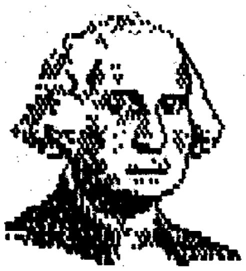
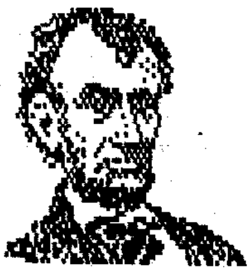
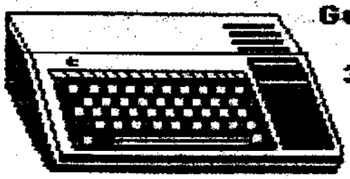
GUILFORD 99'ERS NEWSLETTER

PRESIDENT'S DAY

FEBRUARY 18TH

SUPPORTING THE TEXAS INSTRUMENTS TI-99/4A COMPUTER



GUILFORD 99'ERS UG
3202 CANTERBURY DR
GREENSBORO NC
27408

NC PIEDMONT TRIAD AREA 274
PM
28 JAN
1991

NORTH CAROLIN
PIEDMONT
TRIAD.

TO: [redacted]

VOLUME 8 NUMBER 2 FEBRUARY 1991

George von Seth, Pres. (292-2035)   Bob Carmany, Newsletter Ed (855-1538)
Tony Kleen, Sec/Treas (924-6344)   Bill Woodruff,Pgm/Library (228-1892)
BBS: (919)621-2623 --ROS
+-
The Guilford 99'er Users' Group Newsletter is free to dues paying members (One copy per family, please). Dues are $12.00 per family, per year. Send check to:Tony Kleen c/o 3202 Canterbury Dr., Greensboro, NC 27408. The Software Library is for dues paying members only. (Bob Carmany Ed)
+-

OUR NEXT MEETING

DATE: February 5, 1991 Time: 7:30 PM. Place: Glenwood Recreation Center, 2010 S. Chapman Street.

Program for this meeting will be an instant replay of the database utilities for TI-Writer that was scheduled for last month's meeting cancelled by the sleet and freezing rain. Plan to be there to see a couple of interesting programs!

MINUTES

Guess what? Since the meeting was cancelled by the weather, there aren't any minutes for this month. George and Bob were there but no one else braved the slick roads. No worries, we'll try again this month and hope that the weather cooperates.

DUES

Everyone gets an extension of time to get their annual dues in because of the meeting cancellation. Please bring your dues to the February meeting or mail them to Tony at the UG address. Your subscription will end in March if you don't get them paid by then. Support your local Users Group!

REVIEW: MANCALA

By: Andy Frueh, Lima UG

Several other computers, especially the PC's, have several "foreign" games. These originate as folk games in some other country and are adapted in the USA for play on our computers. Tetris is a good example. These games are few becuase they require thinking and brainpower with coordination. They are not the simple familiar shoot-em-ups. Now, from Africa, comes...Mancala. It is very well-done, and extremely addictive to play. The rules are about as simple as the ones for Tetris.

It is sold as a cartridge, so it is very convenient to load; just plug it in. The title screen slowly forms the word Mancala to a neat "bongo" type beat. You can press any key to abort this display. Upon doing that, you are asked for the number of players. I prefer to play a human, mainly because it's an easier opponent (usually). Next you are asked for the number of stones (explained later) that you want. This is a number from 3 to 8 or a random amount. Finally, you are given the choice of which player moves first. I know some of the strategy of the game, and I can see no reason to want to move first or second. Now the game begins.

On the screen are 6 "cups" and 1 "goal" for each player. Player 1 has the bottom row and the right goal, and the second player/computer has the top row and the left goal. Each cup holds a certain number of stones, as specified in the beginning. You can play with joysticks or the keyboard. Each player is represented by an orange marker, which surrounds the cup. The idea is to get as many stones in your cup as possible.

The rules for acheiving this aren't very difficult. You move the stones by moving to a cup and pressing FIRE. Then, the marker picks up all the stones in the cup. The marker moves towards your goal, depositing one stone in each cup it passes over. If there are more stones left after the goal is reached, then the marker moves to the other player's side and continues going around the board. If you make it to the goal with no stones left in the marker (for example, you move one stone from the cup next to your goal), then you may go again. If you move one stone into an empty space, then you get all of the stones from the opponent's cup opposite of yours into your goal. This provides a good amount of action.

There is a demo mode if you want to see how to play. There are also several levels of play (but I can't get past 2). All of this makes for a great game. It isn't one of those games where after you lose or win you want to go on to something else. It is available in Triton catalogs. BUY IT!

