEP 4
90 CIRCLE3,75,90 X, X,,,,Z
100 NEXT X
110 FORX=60 to 0 STEP -4
120 CIRCLE0,75,90, X, X,,,,Z
```

```
130 NEXT X
140 FOR X= 0 to 60 STEP 4
150 CIRCLE1,75,90, X, X,,,,Z
160 NEXT X
170 FORX=62 to 0 STEP -4
180 CIRCLE0,75,90, X, X,,,,Z
190 NEXT X
200 FOR X= 0 to 60 STEP 4
210 CIRCLE0,75,90, X, X,,,,Z
220 NEXT X
```

Additional line for correct shape:
Answer A:
18 Z = 20
Answer B:
18 Z = 90
Answer for C
18Z = 27.69

Experiment 8.3d

```
10 COLOR1,1:COLOR4,2:COLOR0,2
20 GRAPHIC1,1
30 CIRCLE1,160,100,90,45
40 FOR A1=0 TO 360 STEP 30
50 A2=A1+15
60 CIRCLE0,160,100,90,45,A1,A2
70 NEXT A1
80 GETKEYA$:GRAPHIC0
```

Experiment 8.4a

```
10 COLOR0,2
20 GRAPHIC1,1
30 COLOR1,1
40 REM******BOXES IN DIFFERENT PLACES
50 FOR J=12 TO282 STEP 30
60 FOR K=3 TO183 STEP 15
70 BOX1,J,K,J+25,K+12
80 NEXTK,J
90 PAINT1,1,1
100 GETKEY A$:GRAPHIC 0
```

Experiment 8.4b

```
10 GRAPHICCLR:GRAPHIC3,1
20 COLOR0,1
30 COLOR4,1
40 COLOR1,3
45 FORX=1 TO 160 STEP 20
50 FORY=1 TO 180 STEP 30
60 BOX1,X,Y,X+20,Y+30
65 NEXT Y,X
70 DRAW1,159,1TO159,180
80 FOR X=1 TO 160 STEP 40
85 FOR Y=1 TO 180 STEP 60
90 PAINT1,X+1,Y+1
100 NEXT Y,X
105 FOR X=20 TO 140 STEP 40
110 FOR Y=30 TO 150 STEP 60
115 PAINT1,X+2,Y+2
120 NEXT Y,X
125 GETKEY A$:GRAPHIC 0
```

Experiment 8.4c

```
5 COLOR4,1:COLOR0,1
10 GRAPHIC4,1
15 SLEEP1
20 REM********BASIC HOUSE*********
30 BOX3,20,90,50,120
35 DRAW2,20,90TO35,60TO50,90:PAINT3,25,85,1:PAINT2,40,100,1
40 REM*******HOUSE SIDES*********
45 SLEEP1
50 DRAW2,35,60TO 80,78
55 DRAW2,50,90TO90,94
60 DRAW3,50,120TO90,109
70 DRAW2,50,90TO90,94
80 DRAW3,80,78TO90,94TO90,109:PAINT3,65,80,1:PAINT2,65,115,1
85 REM*******HOUSE WINDOWS*********
90 SLEEP1
95 BOX3,55,98,70,106,1
100 BOX3,75,99,86,105,1
110 DRAW3,62,106TO62,98
115 DRAW3,55,102TO70,102
120 DRAW3,80,99TO80,104
125 DRAW3,75,102TO86,102
130 SLEEP1
135 REM**********DOOR**************
140 BOX3,32,105,38,120:PAINT3,33,110
145 DRAW2,36,112
150 REM********MORE WINDOWS********
155 BOX3,23,95,29,103
160 BOX3,41,95,47,103
165 DRAW3,26,95TO26,103
170 DRAW3,23,99TO29,99
175 DRAW3,44,95TO44,103
180 DRAW3,41,99TO47,99
185 REM*********ROOFING*************
190 SLEEP1
195 DRAW2,83,82TO40,69
200 DRAW2,42,74TO85,85
210 DRAW2,44,79TO87,89
215 DRAW2,47,84TO89,92
220 SLEEP1:COLOR1,6
225 REM********HORIZON***************
230 DRAW1,160,100TO90,95:DRAW1,20,89TO1,85TO1,150TO160,150TO160,100
235 PAINT1,110,110,1:REM PAINT GRASS GREEN   ← Line with answer
240 COLOR4,7:COLOR0,7:SLEEP1
245 CIRCLE2,90,50,10,10
250 PAINT2,90,50
255 CIRCLE2,75,30,15,15:PAINT2,75,30
260 CIRCLE2,95,35,10,10:PAINT2,95,35
270 CIRCLE2,140,35,15,10:PAINT2,140,35
275 CIRCLE2,118,25,12,12:PAINT2,118,25
280 COLOR1,8
285 CIRCLE1,20,10,5,5:PAINT1,19,11
1000 GETKEYA$:GRAPHICCLR:END
```

UNIT:9

Experiment 9.1

