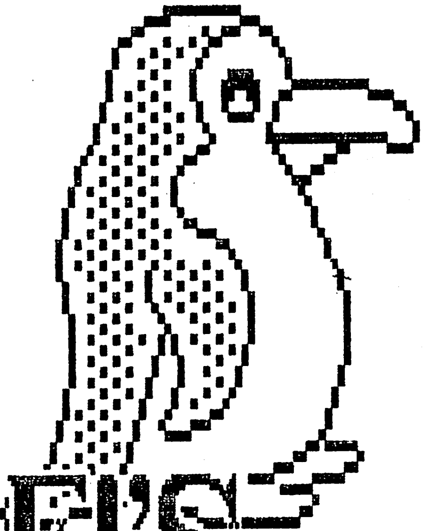
THE COMPUTE!'S GAZETTE DISK

JANUARY 1994



```
*****  *****  ** **  *****  *   *  *****  *****    *  *****
*      *   *  * * *  *   *  *   *    *    *       *   *
*      *   *  * * *  *****  *   *    *    ***    *    *****
*      *   *  * * *  *      *   *    *    *               *
*****  *****  *   *  *      *****    *    *****       *****
```

G A Z E T T E   O N   D I S K
==============================

Compute's Gazette Disk
January 1994
==

UNSCRATCH

By Randy J. Clemmons

During the past nine years of computing, on more than one occasion I've experienced the heart-sinking feeling that follows mistakenly scratching a file from a disk drive. Chances are it's happened to you, too.

Maybe it's the latest version of a program you've been working on for weeks or the research paper that's due tomorrow. Or maybe you scratched an older version of a program, and the newer one has developed unexpected errors the older version didn't have. Good news! You can use disk utility programs to recover a scratched file if you haven't overwritten it, and your chances of reviving the file are excellent.

Armed with a track and sector editor, speaking in hex, time permitting, and disk operating system references handy, you can revive that scratched file. Who knows: You might even enjoy browsing around in your tracks and sectors. Just be sure to make a backup copy before you go exploring. Of course, if you don't feel the need to learn the intricacies of disk directories and DOS in order to save a file, what do you do? Isn't there a simple way to get your file back quickly and painlessly?

Unscratch to the rescue. Unscratch is easy to use with self-explanatory prompts on the screen. Simply type LOAD "UNSCRATCH",8 and then RUN.

The program's first prompt lets you choose a disk drive number from 8 to 11. Single drive users will want to press the Return key to use the default drive number 8. After selecting the appropriate drive number, the filename prompt appears. You then have four options:

1) Enter the name of a scratched file to restore,

2) Enter $ to display the disk's directory,

3) Enter * for a wildcard search of all scratched files, or

4) Press the Return key to exit the program.

Filename Prompts

Entering a filename and pressing Return starts the program searching for a specific scratched filename. If that filename doesn't exist, then an END OF SEARCH message will be displayed, and the program will end. Enter RUN again if you wish to restart the program.

Entering an asterisk at the prompt starts a wildcard search for scratched files. A wildcard search finds and displays all scratched filenames in the disk's directory. An option to restore each filename found will be made available.

When a scratched file is found, you'll be prompted with a (Y/N) decision to restore the scratched file. Press N to look for other scratched files on the disk (in the wildcard mode only). If scratched files aren't found, then an END OF SEARCH message is displayed, and the program ends.

After finding a file and entering Y to restore the file, another prompt appears requesting the user to provide a file type of 1, 2, 3, or 4. There are actually five file types that the disk operating system uses.

0 =	DEL	Deleted (scratched)
1 =	SEQ	Sequential
2 =	PRG	Program
3 =	USR	User
4 =	REL	Relative

Enter an appropriate file type for the file which was scratched. As shown above, file type 0 indicates a deleted or scratched file. You'll have to enter one of the other four appropriate file types (1-4) to unscratch the file.

If you're uncertain about the file type, you may wish to apply the following logic. BASIC programs are type 2 (PRG) files. For files created by your word processor, check the directory of a disk with saved files to determine the correct file type. Some word processors have an option of saving files as either PRG or SEQ file types; the default type is your best choice. Rarely, if ever, will you encounter USR or REL file types. If you chose the wrong file type, simply delete the file and run Unscratch again, entering the correct file type.

Entering a dollar sign and pressing Return at the filename prompt displays the disk's directory. After displaying the directory, press a key to return to the filename prompt.

Pressing only the Return key at the filename prompt ends the program.

Basic DOS

A basic understanding of the disk operating system can be helpful or can at least explain what happens when an attempt to Unscratch a file isn't completely successful. Your odds of restoring a file with Unscratch are almost 100 percent if no part of the file has been overwritten. Even if you have saved files after the file was scratched, you may still be able to restore the file totally or in part. Your odds decrease as the quantity and size of files saved to the disk after the file was scratched increase.

Technical Bits

The Commodore disk operating system performs a Boolean AND operation with the hex value of $80 and one of the five file types. This byte is stored in the disk's directory as the first byte which describes a file. This file type byte is part of the directory data for the filename in the directory. If the filename is for one of the first eight files saved to the disk, then the file type can be viewed using a track and sector editor at track 18, sector 1.

The standard 1541 formatted disk contains 35 tracks, with track 1 at the outside and track 35 at the innermost track. The number of sectors per track varies, with the outer tracks containing 21 sectors each and the innermost tracks each having 17 sectors. Sectors or blocks can hold 254 bytes of a file. Technically, sectors contain other bytes such as sync marks, track and block ID's, checksums, gaps, and so on. Refer to the back pages of a 1541 disk drive manual for additional details.

The first two bytes of any sector (not to be confused with the first bytes of a directory entry) serve as a link which points to the next track and sector. These next track and sector link bytes form a chain of pointers used to connect as many sectors as required to store a particular file. The link pointer in the last sector of the chain is set to track 0. Probably only a few hackers are still with me, but now you're ready to understand what happens when an attempt to revive a scratched file isn't a success.

Crossed linking can occur after files are scratched and other files have been saved. Crossed links exist when two or more files have links (next track and sector pointers) pointing to the same next track and sector. This happens after a file is scratched. Then the disk operating system sees the tracks and sectors of a scratched file as available for use again. This can cause the resurrected file to contain parts of two or more different files.

If a saved file happens to claim the scratched file's first sector, you've got no other choice than to break out a track and sector editor and go hunting. That's your only hope of recovering any portion of the scratched file. With the scratched file's initial sector taken, there's no next link pointer to the rest of your file. So now you have to search track by track, sector by sector in hopes of finding some recognizable portion of the lost file. Not a pretty sight, and you'll need to learn how to rebuild your own directory entries.

Manual linking of the pointers for found sectors of your lost file will probably be necessary. This can be accomplished using the track and sector editor. At this point such words as educational, challenging, frustrating, anxiety, despair, and happy hunting come to mind.

I hope that you never need Unscratch. But the more you use a computer, the more likely you'll someday scratch a file to free up some disk space or be a little too hasty in answering a delete prompt by mistake. However, with Unscratch in your toolbox, you'll be ready to quickly recover from such a mishap.

Randy Clemmons lives in San Diego, California. He is the author of Totalizer and Director-Ease.

Gazette, January 1994

BLOX TROT

By Authur Moore

Blox Trot is a mind-challenging racing game that places you head-to-head with a friend. It's for the 64, and you'll need two joysticks to play.

The object of Blox Trot is to shift blocks within the grid to reach the solution shown in the center of the screen (a complete row of red, followed by rows of cyan, dark blue, and green). Once a grid is complete, you will receive a point. Beat your opponent to your chosen final score, and you win the game. To further complicate matters, if your opponent has a point and you complete a grid, you won't increase your score by one but, instead, decrease his score by one.

To shift the blocks, move your cursor to the block you wish to move, hold the fire button (the cursor reverses), and move the block to the desired position. Moving a block up or down moves all the blocks in that column up or down also; moving a block left or right affects all the blocks in the same row.

When players of different skill levels play, the settings may be adjusted to account for the difference. A larger grid generally takes more time to complete. The size (width) of the grid may be adjusted from four to seven blocks (the default is five).

Perhaps more effective is adjusting the winning score. While the more experienced player can be expected to score four points before claiming a victory, the novice can score a single point to win. The default setting is three. Setting each score to four with two able players can result in a game that takes quite some time to finish.

To change the settings from the main menu, move the highlight box to the aspect you wish to change and then move left or right to change the value. When both fire buttons are simultaneously pressed, the game will begin.

Note: Long-time Gazette readers may recognize the melody used to signify a win. It is borrowed (although rearranged) from Donald Eddington's Improvisor, April 1987.

Arthur Moore lives in Orlando, Florida.

Gazette, January 1994

SPEEDRAM COUNT

By Frank Gordon

Speedram-64 (December 1992) is a modification or patch for SpeedScript that lets Gazette's word processor be used with a 1764 RAM expansion unit (REU) or two drives. Either drive 8 or 9 can be selected from within the program. Press Ctrl-Y to select drive 8 or Ctrl-N for drive 9.

WordCount (available on the Gazette SpeedScript disk) uses the key combination Ctrl-W to count the number of words in the current document. This is a handy feature for students or professional writers. Both programs use the 143-byte area between 9329 ($2471) and 9472 ($2500), the beginning of text. This made the two patches incompatible. Many SpeedScript users expressed a desire for both features in SpeedScript.

To make it possible to combine these two programs, the text area can be raised to 9728 ($2600) by line 20 (POKE 2473,37) in the accompanying SpeedScript converter program. (See SS Converter.) Now 399 bytes are available so that Speedram-64 and WordCount can be combined to give Speedram Count with three commands: Ctrl-N, Ctrl-Y, and Ctrl-W.

Creating Speedram Count

In order to modify your version of SpeedScript, follow these steps.

1. Load and run SpeedScript.

2. Select your favorite background and text colors with Ctrl-B and Ctrl-L.

3. Exit SpeedScript by tapping the Restore key and responding to the prompt by pressing Y.

4. Type POKE44,40:POKE10240,0:NEW and press Return. This will protect SpeedScript's BASIC area.

5. Load but don't run SRC Converter with a ,8 extension.

6. Insert a work disk onto which you want to store Speedram Count. Enter RUN and press Return.

SRC Converter will run and save the modified version of SpeedScript to disk with the filename SPEEDRAM COUNT.

When the save is finished, turn the computer off and back on before using Speedram Count.

Using Speedram Count

If you have a 1764 REU, install it and initialize it as drive 9. At

this point you may also wish to copy any SpeedScript programs to the 1764. If not, return to the menu and quit. Now load and run Speedram Count like any BASIC program. Your familiar Speed
Script screen should appear in the colors that you selected.

You can test Speedram Count by pressing Ctrl-N (for drive 9) and then Ctrl-4 for a directory. You should get a rapid listing of any files stored in the REU (or on drive 9, if you have two drives). Press Ctrl-Y (for drive 8), and Ctrl-4 will list programs from that drive. Also, when you press Ctrl-N or Ctrl-Y, the drive number appears on the command line. Shift from one drive to the other in this rapid manner to load or save SpeedScript files.

You can then use Ctrl-W for the word count in any of these files.

For more information on how Speedram Count works, see WordCount and Speedram-64, since this program is a combination of these.

Gazette, January 1994

Ham Net

By Ralph L. Knight

Ham Net is a utility program that should be of interest to any amateur radio operator who operates a network. Networks consist of a group of ham radio operators who meet at a prescribed time and frequency to relay messages, conduct other activities, or simply chat.

Hams usually check in with a net control station that acts as moderator. With Ham Net, the net control station logs in participants by entering the suffix of the ham's call sign. For example, for WS5A, net control would enter A. For K4BYQ, net control would enter BYQ. Ham Net searches the data for such a call and, if one is found, prints the caller's complete call sign, name, and other information. This provides a means of identifying callers and makes a record of all who check into the net.