[This article/item comes from the January 1991 issue of BITS, BYTES PIXELS (Charles Good, editor), the newsletter of the Lima OH 99/4A User Group, P.O. Box 647, Venedocia, OH 45894

ABOUT SPEECH

By Ron Albright

The more I read about the "new" developments and software for other machines, the more I impressed/infuriated I become with Texas Instruments. Whether you realize it or not, TI was light-years ahead of the remainder of the home computer industry in virtually everything except, of course, consumer marketing and common sense. One of the features which remains the industry leader and is, at the same time, the most neglected and overlooked feature available for our machine is the text-to-speech access. With the speech synthesizer and the Terminal Emulator II cartridge (or disk-based text-to-speech program for XB), you have a feature unrivaled on any other machine. Sure, others have "speech" and some even boast "unlimited vocabulary", but, if you ever heard these facilities on another machine, you realize how far ahead TI was (and still is) in synthetic speech. What I would like to do in this article is to give you an overview of speech synthesis on the TI and, hopefully, revive some interest in this incredible facility.

The chip used in our speech synthesizer is the TMS 5220. A P- channel MOS device packaged in a 28-pin DIP. It is a second generation speech chip, which followed the TMS 5100 used in the Speak and Spell toys appearing in 1977. While the TMS 5220 is capable of all three types of synthetic speech (linear predictive coding, wave-form modulation, and phoneme-stringing), our machine uses the most memory-efficient form: linear predictive coding, or LPC (but has the capability for allophone-stringing). LPC in our machine requires a small amount, 3k of memory, to hold the 128 allophone library, 7K to accomodate the 650 rule text-to-speech set for translating English-language text into allophone equivalents and for contouring inflections with the help of pitch modifiers to make the speech more natural. The allophone library and the rules for stringing them are held in the TE II GROM chips. The synthesizer holds the speech chip and the resident speech vocabulary (memory location >9000). The system is not perfect (as you may have learned hopefully by experience) but even with this small ROM requirement, TI achieved 92% translation accuracy. You can correct the remaining 8% with changing text.

Let us digress for clarity. Of what do we speak when we discuss allophones? Allophones are the most fundamental of any of the other linguistic components, including phonemes, diphones, and morphs. An analysis of the English language shows that about forty allophonic sound characteristics can provide the needed variations for all 45 standard phonemes. For example, the phoneme for the letter "P" in English is rounded and aspirated in the word "Poke", rounded and unaspirated in "Spoke", aspirated in "Pie", slightly aspirated in "Taper", released in "Appetite". These acoustically different "P"'s -so-called voiceless bilabial stops - are allophonic variations of the phoneme "P". Thus, allophonic speech produces better quality than phonemics because the allophones provide the most of the subtle variations each English phoneme can encompass and use each variation in the appropriate relationship. Phonemic speech sounds mechanical and is limited. Allophonic speech is much better though still not perfect...the transition between allophones make the speech sound unnatural and intonations are characteristically monotonic. But allophonic speech is an ideal compromise based on size of vocabulary, memory requirements and quality and versatility of speech.

So, knowing that we use an allophone speech system, how does it work? In general, text from keyboard input is converted into the appropriate allophones which are then converted into LPC data which activates the TMS 5220 to generate immediate speech. Well. it's not quite that simple. For the text to be converted to the "appropriate" allophones, rules must be applied: 650 rules, to be exact. The rules, based on a US Navy laboratory system are complex to say the least. For example, in the process of translating the word "space", the allophone-stringing algorithm looks first at the "s" and supplies a initial allophone for /s/. But for the "p" it finds a rule where the left environment is an "s". Also, since the "p" is not a final sound, the algorithm translates the "p" accordingly. Next the rule is invoked that applies to an "a" where the right-sided environment consists of a single consonant and the word ends with a word-final silent "e". This rule selects the appropriate "long-a" allophone. Finally, the rule for the "ce" inserts an /s/ component in the allophone string to replete the "c" in the text: the rule says the "e" is silent. As we have stated, 92% of the time the rules work... not bad! Compound words give it problems, often easily corrected by hyphenating...e.g. "Base-Ball".

Not only does the TI system convert text to component allophones, it also, through the rule set, translates secondary and primary speech-stress points into pitch variations. Contouring algorithms divide sentences into two major stress profile types: a falling mode where the pitch levels drops following a primary stress point (as occurs in a normal sentence making a statement), and a rising mode which occurs in sentences terminating in a question mark. This adds even more normal quality to speech. Remember how many times you have hears "Ready to start?"...Notice how the pitch varies in a rising tone on 'start'.

So, in all a very complex system that the TI engineers gave us. We have sparse but utilitarian documentation in the TE II manual. It discusses, ever so briefly, how to access both "OPEN #1:"SPEECH", OUTPUT" and the allophone library directly through "OPEN #1:"ALPHON",INTERNAL". It briefly defines the manual override feature to vary pitch and slope through the "//XX YY". Perhaps this feature deserves more comment.

