Laser-Lovers Disk

by K. Dale Sidebottom / Version 0.8

I had a dream...

When I bought my first PostScript printer back in 1991, I had a dream that *PostScript Printing From The Commodore* (PSPFTC) would soon be available to everyone. It thrilled me to think that one day a large number of Commodore enthusiasts would use laser printing to create the finest print documents a Commodore can produce! Instead, what I experienced was years of frustration, reading articles that declared how this or that printer was "as good as a laser." I wanted to protest, but I knew the real problem was the COST! I had paid $1500 for my Hewlett-Packard LaserJet IIP in 1991. How many Commodore users wanted *PostScript Printing From The Commodore* bad enough to pay that kind of price?

Seven years have past and tremendous changes have occurred. As new laser printers beckon the buying public, high-quality used PostScript printers become more and more affordable. This year Maurice Randall purchased the same printer I bought in 1991 for around $100. Willis Patten, who edits the **GEOS Publication**, the only GEOS-specific magazine in America, set himself up with a laser printer for less than $200. These printers required some repair, but this time and energy was well rewarded.

CMD is expecting to have refurbished laser printers in good condition for sale soon for as little as $429, including geoCable. As these printers come with a 90 day warranty, they are a real bargain. You could easily pay around $400 just for a new PostScript cartridge. Adobe Systems makes their money by charging a premium for cartridges which carry their product. To be able to buy a "guaranteed" laser printer for approximately the cost of the PostScript cartridge alone is a good deal.

Increasingly, I am hearing from Commodore users who want to know what this "laser printing stuff" is all about. Perhaps dreams *can* come true!

Once upon a time...

Every story has a beginning. This story began when I asked Maurice Randall to write a "simple" program for me. I wanted something that could send PostScript files to the printer directly from a geoWrite file. He agreed to help me, but it was a little like a musician asking Stradivarius to make a violin. You know that when it is finished, the work will be exquisite. On the other hand, you may be waiting awhile.

Two years later, Maurice sent me a program called *PostPrint*. After modifying it slightly to include some of my ideas, he suggested that I market it. Thus it became the catalyst for the LASER-LOVERS Disk. I thank Maurice greatly for his support and encouragement, without which this disk might be merely a fairy tale.

A Masterpiece in progress...

I like to think of my LASER-LOVERS disk as a "Masterpiece in progress." There may be uncertainty as to whether or not it will ever be a "masterpiece," but there can be no question about the fact that it is still a work "in progress." You may be concerned that it is designated as 'Version 0.8.' Is it a *beta-test* version? No! Except for *TOOLKIT* which is new to me and *PostPrint* which is new to everybody, the files on this disk have been used successfully for seven years!

While I know the programs work, I am not so confident of the presentation. Do I introduce the material in the clearest, most meaningful manner? Before I upgrade this disk to Version 1, what should I change and improve to make PSPFTC as accessible to Commodore users as possible?

Unavoidably, you find yourself somewhat in the role of 'guinea pig.' To compensate for this, I would like to reward you in TWO ways!

1. I eventually expect the LASER-LOVERS Disk to become a two-disk set (or a 1.6 MB HD disk), selling for $35-$40. As an early purchaser, you will be able to receive the second disk for only ~~ONE~~ DOLLAR$ to cover my out-of-pocket expense. This guarantees that, for very little cost, you can enjoy all the improvements which you may have helped to develop by your participation.

(Sorry, This will never happen. Dale)

2. For a reasonable period of time, I offer you my services for FREE as a Commodore PostScript consultant. How much that might be worth is frankly unknown, but it means you can call me anytime between 6:00 p.m. and midnight (EST) to ask questions or make suggestions. If I am not home, leave a message. I will call you back when I can talk, and then you can call me back on your "nickel." You can also email me at <luckykds@iglou.com> or mail me at P. O. Box 303, New Albany IN 47151-0303.

If no one has problems, I shall be shocked and forced to call you on my "nickel." The reason, of course, is that I need your feedback to tailor the upcoming Version 1 so that it best benefits future laser lovers. Thank you for purchasing this disk and for helping me to make progress on my "masterpiece."

Avast, ye Mateys!

Were you a "pirate," the follow section wouldn't make any sense. I hope you're not and that you want to know which programs on this disk may be shared with friends and which, legally, must be reserved for your own private use. It is important to distinguish between the two.

There are ten copyrighted files which are reserved solely for your use.

PSPFTC Booklet   This is the name of a booklet I wrote for the 1995 Commodore EXPO, which has been substantially revised for 1998. The acronym stands for PostScript Printing From The Commodore (pronounced /Pa SPiF'TiC/ or just name the letters). The booklet seeks to introduce Commodore users to PostScript printing. It discusses where we have been and what we need to do to succeed in the magical world of PSPFTC.

PSPFTC pp.7-11   This is a separate geoWrite file which duplicates pages 7-11 of the booklet. You can make text scraps of PostScript code to paste in your *'example'* file.

PostPrint   This program by Maurice Randall will download PostScript code to a PostScript-compatible printer connected in serial or parallel to your Commodore 64 or 128.

WrongIsWrite81   I call this program *WW81,* for short. I purchased it with Joe Buckley's "Storm Systems" disk many years ago. If you own this program legally, you bought Buckley's "Storm Systems" disk or this one.

This program can magically transform any text file from true ascii to pet ascii to any of three geoWrite formats. It can also change the font over an entire document! It is fast, flexible, and user-friendly, and I value Joe's kindness for allowing me to include it here. I believe you will fall in love with *WW81!*

geoPublasr 1.8_GC
geoLaser.GC
geoPublsrDiskA
geoPublsrDiskB
geoLasrDiskA
geoLasrDiskB   The original versions of these programs are copyrighted by Berkeley (now GeoWorks). They were released to the public on Q-Link. However, Jim Collette wrote a program called *PS.Patch* which is required to convert them to the versions on this disk. Jim's patch program is one of many great programs on his "Collette's Utilities Disk" which CMD now owns and copyrights. I would like to explain why they are here. In the past, some have complained that they would run the patch program and their new drivers would not work. These driver-applications have been prepared with special icons to distinguish them. The problem with *geoPublaserDisk* is that you must use it on the same drive, A or B, as the one on which it was first created. Be careful not to get them confused!