```
10 GRAPHIC3,1
20 C1=INT(RND(1)*8+1)
30 C2=INT(RND(1)*8+1):IF C2=C1 THEN GOTO 30
40 C3=INT(RND(1)*8+1):IF C3=C2 OR C3=C1 THEN GOTO 40
60 COLOR1,C1:COLOR2,C2:COLOR3,C3:COLOR4,1:COLOR0,1
70 X=10
80 Y=10
90 XL=4: YL=2
100 BOX1,X,Y,X+45,Y+45:X=X+L:Y=Y+L
110 FOR A=1 TO 90 STEP 10
120 BOX2,X,Y,X+45,Y+45:X=X+XL:Y=Y+YL
130 BOX3,X,Y,X+45,Y+45:X=X+XL:Y=Y+YL
140 BOX1,X,Y,X+45,Y+45:X=X+XL:Y=Y+YL
150 NEXT A
155 Y=Y+5
160 FOR A=90 TO 0 STEP -10
170 BOX2,X,Y,X+45,Y+45:X=X-XL:Y=Y+YL
180 BOX3,X,Y,X+45,Y+45:X=X-XL:Y=Y+YL
190 BOX1,X,Y,X+45,Y+45:X=X-XL:Y=Y+YL
200 NEXT A
205 FOR D=1 TO1000:NEXT D
210 GRAPHIC CLR:GOTO10
```

Experiment 9.2a

```
10 GRAPHIC4,1
20 COLOR0,1:COLOR1,5:COLOR2,8:COLOR3,6  ← Line with answer
30 FORA=1 TO 45:REM DRAW TRI-COLORED RING
40 BOX1,30,30,45,45,A
50 BOX2,30,30,45,45,(22.5+A)
60 BOX3,30,30,45,45,(45+A)
70 NEXT A
80 SSHAPE A$,20,20,50,50:REM SAVE RING SHAPE
90 BOX1,50,20,80,50,,1
100 BOX2,80,20,110,50,,1
110 BOX3,110,20,140,50,,1
115 REM REDRAW SHAPE ON COLORED AREAS
120 GSHAPE A$,62,20,4
130 GSHAPEA$,92,20,4
140 FOR DE=1TO200:NEXT DE:REMDELAY LOOP
150 GSHAPE A$,62,20,4
160 GSHAPE A$,92,20,4
```

Experiment 9.2b

```
10 GRAPHIC 2,1
20 COLOR 0,1:COLOR1,8
30 INPUT"INPUT X CHARACTER POSITION OF YOUR TEXT";X
40 INPUT"INPUT Y CHARACTER POSITION OF YOUR TEXT";Y
50 INPUT"TEXT STRING (LESS THAN 20 LETTERS)";A1$
60 CHAR1,X,Y,A1$
70 SSHAPEA$,(X*8),(Y*8),(X+20)*8,(Y*8)+7
80 INPUT"INPUT X PIXEL POSITION";X1
90 INPUT"INPUT Y PIXEL POSITION";Y1
100 GSHAPE A$,X1,Y1
```


Experiment 9.3

C128

```
10 COLOR0,1
20 COLOR1,7
30 COLOR4,1
40 GRAPHIC1,1
45 FOR I=80TO240 STEP10
60 CIRCLE1,I,100,75,75
65 NEXT
70 SCALE1,400,300
80 COLOR1,3
90 FOR I=100TO300 STEP10
100 CIRCLE1,I,150,75,75
110 NEXT
120 SLEEP3:GRAPHIC 0
```

UNIT:10

Experiment 10.1a

```
10 VOL 7
20 SOUND3,400,600
30 FOR J=1 TO 10
40 SOUND1, 875,5
50 SOUND1,1020,60
60 NEXT J
70 FOR X=400 TO 900
80 SOUND3,X,1
90 NEXT X
100 FOR V=8 TO 0 STEP -1
110 VOLV
120 SOUND3,950,60
130 NEXT V
```

Experiment 10.1b

