

September 5, 1998

Dear Laser-Lover,

Perhaps with tongue-in-cheek, I called the LASER-LOVERS disk I sent to you Version 0.8. I put the disk in the mail two days ago and already I have found two errors. No, *that's* not funny.

However, these errors do not require sending out replacement disks. Rather, you can easily correct them yourself, and I ask you to do so at your earliest convenience.

Pretend for a moment that you are as absent-minded as I am. You try to print out the *ToughTimes* articles by first downloading the *Tymes-Elfin* font. But you forget to hit **Control-D** after sending the font as suggested in the instructions. Instead, you immediately download the next file, *ToughTimesTitle*. The printer "crashes!" Why?

Had you sent the *PS.ErrorHandler* to the printer beforehand, it would report to you that "pop72" was an undefined command. You would then search both the 'font' file and the 'Title' file for the offending string, **pop72**, and you would never find it. Why? Because the error was generated when the last word in the *Tymes-Elfin* file became "glued" to the first text object of the *ToughTimesTitle* file, **72**. If you send the Control-D after the font file, the error will never occur. But if you do not, you still should be able to print the file. The font file contains a small but significant error which needs to be remedied.

Please correct this error by first copying the *Tymes-Elfin* file to another disk or partition. I stuffed the LASER-LOVERS disk so full there is only 1K of space left! After copying the font file to another disk or partition, open it up with *geoWrite*, move to the end of the file on page 15, and then put a space or a carriage return after the last word (*pop*) in the file. By placing a separator or delimiter after "pop," it will never be "glued on" to anything again! ✓

Putting this corrected font file back on your LASER-LOVERS disk may be more difficult than you think. First, I had to erase the offending file from the original disk. Then I had to validate the disk to make sure all the now empty space on the disk could be accounted for. Finally I was allowed to download the corrected file to the disk.

The second error is more crucial. The *ToughTimes* articles were designed to tell a story AND to demonstrate a non-resident font. Therefore, whenever I named the font in the article, I used the *Times-BoldItalic* font. Then before sending it to the printer, I substituted the name of non-resident font, *Tymes-Elfin*, in place of the aforementioned GEOS font. Inadvertently, the *ToughTimesTitle* file on your disk is apparently an old copy. The name of the non-resident font has not yet been substituted.

Therefore, I need you to make this change in order for the non-resident font to be accessed when printing the *ToughTimes* articles. Again, you will need to copy the offending file, *ToughTimes Title*, to another disk. Open it up in *geoWrite* and go to page 2. Look for the first line of the font array as shown below.

```
/fonts[/Times-Roman/Times-Bold/Times-Italic/Times-BoldItalic/Helvetica/Helvetica-Bold
```

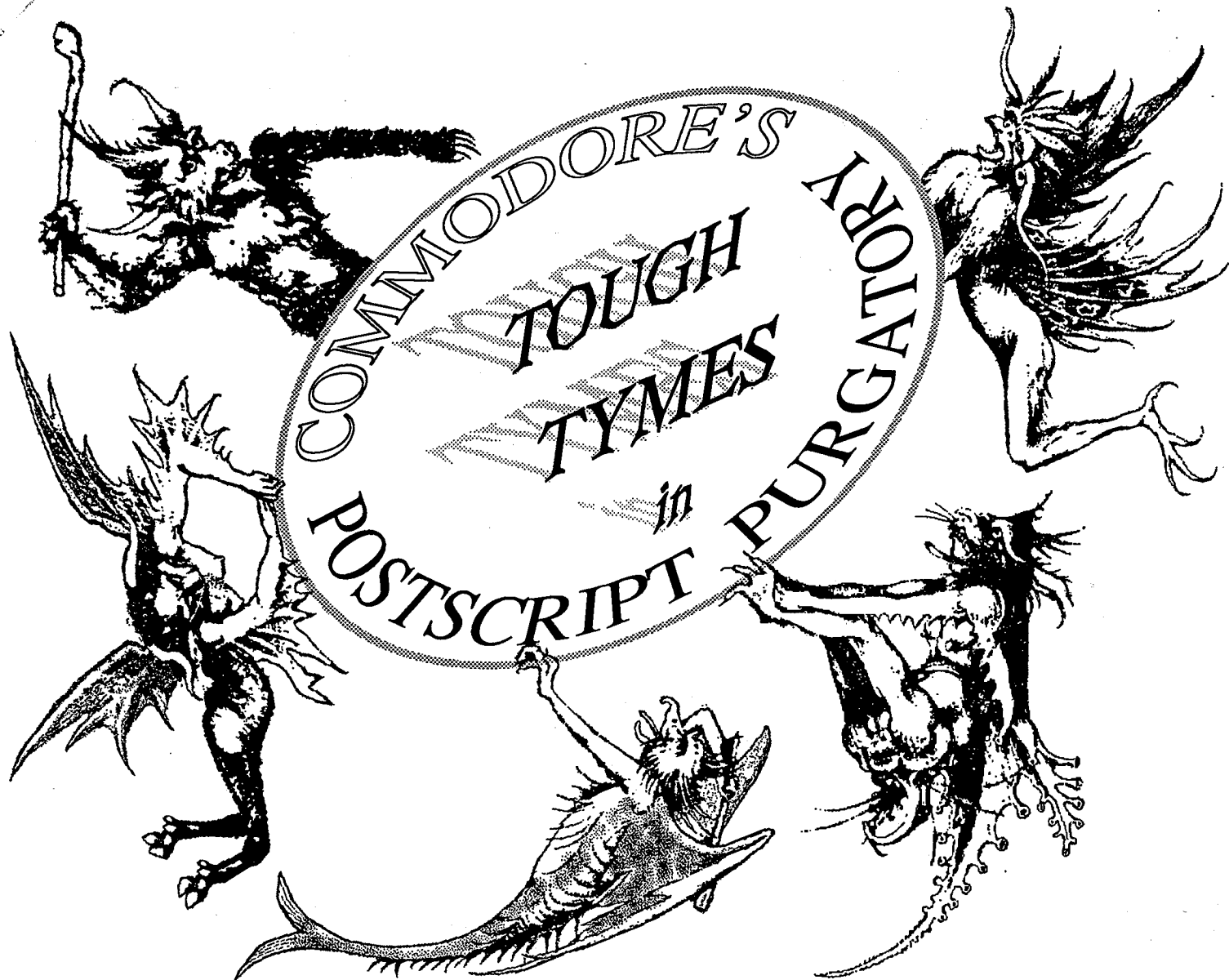
Please substitute the name of the *Tymes-Elfin* file for the *Times-BoldItalic*. It will now read: ✓  

```
/fonts[/Times-Roman/Times-Bold/Times-Italic/Tymes-Elfin/Helvetica/Helvetica-Bold
```