The remaining programs and files may be shared, to the best of my knowledge. *If I find out that I am wrong, I will pull the offending file from this disk as soon as I discover the conflict.* Some files are copyrighted, but have been placed in the public domain. Others are copyrighted by Adobe Systems, Inc. However, you have permission to use their copyrighted material if you are using their products. If you are using a PostScript printer, it is generally assumed that the PostScript interpreter within the printer was licensed through them, and you are legally using their product.

Laser Fonts   These are the LW_fonts created and copyrighted by Berkeley and released to us on Q-Link. The new icons and fontnames were designed by Joe Thomas, author of *GetItWrite*. He also added some smaller point sizes. I added the 11 point size for Times-Roman and Helvetica. (Sorry, but Helvetica still needs help on spacing.)

PS.ErrorHandler   This program from Adobe Systems has been placed in the public domain. It will catch PostScript errors (except in font construction) and print out a report.

GP.Prologue   This is the prologue that *geoPublaser* sends before it downloads your original data. It contains all the defintions and terms which *geoPublish* will need to manipulate your data. The prologue is normally a page and a half, but I reformatted it to five pages in order to make it easier to read and understand. Never use the original! Instead, always work from a copy. That way you will not inject any errors into it.

Tymes-Elfin   This PostScript font is special in that it represents a user-defined font. Hopefully, the Version 1 upgrade will have many more PostScript fonts but there isn't much room on this disk for additional 80+ kilobyte font files.

TOOLKIT   This program can append files and establish new rulers throughout a
TOOLKIT 128   geoWrite document.

PostScript Code Samples   These are samples of PostScript files which may have many purposes. The "text in a circle" coding originally comes from the PostScript Tutorial and Cookbook (pp. 163-165) which is copyrighted by Adobe Systems, Inc. From it, I created a condensed version which can be transferred as a single text scrap. Other examples were written by me and show how I have created special objects in PS code to enhance the newsletter I edit, the ***LUCKY REPORT***. Some examples were downloaded from the internet by Randy Winchester or Roger Lawhorn.
   Most of these may be printed simply by using the *PostPrint* application.

Thinking in PostScript...
   I fear some of you may be disappointed, thinking, "It's great to see many new possibilities for using PostScript, but what have I learned today that will revolutionize my printouts tomorrow?"
   To answer this criticism, I ask you to consider that I have over 50,000 PostScript graphic files and over 3000 fonts (on CD's). These PostScript resources represent nearly fifty times the graphics and fonts available to me within the normal GEOS environment. Yet their total cost, new and used, was less than $50! My point is that PostScript is so popular that its programs are everywhere. The problem isn't how you are going to find resources; rather how will you incorporate them into your documents after aquiring them.
   In order to do this, you have to be able to "think" in PostScript. You need to be willing to give yourself a little time to make this adjustment. Begin by printing out the ***PSPFTC*** booklet, reading it and trying out the examples. If you have problems, call me at 1-812-944-9132. This must print out!
   Just open *PostPrint*, and then select the *PSPFTC Booklet*. It is 20 pages long. The cover page takes about 10 minutes to process. If your printer allows it, I recommend that you print on both sides of each sheet, odd pages on the front and even pages on the back. When the job is complete, press the **Control** key and the **D** key together while still in *PostPrint* to reset the printer.
   Also I recommend that you buy Ross Smith's book, *Learning PostScript, a Visual Approach*. It is, in my opinion, the very best book ever written to explain the basics of PostScript in a way that is fun and easy to understand. If you want to order this PostScript "primer," you can call Peachpit Press, Inc. at 1-800-283-9444. If you are a member of a computer user group, be sure to tell the person taking your order as it will afford you a 20% discount. The cost, sans discount, is $22.95 plus $4.00 shipping. [IDBN 0-938151-12-6]
   Before long, you will be able to "think" in PostScript. You will be amazed at how many wonderful new things your Commodore can do, "when you care enough to print your very best!"

Weird Stuff

You will have to accept the fact that you are a kind of pioneer on the PostScript frontier. I have tried to make your journey easier, but I must still warn you that there are some things in PostScript which have not yet been completely "ironed out." For instance, Maurice Randall and I agree that for some reason, *geoPublaserDisk* can occasionally corrupt a disk or partition. Therefore, we suggest that you isolate it as much as possible. Use it in a separate disk or partition and only after you have backed up the files you cannot afford to lose.

Obviously, when I refer to *geoPublaserDisk*, I am referring to two files...*geoPublsrDiskA* and *geoPublsrDiskB*. Realize that you must select the driver-application according to the drive from which it will be used. DO NOT CONFUSE THE TWO!!! If you do, then it will try to mistakenly access the other drive and that could REALLY corrupt things.

You would expect *geoPublaserDisk* and *geoPublasr 1.8_GC* to handle a geoPublish file in exactly the same way since each is a patched version of the same program. However, I have found that it "tain't necessarily so!" I find that the *geoPublasr 1.8_GC* will allow you to "fudge" or cheat now and then whereas *geoPublaserDisk* is less forgiving and more rigid. You will have to see for yourself what you can and cannot get away with.

In my PSPFTC booklet, I ignore *geoLaserDisk* completely. My reason for doing so is that I am trying to encourage you to get the most out of PostScript. I do not feel that geoWrite alone can do that. For instance, graphics in *geoPublish* can be translated and scaled. (With the help of this disk, they can also be rotated.) Yet, *geoWrite* can only print the original graphic in the middle of the page. You would have to code in everything personally because *geoWrite* will give you no real help with any of these three functions.

If you have used the serial driver-applications, *geoPublaser* and *geoLaser*, then you know that the first dialogue box you see allows you to adjust the baud rate. When Jim Collette developed his *PS.Patch* program, he tried to suppress this "baud rate box" because it was useless for any of the patched driver-applications. However, these programs may (or may not) show you a "baud rate box." If they do, just click on "OK" and ignore it.

As time goes on and the popularity of PostScript printing increases, it is hopeful that we will receive better tools with which to work. Maurice is already making substantial improvements in the *PostPrint* program, but you will have to have *WHEELS* in order to take advantage of it. I suggest that you buy *WHEELS* anyway. It will make your use of storage memory so much easier that you will wonder how you ever lived without it.

I have tried my best to give you a sampling of the best that the Commodore currently can offer in the area of PostScript printing. I strongly believe that with the combination of your support, as well as that of others, more Commodore programmers will seek to bring us improved products. I thank you for the part you will play in encouraging that effort.