You can vary greatly the pitch and slope of speech through the use of the //XX YY command. I have hears a sparse few programs where the computer actually sings. The most recently published was the "ABC Song" seen in the Tigercub Tips (Jim Peterson, Tigercub Software, 156 Collingwood Ave., Columbus, Ohio 43213). Look at the program and see how Jim changes the pitch and slope to produce synthetic singing! The key formula is one where the slope is calculated from the set pitch through y(slope)=(x(pitch)/10). We are told in the manual (p.34), that this gives the best results. So, by changing the pitch to simulate singing of notes and adjusting the slope by this formula, we can approach singing. Further, we can set stress points in our own tex by use of " "(sets primary stress point in a sentence),"_"(sets secondary stress points within a sentence, and ">" (shifts stress points within a word). So, we need not rely on the 92% accuracy TI accomplishes with the rule set...we can achieve realism approaching 100% with manual symbols placed within our text!

Through "OPEN #1,"ALPHON",INTERNAL" we can access the 125 allophones (but we said 128: 126 and 127 are pauses) in the TE II GROM library. They are listed in the manual with a rather spartan description of their use. They are strung together as CHR$ statements: CHR$(10)&CHR$(22)&CHR$(X)...etc. Again, we are allowed to change pitch and slope through manual input by sending a CHR$(252)&CHR$(XX), where the variable "XX" following the CHR$(252) sets a new pitch and CHR$(251)&CHR$(YY) where CHR$(251) changes slope to the following CHR$(YY) value. Stress points can be set with CHR$(253)(Primary stress with rising contour, CHR$(254)(Primary stress with falling contour) and CHR$(249)(Secondary stress point). While you can change pitch and slope of allophones, the only way I know of to increase the duration of the sound is to string allophones, i.e. CHR$(N)&CHR$(N)&CHR$(N) to increase the duration of allophone "N" three fold. A way to implement the RPT$ function in Basic would do the trick.

DISKS AND SUCH

The diskette used on the Home computer Disk Peripheral has the following specifications:

Capacity: 92160 bytes per disk; 2304 bytes per track; 256 bytes per sector; 9 sectors per track

Encoding method: FM Single Density Recording

Mini diskette type SA 104 (ANSI standard 5.25")

The specified diskette contains a total of 360 sectors of 256 bytes each. In the remainder of this chapter each sector will be addressed as if the diskette was a linear medium, i.e. track 0 sector 0 will be designated "sector 0"; track 39 sector 8 equals "sector 359".

The following section contains a description of the logical structure on each diskette in terms of records.

Physical Diskette Format:

The general diskette format used in the TI-99/4 Disk Peripheral is the following:

Sector 0 contains the Volume Information Block (VIB). This block contains general information about the diskette like:

Volume Name

Number of available Allocation Units
Number of sectors/track
Allocation Bit Map

Sector 1 contains pointers to file descriptor records.

Sector 2 thru 359 contain File Descriptor Records and data blocks.

The File Descriptor Records contain general information about the file, such as:

File name
File status data
File data access blocks

Volume Information Block

As mentioned previously, this block contains general information about the diskette. A more detailed description of each entry and its contents will be given in this section.

Bytes 0-9 contain the volume name of the diskette. The volume name can be any combination of ten ASCII characters, except for the space or period (".") characters and the null character (ASCII code 0). The name is space filled to the right in case of less than 10 characters. The volume name must contain at least one non-space character.

Bytes 10-11 give the total number of allocation units (AUs) on the volume. This datum should match the allocation bit map.

Byte 12 indicates the number of sectors per track.

Bytes 13-15 contain the ASCII code for "DSK", which is used by the disk manager software to check if the diskette has been initialized.

Byte 16 contains the ASCII code for "P" if the diskette is protected (a protected disk is also called a proprietary disk), otherwise this byte contains a >20.

Byte 17 indicates the number of tracks per side.

Byte 18 indicates the number of formatted sides on the diskette.

Byte 19 indicates the density of the diskette.

Bytes 20-55 are reserved for future expansions like date and time of creation. In the current version of the disk software these bytes are set to zero.

Bytes 56-255 contain the allocation bit map. This 200 byte map can control up to 1600 256-byte records (total controllable storage capacity = 400K bytes), which make it useable for a double density, double sided diskette. The disk allocation system uses a conventional method of allocating disk space called Bit Maps. Each bit in the bit maps represents one sector on the disk. A logical one in the bit maps means that the corresponding sector has been allocated. A zero means that the sector is still available.

The volume name can be used as an alternative to the actual disk drive name, i.e. the user can specify a disk drive in either of the following ways:

DSK.volname.filename or DSKn.filename

If the volume is specified, rather than the physical drive number, the system will look in sequence on every drive in the system, until it finds the specified volume. If more than one volume of the same name exists, the drive with the lowest drive identification number will be assigned.

