CADET'S COLUMN

From BASIC to COMAL

By Cheryl Peterson

Last month we explored Pascal and got to know a bit about it as a programming language. We took a look at the structured nature of Pascal and found that it is much less forgiving about format and structure than BASIC is. And we found that it has more complicated data and programming structures. The CASE, WHILE, and extended FOR/NEXT and IF/THEN structures give Pascal more flexibility than C-64 BASIC.

This month we'll try a similar exercise with COMAL. But first let's find out something about COMAL.

COMAL was designed by Borge R. Christensen and Benedict Loefstedt in 1973 as a replacement for BASIC on the C-64. While it has many of the statements we're all familiar with from BASIC, other more Pascallike commands are also included. COMAL comes in two versions. COMAL 0.14 is the simpler and less expensive of the two. For $24.95 (shipping included) you get three disks with the language and many sample programs, a dictionary of commands, and four back issues of *COMAL Today,* the COMAL monthly newsletter. This starter system is a great idea for beginners, as its low price means you've lost little if you decide you don't like programming.

With thousands of programs available from the COMAL Users Group, you could actually use COMAL without ever having to do your own programming. Their disks run from $15 to $20 with shipping. Special discounts are offered to user group members and newsletter subscribers.

Those who want to write programs to distribute to friends or for commercial gain should be aware that the COMAL language is required to run COMAL programs. It is not a compiled language. You are allowed to distribute COMAL with your programs, however! No royalties to pay, either.

Version 2.0 comes on a cartridge, allows more free programming space, and has added commands. Because this kit includes a cartridge and more features, it's more expensive: $98.95 plus $4 for shipping.

AS A PROGRAMMING LANGUAGE

COMAL can be as structured as Pascal and almost as forgiving as BASIC. Both at the same time. As with BASIC, the programmer decides how structured he wants to be. Of course, good programming technique and proper structuring never hurt. And making adjustments to a program later will be much easier if the program is written using a modular style.

COMAL has all the programming structures of generic Pascal, but not as many different data structures. While COMAL understands and uses many of the same statements as BASIC, it goes farther by adding options to some of them. For instance, the IF/THEN statement can have ELSE and ELIF (else if) options. PRINT is expanded to include PRINT USING. Of course, BASIC 7.0 on the 128 has PRINT USING and ELSE.

Rather than try to cover all the differences here in the article's text, I have included tables on the following pages that show the commands available to BASIC 2.0, BASIC 7.0, and COMAL. Because Pascal is not as standardized as the BASICs and COMAL, I didn't include it in the table. Instead, a separate table shows some common Pascal keywords and their COMAL equivalents.

A quick look at the table will show the commands that COMAL has in common with the versions of BASIC you are already familiar with. You may find this helpful as you attempt to create your own programs. I have included as many of the commands as I know, and where possible have tried to note similar commands that are found under different names.

I also tried to arrange some of the commands by topics. Graphics and sprites are handled quite differently by COMAL, so I put those commands in a different area. COMAL has a graphics mode that simulates LOGO. Once activated, the graphics screen replaces the text screen and you control the movements of a turtle. This involves a whole series of commands that in no way resemble those used by the two Commodore versions of BASIC.

Another different is the handling of string functions. You may miss the statements MID$, RIGHT$, and LEFT$, but I sure won't. I never liked them. COMAL doesn't use these commands at all. Instead, when you wish to remove a substring from a large string, you specify the string name, the first character position, and last character position of the substring. For instance, let's take a string NAME$ that contains my name, CHERYL PETERSON. To use my last name, we could say LAST$:= NAME$(8:16). This assigns the characters from position 8 to position 16 of NAME$ to the string LAST$.

Though it might take a bit of getting used to, I think this manner of handling characters is actually more precise and easier to use. Of course, you must remember to DIM your strings before they are used.

