"We think it was a glitch..."

And penguins can fly! The first explanation for the computer 'malfunction' that caused the Shuttle launch to be delayed was wonderful! A glitch indeed... a glitch in the software! Rule #1 in computer technology is: "The problem is in the software - always in the software." But how did a 'glitch' (bug) get there? Rule #2: "Any 'little' routine or bug fix done at the last minute will not work." Ignoring Rule #2 results in 'bug fix' fixes (Empire again!*#%#!) and eggy faces. But what's 40ms between CPU's, anyway...

CLOAD
MAGAZINE INC.
P.O. Box 1267
Goleta, CA 93116
APRIL 1981

Side	Title	Turns Count CTR-41	Turns Count CTR-80
****	3D Cover	12 & 262	7 & 154
** **	Level 0	43 & 284	24 & 167
** **	Frog Song	93 & 320	55 & 188
** **	Bug Hunt Solution	144 & 359	84 & 211
****	Mapdisk (Model I DOS only)	199 & 401	116 & 236
**	Medieval Adventure	11 & 258	7 & 152
***	Space Gallery	153 & 366	90 & 215
**	Space Time	197 & 401	116 & 236
**	Gomoku	229 & 427	135 & 251
****			

CLOADing Notes - This tape may load at an ODD RECORDER VOLUME. Set the volume LOWER than normal for your first attempt, then increase it slightly until the tape loads. If the first copy of a program won't load, try the second. That is why it is there. Model I only: Put an AM radio very close to the keyboard, tune it to a non-station, and you can listen to the tape loading in. Adjust the recorder volume so the hash from the computer sounds 'cleanest' during a load.

Model III notes - Load the tapes at the LOW speed (POKE 16913,0). An occasional program will NOT run. There may be upper and lower case goofs in some programs. Arrow keys often are translated as follows: (↑, ↓, ←, →) = ([, \,], ^).

Special glasses are not required to view the 3D Cover, but the thing on the screen looks like it would go good with crackers, melt nicely on a burger, or drive mice to salivate.

With our usual timeliness, here is our April Fool's contribution - the back-talking TRS-80! Level 0 lets the real personality of your computer come through. Then this personality takes over! There is a way to make your machine subservient again, but telling you how would not be as nasty...

On to Calabasas County! Only the frogs in Frog Song don't jump very far. But they do make a sound as they leap. Now, once they have finished jumping, you must prod them to jump again - in the same sequence! If you don't have an auxiliary amplifier to plug the large grey recorder plug into, just plug the grey plug (enough 'plugs', don't you think?) into the recorder, push the play and record buttons, and (dare I say it) plug an earphone into the ear jack to listen.

The sound routine in Frog Song is worth looking at. Since the routine itself can be placed anywhere in memory ('relocatable code'), Mr. Quante decided to place it in a string variable (A$, line 2000) so that the computer takes care of where to put it and we don't have to set memory size. He kindly left the routine that packs A$ with the code in his program (line 2000-2030), but he removed the data that he used. Below is a reconstructed version of his original routine with comments added:

```
1995 REM  MAKE A$ BIG ENOUGH TO HOLD ROUTINE (25 SPACES THIS TIME)
2000 A$="ABCDEFGHIJKLMNOPQRSTUVWXY"
2002 REM  YOU MUST 'GOSUB 2000' FIRST TO ALLOCATE SPACE FOR A$
2005 RETURN
2008 REM  FIND THE ADDRESS OF THE 1ST ELEMENT OF A$
2010 A=PEEK(VARPTR(A$)+2)*256+PEEK(VARPTR(A$)+1)
2015 REM  READ DATA AND PUT IN A$ STARTING WITH FIRST ELEMENT
2020 FOR I=0 TO 24: READ X: POKE A+I,X: NEXT I
2030 STOP: REM  ROUTINE DONE
2035 REM  DATA RECREATED BY PEEKING THE CURRENT A$
2040 DATA 33,37,64,78,33,39,64,70,62,1,211,255,16,254,70,62,2,211,
          255,16,254,13,32,239,201
```