```
3000 REM RIGHT ANSWER
3005 FOR L=1 TO 3
3010 X=834
3020 FOR C=1TO 4
3030 SOUND2,X,3
3040 SOUND3,1023,1
3050 X=X+30
3060 NEXT C
3070 NEXT L
4000 REM WRONG ANSWER
4010 SOUND1,50,60
4020 SOUND3,1010,60
4030 FOR J=1 TO 1000:NEXT J
```

Experiment 10.2

```
10 VOL7
20 READ X,Y:REM GET VALUES FOR NOTE
```

```
30 IF X=0 THEN END
40 SOUND1,X,Y:REM PLAY NOTE
50 SOUND1,1020,2:REM CREATE GAP
60 GOTO 20:REM NEXT NOTE
100 DATA 873,5,854,5,834,5,854,5,834,5,822,5,834,10,722,5,739,5
110 DATA 770,5,798,5,770,5,722,5,770,10,834,5,854,5
120 DATA 873,10,873,10,873,5,854,5,834,5,854,5
130 DATA 873,10,854,10,854,10,873,5,854,5
140 DATA 834,5,854,5,834,5,822,5,834,10,722,5,739,5
150 DATA 770,5,798,5,770,5,722,5,770,10,834,5,854,5
160 DATA 873,5,897,10,911,5,897,5,873,5,834,5,854,5
170 DATA 873,10,854,10,834,10,1020,10
180 DATA 873,5,897,10,873,5,897,10,897,10
190 DATA 873,5,897,10,873,5,897,20
200 DATA 881,5,911,10,881,5,911,10,911,10
210 DATA 881,5,911,10,881,5,911,10,911,5,923,5
220 DATA 929,10,929,10,897,10,897,10
230 DATA 873,10,873,10,854,10,834,5,854,5
240 DATA 873,5,897,10,911,5,897,5,873,5,834,5,854,5
250 DATA 873,10,854,10,834,10,0,0
```

Experiment 10.3

```
10 COLOR0,5:COLOR4,2 :REM COLOR BACKGROUND,BORDER
20 PRINT"****************************************"
30 PRINT"◥****************************************"
40 FOR X=1 TO 21:REM CREATE BORDER
50 PRINT"*                                       *"
60 NEXT X
70 PRINT"****************************************"
80 PRINT"■PRESS ANY KEY FOR NEXT SOUND
90 WINDOW2,3,38,21
100 VOL 7
110 V=INT(RND(0)*3)+1:REM VOICE
120 F=INT(RND(0)*65535)+1:REM FREQ
130 D =INT(RND(0)*32767)+1:REM DUR
140 DR=INT(RND(0)*3):REM DIR
150 M=INT(RND(0)*65535)+1:REM MIN
160 S=INT(RND(0)*32767)+1:REM STEP
170 W=INT(RND(0)*3)+1:REM WAVEFORM
180 P=INT(RND(0)*4095)+1:REM PULSE W
190 C=INT(RND(0)*16)+1:REM CURSOR COLOR
200 IF C=5 THEN 190:REM REJECT BACKGROUND COLOR FOR CURSOR
210 COLOR5,C
220 PRINTV;F;D;DR;M;S;W;P:PRINT:PRINT
230 SOUND V,F,D,DR,M,S,W,P:REM MAKE SOUND
240 GETKEY A$: REM WAIT FOR USER
250 SOUNDV,0,0,DR,0,0,W,P :REM STOP SOUND
260 GOTO 110: REM NEXT SOUND
```

Experiment 10.4