If no record is found, net control can enter the information manually and then save it to the program's data statements or as a disk file.

Ham Net consists of a loader (Ham), a machine language directory reader (Dirbyt), and the main program (Net). Start the program by loading and running Ham.

To start, the user enters information about stations that are expected to participate in the net. Enter this data by selecting item 1 from the menu. Sample names have been inserted in lines 6500, 8100, and 8102. These may be deleted.

Start the Net

Menu choice 5 starts the net. The search for a call suffix is fast since the program need search only through one letter of the alphabet rather than all the data statements. For example, suffixes beginning with the letter A are in data statements in lines 6500 - 6598. Letter B is stored in lines 6600 - 6698, and so on.

Call signs entered through choice 1 are assigned to the applicable data line numbers. The search by suffix letter category is accomplished through the use of the machine language code entered in lines 4004 and 4006.

You can return to the main menu at any time during the check-in process by pressing the English pound key. You can then list all stations that have logged in by pressing choice 6. Should you have numerous stations logged in, Ham Net allows for paging a list too long for one screen.

Menu choice 2 searches all data lines by various categories.

Choice 3 resaves the program to include any changes that may have been made in the data statements.

Choice 4 ends the program.

Choice 7 provides for saving the screen list of check-ins to disk. Files are named by date, for example, 1-19-94. The program places a NET prefix on these files, but you need not enter it to recall them later.

Choice 8 loads a check-in list that has been saved to disk.

Choice 9 lets you read the disk directory from within the program.

Choice 10 lets you send the list to a printer.

Choice 11 provides a means for recording the number of check-ins during any month. The daily total of check-ins is added to the count under the Add option following the PRINT statement in the January sample.

Neither choice 1 (adding a new member) nor choice 11 (adding to the net count) should be used while the net is in progress since the screen list of check-ins will be lost. Use these options only after the list has been saved under choice 7.

Ralph L. Knight lives in Roswell, New Mexico.

Gazette, January 1994

64 Dimensions

By Eric J. Bryant

This curious game adds a new twist to an old pencil-and-paper classic, tick-tack-toe. Unlike the regular game, which you play on a 3 x 3 grid, 64 Dimensions lets you play on four 4 x 4 game grids. The extra grids add a third dimension to the game.

64 Dimensions is a two-player game for the 64 written entirely in machine language. It loads and runs like a BASIC program.

Playing the Game

64 Dimensions can be played with either the keyboard's cursor keys and the Return key or with two joysticks. To make gameplay smoother, either player can enter moves through keyboard or joystick. Play isn't restricted to player 1 using the joystick in port 1 and player 2 using port 2. However, 64 users should all be aware that problems sometimes occur with joystick port 1, so I recommended you avoid using it if you plan to use the keyboard.

After you load and run 64 Dimensions, press any key to exit the title screen. The next screen will be divided into two sections: the game area, which contains the four 4 x 4 tick-tack-toe grids in the top half of the screen and a text window in the lower portion of the screen.

The text window will prompt you to enter the names of the two players. The game will generally assign player 1 to play the X's and player 2 to play the O's. Press any key to begin the game. The game starts with an X placed in the upper left corner of the first grid unless you start the game by pressing the cursor key or joystick.

Move the flashing cursor to position your piece and then press Return or the fire button to play the X or O on the grid. The cursor will cease to flash and turn yellow should you move over a position already occupied by an X or O. You cannot place your piece on a square that's already occupied.

As in tick-tack-toe, players take turns placing pieces on the board. The first player to place four pieces in a row horizontally, vertically, or diagonally on any plane wins.

To restart the game or even quit it, press the Run/Stop key. Move the cursor to Restart to end the current game and start another or to Quit to quit the game altogether. If you wish, you can also cancel either option and return to the game.

Mastering the Third Dimension

Playing a game in three dimensions may take some practice before you get the hang of it. Picture four 4 x 4 grids placed one on top of the other, forming a cube. This essentially is what you are seeing in the game area. The first layer of this cube corresponds to the first grid on the left of the playing screen. The layers descend until you reach the last grid on the right, which would be the fourth or bottom layer of the cube. You can get a better feeling of this cube by moving the cursor about, advancing it past the right or left edges of a board to the next or previous layer. If you move the cursor past the screen . edges, it will wrap around to either the fourth or first boards.

To aid you further, figures 1 through 4 show the four winning combinations that can take place across the boards in third dimension. Use these illustrations and experiment with the game to get acquainted with 64 Dimensions.

Eric J. Bryant is a computer science major at Hampton University at Hampton, Virginia, but is a native New Yorker from the Bronx.

Gazette, January 1994

Link 'Em

By John Cameron

Link 'Em is a fun, easy-to-play strategy game for two to four players. Given markers of different colors (red, blue, green, and yellow), the object is to place them on the game board in such a way as to build a chain of four in a row. At the same time, you should try to prevent your opponents from doing the same thing.

But the fun doesn't end there. As well as earning instant points for placing markers on the board, you may land on special board squares. Here, you may lose moves, gain free moves, or lose your marker to deadly Pit Traps. Also, you may uncover special bonuses of up to 250 points. Link 'Em requires strategy, cunning, and not a little bit of luck to win.

The game consists of two programs: Link 'Em, written in BASIC, and Link.Data, which is written in machine language. Simply load and run the BASIC program and, within seconds, the fun will commence.

Playing the Game

The first thing that you'll see upon starting the game is the title screen. To exit it, press the fire button on your joystick. Note that both joystick ports are active in this game, so players may use either one or two joysticks.

After you leave the title screen, it takes a moment to build the game board. The playing screen is divided into three sections. At the bottom is an empty area where messages will appear. In the middle are four compartments where scores and other information are kept. Initially, these are both blank. The top part of the screen is filled with 126 blocks (7 rows by 18 columns). These are the blank squares that form the game board. The game continues until every block has been revealed by the players.

Once the board is complete, the game is ready to begin. A message appears in the bottom window asking you how many opponents you have. Move your joystick left or right to select the number of people you'll be playing against and then press the fire button to make your selection.

Each player is represented by a different number and colored ball. For example, ball 1 is red. The score for each player is displayed in the compartments in the middle of the screen. To help you know whose turn it is, both the score numbers and the two arrows on either side of their marker icon turn bright white for the active player. The other scores and the icons of the other players are cyan.

Players move a cursor around the board to select a block to uncover. To speed up the selection process, you may also loop around from one side of the board to the other--from top to bottom or left to right.

Upon selecting a block and pressing the fire button to uncover it, a number of things might happen. An instant prize of 10, 15, 25, 50, or 75 points may be uncovered. When this occurs, the bonus is added to your score, and a message stating the increase is given in the message window. This is where luck plays such an important part in winning the game.

Should you uncover the red Stop Light, a message appears stating that you've lost a turn although your marker does remain onscreen. The Pit Trap, in comparison, doesn't allow your marker to stay on the board. On the other hand, by finding the green Go Light, you are given a free move.

Bonus Games
A special block which may be found uncovers a large question mark in the current player's color. This is the Mystery Bonus of 25, 50, 75, 100, 150, 200, or 250 points. Upon finding it, you have an option of selecting the prize that you'd like to try for.

Different points are arranged in a row. By moving left or right and pressing the fire button, you move the yellow arrow to select the points you want. Note that the greater the prize, the more difficult the game. Therefore, if you need the block that you're playing for to complete a chain of four, it's often a good idea to play for one of the smaller prizes. You might not gain as many points, but your odds of winning won't be quite so slim. Only by winning the bonus game will your marker be placed at that block and your score increased. Losing places an empty block in that spot, ruining any chances of building on your chain in that direction.

When you're playing the bonus game, a number of different shapes in different colors appear in a row, with numbers beneath each shape. Three items appear when you're trying for the 25-point bonus, and the items increase on up to nine items when you play for 250 points. The object is to place these shapes in a predetermined order selected by the computer.

You select two numbers for each move, and the numbers will exchange positions. At the bottom of the window, a count of items in their proper places and a total of the number of tries remaining are given. The number of attempts allowed is also displayed. If, after you run out of tries, you still haven't solved the puzzle, the solution is displayed, and a message stating that you've lost the bonus game is given.

There is an easier way than winning a bonus game to earn some big points. As the name suggests, the object of the game is to link your markers. Four markers in a row earn you 250 points. Note that a chain counts only when the final ball is placed on the end of three others. Placing a marker in the second or third positions to link four is not recognized as being a chain, and you'll receive no points for it.

Strategy plays an important role in this game. As you place your markers on the board, you may find yourself in a position where placing one marker will link up a chain of four in more than one direction. When this happens, you earn 250 points for each chain you link up. Thus, two links in any direction earn 500 points, three earn 750 points, and so on. With this many points at stake, it's no wonder that blocking your opponents' chains is of vital importance.

A chain is good only when it consists of four similarly colored markers in a row in any direction. Anything else, a Pit Trap, an empty block, an opponent's marker, breaks the chain.

When four in a row are recognized, the ends are capped, and your score increases. The caps prevent you from being able to build additional links upon the ends of existing ones. Any further linking to this chain would have to be made across the second or third markers in a different direction.

The game continues until every block has been revealed. At this point, whoever has the highest score is the winner. If more than one player has the same score, the first among them is the victor automatically.

Tips For Winning
Don't try to place all of your markers in a row immediately. If you do, your opponents will easily recognize the pattern. Instead, try to get as many as possible in close proximity to one another. That way, as more and more markers fill the screen, your opponents might be less apt to notice the grouping.

When selecting a bonus game, don't try for anything over 100 points unless you have a strong lead. The greater the number of items to be sorted, the more permutations are possible, thus making it more difficult to win. To almost guarantee a win, select the game for 25 points. With six attempts to sort three items, there's virtually no way you can lose.

John Cameron, the author of Mindboggle and Runestones, lives in New Castle, New Brunswick, Canada.

Gazette, January 1994

AUTOGRAPH v1.0

By Fuzzy Fox ,

Welcome to AutoGraph, one of the most versatile graphics convert/display programs to come along! This program allows you to view and manage files created from many different types of art programs, and also allows you to convert between any of the formats supported. The program has been designed to be simple and easy to use, and hopefully, it is.

When you start the program, you'll be asked to insert a disk containing graphics files. The F1/F2 keys let you change the device number that AutoGraph will search for files, and the F3/F4 keys will let you change the drive number. For 1541/71/81 disk drives, the drive number has no meaning, and MUST be left at 0. Dual drives such as the 4040 or 8050, or hard drives like the Lt. Kernal will be able use the drive number to select disks or partitions. 1581 users will use a different method to select partitions, and should simply insert the disk and press RETURN.

AutoGraph will then read the directory and look for graphic files that are in a format it recognizes. Not all graphic formats are supported, but most of the major commercial art program formats are supported. If AutoGraph can't find any files that it recognizes on the disk, it will ask for another disk before allowing you to proceed.

Once AutoGraph reads a directory containing graphic files, it will display the main screen. Most of the screen contains a window showing what files are on the disk. AutoGraph displays the name of each file, without the prefix or suffix associated with that file type, and places a two-letter code after each file, which shows its type. The types recognized by AutoGraph and the two-letter codes shown for each type are explained below:

KO	Koala Paint
RB	RUN Paint Bitmap
DD	Doodle!
RM	RUN Paint Med-Res
GG	Compressed Koala
RH	RUN Paint Hi-Res
JJ	Compressed Doodle
BP	Blazing Paddles
OC	OCP Art Studio
SB	SID/PIC Bitmap
AD	Advanced OCP Art
SM	SID/PIC Multicolor
64	Artist 64
SH	SID/PIC Hi-Res

You will see a "cursor" on one of the filenames, which you can move around with the CRSR keys. The cursor shows which file will be acted upon by a command that you give. The HOME and CLR (Shift HOME) keys move the cursor to the first or last file on the screen, respectively. Pressing HOME or CLR twice will take you to the beginning or the end of the directory.