To run the above BASIC routine, you would type 'GOSUB 2000: GOTO 2010'<ENTER>. A$ will then be filled with the sound routine and you can delete all of the lines except lines 2000 and 2005. To use the A$ routine in a program, you must 'GOSUB 2000' at the beginning (see line 100) to let the program know where to find the noise. So that the A$ routine could be used by disk systems as well, you would want to include a line similar to line 210 to set up the USR function. Finally, before calling USR(0), you would have to POKE values in locations 16421, 16423, and 16425 as is done throughout the program to get the notes you want.

And now we come to last month's promised feature, Bug Hunt Solution. For those new to CLOAD this month, last month's issue contained a program (Bug Hunt) that had built in bugs for the user to weed out. This program should draw a bug that runs back and forth across the bottom of the screen, then (after you hit <enter>) draw a very moody large bug. If you get an error with this one, the program must have loaded poorly. Just to repeat myself at what appears to be a perfect place: If you still find it impossible to load this (or any other program) after trying the loading tips given in the tape index block above, just return the tape for a fast replacement.

Now we have a program just for you Model I'ers with (35 track, single density?) disks - Mapdisk. This program lists the programs on any particular disk. Big deal!? It also will list the disk gran by gran and tell you which program is residing (in part) in that gran. This can be VERY useful if a section of your disk goes bad and you want to try and salvage what you can from the wreckage. It can also be informative:

Have you ever wondered why a program takes an inordinate amount of time to load in from disk? It is probably because that program is stored in little pieces all over the disk. When the program is loaded, the drive head has to jump back and forth to pick up all of the bits of the program. With Mapdisk, you can see just how the program is distributed and see the steeplechase that the head has to go through. But why is a program stored in non-adjacent pieces in the first place?

Let's say that the disk is almost full, with 3 grans (68-70) free at the end of the disk. Now you decide to delete this month's cover from the disk because you want to feed it to your dog (dogs love cheese!). The cover took two grans (11 and 12). Now you have grans 11,12,68,69, and 70 free. Then

you try to save Mapdisk on disk, and it takes four grans. Since there are not 4 connected grans free, the disk operating system (DOS - the 'lawbook' that regulates how information will be saved and loaded to disk) can be written to do 1 of 3 things:

1) An error message can be flashed saying, "Put it in your pocket, I'm stuffed." This is not very clean, is frustrating, and tends to fill disks very fast. However, if you have 100 mega-bytes of storage on line, this method is fast and trouble free. Bubble memory, at this time, works somewhat like this. You can write to it, but you can't change what you've written. So you need a lot of it to last you a while (1 mega-byte of bubble memory would be equivalent to about 10 standard Model I mini-floppies).

2) The disk can be 'packed'. This means moving what is stored in grans 13-67 and moving them all down 2 grans so that grans 66-70 are free and Mapdisk can be saved to grans 66-69. This makes for the quick reloading of a program, but saving a program to disk can be very time-consuming. Many systems require the user to pack the disk, making the DOS a little less transparent to the user (not a feature in my book).

3) Part of the program can be saved in grans 11 and 12, and the rest can be saved in grans 68 and 69. Then a directory has to be set up to point to all of the pieces of a program. Or at the end of each gran there is either a stop flag or a pointer to a gran containing the next part of the program. This way all of the grans get used, but loading times can be slowed considerably.

Radio Shack uses method 3 in their DOS, so you get fragmented program storage (and you get programs like Mapdisk).

Heralding the 2-player Medieval Adventure! You and another adventurer race around a palace looking for all the goodies, then bring them back to your home base. The one that gets ALL of the treasures to his/her base wins. But finding them is not easy, and keeping them safe from your opponent is even harder! A thief in the night...