K. Dale Sidebottom

Address: P. O. Box 303, New Albany IN 47151-0303
Email: luckykds@iglou.com

POSTSCRIPT PRINTING FROM THE COMMODORE

PSPFTC

by K. Dale Sidebottom

A BRIEF HISTORY OF
AND AN EXPLORATION INTO
POSTSCRIPT PRINTING FROM THE COMMODORE

About the Author

My father bought my first computer for me when I was 33 years old. He wanted to prevent "brain decay." Mine! I had recently been laid off from a soft desk job and, desparate to support my family, became a laborer on an assembly line for a soft drink bottler. The job required that I "fill the line" by throwing cases of empty soft drink bottles on the assembly line at a rate of approximately 500 per hour, 10 hours per day, six days a week. When I got home each day, I "died." So my thoughtful father tried to stave off "brain rot" by giving me a Timex-Sinclair computer for Christmas, and I deeply appreciated it!

Perhaps you remember the Timex-Sinclair was advertised as the first computer to be sold for under $100. It was expandable to 16K and required a tape recorder to store programs. I soon outgrew it. In fact, the very first basic program I wrote maxed out the memory. I realized that I needed to upgrade to a computer with more memory and a disk drive. I bought a Commodore.

That was back in 1983, and I have been a Commodore enthusiast ever since. Dad is glad that my brain survives, but I sometimes wonder if he wishes that I had moved to a 16-, 32-, or 64-bit computer somewhere along the line. Today I am a postal carrier and could probably afford it. But I prefer to keep my Commodore as long as it continues to do everything that I want to do with a computer! Instead, I toss my "mad money" into accessories that boost my Commodore's performance. I 'invest' in things like the HandyScanner 64, a RAMLink, a CMD HD converted to a ZIP drive, and (of course) my *LASER PRINTER*!

To learn more about PostScript, I paid $195 in 1994 to join the Adobe Developers Association. I may be the only strictly-Commodore user ever to join the ADA. I left in 1996 when they doubled the dues. However, the technical assistance I received was quite helpful, and I am available for consultation *to Commodore users only!*

Today, most of my computing time and effort is spent studying and promoting PSPFTC (PostScript Printing from the Commodore).

K. Dale Sidebottom

Foreword

PostScript printing from the Commodore is a dream come true for many Commodore enthusiasts. However, the high cost of hardware and the low availability of software have combined to lock out most of us for many years. This booklet is written with the purpose of giving information and encouragement to those interested in PSPFTC! It is also intended to introduce specific knowledge enabling you to more easily master this exciting and powerful technology.

PostScript was created to be an invisible layer of software between the computer and the printer. However, for those of us who love to get the best from our Commodores, particularly in the area of desktop publishing, we need to strip away that "cloak of invisibility" and take control of our PostScript documents. After all, *geoPublish* is over ten years old. By learning just a few simple tricks, we can incorporate some fantastic features that may seem unbelievable! Imagine thousands of exciting PostScript fonts becoming available for *geoPublish*. Envision PS "photo scraps" (normally called EPS files) available in 16 million colors! No, not all the bugs have been worked out yet, but I am confident that all this will be accomplished within the near future.

Please keep in mind that while I constantly speak of printing with a LASER, what I actually mean is *printing from any printer equipped with a PostScript interpreter* (usually on a plug-in board or cartridge) which will enable that printer to decode PostScript's page description language. I actually now own two such printers, a Hewlett-Packard LaserJet IIP and the HP CopyJet M. The first is a PS Level 1, black-and-white laser printer which I have used from my Commodore since 1991. The second is a recent purchase which allows me to translate Level 2 PS files and even print in color. It is not a *laser* printer at all, but rather uses an ink-jet mechanism. It doesn't matter how your printer prints, but since the majority of PostScript printers are 'laser' printers, I will continue to use that term as a convenience.

This booklet will give you a short history of how we got to where we are now. It also honors those who have "paved the way" for all of us. Last of all, I want to give you an overview of 'PostScript, The Language' so that you can take control of your printed documents in a way that you never thought possible.

Good Luck!

Dedication

My version of the Universal Law of Computers goes like this: **Given proper time, program and peripherals, any computer can accomplish any computer task.** Consider what this means to those of us who use Commodores. If we allow ourselves to brush away the constraints of time, made more palatable by the introduction of the SuperCPU, we can do *anything* on a Commodore if we have the right hardware and software!

Creative Micro Designs (CMD) has worked wonders to provide us with the hardware we need. In fact, there are only two peripherals I miss having, a color scanner and a CD-ROM drive. Actually, someone in Europe is currently working on a way for us to access a CD-ROM drive. Because our Commodores can access SCSI devices, anything is possible!

Software is our biggest challenge. The multi-billion dollar company that wrote the C64's original operating system no longer considers us to be a commercially viable market. We've been abandoned and forgotten like a jilted lover. We must therefore write our own software. Fortunately, within the Commodore community, we can find some brilliant programmers. Their products enable us to do far more with these 8-bit machines than their creators ever dreamed possible!

This booklet is dedicated to all the users who give this publication purpose, to the company (CMD) which provides the hardware we need to stay current, and to the programmers whose tireless genius expands the productivity of this platform beyond expectation. Thank you all!

Our "Fairly" Godmother Brings the Laser Home!

Jeanine Cutler deserves a lot of credit. Up until her article appeared in the January 1990 issue of *geoWorld*, PostScript printing with a Commodore required us to send files via modem to another computer. Another possible option was to save the file to disk and use *Big Blue Reader* to transfer it to an IBM formatted disk. Thus, the files could be transferred to another computer for printing. In other words, our Commodores could create PostScript files using GEOS but could not directly print them! Admittedly, if you had an $8000 Apple LaserWriter and an RS-232 interface, you could do it at home, but I never heard of anyone doing that. The title of this booklet, *PostScript Printing from the Commodore*, never really become a reality until Jeanine made it happen!

Working alone, she made a large investment of time and money to find out how a Commodore computer could work one-on-one with a laser printer. The results of her research appeared in the above article entitled "An Affordable Laser Printer." She explained how GEOS users could now do PostScript laser printing right in their own homes! The price tag? Only about *$1500*. Compared to the cost of an Apple LaserWriter, it was a bargain!

Okay, so it's still a lot of money, but several Commodore users were willing to pay it, including me. Soon I was publishing newsletters that impressed all my friends! My user group loved the results, and I found myself falling in love with my "Laser."

I want to interject here that, today, we take all this for granted. I know many people who have purchased one of the newer laser printers, hooked it up to a Commodore, and started printing immediately. Hey, no big deal! Let me assure you that in 1990 Commodore users were hesitant to invest thousands of dollars just to *investigate the possibility* of laser printing. Jeanine gave us the confidence and the knowledge we needed to enter this new arena, and we today still benefit from her efforts.