Like Pascal, COMAL uses indented structuring to track modules. Unlike Pascal, COMAL tracks these structures automatically. Each new level is indented one space further than the last. And until you type the appropriate END

statement, each new line will automatically indent itself the proper amount. One point about the END statement in COMAL: Where simple ENDs are used to terminate subroutines in Pascal, COMAL has specialized ENDs for each different kind of subroutine: ENDCASE, ENDFOR, ENDFUNC, ENDIF, ENDPROC, ENDWHILE, and END (for the end of the program).

When you type the appropriate END for an indented routine, the next line is indented to its proper level without your having to do anything else. The END commands can be followed by the name of the procedure, function, or routine, making it easier to keep track of modules. Provided that you remember to put in the proper END statements where needed, COMAL formats everything with the proper indentation.

From the ENDs listed above, you can get an idea of what programming structures are available to you. In addition to BASIC's three structures (FOR, IF, and FUNC), you have Pascal's CASE, PROC, REPEAT, and WHILE. These function similarly to their Pascal counterparts in most respects.

You should know that COMAL's GOTO doesn't use line numbers. Line numbers are used while programming, but the GOTO command accesses routines by name. As with Pascal, functions and procedures are given names and can be accessed by calls to those names. There is no GOSUB command in COMAL.

COMAL includes an EXEC command to "execute" procedures, but it isn't necessary most of the time. To use a procedure or function, you merely type its name on a line. If you need to pass any parameters, these follow the routine name.

STRUCTURING

As mentioned earlier, your programs can be very structured or a bit slapdash. There are a few necessities. String variables must be defined (i.e., dimensioned) before they are used. As with Pascal, you can use local variables. In COMAL, local variables

```
0010 PRINT "SELF ADDRESSED LABELS"
0020 PRINT "COPYRIGHT 1986 AHOY! MAGAZINE"
0030 PRINT "ALL RIGHTS RESERVED"
0100 //COMAL VERSION OF SELF ADDRESSED LABELS PROGRAM//
0110 //DIM VARIABLES//
0120 DIM NAME$ OF 20, ADDRESS$ OF 40, CITY$ OF 20, ST$ OF 2, PC$ OF 9
0130 PASSES:=0; LINES:=0
0140 LABELFORMAT
0150 PRINTNUMBER
0160 READDATA
0170 PRINTOUT
0190 END
0200 PROC LABELFORMAT
0210  INPUT "HOW MANY LINES PER LABEL?": LINES
0220  IF LINES<3 THEN INPUT "AT LEAST THREE LINES PLEASE.": LINES
0230  IF LINES<3 THEN LABELFORMAT
0240  LINES:=LINES-3
0250 ENDPROC LABELFORMAT
0300 PROC PRINTNUMBER
0310  INPUT "HOW MANY LABELS TO PRINT?": PASSES
0320 ENDPROC PRINTNUMBER
0400 PROC READDATA
0410  INPUT "ENTER FULL NAME--20 CHARACTERS OR LESS.": NAME$
0420  INPUT "ENTER ADDRESS--40 CHARACTERS OR LESS.": ADDRESS$
0430  INPUT "ENTER CITY--20 CHARACTERS OR LESS.": CITY$
0440  INPUT "ENTER 2 CHARACTER STATE CODE.": ST$
0450  INPUT "ENTER ZIP CODE--9 CHARACTERS OR LESS.": PC$
0460 ENDPROC READDATA
0500 PROC PRINTOUT
0505  ZONE 3
0510  SELECT OUTPUT "LP:"
0515  FOR Y:=1 TO PASSES DO
0520   PRINT NAME$
0530   PRINT ADDRESS$
0540   PRINT CITY$;ST$;PC$
0550   FOR X:=1 TO LINES DO
0560    PRINT
0570   ENDFOR X
0580  ENDFOR Y
0590  SELECT OUTPUT "DS:"
0600 ENDPROC PRINTOUT
0700 END
```