File Descriptor Index Record

The File Descriptor Index Record contains up to 127 two byte entries, each pointing to a file descriptor record. These pointers are alphabetically sorted according to the filename in the associated file descriptor record. The pointer list starts at the beginning of this block, and ends with a zero entry.

Since the file descriptors are alphabetically sorted in this block, a binary search method can be used to find any given filename, limiting the maximum number of disk searches to 7 if more than 63 files are defined. In general if between 2**(N-1) and 2**N files are defined, a file search will take at most N disk searches. To obtain faster directory search response times, the system will prefer to allocate data blocks in the area above AU number 34. Only if no AU can be allocated in that area will the disk data block allocator start allocating blocks in the AU area 2-33.

File Descriptor Records.

The File Descriptor Record (FDR) contains general information about the associated file. All the information the system needs to know to access and update the file has to be contained within the file descriptor record.

The physical layout of an FDR is:

Bytes 0-9 contain a filename up to ten characters in length.

Bytes 10-11 are reserved for future extension of the number of data chain pointers through linkage to a data chain pointer block chain. In the current version these bytes are always 0.

Byte 12 contains the file status flags. These flags are to be interpreted as follows (bit 0 is the least significant bit):

0: Program/data file indicator 0 = Data file 1 = Program file

1: Binary/ASCII data 0 = ASCII data (DISPLAY file) 1 = Binary data (INTERNAL or program file)

2: Reserved for future data type expansion

3: Protect flag 0 = Not protected 1 = Protected

4-6: Reserved for future expansion

7: FIXED/VARIABLE flag 0 = Fixed length records 1 = Variable length records

Byte 13 contains the number of logical records per AU.

Bytes 14-15 contain the number of logical records allocated on Level 2 (256 byte records).

Byte 16 contains the EOF offset within the highest physical AU for variable length record files and program files.

Byte 17 contains the logical record size in bytes. In case of variable length records, this entry will indicate the maximum allowable record size.

Bytes 18-19 contain the number of records allocated on Level 3. For variable length records, this entry is replaced with the number of Level 2 records actually used. (NOTE: The bytes in this entry are in reverse order.)

Bytes 20-27 have been reserved for future expansion. They will be fixed to 0 in this implementation of disk peripheral software.

Bytes 28-255 contain three byte blocks indicating the clusters that have been allocated for the file. The first 12 bits in each entry indicate the address of the first AU in the cluster . The second 12 bits indicate the highest logical record offset in the cluster of contiguous records. This indication has been chosen, rather than the number of data-records in the chain, since it reduces the amount of computation required for relative record file access.

ESKTOP PUBLISHING & PAGE PRO

By Bob Carmany

Page Pro is just about the closest thing to a true desktop publishing system for our "orphan" that there is around. It allows one to combine different sized fonts, pictures, and manipulate them to create individual pages of a document. There are some limitations! First the page size is 60 columns wide and 66 lines from top to bottom. The other restriction is that you can only use one large font and one small font per page. On the other hand, the commands are just like TI-Writer for text manipulation and you can use as many as 28 pictures per page!

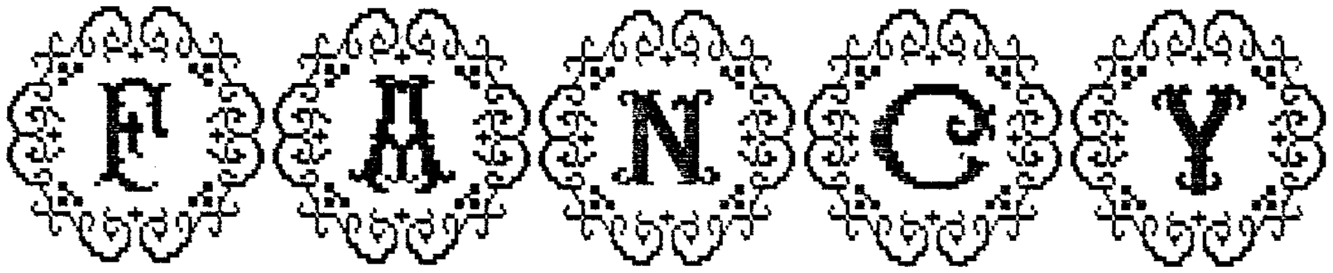
You can also use the line characters to box text to create other effects if you wish. All in all, you can do some interesting things with Page Pro.

You can even use pictures (in this case giant letters to create an effect similar to a document created in medievel times by monks in one of the many monastaries that existed then. These pictures come from many sources but the largest is probably the TI-Artist Instances. It is easy to convert one (or more) TI-Artist Instances into the format that Page Pro uses. In fact, there is a built-in conversion program to do just that. Another source of pictures (the "Y" in fact) are the myriad of TIPS files. They can be converted as well without difficulty by a conversion program. So, there are probably more than 10,000 pictures that you can combine with fonts to create pages of graphics and text.