This will correct the file. Return the corrected file to your original LASER-LOVERS disk for safe-keeping. You will need to do this the same way you handled the corrected *Tymes-Elfin* file. I feel guilty about failing to catch this error earlier, so I am sending with this letter a copy of Part I of the two *Tough Times* articles. It demonstrates how those articles should look when printed properly. Please try to duplicate this to show that you have mastered this section. As always, feel free to contact me if you have questions.

Thanks again for your interest and participation in PSPFTC.





## PART I The TORMENT

by Dale Sidebottom

I consider Commodore computing to be the greatest hobby going. It usually makes me feel terrific; though sometimes, when things go badly, I feel terrible. In 1994, it smacked me up against the wall and nearly threw me over the edge. My wife doesn't want to hear about it; so I thought I'd tell you.

It all started when I decided to learn how to write PostScript, the graphic imaging language created by Adobe Systems, Inc. This was the computer language that Macintosh employed when it first excited the business world with desktop publishing. In 1987, when GEOS introduced its own desktop publishing program, *geoPublish*, they thoughtfully included two programs which enable us to

access PostScript laser printing, *geoLaser* for *geoWrite* files and *geoPublaser* for *geoPublish* files.

When Jeanine Cutler wrote her article, "The Affordable Laser Printer," in 1990 for *geoWorld* (Issue #25), the "affordable" cost of a laser printer with PostScript was about \$1500. Today, Commodore users call me to brag about setting themselves up for less than \$300! I expect this trend to continue as these machines diminish in price. I also expect 300 dpi laser printers (like mine) to become "dirt cheap" when current users upgrade to 600 dpi printers. I predict, that within a few years, quality laser printing will be within the wallet-range of nearly every Commodore user.

Does this sound exciting? You bet! Then why is my title so depressing? I'm getting to that.

As I was saying, I dedicated myself in the spring of 1994 to learning how PostScript code is written. I hoped to accomplish whatever was necessary to bring all PostScript [Level 1] features into the Commodore world. I realized that this could be quite a challenge. With the lone exception of Jim Collette's *PS Processor*, which was written to extend the usefulness of *geoPublish* files, no PostScript programs for the Commodore had been written for the last seven years!!

I wrote about my frustration and my hopes in the January 1994 issue of

*dieHard* magazine. The article is entitled "Commodore's PostScript Revolution." The article explains my journey into this esoteric environment and why I think that we as Commodore enthusiasts should be giving PostScript a lot more attention.

"Where should I start?" I asked myself, "What is the greatest barrier facing us in PostScript, Level 1?"

The question that constantly intrigued me was, "How can we download a non-resident font?"

Consider this. Every PostScript printer has a variety of resident fonts already on board, usually 8-11 different font families. But Adobe System, Inc. has many more fonts for sale, and that doesn't touch the thousands available through third-party and PD sources! Do we want to write them off? *Heavens, no!*

So I accepted the challenge to discover how our Commodores could download non-resident fonts. For this and other reasons, I decided to join the Adobe Developers Association (ADA). This organization was established by Adobe Systems, Inc. to encourage third-party developers.

I sent in my application plus the \$195 in dues for the first year. When I received my membership card, I immediately called the Adobe technical support line and asked for a list of all the support files they had for the Commodore. The technical advisor informed me that I was the only Commodore user that ever joined the ADA! *I sensed the clouds of Purgatory beginning to gather.*

She did send me one program that has been very useful. It is a PostScript error handler. I still remember spending an hour one day trying to find the error I created when I left the "s" off the fontname, Times-Roman. This program, *PS.ErrorHandler*, saves me endless hours when testing PostScript programs.

Of course, she could not send it to me on a Commodore disk. Instead, I received it on an IBM formatted disk and needed to borrow a friend's *Big Blue Reader* to transfer it to my Commodore.

I shared with Randy Winchester my enthusiasm for PostScript, and he kindly sent me a disk full of PostScript files he had downloaded from the internet. I experimented with several of them. A few of them would not work until I corrected a few bugs.

As I was especially interested in fonts, I was excited to see it contained one user-defined font, *Times-Elfin*. I tried to download the original file to the laser printer straight from Randy's disk. It crashed! *The gates of Purgatory were slowly creaking open.*

Before continuing, I need to discuss how I downloaded PostScript programs to the laser printer. In the *dieHard* article, I was able to download "pure" PostScript programs by using Jim Collette's *PS Processor*, even though it was never created for that purpose. It actually worked by accident! Later, I learned why.

All good Postscript programs begin with a prologue. It contains all the definitions and procedures needed to manipulate the data which follows.