COMAL SELF-ADDRESSED LABELS
(PASCAL STYLE)

```
0010 PRINT "SELF ADDRESSED LABELS"
0020 PRINT "COPYRIGHT 1986 AHOY! MAGAZINE"
0030 PRINT "ALL RIGHTS RESERVED"
0100 //COMAL VERSION OF SELF ADDRESSED LABELS PROGRAM//
0110 //DIM VARIABLES//
0120 DIM NAME$ OF 20, ADDRESS$ OF 40, CITY$ OF 20, ST$ OF 2, PC$ OF 9
0130 PASSES:=0; LINES:=0
0200 LABELFORMAT
0210 PROC LABELFORMAT
0220  INPUT "HOW MANY LINES PER LABEL?": LINES
0230  IF LINES<3 THEN INPUT "AT LEAST THREE LINES PLEASE.": LINES
0240  IF LINES<3 THEN LABELFORMAT
0250  LINES:=LINES-3
0260 ENDPROC LABELFORMAT
0300 //PRINTNUMBER//
0310 INPUT "HOW MANY LABELS TO PRINT?": PASSES
0400 //READDATA//
0410 INPUT "ENTER FULL NAME--20 CHARACTERS OR LESS.": NAME$
0420 INPUT "ENTER ADDRESS--40 CHARACTERS OR LESS.": ADDRESS$
0430 INPUT "ENTER CITY--20 CHARACTERS OR LESS.": CITY$
0440 INPUT "ENTER 2 CHARACTER STATE CODE.": ST$
0450 INPUT "ENTER ZIP CODE--9 CHARACTERS OR LESS.": PC$
0500 //PRINT ROUTINE//
0505 ZONE 3
0510 SELECT OUTPUT "LP:"
0515 FOR Y:=1 TO PASSES DO
0520  PRINT NAME$
0530  PRINT ADDRESS$
0540  PRINT CITY$;ST$;PC$
0550  FOR X:=1 TO LINES DO
0560   PRINT
0570  ENDFOR X
0580 ENDFOR Y
0600 SELECT OUTPUT "DS:"
0700 END
```

COMAL SELF-ADDRESSED LABELS
(BASIC STYLE)

are defined as being local by indicating that the procedure or function is CLOSED. The CLOSED statement immediately follows the routine name:

```
10 PROC write'file CLOSED
```

This statement indicates that the procedure name is write file and that the variables used within the procedure are local to that procedure. They do not affect any variables of the same name contained elsewhere in the program.

Procedures, functions, and subroutines are found at the end of a COMAL program. Structured BASIC programs generally put subroutines at the end of the program, as well. As we mentioned last month, Pascal requires functions and procedures to be named at the beginning of programs, so in this instance COMAL more closely resembles BASIC.

There is a big difference between functions in Pascal and BASIC and those in COMAL. To get the results from a function in COMAL you must use a RETURN statement. When used in this way, the RETURN assigns a value to the variable and this variable is returned to the main program for later processing. The RETURN statement can also be used to escape from a procedure and return execution back to the main program, just as it is used in BASIC.

MORE MAILING LABELS

In last month's column we developed a program to create self-addressed mailing labels. Let's do the same thing using COMAL to see how we write a COMAL program. In fact, we'll write two versions: one loosely structured, the other more formal. The first will be organized similarly to a BASIC program. The other will resemble a Pascal program.

In last month's programs we used routines called labelformat, printnumber, readdata, and print. To some extent we'll use similar names, but COMAL balks at a procedure name that is the same as a command state-

COMAL-BASIC Comparison Chart

KEY

Y: yes; N: no; M: modified

1: This keyword performs entirely different functions in each language.
2: In COMAL a whole set of commands add flexibility to this keyword. See COMAL sprite commands.
3: The extended drawing commands of COMAL add flexibility to this command. See COMAL graphics commands.
4: GET# is similar to READ FILE
5: INPUT# is similar to INPUT FILE
6: PRINT# is similar to WRITE FILE