Fonts are just as easy to find. There is a program to convert both large and small TI-Artist fonts into Page Pro format. With literally hundreds of fonts available for the TI-Artist program, it is unlikely that you will ever run out of fonts to use while creating a series of Page Pro pages. The only limitation seems to be the imagination of the user. In fact, the combination of TI-Artist+ and Page Pro seems to be almost unbeatable.

With Page Pro, you can place pictures wherever you want them on the screen and even incorporate labels within the picture itself.

Creating large pictures from several small pictures just as easy. The "party" picture on the cover of last month's newsletter was actually six seperate pictures pla together to form a large picture. It leads to some very interesting possibilities. How about a string of pictures form a word -- like those below:

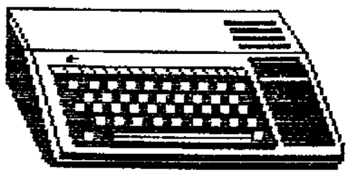


Of course, if you create a textfile that is 60 columns wide you can import it directly into a Page Pro page. You could create your text with TI-Writer or F'WEB and then load it into Page Pro and alter the font to suit your whims. In fact, it would probably be faster to do it that way since both are faster when the screen scrolls than is Page Pro.

Letterheads, invoices, and other documents are all within the capabilities of Page Pro. There are examples of each included with the program. In fact, there is even a utility included to create multi-column text like you find in magazine articles and some newsletters.

I hope that you will take a long, hard look at Page Pro if you are considering a graphics utility. If you combine it with TI-Artist+ and F'WEB, you can create almost any type of document you wish. Besides, the TIPS files are public domain and there seems to be an almost unlimited supply of pictures available.

Where to you get Page Pro? Asgard Software markets Page Pro and several of the auxiliary utilities that go along with it.