Her contributions did not stop there. Because of her article in *geoWorld*, she soon became a "lightning rod" for all the Commodore users who were interested in laser printing. She even received a letter from a Dan Frey in Europe who claimed to be commercially laser printing in Europe with a parallel version of *geoPublaser*. Apparently he had written letters to a programmer in America who had sent him several versions until one of them worked. Through Jeanine's efforts, we finally discovered that this programmer was Jim Collette!

Since Jim had sent several versions to Dan Frey in Europe, he was not certain which one of them actually worked! He asked Jeanine to send him a copy from Mr. Frey so he could more easily recreate the patch for it, which she did immediately. Due in large part to her prodding, the laser lovers of America now have serial and parallel versions of *geoPublaser* (and *geoLaser*) available.

Even that was not enough for her. Jeanine wanted *geoPublish* to become more flexible. She wanted to be able to use *geoPublish* to make pamphlets printed on 8 1/2" by 14 " paper and even do "landscape" documents. So she bought a book for Jim called *Learning PostScript: a Visual Approach* by Ross Smith. She sent it to him and urged him to read it. The result was his program called *PS.Processor*.

Jeanine also nudged me in the right direction. I was becoming unhappy with the limitations of *geoPublish*. I saw so many wonderful things in magazines and such that appeared to have been laser printed. I kept wondering, "Why can't my Commodore do that?"

She reminded me of the book she had sent to Jim Collette and suggested that I order it. She assured me that it would make PostScript much easier to understand, and she was right! I highly recommend that you buy Ross Smith's book, *Learning PostScript, a Visual Approach*, which is available in local bookstores and through AMAZON.COM on the internet, if you have not already done so.

The GeoWizard

Jim Collette is called the "GEOWIZARD," not only because he created a program by that name, but also because he possessed a magical talent for doing what seemed impossible to everyone else. For instance, back in June of 1988, *geoWorld* magazine published a column by Randy Winchester which discussed how easy it was to locate IBM PC's hooked up to Apple LaserWriters in local print shops. A subscriber's letter further explained that everyone knew *Big Blue Reader* could store Commodore ASCII files to an IBM formatted disk. Therefore, PostScript output from GEOS would be easy to laser print if we just had a *geoLaser/Publaser* version that could write PostScript files to disk.

Randy's response was very optimistic. He declared that "Since some of the best programming minds in the world read this column, you can probably expect to see a patch file from one of them on Q-Link any day now!"

A year would pass before Randy Winchester's column could declare victory. The problem which Randy had, perhaps whimsically, presented to the "best programming minds in the world" was at last resolved by a wiz kid in junior high! Jim Collette had agreed to publish his junior high school newsletter. He liked using *geoPublish* but realized that he needed to store the PostScript files to disk if he were going to have his dad laser print them at work. So he "fiddled" around with *geoPublaser* until he got it to do the job he wanted done.

From there, Jim Collette created a program called *PS.Patch* which would alter *geoPublaser* or *geoLaser* to make them send their PostScript files to disk. If you use *PS.Patch* to change *geoPublaser* or *geoLaser*, be sure to rename the file because *PS.Patch* does not rename the file for you! Also, it is important to take note of which drive you created them on, because this is the drive on which they will always attempt to store PostScript encoded files. (To avoid these hassles, I have already placed these files on your LASER-LOVERS disk.)

The next year, he expanded his program to create a parallel driver, as well, and called it *PS.Patch 2.0*. This was important because almost all laser printers made today offer parallel connections. If you have a dot matrix printer that uses a printer interface or a geocable, then you can connect your laser printer to your computer in exactly the same way, if it has a parallel port. If you use a printer interface, use *geoPublaser* or *geoLaser*. If you use a geocable, use Jim's *PS.Patch 2.0* to create parallel drivers and rename them using a *.GC* for geocable. Although Jim's program will allow you to create a parallel driver for either version of *geoPublish*, I anticipate that you will want to get the version that fixed the earlier bugs. Therefore, I only support the 1988 version of *geoPublish*. (The parallel driver-applications, *geoPublsr 1.8_GC* and *geoLaser.GC*, are on your disk.)

I mentioned that he wrote *PS.Processor*. Did I also mention that *PS.Processor* was the only PostScript program written for the Commodore in a seven year period, extending from 1987 through 1994?

PS.Processor is what I would call a "niche" program. It fits a very small niche among laser users, but if you need it, it does some amazing things. First, it allows you to print geoPublish pages in any order. This is vital if you are making a booklet or pamphlet and need to reorder your pages. You might want to make a poster with the long side down, which is called "landscape" mode (as opposed to "portrait" mode). *PS.Processor* allows you to turn your design 90 degrees or upside down. Unfortunately, *PS.Processor* requires a "script file," which may be difficult for the novice to write; so be willing to devote a little time if you are just starting to use it.

In order to use *PS.Processor*, you will need to use *geoPublasrDisk* which sends the geoPublish file to disk as mentioned earlier. This is the only way that *PS.Processor* can manipulate your data. I highly recommend that you buy *PS.Processor*, now available on the "Collette's Utilities" disk from CMD, if you are interested in PSPFTC.

Today Jim is no longer programming for us. He is now programming for Brian Dougherty, who was CEO at Berkeley Softworks when GEOS was introduced but is now a "bigwig" at Wink developing interactive television. We wish them well!

Two Important Reversals

It is easier to adjust to PostScript if you can master the two fundamental reversals between the Commodore and PostScript environments. It may help you to understand them if you realize that these conflicts arise out of differences between conventional usage and the scientific model. For instance, since the typewriter was invented, we have measured the vertical distance on the printed page starting from the top down. Our Commodores continued in this tradition. In GEOS particularly, we have always measured from the top and accepted this as a normal practice.