General Commands	COMAL 0.14	BASIC 2.0	BASIC 7.0
AUTO	Y	N	Y
BASIC	Y	(1)	(1)
BEGIN/BEND	M	N	Y
CASE/OF	Y	N	N
CAT	Y	N	M(CATALOG)
CHAIN	Y	N	N
CHAR	N	N	Y
CLOSED	Y	N	N
CLR	N	Y	Y
CMD	M(SELECT)	Y	Y
CON	Y	M(CONT)	M(CONT)
DATA	Y	Y	Y
DEC	N	N	Y
DEL	Y	N	Y
DEF FN	N	Y	Y
DIM	Y	Y	Y
DIV	Y	N	N
DO	Y	N	Y
EDIT	Y	N	N
ELIF	Y	N	N
ELSE	Y	N	Y
END	Y	Y	Y
ENDCASE	Y	N	N
ENDFOR	Y	M(NEXT)	M(NEXT)
ENDFUNC	Y	N	N
ENDIF	Y	N	N
ENDPROC	Y	N	N
ENDWHILE	Y	N	N
EOD	Y	N	N
EOF	Y	N	N
ERR$	N	N	Y
ESC	Y	N	N
EXEC	Y	N	N
FALSE	Y	N	N
FOR/TO/NEXT	Y	Y	Y
FUNC	Y	M(DEF FN)	M(DEF FN)
GET	N	Y	Y
GOTO	Y	Y	Y
GOSUB	N	Y	Y
HELP	N	N	Y
HEX$	N	N	Y
IF/THEN	Y	Y	Y
IN	Y	N	M(INSTR)
INPUT	Y	Y	Y
INSTR	M(IN)	N	Y
JOY	N	N	Y
KEY	N	N	Y
KEY$	Y	M(GET)	M(GETKEY)
LET	Y	Y	Y
LINEFEED	Y	N	N
LIST	Y	Y	Y
MONITOR	N	N	Y
NEW	Y	Y	Y
NEXT	M(ENDFOR)	Y	Y
NULL	Y	N	M(SLEEP)
OF	Y	N	N
ON	N	Y	Y
OTHERWISE	Y	N	N
PASS	Y	N	N
PEN	N	N	Y
POINTER	N	N	Y
PRINT	Y	Y	Y
PRINT USING	Y	N	Y
PROC	Y	N	N
PUDEF	N	N	Y
RANDOM	Y	N	N
READ	Y	Y	Y
REF	Y	N	N
REM	Y	Y	Y
RENUM	Y	N	Y
REPEAT	Y	N	N
RESTORE	Y	Y	Y
RESUME	N	N	Y
RETURN[3]	Y	N	N
RETURN[3]	N	Y	Y
RUN	Y	Y	Y
SELECT OUTPUT	Y	N	N
SETEXEC	Y	N	N
SETMSG	Y	N	N
SIZE	Y	M(FRE)	M(FRE)
SPC	N	Y	Y
STATUS	Y	Y	N
STEP	Y	Y	Y
STOP	Y	Y	Y
SYS	Y	Y	Y
TAB	Y	Y	Y
THEN	Y	Y	Y
TIME	N	Y	M(TI)
TIME$	N	Y	M(TI$)
TRAP	Y	N	Y
TRON	N	N	Y
TROFF	N	N	Y
TRUE	Y	N	N
TO	Y	Y	Y
UNTIL	Y	N	Y
USR	N	Y	Y
VAL	N	Y	Y
WHEN	Y	N	N
WHILE	Y	N	Y
ZONE	Y	N	N

Disk Commands	COMAL 0.14	BASIC 2.0	BASIC 7.0
APPEND	Y	N	Y
BACKUP	N	N	Y
BLOAD	N	N	Y
BOOT	N	N	Y
BSAVE	N	N	Y
CATALOG	Y	N	M(CAT)
CLOSE	Y	Y	Y
COLLECT	N	N	Y
CONCAT	N	N	Y
COPY	N	N	Y
DCLEAR	N	N	Y
DCLOSE	M(CLOSE)	N	Y
DELETE	Y	N	M(SCRATCH)
DIRECTORY	M(CATALOG)	N	Y
DLOAD	M(CHAIN)	N	Y
DOPEN	M(OPEN)	N	Y
DSAVE	M(SAVE)	N	Y
DVERIFY	N	N	Y
ENTER	Y	M(LOAD)	M(LOAD)
GET#	N	M[4]	M[4]
HEADER	N	N	Y
INPUT#	M[5]	Y	Y
INPUT FILE	Y	M(INPUT#)	M(INPUT#)
LOAD	Y	Y	Y
OPEN	Y	Y	Y
PRINT#	M[6]	Y	Y
PRINT FILE	Y	M(PRINT#)	M(PRINT#)
READ FILE	Y	M(GET#)	M(GET#)
RECORD	N	N	Y
RENAME	N	N	Y
SAVE	Y	Y	Y
SCRATCH	M(DELETE)	N	Y
SPRSAV	N	N	Y
VERIFY	N	N	Y
WRITE FILE	Y	M(PRINT#)	M(PRINT#)