EXTRA, EXTRA! You can use the first 3 letters of an object or command instead of the whole word. For example, you can type 'ope doo' instead of 'open door'. Second, you can use just one letter to go a direction ('s' instead of 'go south'). Last, you can type commands in lower case if you have it.

NOT-EXTRA, NOT-EXTRA! A bug was found after this month's issue went to 'press'. In line 5 of Medieval Adventure, there is a 'CLEAR140'. After a lengthy canvassing of the castle, an 'OS' (Out of String space) error may appear. Change 'CLEAR140' to 'CLEAR200'. This has been tested, and works (where have I heard that before?) as long as you don't type in 40 character names or commands. If you have a Model III, you will have to delete line 10000 in order to have enough memory. Also, the editor ignored his editing responsibilities and let words like 'armor' and 'allgator' slip by. He also let such twelfth century terms as 'OK' (note - sarcasm) stay in the program. He is now on bologna sandwiches (no PEANUTBUTTER!!) until he mends his ways.

Space Gallery is one of them-there arcade type games.

Give me the time - as if I was on my way to Io at Warp .7! Space Time is a little demo that needs the current date, two earth-based people, two ship-based people, and a rate of motion for the ship. The program then calculates the relative passage of time for all of the people, as you watch. The program does increment time a bit faster than standard earth-time, so you don't have to dedicate your computer to this program for more than a

year or so... (more sarcasm!!!)

Gomoku - an ancient game that is now played by an all-to-logical computer. Just try and beat your TRS-80! The object is to get 5-in-a-row first - on a 9 by 9 grid. Tic-Tac-Toe on a larger scale. Good Luck.

The Empire strikes back - and wins...

Last month I published a few bug fixes for February's 1981's Empire program. I ignored Rule #2 (see introduction of this issue) and one of the 'fixes' didn't fix. Also, since then a few other 'not nice' things have been pointed out. The mods for lines 245 and 277 may cause an Out-of-Memory error in the 16K Model III. The mod for line 245 is rather important, so edit the REM lines as described below to get a few more bytes if needed. Now here goes the third installment, 'The Return of the Fixes':

Remodify line 245 (underlined parts are to be added to the original):
E2=0:GOSUB57:...:IFE2<>INT(E2)ORE2<0GOTO245ELSEIFE2=0THEN255ELSE...

You used to be able to get merchants and nobles for free by just buying 0 marketplaces or 0% of the palace. To fix that:
Add 'J=0:H=0:' to the beginning of line 231.
Add '+(E2-1)*(J+H)' after the 'E2' in line 247.
Add 'ELSEA(K,7)=A(K,7)+J*E2:A(K,18)=A(K,18)+H*E2:A(K,3)=A(K,3)-(J+H)*E2' at the end of line 253.
Retype line 233 as '233 E1=1000:J=RND(7):GOTO245'.
Retype line 243 as '243 E1=5000:H=RND(4)'.

So you don't loose fighting turns for trying to send more soldiers than you have:
Move 'IQ=IQ+1:' from the front of line 277 to the front of line 279.

So you can't fight with 0 soldiers:
Add 'IFI1<1THEN259ELSE' after ';I1:' in line 277.

So you can't fight non-existent countries:
Change 'ANDA(K,0)' to 'IFA(I-1,0)' in line 271.

So you don't get weird amounts of money due to a computer's country buying grain from you:
Change 'QR*A(Q,6)*.9' to 'INT(QR*A(Q,6)*90)/100' in line 469.

Model III'ers especially - to get a few more bytes of RAM, you can change the single quote used to mark a 'REM' statement in 11 lines (like line 33) to 'REM'. It turns out that a single quote uses 3 bytes of storage, while 'REM' only uses 1. A savings of 22 bytes if you change all those lines in Empire!

A note to repeat - Mike Gieg discovered that the DOS Repeat program (February 1981) would not work after the first run on his system (TRS-DOS 2.3), but always would on a second attempt (done immediately after the first).

I'm off to see the Shuttle land...

Head in the clouds,

Dave