All of the commands that you will use can work on one file or several files at a time. To specify a single file, simply move the cursor to that file and give the command. To specify several files to be operated on in a batch, use file selection.

The SPACE bar is used to select the file under the cursor. When a file is selected, it appears in reverse video on the screen. You can move the cursor around, selecting as many files as you like, and when you give a command, all files that are selected will be operated on by the command. Those of you familiar with Fast Hack'em and other copy programs will recognized this approach to file processing.

A quick method of selecting files is to use the toggle command, "T", which reverses the selection status of all filenames. Pressing "T" with no files selected causes all files to be selected. Also, the unselect command, "U", quickly unselects all selected files, so that you can select a different batch to process.

Once you know what files you want to use in a command, simply press the key that performs that command, and the command will start executing. Note that in almost every case, you can press the STOP key to cancel a command and return to the main screen, so if you make a mistake, just press STOP. Pressing STOP at the main screen will exit the program and put you back at the Basic READY prompt.

COMMAND SUMMARY
Pressing the RETURN key starts the View command, which displays files on the screen. You can view one file or select many files and view them one after the other. While you are viewing a file, you have several options that you can perform, that are not shown on the screen. Pressing the F1 and F3 keys changes the border color of the picture. If you press "I" while viewing a picture, the bitmap will be inverted. This function has been found to be helpful when viewing files created by scanners or digitizers. Any other key (except "S") stops viewing the file. If there are more files to be viewed, the next one will be loaded and displayed. You can press STOP to cancel a multi-file View.

Pressing "S" while viewing a picture starts a Save operation. The picture on the screen can be saved to disk, either in the same format or in any other format of the same display type (multicolor or hi-res). A menu will appear showing which types you can convert to. If you select one and press RETURN, the file will be saved in the format you selected. Details on source and destination drives will be explained later.

If you press the "S" key from the main command screen, you will start a Slideshow, which is a way of displaying pictures, one after the other, with a timed delay in between. Unlike the View command, which always returns to the main screen when loading the next file, the Slideshow command loads the next picture in the sequence while it is still displaying the previous picture. Only when the picture finishes loading, and then only after the inter-picture delay time runs out, then the next picture is displayed.

When you start a slideshow, you are asked first for a time delay between pictures. You can enter any value from 1 to 255 seconds, although it usually takes longer than 1 second to load a picture. This time delay is measured from the start of displaying one picture to the start of the next picture, including load time. For instance, if you select a 30 second delay, and the loading of the next picture takes 10 seconds, the program will wait 20 more seconds before displaying the picture that was loaded.
You are also asked if you wish to perform a continuous slideshow. This means that after displaying the last picture in a sequence, the first will be re-displayed, and so on. You must use STOP to exit a continuous slideshow.

Note that only the files you select will be included in a slideshow. If you select no files, then this command works exactly like View.

There may be some cases in which you wish to pause a slideshow for a moment, then let it resume. If you press down the SHIFT LOCK key during a slideshow, the next picture will not be displayed until you release the SHIFT LOCK key.

The "DEL" key allows you to delete one or more graphic files from the disk. You are always asked for confirmation before any files are deleted. If you are deleting multiple files, the STOP key can be used to abort the operation.

The "R" key allows you to rename graphic files, with full editing. The old filename and new filename are displayed, and you can edit the new filename to your heart's content, using the CRSR left/right, HOME, CLR, and INST/DEL keys. When you have edited the name, press RETURN to rename the file, or STOP if you do not wish to rename it. Multiple files can be selected and renamed. Note that if the filename you specify is illegal, the rename will not be performed.

Pressing "C" from the main command screen allows you to convert files from one type to another, without having to use the View/Save option described earlier. Note, however, that if you have only one drive, you can only use this command if you save to the same disk as you are loading files from. Make sure there is plenty of space on the disk.

A menu will appear showing you which types you can convert the hi-res and multicolor files that you have selected on the disk. Note that on both menus, an option called "No Conversion" appears. If you select this type, no file conversions will be done, resulting in a Copy operation to the destination disk.

While the conversion is taking place, the files being converted are displayed on the screen while saving, for entertainment value.

Pressing "D" at the main command screen causes the program to ask for a new disk of graphic files. When you press RETURN, the new disk will be read, and the directory displayed on the screen. You should use this command whenever you change source disks.

Either at the command screen, or after pressing "D", you can use the function keys to change the Source and Destination device numbers displayed on the right side of the screen. You can select any device/drive combination, even an illegal one, so be careful that the device AND drive that you select are legal. If you select an illegal source or destination device or drive, the program will probably appear to be ignoring your commands when you reference the illegal drive.

The Source drive is the drive from which graphic files will be loaded, and from which the directory will be read when you changed disks. The Destination drive is the drive to which any files created will be saved, either from the Save option in the View command, or when using the Convert/Copy command. Also, the number of blocks free (displayed near the lower right corner) is displayed for the Destination drive, not the Source drive.

That covers all the commands, and through creative use of this program, you can do some amazing things. For example, using RAMDOS, you can create an impressive REU slideshow by copying files to the RAM disk, then performing a slideshow from that device. The view, scratch, rename, and copy operations make this program very helpful in managing a large number of graphic files across multiple disks. The conversion commands are extremely helpful when you wish to use the features of several different art programs to complete a picture. Also, many other utilities require files to be in a specific format (usually Koala or Doodle) before they can use the files, and this program can convert most pictures to these types. Lastly, this program is well-suited to converting files to and from the compressed JJ and GG formats found so often on BBS's and online services.

NOTE TO 1581 USERS
AutoGraph attempts to support 1581 partitions, in a limited fashion. When you insert a new disk and read the root directory, partition files appear on the main screen just like normal graphic files, but the filetype is shown as an arrow, indicating that the file is a subdirectory. If you move the cursor to the directory and press RETURN, the subdirectory will be opened and read. If it contains graphic files, you will see them. If not, a message will be displayed, and you will be returned to the root directory. At any time, you can also return to the root directory by pressing the "/" key.
You are not allowed to select (via SPACE bar or "T" command) a partition file, and no file operations are allowed on such files, except for Rename, which allows you to change the name of a partition

file. Note that all picture loads and saves will be performed in the current open partition. There is currently no way to copy files from one partition to another on the same disk.

DON'T READ THIS!
This program is distributed as FreeWare, meaning if you like the program, that's great, and if you don't like it, tell me why, and maybe I can help you with it or fix something that's wrong with it.

If you wish to contact me by E-Mail, I can be reached as "Fuzzy Fox" on Quantum-Link. Or, you can contact my by P-Mail (Paper Mail!) at the following address:

Fuzzy Fox
940 Ramblewood Dr.
Lewisville, TX 75067

Also, if you have any drawings or pictures of furry critters that you don't want any more, send them to me! I love cute critter drawings! Also, please don't send any checks or money orders made out to "Fuzzy Fox," as my bank will not accept that name, for some reason.

Please enjoy the use of this program, and look for upgrades in the future, as I intend to support this program.

Special thanks go to all the people who helped me put this program together and make it work (even without a 1581 to test with!). Thanks, and have fun!

BEAM ME DOWN, SCOTTY

By Steve Vander Ark

The bridge may get all the glamour, but engineering is where the real work is accomplished.

I always kind of feel sorry for Scotty on Star Trek. Whenever the starship Enterprise gets into a scrape with anything from Klingons or ion storms to gigantic space monsters, he always sticks to his post in engineering, informing the bridge that it'll take hours to fix whatever Captain Kirk has just broken. Let's face it; when you're stuck in engineering, you're pretty much left out of the good stuff. That's one of the reasons that there is an engineering station on the bridge in the second season of Star Trek: The Next Generation. That lets Geordi get in on all the action. After all, the bridge is where all the big decisions, the amazing discoveries, and the nifty dialogue takes place.

Of course, a starship doesn't run on dramatic discoveries and witty banter or by opening any number of hailing frequencies. When Sulu or Worf fires the phasers, there must be people somewhere making all that equipment work. And no matter what's happening on the bridge, someone still has to keep the environmental and power systems chugging along. OK, OK, so the Enterprise is just pretend, you say. Get a grip, Vander Ark.

It's sort of the same way with GEOS. Up on the bridge, in a premier application such as geoPublish, for example, is where all the exciting stuff happens. That's where you create all those great documents and spit them out of your printer to impress your family and friends. That's where the action is, and, frankly, that's what I usually talk about in this column. But if you never trot down to engineering and make an effort to keep those engines running properly, you're going to end up being blown out of space by a Borg somewhere along the line. I'm talking about things like keeping your disks organized and validated, jobs that no one really enjoys but which are essential.

These jobs are particularly important to GEOS users because GEOS is a disk-intensive system. Besides the required boot disk (or boot disks, if you use Gateway as well as the standard deskTop), you'll need a variety of work disks for different tasks or applications--not to mention disks to store everything from documents to fonts to clip art.

The more you use GEOS, the more disks you have. If you download regularly from GEnie or QuantumLink, you'll have even more disks filled with files and programs. I have so many disks that I've graduated from slick plastic disk boxes to a series of shoe boxes that happen to be just the right width for floppies. To be honest, I have no idea what's on some of those disks, and that's a big part of the problem I'm talking about.

This problem really came to a head when Tom Netsel, the editor of Gazette, asked me to put together a disk of the best GEOS shareware that I've mentioned in my column over the past two years. Then, thunderstorm season came to my home here in Grand Rapids, Michigan, and my RAMLink lost all its files every other day for a week or so. All this meant that I needed to find things, and I soon realized that I had no idea where most of those things were. I needed to take a trip to engineering and get my ship running smoothly again.

First of all, I needed to put all my main applications and the files I usually use with them on a few disks. Now, I have three 1571 disks with the applications, fonts, and desk accessories that I use frequently. This means, for example, that I didn't include Photo Manager because I use Scrap Grab instead.

Now, when one of those pesky power outages zaps my system, I can quickly reload my RAMLink and be back in business in minutes. I don't have to track down my master disks to find a copy of geoWrite or geoPublish, either. That means they can stay safely stuffed in that drawer full of disks.

OK, that drawer full of disks was my next target. There's no way I'll ever have time to copy all those files into some sort of logical system of disks. That would be ideal, but I know I'll never do it. Instead, I found a good alternative system that lets me find things pretty quickly but doesn't require any file copying.

First of all, I made sure that every disk was labeled with the same name as the one that appears as the disk name in the directory. Then I ran geoLogger, a snazzy shareware program that makes disk cataloging almost fun. GeoLogger creates "catalogs," text files listing all the programs on a number of disks along with information about each file and which disk it's on. These catalogs can be printed and stored as a ready guide to what's on which disk. Best of all, the catalogs can be loaded back into the program and searched just like a database.

You can specify a line of text for the program to use as it searches through the filenames or the disk names. This way I can keep a disk of disk catalogs, and when I need a particular file, I can run geoLogger, load a catalog, and search for the file. The catalog then tells me which of my hundreds of disks hold copies of that file.

Notice that I said I would have catalogs--more than one. That's because geoLogger's one drawback is its limit on the number of files per catalog. I always try to turn a minus into a plus, so I create catalogs of easily identifiable groups of disks.