+ 4 + ME

P.S. - All of our newsletter cover sheets (and these two pages) are done with Page Pro. It takes anywhere from 15 minutes up to compose and print a page depending on how complex it is.

```
100 ! **********************
110 ! *     AADVARK        *
120 ! *        BY          *
130 ! * PATRICK PELLETIER *
140 ! **********************
150 !
160 ! 99ER VERSION 2.8.1XB
170 !
180 CALL CLEAR
190 DISPLAY AT(2,11):"AADVAR
K"
200 DISPLAY AT(4,5):"BY PATR
ICK PELLETIER"
210 DISPLAY AT(8,1):"LE BUT
DU JEU *THE GOAL OF"
220 DISPLAY AT(10,1):"EST DE
 MANGER * THE GAME IS"
230 DISPLAY AT(12,1):"LA NOU
RRITURE * TO EAT THE"
240 DISPLAY AT(14,1):"AVANT
D'ETRE * FOOD BEFORE"
250 DISPLAY AT(16,1):"MANGE.
...... * BEING EATEN"
260 DISPLAY AT(23,4):"TAPEZ
UNE CLE*PRESS ANY KEY"
270 CALL KEY(0,K,S):: IF S=0
 THEN 270
280 AH=49 :: AV=104 :: TIM=0
 :: OEUF=2 :: ANT=25
290 CALL CLEAR :: CALL MAGNI
FY(4):: CALL SCREEN(8)
300 CALL CHAR(36,"00")!REPLA
CE
310 CALL CHAR(97,"00")
320 CALL CHAR(96,"1818181818
181818")
330 CALL CHAR(98,"1818181F1F
00")
340 CALL CHAR(99,"000000FFFF
00")
350 CALL CHAR(100,"000000F8F
8181818")
360 CALL CHAR(101,"181818F8F
8000000")
370 CALL CHAR(102,"0000001F1
F181818")
380 CALL CHAR(103,"03060C183
060C080")
390 CALL CHAR(94,"AA55AA55AA
55AA55")
400 CALL CHAR(112,"000000000
010387C")
410 CALL CHAR(120,"287060101
83F4F12")
```

```
420 CALL CHAR(40,"8484A4A5ED
EDFFFF")
430 CALL COLOR(2,13,8)
440 CALL CHAR(128,"03190701
F3E70210300000000000000071F3
FFFFFFFDD8183000000000000000"
)
450 CALL CHAR(132,"80C0F0F8F
CFFDFCFE7000000000000000000
81870E0C0800000000000000000"
)
460 CALL CHAR(136,"0C05071D3
F7320212300000000000000C71F3
FFFFFFFDD8183000000000000000"
)
470 CALL CHAR(140,"80C0F0F8F
CFFDFCFE7000000000000000000
00000000C0F0FE0000000000000
")
480 CALL COLOR(1,8,8)
490 CALL COLOR(8,7,3,9,7,1,1
0,7,3,11,2,1,12,14,1)
500 CALL HCHAR(1,1,32,768)
510 FOR I=9 TO 24 :: CALL HC
HAR(I,1,94,32):: NEXT I
520 RESTORE 700
530 FOR I=9 TO 22
540 READ ZZ$
550 DISPLAY AT(I,1):ZZ$ :: N
EXT I
560 CALL HCHAR(8,1,40,32)
570 FOR I=3 TO 7 :: CALL COL
OR(I,2,8):: NEXT I
580 CALL HCHAR(10,3,112):: C
ALL HCHAR(2,3,112,5)
590 FH=22 :: FV=30
600 CALL HCHAR(FH,FV,120)::
CALL HCHAR(2,26,120,5)
610 CALL SPRITE(#1,128,5,AH,
AV,#2,132,5,AH,AV+30)
620 DISPLAY AT(2,11)SIZE(9):
"AARDVARK"
630 CALL JOYST(1,X,Y)
640 IF X=-4 THEN LET FV=FV-1
 :: GOTO 780
650 IF X=4 THEN LET FV=FV+1
:: GOTO 840
660 IF Y=4 THEN LET FH=FH-1
:: GOTO 900
670 IF Y=-4 THEN LET FH=FH+1
 :: GOTO 970
680 IF TIM>=5 THEN LET TIM=0
 :: GOSUB 1280
690 GOTO 630
```

```
700 DATA "^^^^^^^^^^^^$^^^^^^
^^^^^^^^^^^","$$^^^^^^^^^^$^^^
^^^^^^^^^^^^^^"
710 DATA "^$$$$$$$$$$$$$$$$$$
$$$$$$$$$^^","^^^^^$^^$^^$^^$
^^$^^$^^$^^"
720 DATA "^^^^^$$$^$^^$^^$^^$
^^$$$$$$^^","^^^^^$^$$$$$$$$$
$^$^^^^$^$^^^^"
730 DATA "^^^^^$^^$^^^$^$^$$$
$$$$$^$$^^^","^^^^^$^$$$$$$^$
^^^$^^^^^^$^^^^"
740 DATA "^^^^^$$$^$^^$^$^$$$
$^^^$$$^^^^","^^^^^^^^^$^^$^$^
$^^$^$$$^^^^^^"
750 DATA "^^^^^^^^^$$$$$$$$^^
$$$^$^^^^^^","^^^^^^^^^^^^$^$^
^^^$^^^$^^^^^^"
760 DATA "^^^^^^^^^^^^$^$^^^^
$^^^$$$^$$","^^^^^^^^^^^^^$$$^
^^^$$$$$^$$$$"
770 DATA "AAA","HHH"
780 CALL GCHAR(FH,FV,XX)
790 IF XX=36 THEN GOSUB 1040