ment. So we'll use printout instead of print for that procedure.

In the BASIC-style version, we dimension our variables and then jump right in. Since the labelformat section requires testing a value and then repeating our INPUT statement if the proper value is not present, we have to use a procedure instead of a routine. The GOTO command (as mentioned earlier) does not allow jumping by line number, so we must jump to a label. In this case, we jump to a procedure name.

One difference you may notice is in the syntax of the INPUT command. If you recall, when using this statement in BASIC a semicolon usually follows the prompt you put in quotation marks and a ? appears on the screen at the end of the prompt, whether you want it there or not. One of the more popular "hints" included in programming tips columns is how to suppress the ? in INPUT commands. Without getting into that here, you should know that in COMAL it isn't necessary to do anything except place a colon after the prompt. If you want your onscreen prompt to include a question mark, you'll have to enter it before the closing quotation mark.

Notice in line 230 that our "AT LEAST THREE LINES PLEASE." is followed by a colon. When this appears on the screen it is a statement, not a question. Also, our requests for information in the readdata section appear as statements.

To open the channel to the printer in COMAL, you use the SELECT OUTPUT or SELECT command. Both work equally well. The two valid choices are "LP:" and "DS:". LP designates the line printer, DS the default screen.

Although this is the BASIC-style version, we use the FOR/TO/DO structure to print out information. Another convenience of the program is that if you forget and type the FOR/TO/NEXT routine by accident, it will automatically clean it up for you, adding the DO and changing the NEXT to an ENDFOR. Nice!

Taking a look at the Pascal-style version, we see that all our chores have been assigned to their proper proce-

Graphics and Screen Commands—BASIC 7.0	COMAL 0.14	BASIC 2.0	BASIC 7.0
BOX	N	N	Y
CIRCLE	N	N	Y
COLOR	N[2]	N	Y
DRAW	N[3]	N	Y
GRAPHIC	N[2]	N	Y
LOCATE	N	N	Y
PAINT	M(FILL)	N	Y
RWINDOW	N	N	Y
SCNCLR	N	N	Y
WIDTH	N[2]	N	Y
WINDOW	N	N	Y

Sprite Commands—BASIC 7.0	COMAL 0.14	BASIC 2.0	BASIC 7.0
BUMP	N[2]	N	Y
COLLISION	N[3]	N	Y
GSHAPE	N	N	Y
MOVSPR	N	N	Y
RCLR	N	N	Y
RGR	N	N	Y
RSPCOLOR	N	N	Y
RSPPOS	N	N	Y
RSPRITE	N	N	Y
SCALE	N	N	Y
SPRCOLOR	N[2]	N	Y
SPRDEF	N[2]	N	Y
SPRITE	N[2]	N	Y
SPRSAVE	N	N	Y
SSHAPE	N	N	Y

Sound Commands—BASIC 7.0	COMAL 0.14	BASIC 2.0	BASIC 7.0
ENVELOPE	N	N	Y
FILTER	N	N	Y
PLAY	N	N	Y
SOUND	N	N	Y
TEMPO	N	N	Y
VOL	N	N	Y