To manipulate geoPublish data, *PS Processor* will first dump the geoPublish prologue, nearly 8k in size, straight to the printer and *then* it will start sorting pages. So any PostScript file I wrote could be downloaded, as long as it did not exceed 8K. Anything larger would crash unless it indeed began as a geoPublish document!

The *Times-Elfin* font, hereinafter referred to as *TYMES*, was 82 kilobytes in length. Obviously, if I hoped to download non-resident fonts to the laser printer, I needed help! Just in time, Maurice Randall came to my rescue!

Maurice is a very talented GEOS/Commodore programmer. I first met Maurice by phone several years ago when I called to check his progress on an 80-column DTP program for the C128. When I suggested that I could send him money as an advance to aid in that development, he flatly refused! So I sent him \$25 for something called *geoSHELL* instead.

When it arrived, I read *geoSHELL's* meticulously written manual from cover

to cover. I loved the concept of this program. It could work with GEOS 64 or 128. The catch was that it was also the most un-GEOS-like program I had ever seen. *GeoSHELL* provides a CLI, command line interpreter, that is very much like using PC-DOS before the advent of *WINDOWS*. I appreciated his genius but put the program away thinking, "I'll try it someday."

When *someday* stretched into a year, I received the *geoSHELL* 2.2 upgrade in the mail. The "carrot at the end of the stick" was a command module called *laser*. Maurice informed me that this COM file, as he called it, could be used within *geoSHELL*, allowing me to download a PostScript file of any size. Since he had no laser printer, would I be willing to *beta-test* it? I agreed to try it immediately!

Next I needed to install *geoSHELL* on my Commodore system. That done, I tested the *laser* command, using it to download all the small PostScript files I had used earlier with *PS Processor*. *Laser* performed flawlessly. I was now ready to tackle bigger game.

Remember that my first attempt to tackle "big game" by downloading the *TYMES* font had failed! I assumed that this font must have a bug in it like some of the other PostScript programs I had found on the disk. I expected to fix it the same way I had fixed the others.

Removing the bugs from PostScript files can become rather involved. First I converted the original PostScript file (in TRUE ASCII format) to *geoWrite* format. From *geoWrite*, I opened the file, studied it, and made any necessary changes. Then the corrected file is converted from *geoWrite* format back into TRUE ASCII so that *laser* can download it to the printer. These conversions were always made with Joe Buckley's *Wrongs-Write81* (WW81).

When I converted the original *TYMES* file and attempted to open it with *geoWrite*, the dialog box counted off more than a hundred pages! Finally it opened, and I discovered that I could only access the first seven pages. That left me

over 90 pages short! Obviously, I needed to find another way to do this.

The few pages I *could* access told me an interesting tale. Remember, Postscript fonts are drawn, not bit-mapped. This enables them to be manipulated like graphics. As an example, the simplest letter in the alphabet, the lower-case letter "l", might be graphically defined as follows:

```
l {newpath 150 0 moveto 225 0
lineto 225 750 lineto 150 750 lineto
closepath fill}def.
```

The *newpath* signals a new start. The *moveto* operator is necessary to initiate every object. It says "start right here." The *lineto* instruction is misleading; it tells PostScript to draw a path [not a line] to the next coordinate point. That repeats twice until the *closepath* operator automatically closes the figure and the *fill* operator fills the enclosed path with the current color. The "curly braces" are used to enclose a Postscript procedure. See how easy it is!

But what if it looked like this:

```
l {newpath
150 0 moveto
225 0 lineto
225 750 lineto
150 750 lineto
closepath
fill
}def
```

The **TYMES** font file was constructed with a line-feed after every operator. This created "tons" of white-space. *geoWrite* cannot handle a file with over 60 pages. I needed a program that would take out all the line-feeds and replace them with blank spaces. I looked all over without success until Dave Snyder, the SysOp on our LUCKY BBS, gave me the answer. He suggested a program called *Velveta*. It would remove all the "foreign" bytes in a file and replace them with "benign" spaces. Since, in the Commodore world, text files use only carriage returns, a line-feed becomes a foreign byte. So I thought this could work!

Then I learned that *Velveta* is written in Commodore basic and only works on PET ASCII files. I used WW81

to convert the font file from true ASCII to PET ASCII. *Velveta* then melted **TYMES** down from 100+ pages to a manageable 16!

Now I was in business. With happy anticipation, I converted the PET ASCII **TYMES** into a *geoWrite* file and opened it. Then I realized something was wrong. Remember all those "curly braces" so vital to PostScript coding? *Velveta* thought they were all "foreign" bytes and killed every one of them. What else might be missing, I could only guess!

So I dug into my *PostScript Reference Manual* and every other PS book I could find. I thoroughly read everything concerning fonts, just to learn how to put **TYMES** back together again! I discovered that I still needed more help.

This is where the Adobe technical support line really paid off. With their assistance, I was able to master the finer points of "PS Font Construction 101" and recreate a workable font.

I put a special tag on the end of the **TYMES** file to print out the message, "**The Tymes font is downloaded.**" Everytime I downloaded the file, this printed out to confirm that the download was successful, but a light on the front panel of my printer kept blinking, blinking, blinking... What's wrong?

Again I call Adobe. "Why does my printer keep blinking. When I'm done, why doesn't it go back to sleep like it's supposed to?"

"You must send your printer a Control-D. That is a hex 04 byte. Your printer doesn't know you're done if you don't say so."

"Okay, how do I do that in PostScript?" I asked.

"You don't. Control-D is not a PostScript command. It is a printer convention. It never even reaches the PostScript interpreter."

So I asked Roger Lawhorn to create a one byte program. Now you and I know that a file must have more than one byte just to spell the title. But this file called *CONTROL-D* sent only one 04 byte to the laser printer. Then, magically, the printer returned to normal.