However, the scientific model is based on the Cartesian plane. There we can locate any point on the plane (or the page) by two coordinates measured along an X and a Y axis. The intersection of the X and Y axes is called the *origin* (0,0). In order to place the printed page where all values for X and Y are positive, PostScript places the page in the upper-right quadrant. Thus the lower-left corner of the paper is the origin (0,0) and all values for Y are measured from the bottom of the page going up. Think about it...do you measure your height from the floor or the ceiling? Thus, the PostScript model parallels science. Despite this, it is easy to get confused when you are accustomed to conventional usage.
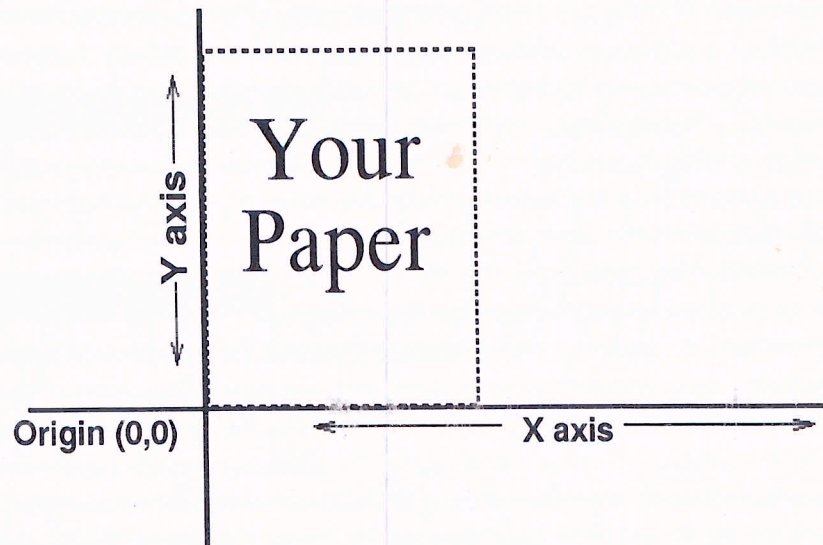
The second great reversal affects graphic interpretation. We know that we create graphics on the Commodore by printing out bit-mapped designs. The bit-map is composed of ones and zeros. Wherever there is no color placed on the page, we place a value of zero. When we want to place ink (color) on the page, we insert a value of one. Therefore, in conventional usage, we use a zero (0) to represent areas of white and a one (1) to signify the black areas. White equals O and black equals 1. Simple, yes?

PostScript is completely the opposite! Remember in science class where you learned that black was the absence of all light? How do you numerically represent nothing? With a zero. White is the presence of all light. Since white contains the whole spectrum of light, it is represented by a value of one.

This distinction is important because PostScript often assesses a 'gray' value to **stroke** or **fill** in an object. This is done with the **setgray** command preceeded by a number between zero and one. If you see **.3 setgray**, remember this will be nearly black, and **.75 setgray** will be closer to white. If you use the **1 setgray** command to outline a design on white paper, you'll never see it!

Understanding these two reversals is vital to working comfortably in PostScript. Continue to you remind yourself that PostScript follows the scientific model, and I believe these two reversals will no longer be a problem.

PostScript, the Language

You have heard it said that a picture is worth a thousand words. PostScript proves the reverse...that a thousand words, more or less, is worth a picture. I believe that it is the greatest "page description language" in the world. It enables you to design with words almost anything that can be pictured on a two dimensional plane.

PostScript is a great language for Commodore because it is a straight-forward 8-bit language much like Basic. As such, its programs can be very simple or extremely complex. I would like to demonstrate for you its simplicity and challenge you to begin using PostScipt to enhance your geoPublish programs.

In order for you to prove the examples here, you must have *geoWrite* and *PostPrint*. Simply place both applications on a GEOS work disk (or in a work partition). Then create an empty geoWrite file called '*example*.' Now you are ready to begin.

Type the three words in bold typeface below into your now empty geoWrite file called '*example*.' Or if you are as lazy as I am, just make a text scrap of this program and paste it into the geoWrite document. Next open *PostPrint* and send this '*example*' to the laser printer. Then hit the 'Control' and 'D' keys together to tell the printer that this job is complete. As a precaution, you might want to change the font style from bold to plain. The reason that you should change the font style to plain is that the ascii value that GEOS uses to signal **bold** print may become a "@" sign in PostScript. If your first attempt to print the example fails, change to 'plain' text and try it again.

clippath stroke showpage

This is an important exercise because it tells you how close to the edge of the page your particular printer will print. It *clips* a *path* along the borders of a page illustrating the furthest reach of your printer. My LaserJet prints to within 7/32" of the bottom and sides, but it has no limits on the top. The reason for this is that I am using 8 1/2" by 11" paper now, but my printer is designed to use sizes up to 8 1/2" by 14". Therefore, the upper limit is not really being tested.

Hopefully, your printer will print to within a quarter of an inch of all edges. Both *geoWrite* and *geoPublish* assume that it will. Some Commodore users have had difficultly when their printers failed this standard. Let's test it again by altering the '*example*.' The zero before the decimal is optional. Change to 'plain' text and press Control-D afterward.

clippath 0.9 setgray fill showpage

Now let's combine these two programs into one result. Alter your *example* file to match the following code:

clippath 0.9 setgray fill clippath 0 setgray stroke showpage

Notice that it is not enough to draw a path. You must also tell PostScript what to do with it! For instance, do we *stroke* or *fill*? We might simply *clip* around the edges which would affect where future designs could appear.

Notice that PostScript is a stack-based language, similar to Forth. When programming the Commodore, we are told not to leave anything on the stack longer than necessary because it may be lost. But PostScript has different stacks for different purposes. The integrity of the stack is assured, and a PostScript programmer must "live or die" by it. This is why we must place the **setgray** command after its value.

We do not even have to place the command immediately after the value. In PostScript, we often leave values on the stack to be used much later. It looks ridiculous, but you can accomplish the same result by placing the setgray values up front. Try this and see if it works!

0 0.9 clippath setgray fill   % Zero before decimal is optional
clippath setgray stroke showpage

Please notice that the percent sign in PostScript is used like the REM command in Basic. PostScript will ignore everything between the percent sign and the next carriage return.

We also see that some PostScript commands require elements from the stack and others do not. As long as the commands that require values or objects from the stack can find them, everything runs like clockwork. But what happens if there is nothing there, or if it is the wrong thing? Then PostScript crashes and you will likely get nothing.

The problem with PostScript is that it has no built-in system for reporting errors. In other words, if you left out one 'p' in the *clippath* command, the page would fail to print and you might not know why.

Fortunately, we can download our own error-handler. I received from Adobe Systems a small public domain program renamed *PS.ErrorHandler*. Using *PostPrint*, you should send this to your printer followed by a Control-D. It will remain active until you reset your printer.

To test it, erase one 'p' from *clippath* and send the 'example' again. You should get an error message saying...