Graphic Commands—COMAL	COMAL 0.14	BASIC 2.0	BASIC 7.0
BACK	Y	N	N
BACKGROUND	Y	N	N
BORDER	Y	N	N
BOX	N	N	Y
CLEAR[1]	Y	N	N
CIRCLE	N	N	Y
DRAWTO	Y	N	M(DRAW)
FILL	Y	N	M(PAINT)
FORWARD	Y	N	N
FRAME	Y	N	N
FULLSCREEN	Y	N	N
HIDETURTLE	Y	N	N
HOME	Y	Y	Y
LEFT	Y	N	N
MOVETO	Y	N	N
PENCOLOR	Y	N	M(COLOR)
PENDOWN	Y	N	N
PENUP	Y	N	N
PLOT	Y	N	M(DRAW)
PLOTTEXT	Y	N	N
RIGHT	Y	N	N
SETGRAPHIC	Y	N	N
SETHEADING	Y	N	N
SETTEXT	Y	N	N
SETXY	Y	N	N
SHOWTURTLE	Y	N	N
SPLITSCREEN	Y	N	N
TURTLESIZE	Y	N	N

Sprite Commands—COMAL	COMAL 0.14	BASIC 2.0	BASIC 7.0
DATACOLLISION	Y	N	N
DEFINE	Y	N	M(SPRDEF)
HIDESPRITE	Y	N	N
IDENTIFY	Y	N	N
PRIORITY	Y	N	N
SPRITEBACK	Y	N	N
SPRITE-COLLISION	Y	N	N
SPRITECOLOR	Y	N	Y
SPRITEPOS	Y	N	N
SPRITESIZE	Y	N	N

Common Functions and Comparitors

ABS	ORD	AND
ATN	PEEK	OR
CHR$	POKE	NOT
COS	RND	+
EXP	SGN	–
INT	SIN	1
LEN	SQR	/
LOG	TAN	=, <, >, <=, =>

Pascal-COMAL Comparison Chart

KEYWORD	Pascal	COMAL
AND	Y	Y
ARRAY	Y	N
BEGIN	Y	M[1]
CASE	Y	Y
CONST	Y	N
DIV	Y	Y
DOWNTO	Y	N
ELSE	Y	Y
FILE	Y	N
FOR	Y	Y
FUNCTION	Y	M(FUNC)
GOTO	Y	Y
IF	Y	Y
IN	Y	Y
LABEL	Y	N
MOD	Y	Y
NIL	Y	M(NULL)
NOT	Y	Y
OF	Y	Y
OR	Y	Y
PACKED	Y	N
PROCEDURE	Y	M(PROC)
PROGRAM	Y	N
RECORD	Y	N
REPEAT	Y	Y
SET	Y	N
THEN	Y	Y
TO	Y	Y
TYPE	Y	N
UNTIL	Y	Y
VAR	Y	N
WHILE	Y	Y
WITH	Y	N

COMMANDS	Pascal	COMAL
GET	Y	M(INPUT)
INUPT	Y	M(INPUT FILE, READ FILE)
NEW	Y	Y
PACK	Y	N
PAGE	Y	N
PUT	Y	M(PRINT)
READ	Y	M(INPUT)
READLN	Y	M(INPUT)
RESET	Y	M (RESTORE)
REWRITE	Y	N
UNPACK	Y	N
WRITE	Y	M(PRINT)
WRITELN	Y	M(PRINT)

[1]This command is simulated in several other commands.

dure name. The main program thus becomes just a list of procedure names. All our routines function in the same way as their counterparts in the BASIC-style version; they are just called differently.

As you can see, COMAL falls somewhere between Pascal and BASIC. It's not quite as rigidly structured, and it retains many of BASIC's fundamental statements. For more information on COMAL, you can contact the COMAL Users Group (address below). Also, Captain C on PlayNET can also help you with any questions you may have. Send him Online Mail or reach him in the "Let's Talk COMAL" room at 9:30 p.m. EST on the first Thursday of each month.

Numerous books are available to help you learn COMAL. Most are offered by the COMAL Users Group and run in the neighborhood of $20. Most of these have sample programs, a disk of which is available for an extra fee. For those who think they may seriously use COMAL, I would suggest a subscription to the newsletter. Besides giving excellent tips on COMAL programming, subscribers get substantial discounts on disks, books, and other supplies.