Picture if you will the following sequence. I download the *PS.Error-Handler* and then the *CONTROL-D*. I download the **TYMES** font, and then *CONTROL-D* again. My laser printer prints out the message, "**The Tymes font is downloaded.**" Now I send another file called *Tymes Testor* to check and see how well **TYMES** prints after it's downloaded. The *Testor* program prints out in the default font, Courier. **TYMES** cannot be found! I repeat this process over and over, unable to accept that the font is there, and then it's gone!

Well, by now, you know the drill. I call Adobe's technical support line and ask the advisor why my font keeps "going south." She asks me, "Did you download the font *under the server loop*?"

"Lady, I have six PostScript books here, and none of them say anything about having to download the font under the server loop!"

She explained, "The PostScript interpreter automatically puts a save and restore command around every incoming 'job.' When you downloaded the font, it stays in memory until you send the Control-D. Then it is immediately erased."

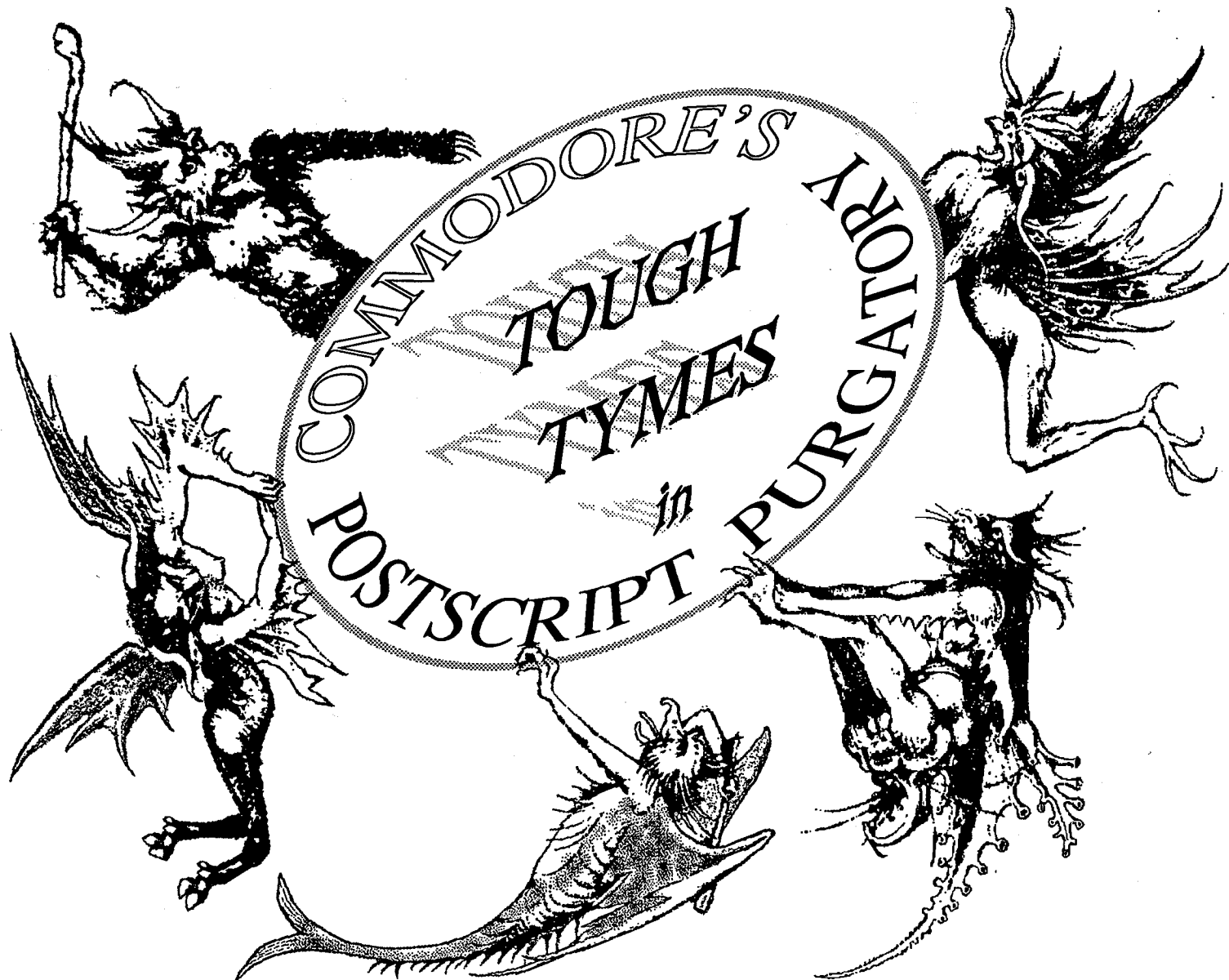
So how does one go *under the server loop*? She gave me the following two lines which go at the beginning of the font file.

```
serverloop begin
0 exitserver
```

This allows you to exit the server loop in the PS interpreter and then download a font so that it will not be erased until you reset the printer.

Now that I had all the answers, I began writing my "Victory Letter" to be printed out in my newest laser font, **Tymes-Elfin**! I downloaded it to my laser printer and it crashed! I sent it, again and again, and it crashed every time. *Along with it, Purgatory crashed in all around me.*

The letter simply refused to print. I couldn't believe it! How utterly devastating to realize that you have somehow managed to pull *defeat* from the jaws of *victory*! The "Victory Letter" had to be scrapped. I was left wondering if anything could be salvaged. ■



## PART II The TRIUMPH!

by Dale Sidebottom

If you have read Part I, then you know that I'm in trouble! Having learned to download a non-resident font called **Tymes-Elfin**, I suddenly run smack into the wall. While I can print out "**The Tymex font is downloaded**" on my laser printer, the letter I'm writing keeps crashing! Clearly, there are a few characters in the font that simply refuse to print.