For example, I now have a catalog of all my miscellaneous downloads. These are the ones I sock away without rhyme or reason. The same holds for my regular disks, those ten or so that I keep in the disk holder on my desk and I actually use most of the time. See how organized I'm getting? And it all happens with a few easy keypresses and disk swaps! That means that I now have ten catalogs on a directory disk, but, hey, any way you look at it, it's easier to quickly glance through my catalog printouts or run a few searches than to open disk after disk looking for a copy of some desk accessory or printer driver.

GeoLogger is shareware; the fee is a mere $5.00, payable to the author, Mike Craig, 12001 Lockton Drive, Richmond, Virginia 23233. It runs in 128, 80-column mode only. You can find it on Q-Link. There are other excellent directory printer programs if all you want is a printout of the directories with no searching or cataloging. A fine example is geoList, written by John Howard. This shareware program is available on Q-Link (filename: geolist v2.2, uploaded by REBASAK) and GEnie (file number 6672).

GeoList will dump the directory of a disk either to a printer or to a geoWrite text file. The shareware fee for geoList is $5.00, payable to John Howard, 4433 Clemsford Drive, Virginia Beach, Virginia 23456.

OK, so my disks and files are organized. There's still plenty of work left to do down in engineering. For one thing, we Gateway users desperately need a utility to rearrange our directories. It's nigh to impossible to get things set up just the way we like them within Gateway itself. Then there's the matter of disk doctoring, validating, and fixing directories. Next month I'll see what I can turn up to take care of those chores.

Gazette, January 1994

End of file? Then back to BASIC, where the input and output files will be closed.

```
        RTS
```

The BASIC Program.
The BASIC program pokes the machine language into place. It then opens the input file as logical channel 1. Logical channel 2, output, is set to the screen, device 3.

When the program runs, supply the name of a text file; other files may put strange characters on the screen. If you don't have a text file on your disk, you can create one easily. Load any BASIC program (this one is OK), and type the following.

```
OPEN 8,8,8,"0:MYTEXT,S,W" :CMD 8: LIST
```

When the cursor returns, type the following.

```
PRINT#8: CLOSE 8
```

Now you have a sequential text file called MYTEXT on disk.

Here's the program.

```
100 DATA 162,1,32,198,255,160,0
110 DATA 140,0,34,32,228,255,172,0,34
120 DATA 153,0,33,166,144,208,3,200
130 DATA 208,237,142,1,34,32,204,255
140 DATA 162,2,32,201,255,160,255
150 DATA 200,185,0,33,32,210,255
160 DATA 204,0,34,208,244,32,204,255
170 DATA 174,1,34,240,197,96
200 FOR J=8192 TO 8251
210 READ X:T=T+X
220 POKE J,X
230 NEXT J
240 IF T<>7628 THEN STOP
400 PRINT "BUFFERING DEMO - JIM BUTTERFIELD"
410 PRINT
420 INPUT "NAME OF DISK (TEXT) FILE";F$
430 PRINT
440 PRINT "(OUTPUT TO SCREEN)"
450 OPEN 1,8,2,F$
460 OPEN 2,3
470 SYS 8192
480 CLOSE 2
490 CLOSE 1
```

(Note: A runnable version of this program is on the flip side of this disk.)

Beginner BASIC~~y 1994~~
By Larry Cotton

REDEFINED CHARACTERS

Happy New Year! Last month I challenged you to write (or just submit) one of three kinds of programs--onewhich plays a Sets-like game, one which sounds Westminster chimes, or one which rounds numbers. When I attempted to take on the first challenge myself, I used a technique that I believe has escaped coverage in the first seven years of this column: programming special characters.

Special characters take the place of ordinary characters which normally appear on your TV or monitor screen as you type. In my Sets program, I've replaced the Shift-A through Shift-I characters with circles, squares, and triangles. I'll show you part of the program which illustrates how this is done.

```
0 REM
10 PRINT"[CLR][BLK]":V=53248: POKEV+32,15: POKEV+33,12
100 POKE214,9:PRINT:PRINT" WHILE SETTING UP, PLEASE ADJUST COLORS
110 PRINT"[DOWN] AS FOLLOWS:[RED] RED[YEL] YELLOW
 BLUE[BLU] THEN
120 PRINT"Qp PRESS ANY KEY TO CONTINUE.
130 GETA$:IFA$=""THEN130
140 PRINT"[CLR]"
150 POKE214,9:PRINT:PRINT"    PLEASE WAIT FOR THESE CHARACTERS
160 POKE211,11:
PRINT"[DOWN][RED][SHIFT-A][SHIFT-B][SHIFT-C][SHIFT-D][SHIFT-E][SHIFT-F
[SHIFT-G][SHIFT-H][SHIFT-I]
170 PRINT"[DOWN][BLK]  TO CHANGE TO THOSE USED IN THE SETS!
180 POKE56334,0:POKE1,51
190 L=12288: FORI=8TO472: POKEI+L,PEEK(I+V): NEXT
200 FORI=12808TO12879: READD: POKEI,D: NEXT
210 DATA 255,195,195,195, 195,195,195,
255,255,129,129,255,129,129,129,255,255,255
220 DATA 255,255,255,255,255,255, 60,
102,195,195,195,195,102,60,60,102,195, 255
230 DATA 195,195,102,60,60,126,255,
255,255,255,126,60,24,24,36,36,66,66, 129,255
240 DATA 24,24,36,36,126,66,129,255, 24,24, 60,60,126,126,255,255
250 POKE1,55:POKE56334,1:POKE53272,28
```

(Note: The runable version of this program can be found on the flip side of this disk.)

Line 0 is a little trick that I discovered sometime back while writing BASIC programs which need repetitive testing. (They all do, don't they?) It lets you run a program simply by typing RUN anywhere on the left side of your TV or monitor screen.

I/O BUFFERS

By Jim Butterfield

Machine language programs can easily handle devices such as a disk or a printer. Simple programs handle a single character at a time by connecting to the device, sending or receiving a character, and then disconnecting. When you want to handle another character, you have to do it all again.

That's simple, but not efficient. Connect and disconnect signals take time to be transmitted over the serial bus. Sometimes, it's better to buffer: grab (or send) a bunch of characters at a time.

The Buffer

A stream of characters can be stored in a buffer, a portion of memory set aside to hold them. There needs to be bookkeeping to say how much of the buffer is occupied at any time. Of course, the buffer has limited size, so the program needs to take action
when the buffer is full.

A buffer that isn't larger than 256 bytes is quite easy to program. Simple indexing with the X or Y registers will reach any part of the buffer. The index register serves as a pointer within the buffer.

One of my utility programs, Copy-All, uses a 254-byte buffer. That exactly matches the amount of data stored in one disk block. The user can watch the blocks being "counted" as they are copied.

Simple Copy Program

The following program runs on the 64, as well as the 128 and VIC-20. It is easily modified to run on earlier models (PET/CBM) by changing the address of the ST variable from $90 to $96.

We'll use a 256-byte buffer to copy a file of any size from one place to another. Our BASIC program sets up copying from disk file to screen; you can easily modify this to suit your needs. The program has been poked into place by BASIC. Let's look at the code.

BASIC has opened logical file 1 as the input channel. We must first connect to this channel by a call to CHKIN at $FFC6. The code starts at hexadecimal address 2000.

```
        LDX #$01       (logical channel 1)
        JSR $FFC6      (CHKIN, connect input)
        LDY #$00       (set buffer pointer to zero)
```

Our input loop starts at address $2007. We briefly save our Y pointer, grab a character from the input stream, and then restore Y.

```
(inloop)  STY $2200     (save pointer)
          JSR $FFE4     (GETIN, get character)
          LDY $2200     (restore pointer)
```

We put the character into the buffer, which is located in the area from $2100 to $21FF. Then we test ST, our status word, to see if we're at the end of the file. If so, we hop out of the input loop. If not, we bump the Y pointer and loop back.

```
          STA $2100,Y
          LDX $90
          BNE $201A
          INY
          BNE $2007
```

At $201A, we have finished the input cycle. Maybe the buffer is full, or maybe we're at the end of the file. In either case, we save the ST status value and disconnect from the input by calling CLRCHN, $FFCC.

```
         STX $2201
         JSR $FFCC
```

Time for some output. Connect to logical channel 2 with a call to CHKOUT, $FFC9. We'll set the output buffer pointer to a value of -1; it will be bumped up to 0 right away. Keep in mind that our input buffer pointer is stored at address $2200.

```
         LDX #$02
         JSR $FFC9
         LDY #$FF
```

Address $2027 starts our output loop. Bump the output buffer pointer, grab a character, and send the following.

```
(outloop) INY
         LDA $2100,Y
         JSR $FFD2
```

If the output loop pointer doesn't yet match the input pointer, keep going around the loop.

```
         CPY $2200
         BNE $2027
```

At $2033, we're out of the loop and can disconnect from the input channel. Check the previously stored value of ST again to see if we're at the end of the input file. If not, back we go and do it all again.

```
        JSR $FFCC
        LDX $2201
        BEQ $2000
```

Sure, you could add a colon after RUN and omit line 0, but I like this technique better. Try it. List any short program that you know works. Add line 0. Then place your cursor on any line on the left side of your screen and type RUN. That's all there is to it. If you already have a line 0, change its number to 1 first.

But I digress. Line 10 clears the screen and then prepares to print black on a two-tone gray background. Lines 20-90 of my program aren't necessary to define the special characters, so I've left them out. Lines 100-130 print a message and wait for any key to be pressed.

Lines 140-170 then print another message with the old/new characters midscreen. While displaying this message, subsequent lines cause the computer to shut down its keyboard and redefine part of its character set.

Normally, the 64 gets its characters from part of its read-only memory (ROM). However, you can force the computer to look at random-access memory (RAM) instead, which you can program yourself! Among other things, the process involves copying at least some of ROM's character set to RAM so words, numbers, and most punctuation appear on the screen normally.

To accomplish this, you must make judicious use of PEEK and POKE to turn off what is called the interrupt timer and switch out input/output--which just happens to make the keyboard temporarily unusable. Line 180 does this. Line 190 copies a minimum amount of character data (only the alphabet, numbers, and some of the punctuation) from ROM to memory locations 12288-12760.

We now read some special data which defines our nine new letter-sized characters. I'll show you how to write that data next month. Line 200 puts the special character data (contained in lines 210-240) in memory registers 12808-12879.

The only thing left to do is write line 250, which turns the interrupt timer back on and switches in the input/output--thus awakening the keyboard. Then we poke a 28 into 53272, which causes the computer to look at RAM beginning at 12288 for the character data.

Run the program, adjust the colors, press any key, and then wait a few seconds for the redefined characters to appear. If you list the program after running it, some characters (such as =) will be random blobs. To see a normal listing, hit Run/Stop-Restore, or type POKE 53272,21 (its default value). We'll finish up next month.

Gazette, January 1994

D'Iversions
Sticky-Tab Computing

By Fred D'Ignazio

I turned on the TV this morning and watched a slick 30-second commercial from Apple Computer praising its new Newton MessagePad computer. According to the ad, this paperback-sized device can recognize handwriting, send faxes over phone lines, and receive wireless pager messages. It functions like an electronic notebook filled with endless scrolling virtual pages on which you can type or draw or scribble with a plastic stylus. The program inside the Newton (dubbed Newton Intelligence) is capable of turning your handwriting into clean typewritten words. It also functions as a calendar for entering appointments and reminders and as an address book.

I had hardly finished digesting all this when I opened that morning's Wall Street Journal and spotted a headline that read, "Compaq, IBM To Unveil New Pen Computers." It seems that everyone is getting into the pen-computing business. Apple, Compaq, and IBM are already doing battle with other competitors, including Tandy, AST Research, Motorola, Sharp, Casio, and AT & T.

Pretty soon the pen-computing business will be as crowded as the personal computer business.