ERROR: undefined

OFFENDING COMMAND: clipath

In other words, 'clipath' is a term that has not been defined. Until you define the term (or correct your spelling), PostScript will send you nothing!

One advantage of PostScript is that it will allow you to save the *graphic state*. If you draw a path and stroke it, it disappears from memory. But if you save it first, then you can use the same 'path' over and over again. To store it in memory, first use the **gsave** (save the graphic state) command. When you want it to be restored, follow with the **grestore** (restore the graphic state) command. Again we use the same example to illustrate. (Notice how the *gsave* command saves the path <u>and</u> the default *setgray* value.)

clippath gsave .9 setgray fill grestore stroke showpage

Now that you have outlined the print-limits of your printer (several times ;-) and discovered the value of a basic error-handler, let's print something. PostScript does not use any *print* command to put anything on a page. Instead, we must use the **show** command. Notice that **showpage** means *show (or print) the page* in PostScript. The **show** command similarly means *show (or print) the text* on the page. (Remember to place your name inside while *keeping the parentheses*.)

/Times-Roman findfont 50 scalefont setfont

72 576 moveto (Insert Your Name) show showpage

If your name goes off the page, change the number '50' to a smaller figure until your full name stays on the page. (By the way, you must have a really long name. ;-) Then substitute this new value before *scalefont* in future examples.

PostScript uses the parentheses to set off or define a *string*. In the PostScript language, only strings can be printed. In Commodore Basic, we would use quotation marks, but here we must use parentheses to enclose any text to be printed on the page.

Another curiosity is worthy of note. Up to this point, we have used spaces and carriage returns as delimiters or separators. When we put them inside of parentheses, the spaces will be printed, but the carriage returns will be ignored. Try this example. Make sure you leave a space between **FirstName** and the carriage return on line 2.)

/Times-Roman findfont 50 scalefont setfont

72 576 moveto (FirstName

LastName) show showpage

After the initial *moveto* command, PostScript can "automatically" determine where to place letters in a line of text. However, it has NO facility to format multiple lines of text. If you have 50 lines of text on a page, then you must use 50 '*moveto*' commands to initially set the placement of each line.

A PostScript file is sent to your printer in a data stream. The PostScript interpreter must handle this data as soon as it receives it. Everything must be either stored or executed immediately. In order for the interpreter to know whether to store or execute various text objects, we place a diagonal line or 'slash' in front of those we do not want to be immediately executed.

Notice that the name of the font is always preceeded by a slash (/). No space is allowed in between. Try using the fontname (Times-Roman) without the slash and see what happens. (Make sure your PS error-handler is in place.)

We also use the slash when we want to define terms. After the terms are defined, then you can use them without the slashmark. The example on the next page shows how to define an inch (in.) so it can be used to place text in PostScript.

```
/Times-Roman findfont 50 scalefont setfont
/in. {72 mul} def
1 in. 8 in. moveto (FirstName LastName) show
showpage
```

If you run the above program without the curly braces, you will get a typecheck error. (If you are using GEOS, the curly braces are formed by holding down the Commodore key and pressing the colon or semicolon key.) These curly braces are vital to PostScript because they tell the PostScript interpreter to save everything inside for later. The term 'in.' is stored as a *key* in the current dictionary and '72 mul' is stored as its *value*. Thereafter, whenever the term 'in.' is used in the program, the value '72 mul' is substituted on the stack and executed. Thus the value of **1 in.** becomes **1 72 mul** (or 72). The value of **8 in.** becomes **8 72 mul** (or 576). Therefore, our last two examples will print your name in exactly the same place.

The tricky thing about doing math on your laser printer is that the answer is stored on the top of the stack, but it will not automatically print out the answer for you. Let's experiment and see if we cannot get it to solve a few simple math problems.

```
/Times-Roman findfont 50 scalefont setfont
/in. {72 mul} def  1 in. 8 in. moveto
/str 3 string def  8 72 mul str cvs show
showpage
```

Above we defined 'str' as a 3-element string. Later, after multiplying 8 times 72, we know that value of 576 is on top of the stack. As 576 has only three number places, it will fit nicely into our 3-element string. Since the value '576' is on top of the stack, we place the now defined *string* on top of it and call for the **cvs** command. It will convert the value on the stack into a string and store it as a substring in 'str'. It will also leave a copy of the substring on the stack. Since it is now a string, we can print it with the *show* operator. Wasn't that easy! Now test all of these operators.

```
/Times-Roman findfont 50 scalefont setfont
/in. {72 mul} def  /str 3 string def
1 in. 8 in. moveto    8 72 mul str cvs show
1 in. 7 in. moveto    72 8 div str cvs show
1 in. 6 in. moveto    8 72 add str cvs show
1 in. 5 in. moveto    72 8 sub str cvs show
showpage
```

PostScript offers an exhange (*exch*) command that will take the first two objects on the stack and reverse their order. For instance, we might make a procedure to simplify establishing fonts. This is important if we plan to use several fonts in a document. Notice how tabs can be used to clarify, without changing, the PostScript code.

```
/Font {exch findfont exch scalefont setfont}def
/in. {72 mul} def
/Times-Roman 50 Font                    1 in. 8 in. moveto    (Insert Your Name) show
/Helvetica 50 Font                      1 in. 7 in. moveto    (Insert Your Name) show
/Courier 40 Font                        1 in. 6 in. moveto    (Insert Your Name) show
/ZapfChancery-MediumItalic 50 Font      1 in. 5 in. moveto    (Insert Your Name) show
showpage
```

In this example, we place the font and the font size on the stack in normal order. However, the procedure requires the name of the font first, so we switch them with the **exch** command. The findfont operator finds the proper font in the font dictionary and places that pointer on the stack. To get the font size, we must switch them (**exch**) again. By creating the '*Font*' procedure, we eliminated a lot of redundant coding.

We can also use PostScript to draw a box as in the example below.

% This will draw a one inch square box. What happens if you replace 'stroke' with 'fill'?
200 300 moveto 272 300 lineto 272 372 lineto
200 372 lineto closepath stroke showpage

The example below is much easier. It makes use of the command, **rlineto**. This means it will draw a *line* in *relation* to the current point. See how much easier it is to draw this box.

% This will also draw a one inch square box in the same location
200 300 moveto 72 0 rlineto 0 72 rlineto
-72 0 rlineto closepath stroke showpage

Now we can express this design as a single procedure which can be used again and again. We will use a variable called 'side' which will be used to determine the size of the box. Check out and try the procedure below.

