The storm before the storm...

We have had the production room organized before - once. Rose had it fairly well under control (at least she knew where to find stuff). With her gone and the holiday season in full swing, the production area now looks like the living room of a family of 12 after the presents have been opened on Christmas morning. Just when one of us has straightened things a bit, another batch of tapes, papers, etc. gets dropped somewhere in the middle and spread around. It's the deep-dish theory of production...

CLOAD MAGAZINE
P.O. Box 1448,
Santa Barbara,
CA 93102

December 1983

Side	Title	Filename	Turns Count CTR-41	CTR-80	CCR-81
****	History of the Computer Part 1	A	8/245	5/145	3/121
** **	Gravitron (SYSTEM /)	GRAVTR	107/318	63/188	45/172
** **					
****					
**	Project Triad Adventure	A	8/254	5/150	3/127
***	Maze Race	B	144/355	85/209	63/201
**	Battery Charging	C	219/414	129/244	105/256
****					

Tape 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 only: Load the tapes at the LOW speed (POKE 16913,0).

Subscribers - The month on the mail label is the last month of your subscription. If you have a cassette subscription, the number next to the month is the amount it would cost to convert the rest of your subscription to the disk version ($4.20 per issue for 6 or less months, $3.75 per issue if more than 6 months).

If Adam had a computer, would it be an Osborne? See what it could have been like with **History of the Computer Part 1** (by Jim Korzun).

If you were a bit disappointed when you discovered that this issue of CLOAD only contained 5 programs, smile! **Gravitron** (by Dennis Lo) is worth it! Not only is it the longest program we ever published (251 seconds to load from tape), it is probably the BEST game we have ever published. Eat your heart out, Big Five! The object is to pilot your spaceship over the land and through a cave to rescue a happily waving person. Unfortunately, the gravity of the 'world' is trying to pull you down to destruction, rockets are trying to collide with you, two types of gunners are trying to shoot you down, and an annoying spaceship comes on the scene to make things even more difficult.

When you start, you choose your level of play. The higher the level, the more bonus points you can receive for rescuing the person on your starting level (you lose bonus points every time you lose a ship, however). To play, use the following keys (also compatible with some joysticks):

<Arrow keys> or **QAOP keys** move ship.
<Space bar> or **<shift>** to fire.
<Clear> destroys all ships and their missles on the screen (you get a limited amount of these 'smart bombs').
S suspends the game (hit a fire key to restart).
<Shift break> to quit the current game and start another.

Notes: The game has sound, so connect the large, grey AUX plug to an amplifier. It is in machine language, so to load it from tape, type **SYSTEM**<enter>, answer the ***?** with **GRAVTR**<enter>, and after 251 seconds answer the next ***?** with **/**<enter>. The tape version loads from 17408-32255 and executes at 31905 (in hex 4400, 7DFF, and 7CA1). Disk users - As you may see, the tape version will crash your DOS if it is loaded at 4400 hex. However, you can load and run it using our **Disk Exec** (August 1981) or give it a load offset with some utility like NEWDOS LMOFFSET. The version of **Gravitron** on the CLOAD Disk Version has the offset and initially loads from 24832-39700 and executes at 39680 (in hex 6100, 9B14, and 9B00).

Three, three, three adventures in one! It's **Project Triad** (by David Lo - yes, brother of Dennis). A seemingly standard adventure where you must find the bomb. But try this... When you begin, go South, **LOOk** at the **TERminal**, and **PLAy** the **ADVenture**. An adventure within an adventure! Good luck (three times).

Adventure notes: You can use one letter abbreviations for directions (ie: N for **GO NORTH**), I to get an inventory of the things you are carrying, and 3 character abbreviations for commands (ie: **LOO TER** for **LOOK TERMINAL**). This adventure (or should it be, 'These adventures') has a 63 verb vocabulary so you will get fewer 'What?' answers to your commands than you do from most BASIC adventures.

Lost in a **Maze Race** (by Graham Wilson). This is another in the long series of you-are-the-rat maze games. However, there is a machine language routine that does the corridor graphics, making this game fast, interesting, and (for me) frustrating. Try the 6 by 6 maze to get hooked, and the 14 by 14 maze if you're the type of person who goes backpacking in the Grand Tetons in December without a map. Features - You are timed as you go through the maze. After you FINALLY make it, you get to play the same maze again, this time to better your last fumbling time. You use the **right/left arrow keys** to turn (you do not move, just turn) and the **spacebar** to go forward.

Oops, you left your car lights on (and your keys are locked in the car, but that's another program) and your battery bought the farm. Recharge it correctly with **Battery Charger** (by Charles Evans). Charles wrote it to give a helping boost to his friends who own battery-powered golf carts, but it can be a help to anyone who recharges their own car battery during the winter freeze (it got down to, heavens, 43 degrees last night in Santa Barbara). In the instructions he gives the technical aspects of battery charging and a method for finding the charger peak voltage. Then the program simply asks you for the charger peak voltage, the battery voltage, and the resistance or current. Note: The question, "Enter the resistance or current?" may be confusing. If you know the resistance, enter that value and you will get the current. And if you know the current, enter that value and you will get the resistance.