The User-Friendly Sticky Tab

The downside of all this is that the machines are only mediocre at recognizing handwriting, they're enormously expensive (a fully configured Newton costs $1,500), and many of their functions aren't yet available. One observer summed it up by calling these machines "pocket calculators on steroids."

And, truly, who in their right mind would want to pay over a thousand dollars for a notepad or an address book? Even one with virtual pages?

Maybe the industry should think creatively and radically shift its message-pad mindset into new directions. For starters let's look at sticky tabs (or Post-it Notes) and all of their spin-offs. These little pads cost a couple of bucks, and they come in a variety of colors to suit your mood or personality. They let you scribble a message in just a second or two. And they stick like superglue--yet they're cleanly and easily removed.

In computer terms they are user friendly; the interface requires almost no sticky-pad literacy. They can be reconfigured into any sort of virtual sticky tab simply by chaining multiple sticky tabs together to form a long sticky-tab tail. They are portable and inexpensive and can be lifted and reattached to new locations and then left behind because of their negligible cost.

On the downside, I don't know of anyone who has ever sent a fax using sticky tabs. Also, sticky tabs don't convert your handwriting into neat printed text. And users are required to purchase their own stylus (pencil or pen) to operate a sticky tab. Perhaps the sticky-tab package should come with a label that says, "Stylus not included."

On the positive side, I probably wouldn't want to spend the money to fax my messy, handwritten sticky-tab drawings and notes, and the stylus I use with my sticky tabs is a general-purpose device which works equally well signing checks, filling out job applications, and drawing cartoons on restaurant napkins to amuse my four-year-old daughter. On the other hand, since the plastic stylus supplied with most electronic message pads lacks an ink or lead tip, it would be useless on most surfaces other than a sandy beach or the muddy corner of my flower garden. However, it might come in handy as an ear scratcher, coffee stirrer, or as a backup oil dipstick for my car.

As you can see, there are pluses and minuses on all sides.

Sticky-Tab Computers

Now what would happen if computer designers tried to make their message-pad computers more like sticky tabs?

Sticky-tab computers would have to be very cheap, probably only a few pennies apiece. They would be ultrasimple to use--just like a real sticky tab. You could buy a whole pad of sticky-tab computers at a local drugstore for a buck or two. And each computer would come with magnets or magic glue on its underbelly so you could slap it on walls, refrigerators, doors, people's chairs, or on another computer!

Sticky-tab computers and real sticky tabs would soon be in hot competition for space on local drugstore shelves. Their manufacturers would engage in a cutthroat attempt to mark down or discount their company's product in order to undercut their rivals' products. And in people's offices and homes sticky-tab computers would become just as ubiquitous as real sticky tabs. You would soon find these computers everywhere: on your desk, chair, walls, sink, closet, toaster oven, or toilet--anyplace where people think they might catch you with an urgent message.

Sticky Tabs That Talk

Soon sticky-tab computer manufacturers (such as IBM, Apple, and AT & T) would have to come up with features that customers really want which would differentiate their products from normal paper sticky tabs. They had already tried Sticky Tabs That Fax but that hadn't worked. They had tried Sticky-Tab Beepers but people complained that the hefty sticky-tab computers pulled down their pants or skirts when attached to the waist and looked cumbersome and bulky under a sports jacket or sweater.

Perhaps they could invent Sticky Tabs That Talk. Talking sticky tabs would come equipped with microthin DSPs (digital signal processor chips) which do on-board speech synthesis; speech recognition; and motion, infrared, and thermal sensing. A single sticky-tab computer sheet would not be much thicker than a piece of paper, yet it would be an intelligent sensing flake of matter, ready to serve its user as a message transponder.

The way to use the new computers would be simple yet powerful. When you had a message to leave someone, you would yank a computer sticky tab off the pad of computers you carried in your pocket or purse. You would hold the computer up to your mouth and talk into the tiny condenser microphone in the center of the tab. Next, you would pinch the corner of the tab to hear an instant oral replay of your message. Then, you would slap the tab onto any surface (yes, the computer is prestuck with magic glue), and you would know that the tab would faithfully deliver your message ASAP.

Imagine This Scenario

A couple of minutes later, your message recipient innocently and unknowingly saunters into the sticky tab's 3-D message space. Thermal, motion, and infrared sensors lock onto his or her moving body. First, the sticky tab ensures that the message recipient is within hearing range.

"Phweet! Phweet!" it might whistle.

"Uh!?" exclaims the designated man, a perplexed look on his face.

The sticky tab performs a perfunctory I.D. "Are you Mary's secretary?" it might ask.

"Why, yes," the man says, spinning around, trying to figure out who is speaking. "Who are you? And where are you?"

The sticky tab would ignore these irrelevant questions and efficiently deliver its message. "Mary is in an important meeting until 3:00 p.m.," it would say (in a 16-bit 44-kHz high-fidelity digital playback of Mary's own voice), "and she wants to make sure you send the flowers to her husband for his birthday! Try to get them delivered to the house before lunch. That is all. Message delivered!"

"Oh," says the man, finally coming upon the sticky tab which has been stuck unobtrusively to the side of his old Newton MessagePad. He tosses the sticky tab into the Sticky-Tab Recycle Bin where its message is immediately erased. The sticky tab is now ready to be attached to a new pad.

"Now where did I put that Alexander Graham Bell?" he says, shuffling through dozens of programmable digital assistants on his desktop. "I think it contains the phone number of the florist."

Programmer's Page

MATHEMATICAL LIMITS

By David Pankhurst

Since this is my on-disk first column, I thought I'd introduce myself. I was born many years ago on a small turnip farm in Buenos Aires, Argentina. No, scratch that; my life hasn't been that exciting. I'm 30; married; live in Montreal, Canada; and adore my 64.

I've been into computers since the early 1980s, starting out with a Timex Sinclair. I learned Z80 machine language by hand assembling code on it, trying to shoehorn something useful into 2K of memory. Compare that to today's programs, when 640K is considered too small on an IBM!

My next computer was the Radio Shack MC-10. From there, I settled on my first real computer, the 64, which I've used ever since.

Although I've programmed in a variety of languages, I prefer machine language for its speed and compactness. Aside from the occasional submission to this magazine, my last program of note was Calc II, a spreadsheet for the 64 written in machine language.

Although I've played on the IBMs, I don't believe that power programming is as necessary on them as it is on our 8 bitters. We need every trick and tip we can get to wring out better performance. Bytes and cycles really count on a 64K, 1-MHz machine.

"Programmer's Page" has always been dedicated to the Commodore programmer, that intrepid soul, and I plan to continue the tradition of sharing programming tips. I do need your help, however. I hope you'll continue to share your tips with us. In addition to having your tip and name published here for all the world to see, there's the monetary reward. We pay $25<n->$50 for each tip published. So send in anything you find useful or interesting. Send your tips to Programmer's Page, COMPUTE, 324 West Wendover Avenue, Suite 200, Greensboro, North Carolina 27408.

I only ask that you explain your code briefly--what it does and what it requires. With this new disk format, we now have room for larger pieces of code. If your tip is more than a few lines long, please send it on disk (which I unfortunately can't return). And don't hesitate to drop a postcard or letter telling me what you want to see in this column.

Now let's look at two novel uses for the 64. For your convenience, the programs mentioned are listed in the article so you can examine the code. The programs themselves are ready to run on the flip side of this disk. I've included the code here so you can examine it as we go along.

Calculating Pi

Pi calculation has always been a lengthy and demanding task for a computer, with calculations running over and over thousands of times. The program here shows one of the latest algorithms (from an article titled @@3.1416 and All That'') which converges very rapidly.

```
10 REM FAST PI CALCULATION
15 X0=SQR(2):Y1=SQR(X0):P0=2+X0
20 A=SQR(X0):X=0
25 X1=(A+1/A)/2
30 A=SQR(X1):B=Y1+1
35 Y2=(Y1*A+1/A)/B
40 P1=P0*(X1+1)/B
45 X=X+1:PRINTX"="P1
50 X0=X1:Y1=Y2:P0=P1:GOTO25
```

When you run the program, you'll see pi displayed in only two loops, correct to seven digits. On some computers, double precision numbers are available, giving you greater accuracy. The number of correct decimal places effectively doubles with each loop; 12 loops would get pi correct to about 5000 decimal places, for instance.

The Catch

So if it's this simple to program, why doesn't everybody write pi programs? The problem is that even simple operations take up a lot of time when the numbers are big. Nobody wants pi to only 600 decimal places; they want it to millions. Multiplying and dividing million-digit numbers takes a long time.

To see why this is so, consider multiplying two numbers by hand on paper. If the two numbers are four digits long, by hand we have to make 16 one-digit multiplications. If the two numbers are five digits, we end up with 5 x 5 or 25 multiplications, almost twice as many calculations. With pi calculations, the variables used must all be the same length, making the speed proportional to the number of digits squared.

If we can do 7 digits of accuracy in a set time, we can predict that a double precision number, or 14 digits, would take not twice but four times as long to compute. And what if we wanted to do a number 1,000,000 digits long? That's roughly 140,000 times the precision of our BASIC variable and would take about 140,000 squared, or 20 billion times as long! Now you can understand why we always hear about pi being calculated on Cray supercomputers, and not 64s.

Not that it isn't possible; a program for doing machine language math on large numbers would let you calculate pi to about 1800 places and tie up your 64 for the better part of a week.

Large Primes

Now that we've seen what we can't easily do on a computer, let's look at what we can. Ever wondered what all the prime numbers less than one million are? Well, now you can find out with your very own sieve of Eratosthenes, a device for finding prime numbers named for the Greek geographer and astronomer. It calculates all the odd numbers above 3 and then deletes every third number after 3, every fifth number after 5, every seventh number after 7, and so on.

The highest number to search is set by M in line 40 and can range up to one million. It can take a long time until the results start to appear, but as more and more primes are found, the numbers print out faster. Let M equal 999, and the program runs in about 13 seconds. Let M equal 1,000,000, and it takes about six hours.

```
10 X=896:FORJ=1TO9:READX$
20 FORI=1TOLEN(X$)STEP2: Z=ASC(MID$ (X$,I,1))
*16+ASC(MID$(X$,I+1,1))-1105: POKEX,Z
25 Y=Y+Z:X=X+1:NEXT:NEXT:IFY<>11208THEN STOP
30 POKE55,0:POKE56,11:CLR:SYS899
40 X=0:Z=0:I=0:M=999
45 POKE785,128:POKE786,3
50 TI$="000000":PRINT2;
55 FORI=3TOMSTEP2:IFUSR(I)=0THENPRINTI;
60 FORX=ITOMSTEPI*2:Z=USR(X):NEXT:NEXT
65 PRINT:PRINT"<DOWN>FROM 1 TO"M"IN"INT(TI*5/3)/100"SECONDS
70 DATA BIJACOBIJAAKABACAEAIBACAEAIA
75 DATA AAAAHIKJDAIFABKJALIFAFIFADKJ
80 DATA AAIFAEIFACKIJBAEMINAPLOGAFNA
85 DATA PHKJDHIFABFIGACAJLLMHIKJDAIF
90 DATA ABKFGFCJAPEKKKEGGDGGGEGGGFEG
95 DATA GDGGGEGGGFEGGDGGGEGGGFEGGDGG
100 DATA GEGGGFKFACBIGFGFIFAEKFGEGFAD
105 DATA IFAFKAAALBAEDNIGADEILBAEBNIG
110 DATA ADJBAEGIKIKJDHIFABFIEMKCLDAA
```

To run the program again after changing the value of M, enter RUN 30.

Remember, both of these programs can be found on this disk. Load and run PI.PD and PRIME.PD.