% This procedure will draw a square box any size.
/Box {/side exch def side 0 rlineto 0 side rlineto side neg 0 rlineto closepath stroke} def
/in. {72 mul} def
1 in. 5 in. moveto 5 in. Box
2 in. 6 in. moveto 4 in. Box
3 in. 7 in. moveto 3 in. Box
4 in. 8 in. moveto 1 in. Box showpage

Now let's really strut our stuff. We will center the squares on the paper and draw them starting with the darkest diminishing to the lightest. To do this, we must use the translate command. This allows us to move any object to any position on the page.

/in. {72 mul} def %This draws centered squares
/CenterSquare {/side exch def newpath side 2 div neg dup moveto side 0 rlineto 0 side rlineto
      side neg 0 rlineto closepath fill}def
306 396 translate   5 in. CenterSquare
.7 setgray   4 in. CenterSquare
.8 setgray   3 in. CenterSquare
.9 setgray   2 in. CenterSquare
1 setgray   1 in. CenterSquare   showpage

PostScript not only allows you to 'translate' or transfer objects to any location, but it also lets you scale them to any size. Try this program. Notice that the variable 'side' has a value of 72 (one inch). The variety in the sizes of the squares is due to scaling only. Also notice that a gsave/grestore is placed around each **scale** command to keep them from accumulating.

/side 72 def   % Squares centered but scaled differently
/CenterSquare {newpath side 2 div neg dup moveto side 0 rlineto 0 side rlineto
      side neg 0 rlineto closepath fill}def
306 396 translate
gsave   5 5 scale CenterSquare grestore
gsave .7 setgray   4 4 scale CenterSquare grestore
gsave .8 setgray   3 3 scale CenterSquare grestore
gsave .9 setgray   2 2 scale CenterSquare grestore
gsave 1 setgray   1 1 scale CenterSquare grestore   showpage

Last of all, we illustrate the **rotate** command. If you are using all three commands...translate, rotate, and scale...then you should use them in that order, as we do in the next example.

```
/side 72 def   /in. {72 mul}def          % Illustrating translate, rotate, and scale.
/DrawSquare {newpath 0 0 moveto side 0 rlineto 0 side rlineto side neg 0 rlineto closepath gsave
     fill grestore 0 setgray stroke}def
3 in. 3 in. translate
gsave                                      5 5 scale DrawSquare grestore
gsave 30 rotate      .7 setgray            4 4 scale DrawSquare grestore
gsave 60 rotate      .8 setgray            3 3 scale DrawSquare grestore
gsave 90 rotate      .9 setgray            2 2 scale DrawSquare grestore
gsave 120 rotate     1 setgray             1 1 scale DrawSquare grestore   showpage
```

I have given you a brief overview of the PostScript language. Yes, it is only the tip of the iceberg, but perhaps it will encourage you to start learning PostScript on your own. If not, you will at least have a basic knowledge of how it functions and how it can benefit you.

I wanted to close this section with two small illustrations from the book, *Learning PostScript, a Visual Approach* by Ross Smith. The first example is found on page 6-10. It illustrates the circle using the **arc** command and the for-next loop using a **repeat** command. The **eofill** command (even-odd fill) is also used. (This may take ten minutes or so to process, so please be patient.)

```
306 396 translate
2{
   24{
      0 0 moveto
      72 72 72 -72 144 0 curveto
      360 24 div rotate
   }repeat
   -1 1 scale
}repeat
eofill


0 0 150 0 360 arc stroke
0 0 160 0 360 arc stroke
3 setlinewidth  0 0 155 0 360 arc stroke
1 setgray  0 0 10 0 360 arc fill
showpage
```

The second example from Ross Smith's book is found on page 4-42. It illustrates the translate, rotate, and scale procedures in a very effective fashion.

```
612 0 translate   90 rotate
/Helvetica-Bold findfont 84 scalefont setfont
gsave 396 190 moveto 1 4 scale (Debate) show grestore
gsave .8 setgray 396 190 moveto -1 4 scale (Debate) show grestore
/Times-Bold findfont 72 scalefont setfont
54 504 moveto  .8 setgray (FORBES v) show
0 setgray (s FULLER) show
/Helvetica-Bold findfont 32 scalefont setfont
36 72 moveto .8 setgray 1 2 scale (CIVIC AUDITORIUM, FRI) show
0 setgray (DAY, MAY 24, 7:30) show
/Helvetica-Bold findfont 30 scalefont setfont
0 6 rmoveto (p.m.) show   showpage
```

The GEOS/PostScript Connection

PostScript printing from the Commodore began with *geoPublaser*. This program has an almost magical ability to pull all the various elements of a geoPublish document together and send them to the printer as a single PostScript file. Even today, after using it for seven years, I still find it amazing!

I remember when *geoPublish* was first introduced. I bought the original 1987 version, the one with all the bugs. I later upgraded to the 1988 version, which I found much more reliable. My first suggestion to anyone who wants to print PostScript is to buy *geoPublish* if you don't already have it. If you have the 1987 version, you can mail in your original disk plus $20.00 to CMD, and they will send you a geoPublish upgrade in a two-disk set.

In 1988, *geoPublaser* meant nothing to me. I was unfamiliar with desktop publishing (DTP), but I loved GEOS, especially fonts! The more the better. Some of my GEOS documents looked like ransom notes! I enjoyed uniting GEOS fonts and various graphics into one "seamless" document. Only *geoPublish* made this possible!

Being anxious to master *geoPublish*, I volunteered to be editor of my user club's newsletter. That way I would force myself to use it regularly. I had no journalistic background; so I found *geoPublish* to be challenging and complex. However, I persevered and soon published my first issue of the newsletter for the Lousiville Users of Commodore of Kentucky (LUCKY) on a dot matrix printer. The response was immediate. They said, "Thanks for trying, but we can't *read* it!"

The Write Stuff, perhaps the Commodore's best word processor, echoes these same thoughts in its documentation. It concludes that if you do not laser print your geoPublish files, you would be better off to use any regular word processor and cut-and-paste your graphics into the final product. **I agree!**

In the beginning, the inclusion of *geoPublaser* with *geoPublish* seemed like an act of kindness on the part of Berkeley Softworks. But with 20/20 hindsight, it now appears to have been a necessity. Without *geoPublaser*, geoPublish documents never see their full potential. PostScript spins a cocoon that converts geoPublish caterpillers into butterflies!