```
10 TEMPO 8
20 FOR X= 1 TO 2
30 PLAY"03 IA 04 SD#FED 03 IBB AA 004 0D IEE #F#F
40 PLAY"SE#FED 03 IBA 04 SD#FED 03 IB B A A 04 0D SD#CD 03 AB 04 D 03 AG
50 PLAY"04 #F DE#C ID
60 NEXT X:REM REPEAT FIRST HALF
70 FOR Y= 1TO 2
80 PLAY"S#FGAG#FAG#FEG#FED#FE#C 03 IA"
90 PLAY"04 SD#CD#FEDEG#FE#FA IE S#FG AG#FAG#FEG #FED#FE#C 03 IA"
100 PLAY"04 SD#CD 03A B 04 D 03 AG#F 04 DE#C ID
110 NEXT Y:REM REPEAT SECOND HALF
```


Experiment 10.5

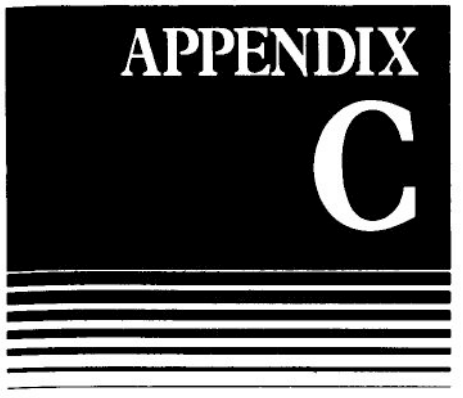
```
10 FILTER1400,1,0,0,5
20 ENVELOPE0,4 ,5,9,3,2,2000
30 ENVELOPE8,0,3,9,6,2,2000
40 TEMPO 15
50 PLAY"V1T0 X1 04 0A.0A V2T8 X1 .0#F V1T0 IG V2T8 IE V1T0 0#F V2T8 D
60 PLAY"V1T0 0G V2T8 E V1T0 A V2T8 #F V1T0 G V2T8 E
70 PLAY"V1T0 .0#F V2T8 D V1T0 IE V2T8 03 A V1T0 04 0D V2T8 03 0#F
80 PLAY "V2T8 03HA  V1T0 04 0#C 03 A V1T0 A V2T8 G
90 PLAY "V1T0 04 .0D V2T8 03 #F V1T0 04I#C V2T8 03 A V1T0 04 0D V2T8 03 #F
100 PLAY"V1 04 E V2T8 03 G V1T0 04 #F V2T8 03 A V1T0 04 G V2T8 03B
110 PLAY"V2T8 HA V1T0 04 0#F E V2T8 03 #F V1T0 04 D
120 PLAY"V1T0 04 HE V2T8 03 0G 04 #C V1T0 A V2T8 #C
130 PLAY"V1T0 .0A V2T8 D V1T0 IG V2T8 03 B V1T0 03 0#F V2T8 03 A
140 PLAY"V2T8 03  HB V1T0 04 0GA V2T8  #C V1T0 G
150 PLAY"V2T8 04 HD V1T0 .0#F IE V2T8 03 0A V1T0 04 D
160 PLAY"V2T8 03 HG V1T0 04 0#C 03 A V2T8 E V1T0 A
170 PLAY"V1T0 04 .0D V2T8 03 #F V1T0 04 ID V2T8 03 E V1T0 04 0D V2T8 03 #F
180 PLAY"V1T0 04 0E V2T8 03 G V1T0 04 #F V2T8 03 A V1T0 04 G V2T8 03 B
190 PLAY"V2T8 03 HA V1T0 04 0#F #C V2T8 03 0G V1T0 04 E V2T8 03 H#F V1T0 04 HD
```

UNIT:11

Experiment 11.2

```
5 REM***** SET COLORS AND GRAPHIC MODE *****
10 COLOR0,7
20 COLOR4,7:COLOR1,7
30 GRAPHIC1,1
40 REM***** DRAW FISH TANK BOTTOM *****
50 COLOR1,13:DRAW1,0,144TO320,147
51 FORN=1TO200
52 X=(319*RND(0)):Y=(146+(RND(0)*39))
53 C=(1+(15*RND(0)))
54 COLOR1,C
55 DRAW1,X,Y
56 NEXT N
70 REM***** DRAW A BRIDGE BY DRAWING TWO CIRCLES *****
75 COLOR1,6
80 CIRCLE1,280,160,30,30,270,90
90 CIRCLE1,255,160,30,30,270,90
100 DRAW1,250,160TO225,160
110 DRAW1,310,160TO285,160
120 DRAW1,280,130TO250,130
130 PAINT1,245,157
140 PAINT1,250,144:PAINT1,290,159
150 PAINT1,285,141:PAINT1,265,131
160 REM***** DRAW A TANK CASTLE *****
170 COLOR1,1
180 BOX1,100,100,195,149
190 CIRCLE1,148,85,55,30,,,,120
200 BOX1,100,50,117,100
210 BOX1,195,50,178,100
220 BOX1,95,50,122,60
230 BOX1,200,50,173,60
240 CIRCLE1,147,80,7,5
250 DRAW1,147,80 TO 147,75
260 DRAW1,147,80 TO 154,80
270 DRAW1,147,80 TO 147,85
280 DRAW1,147,80 TO 140,80
290 BOX1,132,145,162,125
295 CIRCLE1,147,125,15,15,270,90
```