If you have a programming hint or tip that you'd like to share with our readers, we'd like to publish it. Send your tips on disk along with documentation to Programmer's Page, COMPUTE's Gazette, 324 West Wendover Avenue, Suite 200, Greensboro, North Carolina 27408. We pay $25-$50 for each tip published.

Gazette, January 1994

PD Picks
Graphics Converters

By Steve Vander Ark

This is frustrating. It just goes to show how much I rely on technology. My 1670 modem is on the fritz, and now I can't log on to QuantumLink. It's driving me nuts!

For one thing, it makes my job harder. I was writing my GEOS column for this month, and I needed the filename of one of the programs I mentioned. Normally, I just log on to Q-Link, do a quick search for the filename, and I'm set. Without the modem, I couldn't get on Q-Link. I had to upload that column without the Q-Link filename! That's frustrating.

But that isn't the worst of it. The really irritating part is that I miss the chitchat in the GEOS support area, all the messages and questions people leave there, and (worst of all) the late night bingo games in Rabbit Jack's Casino. Yep, the bingo games. OK, so I admit it. I'm addicted to Q-Link.

Of course, as you can imagine, I use Q-Link quite a bit as a reference for this column, too. And now I'm stuck without access to all those great files in the libraries. I do have a number of files downloaded that I haven't written about yet, and maybe I'll find something there. I do have access to GEnie, which always has new files, as well as a few year's worth of uploads to draw on. So somehow I'll manage to find another Commodore treasure or two for you. But I'd better get another 1670 and fast, or I might go into withdrawal.

Here's what I recommend this month: a couple of great files for graphics fans.

Autograph by Fuzzy Fox
Blocks: 39.
Documentation file blocks: 60.
Q-Link filename: autograph.sda. Uploaded by Fuzzy Fox.
GEnie file number: 8594.

Autograph is a program I've had lying around for years. I even wrote it up as one of the best Commodore public domain/shareware programs of all time in an article for another magazine. It's a utility that every graphics enthusiast needs.

In case you're still bewildered by Commodore graphic formats with all the letter codes and whatnot, here's a brief overview. There are two main varieties of Commodore graphics: high resolution and multicolor. Sometimes the latter is called medium resolution. Different graphics programs generally use one or the other of these modes. Doodle and GEOS, for example, create graphics in high-resolution mode. The program Koala Painter works in multicolor mode. The images that these programs create are stored in their respective formats.

Besides that, many graphics programs have their own particular way of tagging filenames on disk. Programs often use a particular letter combination at the front of the filename so they can recognize their own files. On top of that, some graphics files can be compressed to save disk space, and they have their own codes so they can be identified.

If you're confused, don't worry. With Autograph, you can just relax and let that outstanding program figure out how to handle all your graphics files. Autograph will recognize and display both standard and compressed files in just about every commonly used format. Not only that, it will easily convert a file from one format of the same mode to another, all with a few keystrokes.

Let me give you an example of just how useful this is for the graphics fan. Suppose you spend an hour or so on Q-Link some evening downloading eight or nine graphics files for your collection. They will almost certainly be in one of two or three standard formats. So in order to view your new pictures, you would have to load each of these two or three programs into a separate viewer. Instead, fire up Autograph and hit Return. You'll get a directory of every graphics file on your disk, no matter what its format. Then, cursor down to the one you want to see and hit Return. Voila!

There's another way this program can be useful. Suppose you want to store all your graphics in compressed Doodle for high resolution or compressed Koala for multicolor. No problem: With a few more single-key commands you can convert them all to the format of your choice. How's that for being slick? If you're one of those people who can't have enough beautiful Commodore graphics in your collection and you want to view your collection with ease, Autograph is for you.

You'll find Autograph and its documentation on the flip side of this disk

Gifvert
Blocks: 25.
Q-Link filename: gifvert.sda. Uploaded by RandyW18.
GEnie file number 8169.

Let's face it: The 64 isn't state of the art anymore when it comes to computer graphics. You only have to take a look at a 256-color Super VGA image or perhaps some full-motion video running under HyperCard on a Mac to realize that fact. But even so, Commodore users aren't nearly so far out of the running as they might think. Most IBM programs use only 16 colors just like the good old Commodore, albeit with higher screen resolution. Comparing art done with one machine to art done with another is like comparing a watercolor to a photo collage. Sure, you can point out that the photos are clearer images, but the quality of the artwork is much more a function of the way the elements are put together. In other words, talent is compared to technology.

Here's another thing to consider. Nowadays you can buy programs that'll help you create incredible images with nearly photographic realism. These images are dazzling to look at and are a stunning example of what can be accomplished with computer graphics. Bear in mind, however, that just about any casual computer user can create an image; the program does the lion's share of the work. If, on the other hand, a computer artist, using whatever software tools and computer system, creates an outstanding image, that's an example of talent and skill. Commodore artists have shown over the years that they're certainly not short on those two commodities.

So what's my point? Just that Commodore artwork deserves to be shared with the computer world at large, just like any other type. The problem, of course, is one of compatibility. That's where Gifvert comes in.

Gifvert will take a multicolor image and convert it into GIF format. GIF, for those of you who aren't familiar with it, is a format which originated on the CompuServe network. It's designed to provide a standard for storing graphic information which any computer system can interpret and display. Images stored in GIF format can be swapped between computer platforms by way of bulletin board systems and networks. GIF viewer programs for each machine then translate the image into a form that its machine can display. In the case of the Commodore, the image is usually converted into a multicolor, Koala-format image.

Some of the finest Commodore artwork is also created in that format, and Gifvert will convert it into a format that you can share with the world. Gifvert does the job with a minimum of fuss. It lacks a file requester box feature (so you have to remember the filenames), but it makes the conversion quickly. It converts even more quickly if you run it in 64 mode on a 128, where it blanks the screen and utilizes the 2-MHz clock speed. The package even includes a short, self-running documentation file. Both are on the flip side of this disk.

These two programs, Autograph and Gifvert, are only two of a whole library of graphic aids you can find on Q-Link or GEnie. If you need graphics, you'll find a treasure trove of them there; look for files by DocJM for a selection of some of the best. (Many of his pictures have been featured in Gazette Gallery on earlier Gazette Disks.) Then fire up Autograph, sit back, and enjoy the show!

Gazette, January 1994

FEEDBACK

WHEN TO USE CONT?
I've done a little programming, and I'm familiar with most of the BASIC commands. But when does one use CONT? I can't see any real use for it in a program.
MARK SILVERMAN
PARAMUS, NJ

CONT (for Continue) is often used with STOP when you're debugging a program. It can be used to isolate bugs.

Suppose you have a program that runs for a while and then crashes. When you've narrowed the problem down to a few lines of code, put STOP and a colon before a suspected bug and then run the program.

If all goes well, the program will halt when it reaches the STOP command, and you'll get a message onscreen that says BREAK IN XXX. This indicates the line number where the program halted. Type CONT and press Return and watch to see what happens. If your bug doesn't appear almost immediately, move the STOP to another place in the program.

STOP acts like END, but it delivers a message telling you the line number at which the program halted. You can examine the code and check the value of variables. CONT lets you continue the program from the next instruction following STOP.

CONT works only when certain criteria are met: You started the program with RUN; you haven't pressed the Run/Stop-Restore keys; and you haven't edited any lines or pressed Return over any line number, leaving it unchanged.

You'll get a message that says CAN'T CONTINUE if you've changed any code or if the program halted due to an error.

MLX LINE NUMBER
I am interested in Screen Pointer that appeared in the January 1990 issue of Gazette. Having been busy chucking shrimps on the barbie, stoning the crows, and mooching the sheilas, it has taken me all this time to get MLX from the magazine onto disk.

When I try to enter Pointer, MLX lets me enter the starting address but makes a rude noise whenever I try to enter the ending address. I've seen so many INVALID ENTRY messages that I get nervous passing hospitals. Could you point me in the right direction, mate?
VIC WOODLEY
MOUNT LAWLEY, W.A.
AUSTRALIA

That goes back a bit, but we checked the addresses of that program and couldn't find any problems using starting address C000 and ending address C20F.

MLX helps you enter machine language programs, but it's not too particular about what addresses you give it. Actually, any address that is beyond the ending address of the program will work. If you're not familiar with MLX, however, you might have mistaken a 0 (zero) for the letter O in address C20F. The next line that follows would be C210; you could enter that line as the ending address. Just remember that the last digit is zero.

MISSING PROGRAM
In the May 1993 issue, the program Drop-Down Macros was listed in the contents, but it was not published in the magazine. Also, there is never any indication that any back issues of Gazette are available or where to send for them. What gives?
WALTER C. WARMAN
S. BURLINGTON, VT

Each column in the printed versions of Gazette contained special typesetting codes, and the Table of Contents was no different. Instead of entering all these codes from scratch each month, editor Tom Netsel would often copy the previous month's file and use it as a template, overwriting or deleting the old material. You didn't miss the program unless you didn't get the previous issue. Drop-Down Macros appeared in the April issue of Gazette. Unfortunately, it wasn't deleted when the contents page was used for the May issue. Mr. Netsel regrets the error.

We have many back issues of Gazette, and they are still available. Space prevents us from listing them and their contents. You may request any back issue of Gazette or COMPUTE by writing to Gazette Single Issue Sales, COMPUTE Publications, 324 West Wendover Avenue, Suite 200, Greensboro, North Carolina 27408. The price for one back issue is $6, or you can order two for $9.

SMALL TYPE
I have typed in numerous Gazette programs and have found the print in the listings quite small. After typing for about an hour, my eyeballs feel as if they're going to fall out of my head. Is there anything you can do to enlarge the print in the listings?

Also, is there a program or utility that changes BASIC programs to machine language?
ERIC ROY
TERRANCE, B.C.
CANADA

Well, starting with this issue of Gazette, small type is a thing of the past. All of the Gazette programs are ready to run; there's nothing to type in.

In order to fit as many program listings as we could in each issue, we had to keep the type fairly small. If we made the type larger, then we would have had to reduce the number of programs. The listings were made on a daisywheel printer using uppercase letters. These listings were then reduced to about 70 percent of their original size for printing in the magazine. Printing the listings on unglossy paper helped reduce glare and reflections from the page, and many readers said they appreciated that.

Some listings did get quite small when they appeared in one of our columns such as "Programmer's Page." If the program was fairly long, we usually included The Automatic Proofreader checksums. These listings had to be reduced considerably in order to fit in the narrow columns used in that part of the magazine. If the program was fairly short, we would type in the listing without the checksums. These programs were in bold and somewhat easier to read.

If you still have programs that you want to type in, here's a tip that might help if you have access to a copy machine. Copy the page but enlarge it. Many copy machines let you double the size of the image or make it even larger. Several readers have passed along this tip.

As for turning BASIC programs into machine language, you'll need a compiler. A compiler can make your BASIC programs run from 5 to 50 times as fast. There have been a number of compilers on the market over the years. Blitz from Skyles Electric and Compiler 64 and 128 from Abacus are two popular ones. You might check with Creative Micro Designs, (800) 638-3263, for either of these products.

FALLING LETTERS
I am trying to write a title page for a program, and I'd like the text to fall down from the top of the screen. Can you help me with this?
BRIAN MILLER
SACRAMENTO, CA

Here's a short routine that should do the trick. M$ is the title of your program or any message that you want to appear onscreen. It can contain any text or color that you want as long as it's not more than 40 characters long. LN in line 30 is the line number to which the text falls.

```
10 PRINT"(CLR/HOME)"
20 M$= * * * FALLING LETTERS * * *"
30 LN = 5
40 B= LEN(M$);CT=(40-B)2-1: FOR A= 1 TO B: X$ = MID$(M$,A,1):
PRINTCHR$(19)
50 IF X$<>" " THEN FOR T = 1 TO LN-2: PRINTTAB (CT+A) X$CHR$(145):
PRINTTAB(CT=A)" "
60 NEXT: PRINTTAB(CT+A)X$: NEXT
```

For variations, you can repeat the program with another message, moving the cursor HOME rather than clearing the screen. You can also use a FOR-NEXT loop to alter the text and colors for other effects. Load and run FALL LETTERS1, 2, and 3 on this disk for examples.