Planets out of orbit...

William Oaks, the author of **Planets over Seattle** (November 1981) had a bug pointed out to him that caused small errors for some planet locations. Line 155Ø should have been:

```
155Ø LI=(ATN(TAN((HL-AN)*DR)*COS(I*DR)))/DR+AN
```

Route 66...

Last month I said to change the **RUN** at the end of line 9Ø in October's **Interstate** to **GOTO14** to allow you to play another game without going through the long READ process. It worked as long as you didn't hit a city that you visited in the last game. Not good enough? By coincidence, if you make the change **GOTO41**, the offending array will be reinitialized and the game will work correctly the next time!

Call the PILOT...

John C. Fowler of Los Alamos, New Mexico found a couple of interesting disk I/O features not documented in October's **PILOT**:

1) The **KILL** command with the format **KILL"filename/P"**.
2) To save, load, or delete files with extensions, passwords, and drive specifications in the filename use the syntax **filename/ext.password:d/P.** Ie: To delete a file called JUNK/BAS on drive 1 with the password CLOAD use the command **KILL"JUNK/BAS.CLOAD:1/P"**.

Machine code conversions...

I'm sure that most of you have a machine language routine in a BASIC program or two that won't work on your system since it writes over DOS or loads into the top 16k of a 48k machine, and you only have 32k. We have that problem a lot. And although we don't always succeed, we usually can, through certain techniques, get a machine language routine to load where we want it to. This month's **Maze Race** is a prime example. It was written for a 16k tape system and it had a machine language routine that loaded into the top of 16k. This played havoc on a disk system, however, since Disk BASIC program area starts about 12,ØØØ bytes higher than the standard BASIC program area. The program is about 8ØØØ bytes, so the machine language routine loaded right over the BASIC program, crashing the system. If the machine language routine could be loaded higher in memory when it was used in a disk system, then things would be ok. Grady went to work...

First, he had to disassemble the routine (I'm going to switch to hex now whenever I talk about numbers and locations). So he looked at the code and found, in line 46, that the BASIC program loaded the routine at 7918 (31ØØØ decimal). Most disk operating systems start free memory at or below 7ØØØ, so the machine language routine would not be destroyed by the DOS (the routine still would destroy the BASIC program, however). He then loaded **Maze Race** from standard BASIC (when in DOS, hold the <break> key and hit the <reset> to drop down to standard BASIC) and ran it to get the machine code into RAM. Now he jumped back into DOS (hitting the <reset>, pulled out a disassembler (Multidos has a nice one built in), and disassembled the code from 7918 to 7CFE, or where he calculated the end of the routine was. Now came the fun part.

Grady carefully looked at each instruction. If there was a JP or CALL that accessed a location between 7918 and 7CFE, he wrote down the

location and value of the HIGH ORDER BYTE. There is no need the change the whole address, we just want the routine to load higher in memory. For instance, at 791E there was a JP Z,H7CA9. In machine code the instruction was CAA97C, so he wrote down 7920 7C.

Next, he looked for any LD or ST that looked at or modified memory between and/or close to 7918 and 7CFE. Again he wrote down the locations and values of the high order bytes that he found. Ie: At 7C6D he found an LD HL,H7CE6. In machine code this was 21E67C and he wrote down 7C6F 7C.

Now that he knew all of the bytes to change, he had to modify the BASIC program to recognize when it was running in Disk BASIC and to act accordingly. In line 46 he set up the entry points (DEFUSR) for the disk version (2000H bytes higher than the cassette version). In line 104 he set up the start address of the machine language routine and the offset (AA=2000H) if it was a disk system. Line 106 READ and POKEd the routine in the appropriate place in memory and returned if it was a cassette system. If it was a disk system, however, he went on to line 108 which read in new DATA that Grady added in line 310, giving the location in memory of every high order byte to be modified (offset from 9900H). He then PEEKed the value at that location and added 20H (32 decimal) to it, effectively adding 2000H to each critical address.

Next, Grady looked through the program to see where else there might be PEEKs and POKEs into the machine language area and modified them to work with tape or disk. Line 22 makes sure that the value of B is negative if it happens to be greater than 32767 (the greatest positive integer allowed), and line 32 sets the value of B plus the offset AA (0 if cassette, 2000H if disk).

Finally, he ran the new program. Of course, it crashed and he had to repeat some of the above steps until the program worked.

Another first...

Since we got the December issue out before Christmas, for the first time in CLOAD history, I'd like to wish you a timely happy holiday season. But that would be out of character.

Happy Thanksgiving,

Dave

ed.