The **TYMES** font is constructed of 96 characters and most of them appeared to print perfectly. I decided that I would find which ones did not. If the errors in the file formed a pattern, then the *laser* download program might be at fault. If there was no pattern, then the code in the program must still be buggy.

This was a long drawn-out task. Even dividing the characters into groups and sub-groups, testing takes awhile when you are bouncing a 82K file back and forth on a Commodore. I finally found six of the 96 characters that would not print. Since the errors formed no discernible pattern, it seemed to say that *laser* was downloading properly.

Therefore, I surmised, the problem must be caused by "killer bytes" hidden in the PostScript code. Even though *geoWrite* is a wonderful word processor, it does not know how to handle non-printable bytes. It basically ignores them. It was possible that the PS code could have been contaminated with bytes that might be invisible in *geoWrite*, but

could cause PostScript to crash as soon as they hit the interpreter.

So I asked Roger Lawhorn to use his sector editor and search for "killer bytes" within the definitions of those six unprintable characters. Roger told me the code was clean. How could that be possible?

I simply could not rest until I had checked it out for myself. Fortunately, Randy Winchester told me about a GEOS 128 sector editor available on GENIE. Having never used one, I thought even I could learn to use a sector editor if it was GEOS-compatible!

I asked our LUCKY SysOp, Dave Snyder, to download the program for me from GENIE. Soon he sent me a program

by Mike Craig called *geoDiskEdit*. It's terrific. Soon I was using a sector editor for the first time in my life and enjoying it. However, I did not enjoy the results. Roger was right. The code was clean.

Next I created a font, exactly like the first, but with this important difference. It contained only those six characters that would not print in the original font. I downloaded this abbreviated font and tested each letter. *They now printed perfectly!*

I was thunderstruck with the prospect of the whole not being equal to the sum of its parts. This was crazy! My immediate problem, of course, might be solved by downloading the font in sections, but would I want to do this for thousands of future fonts? *Hell, no!*

By now, I had concluded that the *laser* program must be doing *something* wrong. It sent down all the smaller PS files without error, but a 82K font might present a different challenge. I had to find out why! It was not enough to send it back and say, "Maurice, there seems to be a problem somewhere." Why did small PS programs work well while the "big one" kept crashing!

Perhaps I might be able to write a PostScript program that would make the laser printer disclose exactly what was stored in its memory. If I can find out how the code I am sending differs from the code the laser has received, I have a good chance of pin-pointing the problem.

This plan ran afoul of PostScript "protocol." In PostScript, *fonts are sacred and inviolate*. In art, a picture can vary from a simple "smiley face" to a masterpiece *Mona Lisa*. In PostScript, the same is true of fonts. I was working with a fairly simple font, but some are extraordinarily complex and masterfully constructed. PostScript will share almost anything you want to know except how its fonts are made. (This was the reason *PS.ErrorHandler* never told me why my font was crashing.)

It seemed like a dead end, until I realized that I was downloading the font. It was a user-defined font. All I, the user, needed to do was to *re-define* the program

so that the PostScript interpreter could not recognize a font. Then each individual font character would be stored as a separate definition. I also made access easier by exchanging the "curly braces" for parentheses, because parentheses are used to enclose a string. In PostScript, only strings can be printed. I felt certain this plan would work.

Like every other plan along the way, I experienced failure, failure, failure...and finally SUCCESS! The eventual PS code is so ugly, I'm ashamed of it, but it worked!

I was able to investigate the problem with a two-pronged approach. First, I printed out the code of each "bad character" before it was sent down. Secondly, I tagged the end of the **TYMES** font program to print this identical information after it had been downloaded into printer memory. The end effect was magic. I simply held the two print-outs together up to the light, and I could tell instantly if one varied from the other. And they all did!

In each and every instance, a single space had been added to the definition. In each of the six unprintable characters, that extra space landed right in the middle of an operator. For instance, our example definition started out *150 0 moveto*. If the space falls within the operator, *mov eto*, the PostScript interpreter will read *mov* as an undefined command and kill font processing.

I was shocked now to realize that extra spaces could be anywhere! If one space is followed by an extra one, it is ignored. Or it could fall within a number. For example, *150 0 moveto* would start an object at 150x, 0y on the coordinate scale. But if the space fell in the first number, *1 50 0 moveto*, the character could still print, but it would be distorted, starting at 50x instead. This explained to me why some of the characters looked a little weird, even in a font designed for druids!

At this time, a fortuitous event occurred. I was able to reach Jim Collette by phone, explain my problem, and ask his advice. "Look for boundaries," he suggested. So I used that GEOS-friendly

sector editor, *geoDiskEdit*, to check for boundaries. Where did the errors occur?

A GEOS-formatted 1581 disk has 80 tracks and each has 40 sectors, numbered 0-39. I discovered that all the errors occurred immediately following the 19th or the 39th sector. Since a sector equals a block, and four blocks make one kilobyte, the errors occurred 5 kilobytes apart. I did not know if an extra space was inserted every 5K, but I did know every error occurred immediately after that 5K boundary.

I should have felt great jubilation at this point, but I was simply relieved. It had been three months since I began this project, and the end was finally in sight. I also felt a sense of peace, knowing that the problem would soon be in better hands. *Purgatory began to fade as light filtered through the murky stillness.*

I called Maurice Randall and reported what I had found. He thanked me and said that he would check into it soon. Three days later, he called me back to explain the bug. His *laser* program, when used in parallel mode [I always use the geocable], first checked to see if the printer was connected. Then it filled a 5K buffer with data and sent it. Before sending the next 5K, it rechecked the printer connection. Apparently, this innocent but unnecessary request was creating an extra byte which PostScript interpreted as a space.