If you have a question, comment, or suggestion, send it to "Feedback," COMPUTE's Gazette, 324 West Wendover Avenue, Suite 200, Greensboro, North Carolina 27408.

Gazette, January 1994

Review
ELEMENTARY MATH SKILLS AND SAT MATHEMATICS

Reviewed by Alison Netsel

Parents with children in elementary grade levels should take notice of Microphys Programs. This company has a wide range of educational programs still available for the 64. Here's a look at two of them.

Elementary Math Skills offers help with addition, subtraction, multiplication, and division, allowing the child to practice problems like those studied in class. It also provides the student with some basic computer experience. Even if the child has had no previous interaction with a computer, he or she will be able to use the program because it's simple enough, with clear instructions, for the most inexperienced of users.

The program provides four different levels of practice of these basic math skills, ranging from simple two-digit addition to complicated long-division problems. Therefore, if more than one child in the household is in elementary school, you can set the proper difficulty level for each of them.

The menu-driven program offers six courses of action from which the student may choose. Each of the basic skills may be practiced separately at any of the skill levels, or the child may choose to practice all of the skills at random and at the desired skill level.

Once the students have had some practice, they may then test themselves by determining the number of questions to be tried at their skill level. A hardcopy may be made at this point, and I recommend it. The test is set up so that each problem is shown individually, but the answer isn't entered onto the screen. The program moves to the next problem when the student is ready. If a printer isn't available, then the child may simply solve the problem and write down the answer.

Once the students have answered all the questions, the answers may be checked by calling up a different part of the program which has kept track of and answered the questions selected from the test.

This program not only offers children a way to practice the kinds of problems that are being taught in class, thus improving their math skills, but it also gives the parents a chance to work with their children. With the unlimited number of problems, parents may easily create any number of appropriate tests for their children. Also, because of the simplicity of the program, parents can create their own games for the children which incorporate the problems generated by the program. In this way, parents can keep track of their children's progress while supporting their efforts to improve these skills.

Still, the program is self-contained and offers encouragement to the child while also showing a record of the overall performance at the end of each category. For example, when the child decides to end the multiplication section, the computer lists the number of problems attempted, the number of correct and incorrect choices, and an overall percentage, with words of encouragement. During the problems themselves, the program supplies a prompter for each answer and tells the participant whether or not the answer is correct.

This program offers a wonderful chance for children in elementary school to practice their math skills, preparing them for more advanced work.

For high school students, Microphys Programs also offers an SAT Mathematics program. In addition to helping prepare for the math portion of the SAT, this program can be used for the general practice of concepts studied in class. (MicroPhys also offers a verbal SAT preparatory program for students.)

The program offers exposure to six different fields of mathematics: algebraic expressions, fractions, decimals, geometry, ratio and proportions, and coordinate geometry. The student must choose from one of these six sections and then decide on the number of problems to attempt. The problems themselves are set up as they are on the SAT. The question is given along with all relevant data, and then the student must decide on the correct answer from the five possible answers supplied in a multiple-choice format.

Once a choice is made, the computer shows whether or not it's correct. If the answer is correct, the student moves on to the next question. If an incorrect answer was chosen, the problem may be tried again, or the student may see an explanation of how the problem should have been solved. The student may try the problem again, but the program won't simply supply the correct answer. It will only explain the proper method of solving that type of problem. Knowing that one of the choices is the correct solution, the student should eventually hit the right answer, even if out of pure luck. At that point, though, the student may use the correct answer along with the information provided to understand how the problem should be solved. Of course, if it's still unclear, the student should ask his or her instructor for help, just as the program itself encourages.

Unlike the Elementary Math Skills program, there is no random testing section, nor is there an option of choosing the degree of difficulty. But the explanation of the proper problem-solving procedure makes up for that since the material being tested should have been already covered in class.

The program could still be useful even after the student has taken the SAT. If the decision is made to take the GRE at the end of the four years of college, then the program could be extremely helpful in refreshing the memory of someone who has taken only the minimum of math courses in college, as many students do.

Overall, the program is very simple, and while it can hardly be considered a program that students will jump to use, if students are motivated, then SAT Mathematics will probably be considered quite helpful, especially for students who aren't comfortable with the mathematics section of the SAT. Also, this program would be ideal for those students who don't have the time for SAT preparation classes. It would be particularly helpful since the student decides on the number of questions to be tried and from which section the questions are to come.

Microphys Programs
12 Bridal Way
Sparta, NJ 07871
(201) 726-9301
Elementary Math Skills--$25
SAT Mathematics--$69.95

Gazette, January 1994

64/128 VIEW

By Tom Netsel

Welcome to the new, improved Gazette Disk. I'm sure that those of you who already subscribe to Gazette Disk will notice the changes that have taken place within the past month. We've added a number of features to the disk, and we felt that the disk needed a new interface.

For one thing, the disk is now double-sided. This extra space will allow us room to expand without worrying about running out of space quite so quickly. When we started adding Steve Vander Ark's "PD Picks," we were worried about how large some of the programs were and whether we had room on the disk. With both sides of the disk at our disposal, that's no longer a problem.

For the time being we'll probably have programs that are about the same size as those that have been appearing in the magazine. We usually kept them to 25 blocks or less. Trying to type in longer programs could be a real strain.
I also wondered just how many of you actually typed in the programs that we listed in the magazine. A number of readers have told me that they gave up typing in programs years ago. If you're like me, you'd keep back issues and then enter a utility or program when you had a need for it. I'm sure many subscribers would get the programs on disk from user groups.

In the past, we had a policy of letting user groups distribute Gazette's copyrighted programs to members who owned the magazine. We figured that if you bought the magazine, then you had the rights to the programs that it contained.

Gazette Disk is still copyrighted, and our policy will have to change starting with this January 1994 issue. Now that the programs and documentation are together on disk, user groups should not make the new Gazette Disk available to its members. Owning a copy of COMPUTE does not entitle anyone to the contents of Gazette Disk.

Getting back to our programs, let me tell you what to expect in the months to come. As I said, you can look forward to larger, more complex programs on disk than those that appeared in the magazine. These will start to appear as we use up the programs that we've already purchased and as programmers submit their larger efforts for publication.

Let me take this opportunity to thank all of you who have sent program submissions. Once again, we have a large number that are still pending. I can buy only so many each month, and I hate to reject good programs simply because I can't use them immediately. In any event, Gazette is still looking for good programs, but now you can send us your larger ones. If it's a good program, we won't reject it simply because it's too large to offer as a type-in.

As I write this editorial, the interface for this disk is still being designed, so I'm not sure how things will look when they all finally come together. Bruce Bowden, our technical editor, is handling that chore. Right now, we plan to have articles, features, and program instructions on one side of the disk, with the programs themselves on the flip side. You should be able to read the text onscreen or send it to your printer for a hard copy.

Since this is a transition period for us, moving the magazine from paper to disk will undoubtedly uncover some surprises, both good and bad. I hope you'll be quick to let us know what works and what doesn't, what you like and what you don't. If you have suggestions or comments, please send them to me in care of our usual address. Although our format has changed, we still want to be a major resource for Commodore information and entertainment.

Send any comments, questions, or suggestions to Gazette, COMPUTE Publications, 324 West Wendover Avenue, Suite 200, Greensboro, North Carolina 27408. I can also be reached on QuantumLink where my handle is Gaz. I am also COMPUTE's Online editor on America Online. You can send e-mail to me there by addressing it to Tom Netsel.

Gazette, January 1994

BETTER BASIC PROGRAMS

By Cameron Kaiser

Here are some tips from a veteran 64 programmer that can help your programs run better, faster, and friendlier.

Getting the most out of your Commodore programming means you have to learn how to program efficiently. This may seem obvious, but in the ten-odd years I've been programming my 64, I've seen numerous programs that could compete for Ugliest Code of the Year. This article is largely for BASIC programmers struggling to get their code out of the mire, but even old hands may find some tips worth remembering, too.

First on the list: Crunch it. Condense your code wherever you can. Programs like the following take up unnecessary space.

```
1 PRINT: REM THIS PRINTS SOMETHING
2 GOTO 10
```

First, toss out REMs. The only exception to this is if you want your program to be easily readable. We'll look at this use later. Otherwise, can them completely, or at least keep them to a minimum.

Second, if you can put it on one line, do it. Line numbers take up additional room. While it might be splitting hairs to do this in a short program, in a larger one where memory constraints could pose difficulty, the saved bytes could add up.

Third, ask yourself if there's a better way you could write the code. Look at the following lines, for example.

```
1 GETA$:IFA$=""GOTO1
2 IFA$="Z" GOTO4
3 GOTO1
4 Program resumes
```

This snippet has waste written all over it. First of all, you can get rid of line 3 completely by changing line 2 to: IF A$<>"Z" GOTO 1. In fact, you can even eliminate line 2 and make the whole program one line: 1 GET A$: IFA$<>"Z" GOTO 1. That's it. You've just compressed three lines into one. And remember; there's more than one way to skin the proverbial cat. Instead of GETA$:IF A$="" GOTO 1, you can just say WAIT 198,1:GET A$. It's much simpler and a heck of a lot more elegant. Plenty of alternatives exist for other commands, too. Look for them.

Finally, drop anything you don't need to have in your program. For example, the variables in a FOR-NEXT loop don't need the variables repeated following NEXT. Just leave them off. The loop will run more quickly, too.

Your 64 doesn't care about spaces (except in one rare situation; we're getting to that), so drop them. Even some keywords are optional. You don't need LET, but I still see it occasionally. Just leave it out. More on efficiency later.

The exception to leaving spaces: Use them where there could be some ambiguity with your keywords. For example, if you wanted to say IF X = T AND Y = 3, you might be tempted to run it together as IFX=TANDY=3. Run it like this, however, and you'll get a SYNTAX ERROR. The computer sees the TAN as a keyword for tangent and reads the statement as IF X=TAN D Y=3, which is obviously a error.

If you keep getting errors in a line such as this, try spacing it out. It may help. When you can, however, eliminate unnecessary spaces; they're wasteful.

Code Efficiently
Make the code efficient. To do this, start by following the first tip. After all, shorter code means code that will run faster. By exploiting a few tricks with BASIC, however, you can make the code that remains run faster.

First, DIMension every variable you can think of, even nonarrays. The DIM statement forces the computer to set aside space for these variables and can help speed access, even if only slightly. Many of my programs have a line similar to the following at the beginning.

```
10 DIM X$(32), FF$(17), A$, X$, C$, X, Y, T
```

Also, any value you use frequently should be assigned a variable. Every time you specify a constant in your program, BASIC must work to decipher it. With this constant in a variable, the value is in its optimal form. For example: the commands NU=0:TD=1000:FOR X=NU TOTD:NEXT will run much faster than simply FOR X=0 TO 1000:NEXT. Like DIM, it helps to assign variables at the beginning of a program, following a DIM statement.

Lumped into variable management is garbage collection. If your program uses numerous arrays, try putting Z=FRE(0) into a nonspeed-dependent routine before getting to the heart of your code.

Calling the FRE(x)function forces the computer to clean out extraneous variable junk. With this off the computer's stress list, it has more time to devote to your routine.