800 IF XX=94 OR XX=32 THEN L
ET FV=FV+1
810 IF XX=112 THEN LET FV=FV
+1 :: GOSUB 1080
820 TIM=TIM+1
830 GOTO 680
840 CALL GCHAR(FH,FV,XX)
850 IF XX=36 THEN GOSUB 1050
860 IF XX=32 OR XX=94 THEN L
ET FV=FV-1
870 IF XX=112 THEN LET FV=FV
-1 :: GOSUB 1080
880 TIM=TIM+1
890 GOTO 680
900 CALL GCHAR(FH,FV,XX)
910 IF XX=36 THEN GOSUB 1060
920 IF XX=32 OR XX=94 THEN L
ET FH=FH+1
930 IF XX=112 THEN LET FH=FH
+1 :: GOSUB 1080
940 IF XX=96 OR XX=98 OR XX9
9 OR XX=100 OR XX=101 OR XX=
102 OR XX=103 THEN LET FH=FH
+1 :: GOSUB 1190
950 TIM=TIM+1
960 GOTO 680
970 CALL GCHAR(FH,FV,XX)
980 IF XX=36 THEN GOSUB 1070
990 IF XX=32 OR XX=94 THEN F
H=FH-1
```

```
1000 IF XX=112 THEN LET FH=F
H-1 :: GOSUB 1080
1010 IF XX=96 OR XX=98 OR XX
=99 OR XX=100 OR XX=101 OR X
X=102 OR XX=103 THEN LET FH=
FH-1 :: GOSUB 1190
1020 TIM=TIM+1
1030 GOTO 680
1040 CALL HCHAR(FH,FV,120)::
 CALL HCHAR(FH,FV+1,36):: CA
LL SOUND(-100,6000,0,8000,0)
:: RETURN
1050 CALL HCHAR(FH,FV,120)::
 CALL HCHAR(FH,FV-1,36):: CA
LL SOUND(-100,6000,0,8000,0)
:: RETURN
1060 CALL HCHAR(FH,FV,120)::
 CALL HCHAR(FH+1,FV,36):: CA
LL SOUND(-100,6000,0,8000,0)
:: RETURN
1070 CALL HCHAR(FH,FV,120)::
 CALL HCHAR(FH-1,FV,36):: CA
LL SOUND(-100,6000,0,8000,0)
:: RETURN
1080 CALL SOUND(100,2000,0):
: CALL SOUND(100,1000,0)
1090 LET OEUF=OEUF+1
1100 CALL HCHAR(2,OEUF,97)
1110 CALL HCHAR(FH,FV,36)
1120 IF OEUF=7 THEN 2160
1130 FH=22 :: FV=30 :: CALL
HCHAR(FH,FV,120)
1140 TIM=0
1150 RETURN
1160 CALL GCHAR(AA+1,BB,XZ)
1170 IF XZ=120 THEN GOSUB 11
90
1180 RETURN
1190 CALL SOUND(100,110,0)::
 CALL SOUND(100,500,0)
1200 LET ANT=ANT+1
1210 CALL HCHAR(2,ANT,36)
1220 GOSUB 2280
1230 TIM=0
1240 IF ANT-30 THEN 2160
1250 FH=22 :: FV=30
1260 CALL HCHAR(FH,FV,120)
1270 RETURN
1280 RANDOMIZE :: TONG=INT(R
ND*9)+1
1290 ON TONG GOSUB 1800,1820
,1840,1860,1880,1900,1920,19
40,1960
```

```
1300 REM
1310 FOR I=1 TO 13
1320 READ AA,BB,CC
1330 GOSUB 1160
1340 CALL HCHAR(AA+1,BB,CC)
1350 NEXT I
1360 ON TONG GOSUB 1980,2000
,2020,2040,2060,2080,2100,21
20,2140
1370 CALL SOUND(1500,-5,10)
1380 FOR I=1 TO 13
1390 READ AA,BB,CC
1400 CALL HCHAR(AA+1,BB,CC)
1410 NEXT I
1420 GOSUB 2340
1430 RETURN
1440 DATA 20,14,36,19,14,36,
18,14,36,17,14,36,16,14,36,1
5,14,36,14,14,36,13,14,36
1450 DATA 12,14,36,11,14,36,
10,14,36,9,14,36,8,14,36,0,0
,0,0,0,0
1460 DATA 14,20,36,14,19,36,
14,18,36,13,18,36,13,17,36,1
3,16,36,13,15,36
1470 DATA 13,14,36,12,14,36,
11,14,36,10,14,36,9,14,36,8,
14,36,0,0,0,0,0,0
1480 DATA 10,24,36,10,23,36,
10,22,36,10,21,36,10,20,36,1
0,19,36
1490 DATA 10,18,36,10,17,36,
10,16,36,10,15,36,10,14,36,9
,14,36,8,14,36,0,0,0,0,0,0
1500 DATA 10,4,36,10,5,36,10
,6,36,10,7,36,10,8,36,10,9,3
6,10,10,36
1510 DATA 10,11,36,10,12,36,
10,13,36,10,14,36,9,14,36,8,
14,36,0,0,0,0,0,0
1520 DATA 12,8,36,12,9,36,13
,9,36,13,10,36,13,11,36,13,1
2,36,13,13,36,13,14,36
1530 DATA 12,14,36,11,14,36,
10,14,36,9,14,36,8,14,36,0,0
,0,0,0,0
1540 DATA 15,9,36,15,10,36,1
4,10,36,13,10,36,13,11,36,12
,11,36,11,11,36
1550 DATA 10,11,36,10,12,36,
10,13,36,10,14,36,9,14,36,8,
14,36,0,0,0,0,0,0
```

```
1560 DATA 13,7,36,12,7,36,12
,8,36,11,8,36,10,8,36,10,9,3
6
1570 DATA 10,10,36,10,11,36,
10,12,36,10,13,36,10,14,36,9