That evening, Maurice sent the amended version to the LUCKY Bulletin Board, so I could easily download it from there. Not surprisingly, *laser* now works perfectly!

Everything is back to normal. So why did I write this article? Well, I felt the need to exorcise whatever demons might remain. I also wanted to apologize to anyone who may have requested anything from me during that period. I did not want to procrastinate, "the devil made me to it!"

I also believe my experience re-emphasizes three things that Commodore enthusiasts need to remember:

3. *You do not have to do it alone!* Perhaps the area we most often ignore is this one: the importance of cultivating and maintaining a support group for ourselves. We all need to share with Commodore friends in times of torment and triumph. Without the timely assistance of those friends mentioned in this article, *I would still be in Purgatory.*

This project surprised me in two ways. First of all, I thought the biggest challenge facing PostScript users would be downloading non-resident fonts. However, after three months of turning the **TYMES** font "any which way but loose," I never found a single error in the font that wasn't mine! Downloading fonts was never the problem; *downloading* was the problem! I had failed to adequately *beta-test* the *laser* command within *geoSHELL*, and that is a crime for which I paid big-**TYME**!

## ACKNOWLEDGEMENTS

However, don't applaud. Send him money! His program *geoSHELL* is just

**WHEELS**, his newest upgrade for GEOS, enables us to operate from any CMD device, even within native-mode partitions! It costs \$36.00. (Add \$4.00 for shipping.) Order from Maurice Randall, P.O. Box 606, Charlotte, MI 48813-0606.

I also owe a debt of thanks to CMD. I used a 4 MB RAMLink and a 20 MB CMD HardDrive throughout this project. I personally believe that without them, this project would have taken years, instead of months. They also distribute *Big Blue Reader*. For product information, write: CMD, P.O.Box 646, East Longmeadow, MA 01028.

**Downloading fonts  
was never the  
problem;  
*downloading* was the  
problem! I had  
failed to adequately  
*beta-test* the *laser*  
command, and that  
is a crime for which I  
paid big-TYME!**

◆ ◆ ◆ ◆ ◆ ◆ ◆ ◆ ◆ ◆ ◆ ◆

This article was created and printed entirely within the Commodore environment. The "visions of loveliness" surrounding the title are slightly enhanced images gleaned from an illustration of a 1470's engraving by Schongauer entitled "Temptation of St. Anthony." Each figure was enlarged on a copy machine so that I could capture maximum detail with the HandyScanner 64. Each figure represents the compilation of 3-6 handyscanned photo scraps. These five figures total 64K but balloon to 128K when converted to hex!

This article illustrates the first of a new breed, the GEOS/PS hybrid! I use this term to describe a PostScript document in which everything possible is done in GEOS, usually with *geoPublish*. Then the PostScript version is ported over to *geoWrite* where all the "special touches" can be added.

For instance, the added "touches" in these articles include the "text in a circle" feature which appears ingenious, but I just copied it word for word from the *PostScript Tutorial and Cookbook* (known as the Blue Book), pages 163-165. The shadows were created with the "concat" (for concatenate) command.

My favorite innovation is invisible. Do you imagine that I scanned five "demons" at precisely the perfect angle? Actually, no. Only one of them is used as scanned. Instead, I changed the "PP" or PhotoPrint procedure in the geoPublaser prologue to "PPR", PhotoPrintRotate. I enhanced the code so that I could spin these "bad babies" into any position I pleased. I firmly believe that, without this added feature, the opening title would have been impossible for me to organize as I envisioned it.

I also changed the geoPublaser prologue so that the **Times-Elfin** font could be printed wherever Times (bold-italic) is formatted in the document. Of course, the **TIMES** font must be downloaded to the printer before sending these articles. Failure to do so will not cause a crash, because PostScript will simply substitute *Courier* as a default font in its place. ■

PostScript Printing  
from the Commodore

*featuring*  
PostPrint  
*PostPrint*

PostPrint



PostScript Printing  
featuring  
PostPrint  
PostScript  
from the Commodore

PostPrint

PostScript Printing  
featuring  
PostPrint  
PostScript  
from the Commodore

PostPrint

A word cloud of the word "POSTSCRIPT" arranged in a circular pattern, with the word "Programming" written in large, bold, black letters across the bottom.