Something else of note. Don't branch in FOR-NEXT loops, using GOSUB, GOTO, IF-THEN, or other commands. This is because BASIC is inherently stupid. Whenever you specify a line number, BASIC must translate it before it can branch. This takes a relatively good amount of time. BASIC doesn't have the foresight to realize that, in this loop, it will be doing this same operation over and over. So, it blithely translates the number over and over and wastes time.

GOTO and IF-THEN in a loop are particularly bad because they abort the loop altogether. This leaves junk on the computer's runtime stack and, if done too often in a program, causes a spurious ?OUT OF MEMORY ERROR message down the road. Besides, who wants to GOTO out of a loop, anyway? So, never branch in a loop. If you need the code in the loop, it's actually more efficient to make a carbon copy of the code in the loop than to branch to the code elsewhere.

As well as this, be aware that certain BASIC operations are faster than others. The difference between addition and subtraction operations isn't much, so there's no real incentive to change these. However, both are much faster than multiplication and division. Of these, multiplication is faster than division, and compared to these, exponentiation is positively molasses.

Exploit this knowledge in your programs. For example, if you want to divide by 10, you may find it faster to simply multiply by 0.1. Or, if you want to square something, instead of entering X ↑ 2, use X * X. In a FOR-NEXT loop, the milliseconds you save with a single operation could add up significantly.

Debug

Debug your programs effectively. Most programmers, whether they know it or not, debug by hand-execution. That is, they run the code through in their heads by hand and try to anticipate what the computer will give in a certain situation. Obviously, this isn't the best strategy, because there is always room for error. Other techniques exist that can even make the debugging process seem almost automated.

First: Trace the code. This can be as simple as adding PRINT statements that print "line 100 here," "line 200 here," and so forth or as complex as actual physical logic utilities, such as Gazette's recently published XLogic (September 1992). This way you can monitor the performance of the program down to the line number, and when an error does occur, you'll have an idea of where to start searching for it.

The second technique is called wolf fencing. When hunting for a wolf, to cut down its area to roam, you cut its habitat in half by using a fence. Then, when you determine which side of the fence the wolf is on, you halve that. And so on. In your program, the "wolf" is the bug.

To build a "fence," add a marker statement. If you want to monitor a certain variable, add a PRINT statement to put the variable onscreen. If you want execution to halt so you can examine code, insert a STOP. (Be sure you don't edit program lines after the program has stopped or you won't be able to continue with the CONT command.)

If the error hasn't occurred to that point, you can be assured that the wolf isn't on that side of the fence. Once the wolf does show up, you can eliminate the code you've just checked as clear.

Take this following program, for example. It has a bug in it.

```
0 PRINT"LOOK AT ME, MOM!"
20 READ A$, B$, C
30 PRINT C$+B$+A$
40 DATA "OW!","ORK N","I W"
```

As you probably have already determined, the error lies in line 20. But wait! Run the program, and the error perplexingly pops up in line 40.

Let's say you didn't realize that line 20 was at fault. This is where wolf fencing comes in handy. We know the error can't be in line 40. Line 40 is just data. So, we add a couple of wolf fences to narrow the remaining suspects.

```
1 STOP
11 STOP
21 STOP
```

Now, we run it again.

```
BREAK IN 1
```

So far, so good. We continue by typing CONT.

```
LOOK AT ME, MOM!
BREAK IN 11
```

Still OK. Let's continue.

```
SYNTAX ERROR IN 40
```

There it is: line 20. We know this because the program stopped before it hit the wolf fence at line 21. We fix line 20, changing C to C$, and then run the program again. After you skip over the wolf fences using CONT, you will see a message onscreen. Now that you know the program works, you can delete the fences.

Of course, wolf fences are really good with programs with vast numbers of subroutines that are difficult to track. Short programs such as this one example can be traced by hand. If you're not sure whether it's a SYS command or a GOSUB in line 300 that's giving you trouble, a wolf fence can help you find the pesky error.

A third method helps with long-term programming projects and is the only sanctionable use for REM statements. Adding comments to your program assists you and any people updating or investigating your code. I've had to scrap many old projects because I failed to include proper internal documentation. A year or so later, I've forgotten what I've written, where, and why.

For example, can you tell what this does?

```
0 GOSUB400
20 REM Your program here
398 END
400 READA$:POKE2,LEN(A$)
410 FORX=1TOLEN(A$):POKE827+X,ASC
    (MID$(A$,X,1)):NEXTX:SYS944
420 RETURN
```

What if the programmer added this?

```
399 REM LOAD FILE
```

Much easier, huh? This REM is fulfilling its purpose. Although it takes up space, it doesn't get executed, so it slows nothing down. And it's clear and concise.

A programmer investigating your code would clearly understand what's going on now. You can see by the SYS that this is a machine language file routine. Location 828 must be where the filename starts, and 944 the load routine starting address. If a bug appears, the programmer has a rough idea of what should be happening and can program to accomplish it.

To reach this conclusion in this particular example would require a starting knowledge of ML, but even with a BASIC algorithm, a simple REM clue could save a lot of time in deciphering. It just makes good debugging sense when you know your code is going to be studied, and it's really helpful for future reference. Once you finish your program, you can take out the REMs.

Add Crash-Proofing
Make your code robust. Robust code is code that an enterprising user can't crash. There are a number of ways to do this.

First, watch for the INPUT trap. INPUT is a messy command because it lets the user type anything he or she wants. This means that when INPUT expects numbers and gets something else, the user gets the cryptic ?REDO FROM START message. It also means the user gets ?EXTRA IGNORED when INPUT gets commas or more input than requested. And, it also means that programs can get crashed when the user enters something that the program wasn't expecting.

If you want to go the easy route, programs like Gazette's XINPUT (March 1993) can help. You can also use the GET loop trick. Here's an example that accepts only numeric input, up to five characters.

```
1 PRINT "ENTER A NUMBER"
2 NM$="":POKE204,.
3 WAIT198,1:GETA$:IFA$=CHR$(13)    ANDNM$=""GOTO3
4 IFA$=CHR$(13)THENPOKE204,1:END
5 IFA$<"0"ORA$>"9"GOTO3
6 IFLEN(NM$)=5 GOTO3
7 PRINTA$;:NM$=NM$+A$:GOTO3
```

When executed, this program displays a cursor and prompts for a number. This rudimentary example is reasonably userproof, although it lacks the ability to correct for improper use of the Inst/Del key. It won't accept a null string or anything other than a numeric digit. All that remains is to use VAL(x$) to extract the value.

The second trick is to thwart tricky users. A user who really wants a look at a program will go to almost any length to do so. A simple POKE 808,225 at the beginning of your program will stop people breaking out with Run/Stop or Run/Stop-Restore. (This POKE also fouls up LIST.)

Another tip is the REM Shift-L trick. When the program is listed, it stops abruptly with a ?SYNTAX ERROR. There are numerous ways of keeping nosy users out of your programs. I sometimes use a fake compiler that makes my programs look as though they've been compiled. When users list the program, all they see is 10 SYS2061. Actually, there's really a BASIC program inside. Somebody who goes hunting for ML code is going to get a surprise.

Next, check your inputs. I don't know how many programs I have inadvertently disabled when I entered something the program didn't like and the program crashed. For example, can you see the problem in this program other than the INPUT?

```
1 INPUT "NUMBER TO DIVIDE 49  BY";Z
2 PRINT 49/Z
```

As far as code goes, it's bug-free. But run it and try entering a 0 for the requested number. Bingo! You get a ?DIVISION BY ZERO error message. See what I mean? In this instance, the error could be corrected by a simple IF-THEN. In other cases, the fix might have to be more extensive.

Another way to make your programs crashproof is to program defensively. Try to anticipate every outlandish situation. If you think the user will enter your mother's birthday for a phone number, program for that possibility. I guarantee that the situation you don't anticipate is precisely the situation that 50,000 users will try. This can be cumbersome, but do you want your program to work or not?

Be Nice

Be nice to your user. Even if you're writing programs for your own use, read this anyway. You never know when your program might get used by someone else.

Unless you're in the CIA, you'll want your programs to be easily understood. You'll also want them to work as soon as the user types RUN. This becomes more imperative as a program grows in complexity. This is where the term "user friendly" comes into play. If your program is difficult to use or understand (user hostile), then people will avoid it no matter how useful it might be. People don't like to struggle with something they can't understand.

Here's an example. If today were Wednesday, December 15, and a program simply prompts you for the date, what would you enter? What format would you use? You might enter DECEMBER 15, 1993, but the program might want 12/15/93? Without a proper prompt, the user can only guess, and that can become frustrating.

Always, always, always tell the user exactly what the program wants and how it wants it. In the example I gave, an ideal INPUT prompt would be something like the following.

ENTER THE DATE IN (MM/DD/YYYY) FORMAT.

Is this excessive handholding? Not a bit. You would be surprised at how many users can overlook the obvious. If there are any doubts about a program's input requests, it's up to the programmer to clarify them.

The second item to remember in order to keep your programs friendly is to reassure the user. Many users get nervous with computers. If the computer isn't doing something that they can see or hear and the computer won't respond to their keypresses, they may assume the program has blown up. An error message is even more disheartening.

If nothing else, keep the screen busy. Blink a message on it, such as LOADING DATA. Simple animation is even more reassuring than the most friendly message. It lets the user know that computer hasn't locked up.

Never let an error message appear! If the user gets an error message, it's the programmer's fault. There is no excuse for buggy code.

Keep your program consistent. If the user is used to one pattern in a program, then keep it that way. I have encountered programs where not five minutes away from one prompt, I get another wanting the same data, but in a completely alien format. I spotted this quirk relatively quickly, but do you think your average user is going to be as lucky? Establish a clear system of rules governing how data is to be displayed and accepted. Failure to do so will result in a lot of angry users (and a lot of headaches for you).

Where INPUTs are concerned, always allow the user to verify his or her entry. Few things are more annoying than a program that jumps the gun when you've entered something incorrectly.

Whenever you can, add some assistance for the user. Help facilities are nice, but they do eat memory. You can give cheap help in simple ways, like making your commands easy to remember. For example, let M stand for Move or add cursory tips in prompts and screen displays. A user will appreciate what help you can supply.

Get a pattern going. Too many programmers waste precious time reprogramming the same stuff over and over into every program they do.

Too many other programmers use cumbersome methods to do things that can be done for them by outside utilities faster and (often) better.

First of all, if you have even a rudimentary knowledge of machine language, write your speed-dependent routines in ML. Obviously, some tasks would be easier to write in BASIC. But if the routine can go in ML, use it. ML doesn't have the memory restrictions of BASIC, and it can run very fast.

Second, get a basis disk or file. When I get going on a new project, I have something called a basis disk that has all the routines and programs set out that I'll need.

Typically, the disk has a memory mover routine, a memory fill routine, a fastload routine, and then a BASIC template with all the support utilities to drive these ML routines. I load the template, and away I go. It saves my having to sort through my library, picking and choosing necessary routines.

Keep an eye out for automated utilities. To write music for my games, I use a program that I found in Gazette called BASICally Music. It can create music files that are independent of the parent program.

To create custom characters, I employ Gazette's Ultrafont Plus. This program makes code that doesn't require the host utility, and what's more, the fonts are easily portable. A word of caution: Many utilities of this nature do need the host program in order to operate. Also, some routines are copyrighted.

Never pirate! You may not get caught, but you will be depriving the author of the credit he or she deserves. Take this advice from an author who sees this happen to his own creations.

These tips are those gained from my own experience. Use them or forget them, but if you do use them, I hope they will bring you lots of luck and better programs. After all, I haven't been programming on my Commodore almost ten years for nothing.

Gazette, January 1994

@ Nice
Be nice to your user. Even if you're writing programs for your own use, r
h@dad@a b!a8d@a£a8aga8d@a8a8bad@a£a8aga8d@a8a8f@dad@c