```
296 DRAW0,133,125TO161,125
300 CIRCLE1,99,46,5,5,,,,120:CIRCLE1,108,46,5,5,,,,120
310 CIRCLE1,117,46,5,5,,,,120:CIRCLE1,177,46,5,5,,,,120
320 CIRCLE1,186,46,5,5,,,,120:CIRCLE1,195,46,5,5,,,,120
330 REM***** PAINT THE TANK CASTLE *****
340 PAINT1,101,101
350 PAINT1,105,99:PAINT1,167,85
360 PAINT1,101,70:PAINT1,190,39
370 PAINT1,194,70:PAINT1,194,59
380 PAINT1,174,59:PAINT1,199,59
385 PAINT1,193,147
390 PAINT1,199,55:PAINT1,99,59:PAINT1,116,59:PAINT1,121,59
400 PAINT1,100,47:PAINT1,109,47
410 PAINT1,118,47:PAINT1,178,47
420 PAINT1,187,47:PAINT1,196,47
430 COLOR1,10
440 REM***** DRAW SEAWEED *****
445 WIDTH 2
450 DRAW1,50,144TO35,100:DRAW1,50,144TO55,110:DRAW1,50,144TO65,90
455 WIDTH 1
460 REM***** SET COLOR FOR THE SPRITE *****
470 COLOR1,15
480 REM***** DRAW FISH SPRITE BY COMMANDS *****
490 BOX1,2,2,45,45
500 COLOR1,5
510 CIRCLE1,19,21,7,3:REM **FISH BODY**
520 DRAW1,26,19TO32,17TO29,20TO32,23TO26,21:REM **FISH TAIL**
530 REM***** DRAW FISH FINS *****
540 DRAW1,18,24TO24,30TO22,24
550 DRAW1,16,18TO22,13TO21,18
560 DRAW1,14,24TO18,27TO16,24
570 REM***** FILL IN BODY AND DRAW EYES *****
580 PAINT1,19,21:CIRCLE0,15,20,1,1:PAINT0,14,20
585 BOX0,2,2,45,45
590 REM***** DEFINE SPRITE COORDINATES IN F$ AND SAVE AS SPRITE 1 ******
600 SSHAPE$,12,14,33,38:SPRSAVE$,1:SPRITE1,1,5,1,0,0,0
610 REM***** CLEAR THE SCREEN *****
620 COLOR1,7
630 BOX1,10,12,33,30,,1
640 BOX0,2,2,45,45
650 PAINT0,3,3
700 REM***** PLACE FISH SPRITE ON THE SCREEN *****
710 MOVSPR1,150,150:MOVSPR1,270#2
720 REM***** MOVE BUBBLE SPRITES *****
730 SPRITE7,1,15,1,0,0,0
740 MOVSPR7,90,100:MOVSPR7,360#3
750 SPRITE8,1,6,1,0,0,0
760 MOVSPR8,260,100:MOVSPR8,360#1
765 COLOR2,1:COLOR1,2
770 REM***** MOVE FISH SPRITES *****
775 FOR T= 1 TO 10
780 FOR N=2 TO 6
785 S=INT(RND(1)*3+1)
790 A=INT(RND(1)*10)
795 I=INT(RND(1)*12+1)
800 C=INT(RND(1)*13+3)
805 SPRITEN,1,C,0,0,0,1
810 LA=270-A:RA=90+A
815 REM***FISH FACING LEFT MOVE LA**FISH FACING RIGHT GO RA
820 IF N>4 THEN 840
825 MOVSPRN,LA#S:LA=270+A
830 SLEEPI
835 MOVSPRN,LA#S:GOTO 850
840 MOVSPRN,RA#S:RA=90-A:SLEEP I
845 MOVSPRN,RA#S
850 NEXT N
855 NEXT T
860 GOTO775
```

GLOSSARY OF COMPUTER TERMS

A

address — a number or name for a memory location

algorithm — a series of steps which solve a problem

arithmetic operator — a symbol used to represent an arithmetic function; for example, +,-

array — an arrangement of data in the computer's memory

ASCII — code that translates computer symbols into letters, numbers and other characters. Short for American Standard Code for Information Interchange

B

BASIC — a common programming language; short for Beginner's All-purpose Symbolic Instruction Code

binary — a numbering system where all digits are 0 or 1; also called Base 2

bit — the smallest piece of information a computer can understand; short for Binary Digit

bubble sort — a method of arranging a list; usually used for short lists

bug — a programming error

C

cassette tape — a magnetic tape used to store computer data

chained — linked together one after another; also called daisy-chained

channel — a connection between your computer and other hardware

code — instructions for a computer

command — an instruction for a computer to execute

concatenation — joining strings of characters to form a longer string

condition — an expression that is evaluated as true or false

coordinate — the exact location of a pixel or other graphic element; defined as a single point at the intersection of a row and column on the screen

corrupted — damaged and unusable (refers to a disk)

counter — a variable which keeps track of the number of times an event has occurred in a program

crash — program failure

cursor — the flashing square or underline character that indicates where the next character you type will appear

O

data — information entered into a computer

debug — to remove errors from a program

decrement — to decrease

dedicated — set aside for a specific task

disk — a flat plastic sheet with a magnetic surface, used to store computer data

diskette — floppy (flexible) disk

display — 1) to print on the screen of a monitor or TV
2) any characters printed on the screen of a monitor or TV; the screen itself

driver program — a short program demonstrating a subroutine's purpose

E

edit — to make changes to a program

equation — an arithmetic statement consisting of two values defined as equal by the equal sign (=)