# Programming

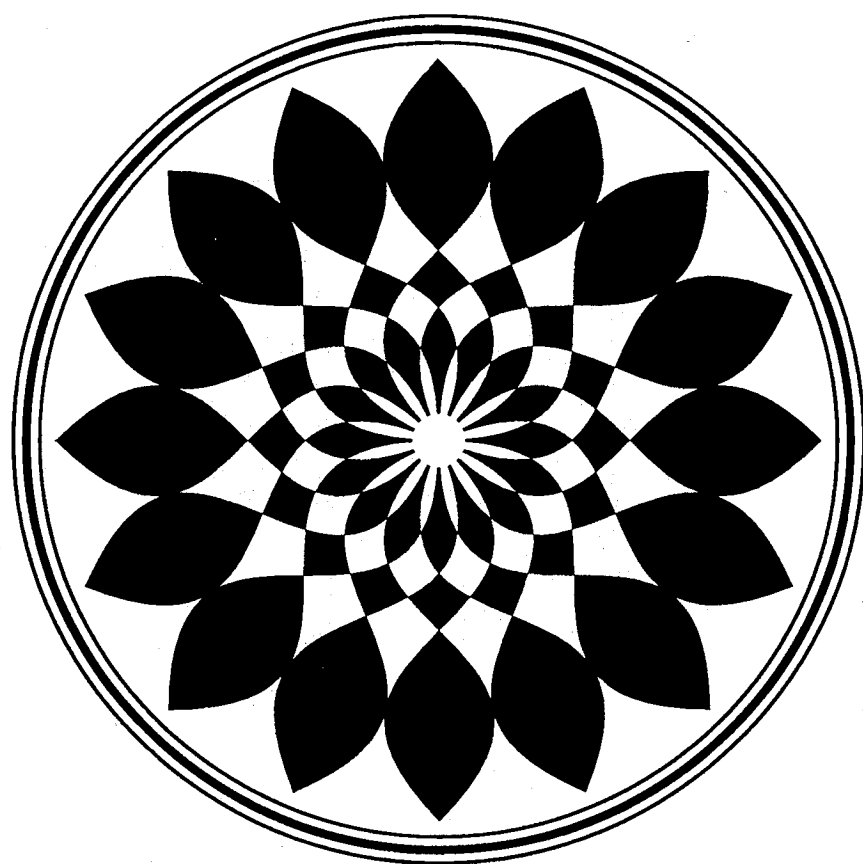


SHADOW

Of a

Commodore

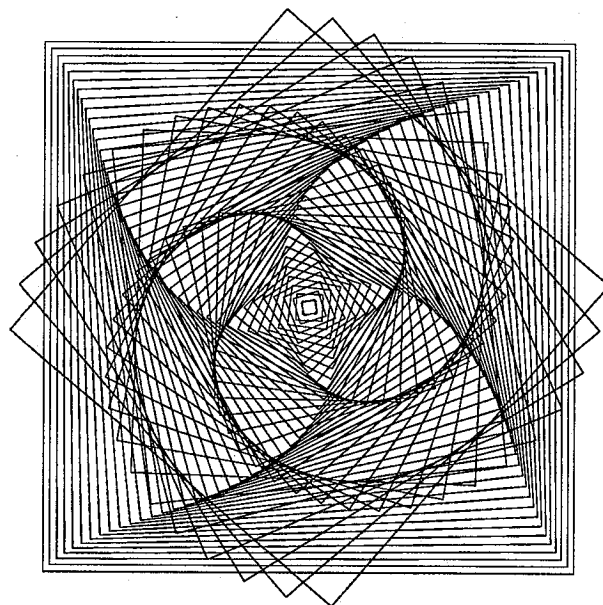
*Nuther Big PostPrint Test*



**FORBES vs FULLER**

# **Open Debate**

**CIVIC AUDITORIUM, FRIDAY, MAY 24, 7:30p.m.**



BOXED In  
*BOXED In*

**Merry Christmas  
and Happy New Year**

**PostPrint**

*The Commodore of the 80's  
is equipped with only CBM  
peripherals. It fails at the Wall  
because of how it was made.*

*The Commodore of the 90's  
includes CMD peripherals,  
providing those things which  
CBM intentionally denied us.*

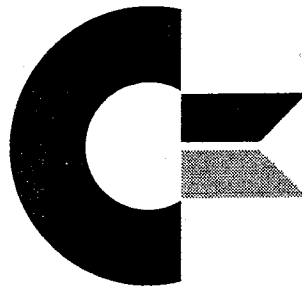
# SUNRISE

# WALL



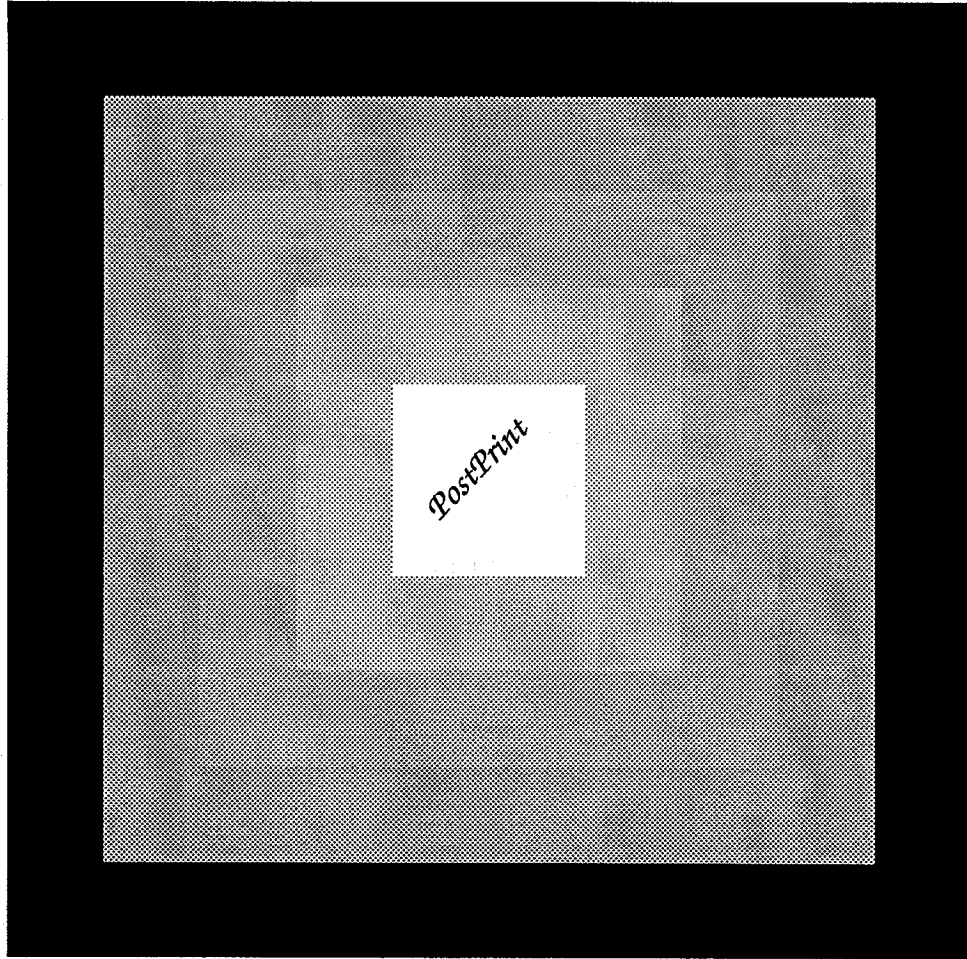
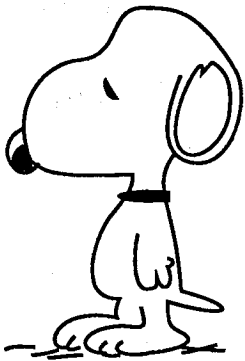
*PostScript Printing from the Commodore*