,14,36,8,14,36,0,0,0,0,0,0
1580 DATA 18,16,36,17,16,36,
16,16,36,15,16,36,14,16,36,1
3,16,36
1590 DATA 13,15,36,13,14,36,
12,14,36,11,14,36,10,14,36,9
,14,36,8,14,36,0,0,0
1600 DATA 17,11,36,16,11,36,
15,11,36,15,12,36,15,13,36
1610 DATA 15,14,36,14,14,36,
13,14,36,12,14,36,11,14,36,1
0,14,36,9,14,36,8,14,36,0,0,
0,0,0,0
1620 DATA 8,14,96,9,14,96,10
,14,96,11,14,96,12,14,96,13,
14,96
1630 DATA 14,14,96,15,14,96,
16,14,96,17,14,96,18,14,96,1
9,14,96,20,14,96,21,14,96,0,
0,0,0,0,0
1640 DATA 8,14,96,9,14,96,10
,14,96,11,14,96,12,14,96,13,
14,98,13,15,99,13,16,99,13,1
7,99
1650 DATA 13,18,100,14,18,98
,14,19,99,14,20,99,0,0,0,0,0
,0
1660 DATA 8,14,96,9,14,96,10
,14,96,10,14,98,10,15,99,10,
16,99,10,17,99,10,18,99,10,1
9,99
1670 DATA 10,20,99,10,21,99,
10,22,99,10,23,99,10,24,99,0
,0,0,0,0,0
1680 DATA 8,14,96,9,14,96,10
,14,101,10,13,99,10,12,99,10
,11,99,10,10,99
1690 DATA 10,9,99,10,8,99,10
,7,99,10,6,99,10,5,99,10,4,9
9,0,0,0,0,0,0
1700 DATA 8,14,96,9,14,96,10
,14,96,11,14,96,12,14,96,13,
14,101,13,13,99,13,12,99,13,
11,99,13,10,99
1710 DATA 13,9,98,12,9,100,1
2,8,99,0,0,0,0,0,0
1720 DATA 8,14,96,9,14,96,10
,14,101,10,13,99,10,12,99,10
,11,102,11,11,96
```

```
1730 DATA 12,11,96,13,11,101
,13,10,102,14,10,96,15,10,10
1,15,9,99,0,0,0,0,0,0
1740 DATA 8,14,96,9,14,96,10
,14,101,10,13,99,10,12,99,10
,11,99,10,10,99,10,9,99,10,8
,102,11,8,96
1750 DATA 12,8,101,12,7,102,
13,7,96,0,0,0,0,0,0
1760 DATA 8,14,96,9,14,96,10
,14,96,11,14,96,12,14,96,13,
14,98,13,15,99
1770 DATA 13,16,100,14,16,96
,15,16,96,16,16,96,17,16,96,
18,16,96,0,0,0,0,0,0
1780 DATA 8,14,96,9,14,96,10
,14,96,11,14,96,12,14,96,13,
14,96,14,14,96,15,14,101,15,
13,99,15,12,99
1790 DATA 15,11,102,16,11,96
,17,11,96,0,0,0,0,0,0
1800 RESTORE 1620
1810 RETURN
1820 RESTORE 1640
1830 RETURN
1840 RESTORE 1660
1850 RETURN
1860 RESTORE 1680
1870 RETURN
1880 RESTORE 1700
1890 RETURN
1900 RESTORE 1720
1910 RETURN
1920 RESTORE 1740
1930 RETURN
1940 RESTORE 1760
1950 RETURN
1960 RESTORE 1780
1970 RETURN
1980 RESTORE 1440
1990 RETURN
2000 RESTORE 1460
2010 RETURN
2020 RESTORE 1480
2030 RETURN
2040 RESTORE 1500
2050 RETURN
2060 RESTORE 1520
2070 RETURN
2080 RESTORE 1540
2090 RETURN
2100 RESTORE 1560
2110 RETURN
2120 RESTORE 1580
```

```
2130 RETURN
2140 RESTORE 1600
2150 RETURN
2160 CALL CLEAR :: CALL DELS
PRITE(#1):: CALL DELSPRITE(#
2)
2170 CALL CHAR(140,"780B0C1F
3F373F0F1E46AF532519020C8000
0000000000000F0FCFEFEF850CF
")
2180 CALL SPRITE(#1,128,5,49
,180,#2,132,5,49,207)
2190 CALL SPRITE(#3,140,14,1
08,180)
2200 CALL COLOR(8,2,1)
2210 DISPLAY AT(6,5):ANT-25,
" ANTS
   FOURMIS"
2220 DISPLAY AT(15,5):OEUF-2
,"FOOD
   NOURRITURE"
2230 DISPLAY AT(21,1):"TAPEZ
 UNE CLEF#PRESS ANY KEYPOUR
ENCORE...# TO START
        # AGAIN."
2240 CALL KEY(0,K,S):: IF S=
0 THEN 2240
2250 CALL DELSPRITE(ALL)
2260 GOTO 180
2270 END
2280 FOR R=1 TO 10 :: CALL P
ATTERN(#1,136):: CALL PATTER
N(#2,140)
2290 CALL SOUND(-1,3000,0,-8
,0)
2300 CALL PATTERN(#1,128)::
CALL PATTERN(#2,132)
2310 FOR Z=1 TO 25 :: NEXT Z
2320 NEXT R
2330 RETURN
2340 FOR I=1 TO 4 :: CALL PA
TTERN(#2,140)
2350 CALL SOUND(-1,110,0,-8,
0)
2360 CALL PATTERN(#2,132)
2370 FOR Z=1 TO 25 :: NEXT Z
2380 NEXT I
2390 RETURN
```