equivalent — two values that can be considered equal

execute — to carry out a command or program

expression — a combination of values, variables, and operators

F

file — information stored on disk or cassette

file name — a label for a file chosen by the user

flow chart — a diagram showing the purpose and steps of a program

format — 1) to prepare a disk for use
2) to arrange output

function — an operation, indicated by a keyword, performed on a number or string

G

generate — to produce results

GIGO — Garbage In Garbage Out; incorrect data produces useless results

graphics — designs and pictures created by the computer and displayed on a screen

H

high-resolution mode — graphics mode that allows precise control in drawing

I

increment — to increase

initialize — to set the starting value of a variable

input — information entered into a computer for processing

invert — to reverse in position; usually a mathematical relationship

J

jiffy — a 60th of a second

K

keyword — a BASIC command word

L

language — a set of words and rules for the computer to follow in creating programs

logical operator — a word that links two parts of a conditional statement; for example, AND, OR

loop — a set of instructions executed repeatedly

empty loop — a loop which serves to delay a program's execution

infinite loop — a loop that continues until the user stops the program; also called an endless loop

nested loop — a loop executed within another loop

M

memory — the part of the computer that stores data

microprocessor — the part of the computer that controls the operations

mode — a state of operation; each mode has unique characteristics, for instance text and graphic mode are examples of screen output modes

multicolor — using colors in addition to foreground and backgrounds colors on the monitor

N

null string — a value for a variable with no characters; also called empty string

O

output — results created by a computer

overwrite — to replace a value in memory or storage with a new value

p

parameter — value or values following a keyword which affect the command's execution; pre-set parameters are called defaults

peripheral device — computer attachments used for input/output; for example, disk drives, printers, joysticks

permutation	rearrangement of a set of characters; for example, abc, bea, and cab
pixel	a single point on the display screen; short for picture element
pointer	a computer's internal method of keeping track of the next command to be executed
process	to execute commands
program	a series of instructions that perform a specific task

Q

quicksort	a method of arranging a list; usually applied to long lists

R

RAM	the part of the memory a user can change; short for Random Access Memory
random number	a mathematical value produced by chance
recursive	a series of procedures; each step based on the results of the previous step
resolution	sharpness of detail on the monitor screen, can be high or low depending on the number of pixels
return address	the line number to which program control is returned following a subroutine's completion
ROM	the permanent memory in a computer; short for Read Only Memory
routine	a set of computer instructions
rules or precedence	rules determining the order for carrying out arithmetic operations

S

sequence	a series of events occurring one after another
sprite	programmable, movable graphic elements
statement	a program line
string	set of letters, or letters and numbers, used as the value for a variable
subroutine	a set of program lines that perform a task independent from the rest of the program
subscripts	elements that identify a particular value in an array
syntax	the correct form for BASIC commands and statements

T

terminator	the ending value of a loop variable; when the terminator is reached, the loop is completed
text	output consisting of letters and numbers
trace	a debugging tool which imitates the computer by testing program lines one by one
truncate	to cut off; for example, the INT function truncates decimal places and returns only whole numbers

V

variable	a label for a memory location in the computer

W

window	part of the screen identified as a work area

COMMODORE

Commodore Business Machines, Inc.
1200 Wilson Drive • West Chester, PA

Commodore Business Machines Limited
3470 Pharmacy Avenue • Agincourt, Ontario, M1W3G3

C128301

Printed in USA