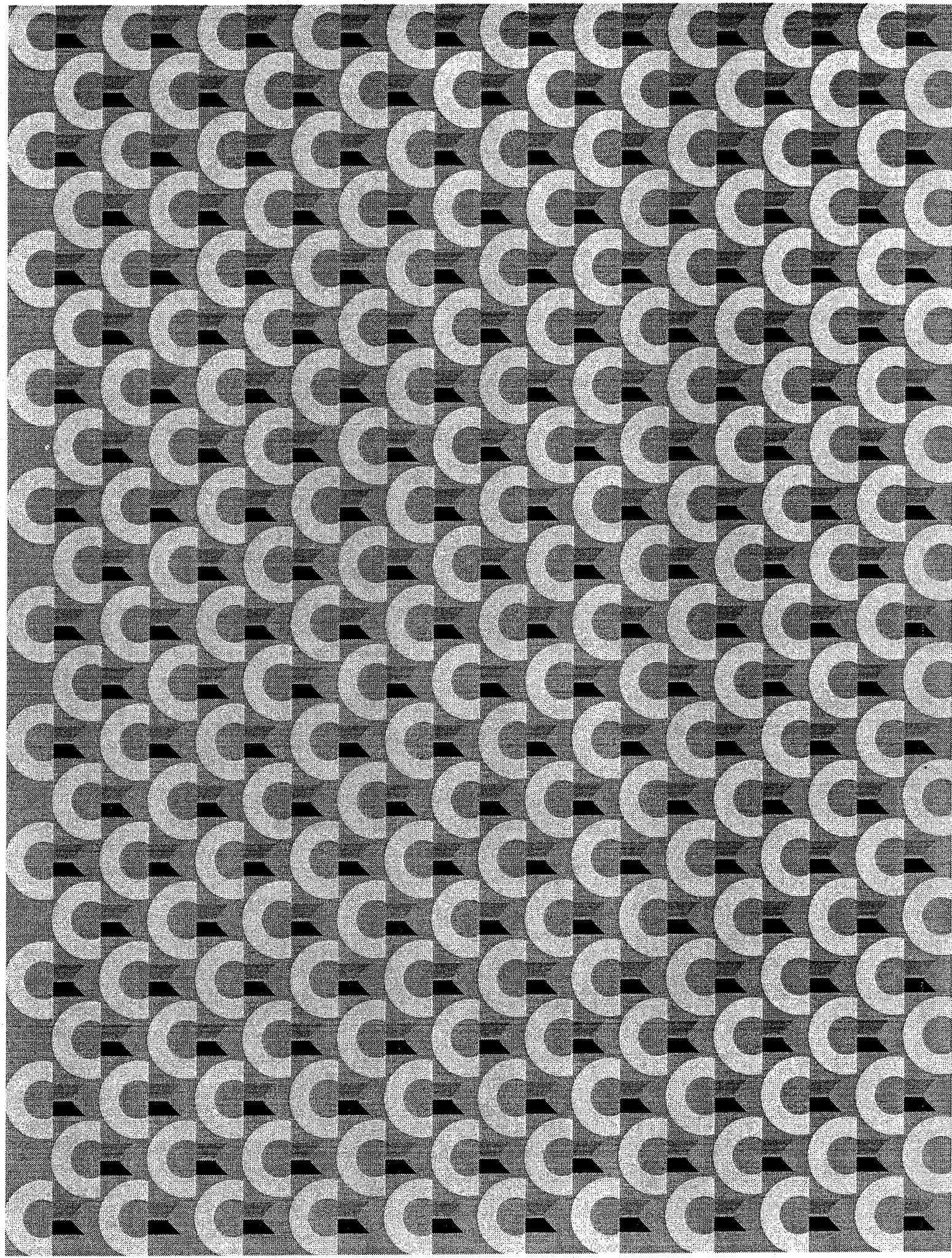
**Fade Away!**



*PostScript Printing from the Commodore*

**Fade Away!**







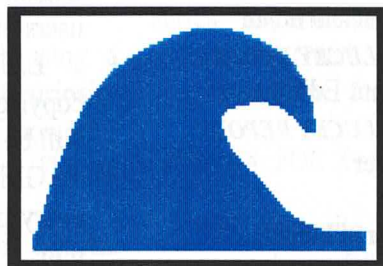
AUGUST  
ISSUE

# LUCKY REPORT

2000

## Maurice Randall and

## The



## WAVE

have re-ignited the  
**BURNING QUESTION**  
that's Smoldered  
for a Decade!

# WHAT

# IS

# A

# Commodore?

*This cover was totally created on and printed from a Commodore computer. At least, it is to me. Will others agree?*

K. Dale Sidebottom Editor