Those who have the disk subscription to *Ahoy!* will be glad to hear that COMAL 0.14 and several sample programs are included on each month's disk. See page 38 in this issue for disk ordering information, or page 10 for details on getting your disk with your magazine.

Next month we'll take a look at yet another language for the C-64. □

COMAL Users Group USA, Ltd.
6041 Monona Drive, Room 111
Madison, WI 53716
(Phone: 608-222-4432)

SCUTTLEBUTT

Continued from page 14

gang attacks, and distillery raids are just part of the fun.

Defender of the Crown begins with the death of King Richard, as the Saxon knights prepare to clash with the plundering Normans. More than 30 animated screens are included, with graphics by Jim Sachs, who created some of the best screens ever seen on a C-64.

Sinbad and the Throne of the Falcon follows the sailor on his quest in search of the missing Caliph, as he battles black magic, man-eating monsters, and other unpleasantness.

SDI requires the young general in charge of the Strategic Defense Initiative, America's ultimate space project, to defend it from a squadron sent to destroy it by a fanatical new Soviet regime.

All four are scheduled for release in the fourth quarter of 1986, and a fifth, *Star Rush*, for first quarter 1987.

Mindscape, Inc., 800-221-9884 in US; in IL 800-942-7315; elsewhere 312-480-7667 (see address list, page 14).

10th Frame ($39.95) simulates professional bowling with a true player's perspective, 3D animation, computerized scoring, and league competition allowing up to eight bowlers to play at once. For the C-64.

Access Software, Inc., 801-298-9077 (see address list, page 14).

Four new C-64 games from Epyx:

World Games packs players off to eight different countries to compete in events specific to those locales: cliff-diving in Mexico, sumo wrestling in Japan, barrel jumping in Germany, bull riding in the US, weight lifting in the USSR, caber toss in Scotland, log rolling in Canada, and giant slalom skiing in France.

The *Super Cycle* motorcycle racing game provides a first person perspective of an obstacle-laden course which the player shares with other cyclists. A series of progressively more difficult courses are provided.

Championship Wrestling simulates the world of professional wrestling with handsome heroes and nasty villains, the use of tactics like the Atomic Drop and the Pile Driver, a boisterous, rowdy crowd, and points awarded for showmanship. Ringside ropes vibrate on impact, and grapplers are often thrown from the ring or to the mat.

Mentioned in August's *Scuttlebutt, The Movie Monster Game* stars Godzilla and five other city-stomping movie monsters. The player assumes a persona, picks a city to destroy, and begins battling the police, army, and other humans that try to stop him.

Epyx, Inc., 408-745-0700 (see address list, page 14).

MATCHBLOCKS

Continued from page 41

in the SET 1 column.

Reversed images can be obtained by adding 128 to the POKE value. The data to create the diamond shape is 233, 223, 95, 105. The program converts these numbers into the diamond shape by placing the triangle shape for POKE value 233 (reverse image of 105) to the left of the triangle shape for POKE value 223 (reverse image of 95). Below these, the triangle shape for POKE value 95 is placed to the left of the triangle shape for POKE value 105 to complete the diamond shape.

You can experiment with changing the color and shape of the block pictures by typing LIST 700-780 (after a READY prompt) and pressing the RETURN key. This will list the program lines as they appear now.

At this point there are two ways to modify a program line. One way is by retyping an entire revised line and pressing the RETURN key. The new line will be substituted for the old line. The other way is by using the two CRSR keys and the INST/DEL key. Use the CRSR keys to move up and over to the data that you want to change. Then type over just the numbers in the line that you want to change and press the RETURN key to enter the changes. Use the INST/DEL key to insert or delete extra digits. In both methods you can verify the changes by either typing LIST 700-780 to look at the program data lines or typing RUN 800 to display the entire new block set on the screen. It is possible to save different block sets by simply saving the entire program under different names.

Matchblocks was designed to be fun for the entire family. I hope you enjoy playing it and experimenting with it. □

SEE PROGRAM LISTING ON PAGE 112