I first discovered this by reading the now deceased ***GeoWorld*** magazine. There, for the first time, Commodore users could see what PostScript laser printing could do for geoPublish documents! Not only could we see the results, but there were advertisers in there who promised to do the same for us! However, my natural proclivity for procrastination prevented me from ever using these services. What I needed was a way to laser print my newsletter at home, and I was willing to do almost anything to make that happen.

As reported earlier, Jeanine Cutler made it happen for me and many others. Encouraged by her, I have been studying PostScript for several years now, and I have been impressed with how cleverly *geoPublish* was designed with PostScript in mind.

For instance, GEOS offers several **LW** (LaserWriter) fonts which can be used in several styles--- plain, **bold**, *italic*, or ***bold italic***. Let's take the Roma font as an example. Within PostScript, this font is called Times-Roman. PostScript also offers several styles, but each 'style' is actually a different font resident within the laser printer. For instance, we have Times-Roman [plain], **Times-Bold**, *Times-Italic*, and ***Times-BoldItalic***. GEOS has a single font with four styles, while PostScript has four separate fonts which are all considered to be part of the same *font family*.

If you are shopping for a laser printer and you find one with 32 fonts, do not be fooled. You might think there are 32 unique fonts within, but more likely it has eight or more font families. Check to be sure what you are buying.

One small irritant in GEOS is that Berkeley Softworks did not name their LW_fonts to match the font families resident in PostScript printers. David Ferguson makes this point so well in his booklet, *The geoPublish Compendium*. "I wondered for a long time why the creators of GEOS couldn't call the LW_fonts by their PostScript names...To make things easier, especially if you are using ideas from desktop publishing books, rename the fonts!"

I agree. Therefore, I have renamed the LW_fonts accessible from within GEOS. The chart on the next page shows how the LASER-LOVERS disk renames these fonts to more closely match their PostScript counterparts. There are eleven of them.

GEOS Fontnames	POSTSCRIPT Fontnames	LASER-LOVERS Fontnames
LW_Bacon	Bookman-Light	Bookman
LW_Barrows	Courier	Courier
LW_Cal	Helvetica	Helvetica
LW_Cowell	Palatino-Roman	Palatino
LW_Galey	ZapfChancery-MediumItalic	Zapf Chancery
LW_Giannini	AvantGarde-Book	Avant Garde
LW_Greek	Symbol	Symbol
LW_Haviland	Helvetica-Narrow	Helvetica Narrow
LW_Piedmont	NewCenturySchlbk-Roman	N Cent School
LW_Roma	Times-Roman	Times Roman
LW_Shattuck	ZapfDingbats	Zapf Dingbats
LW_Zapf (same as Galey)	ZapfChancery-MediumItalic	Zapf Chancery

GeoPublish also allows text and graphics objects to be printed in different patterns. When you look at the program on your monitor, it appears to offer you the 32 patterns used in your geoPaint applications. Not so! PostScript can print those patterns, but it is fairly complex and *geoPublish* does not support it. What *geoPublish* will do is to substitute a suitable gray-scale. Check out the chart below for some samples. (Photo scraps print "as is" and remain unchanged.)

PostScript Gray Scales
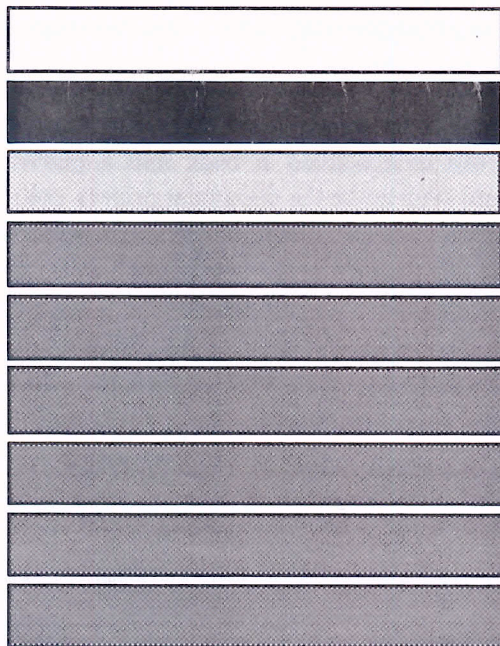
GEOS Patterns
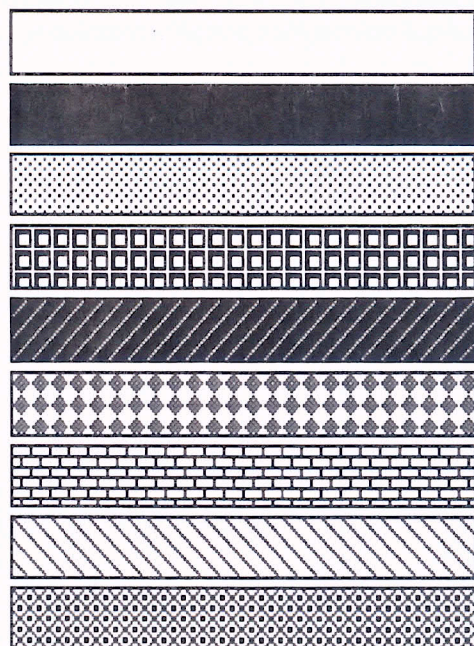
When drawing lines in *geoPublish*, the attibutes box allows you to set the line's width. PostScript does the same thing with the **setlinewidth** command. You may also determine if the ends of your lines will be round or square. The PostScript equivalent is **setlinecap**.

Thus it appears that *geoPublish*, from its inception, was designed with PostScript in mind. And why not? It was PostScript that first set the business world on its ear with amazing technology in desktop publishing. Since Adobe Systems practically wrote the book on the subject, it would have been ludicrous for Berkeley to create their desktop publishing software without a PostScript option. Thus have we profited from their pragmatism.

PostScript was originally intended to be an invisible layer of software between your computer and your printer. In that sense, *geoPublaser* has served valiantly for many years. For how many years? Eleven! Those of us who want to get the most out of our machines have no choice but to strip away the "cloak of invisibility" and to begin extending the power of PostScript programming.

PostScript offers so much that *geoPublish* does not share with us. To see what I'm talking about, I ask you to print out the *Tough Tymes* articles, Parts I and II. These articles describe my torment and triumph in an effort to download a non-resident font to my laser printer in 1994. Simply open *PostPrint* and send the *Tymes-Elfin* font followed by a Control-D. Then send *ToughTymesTitle*. Be aware that it will take 5-20 minutes to process, depending on the age (and speed) of your printer. When it is downloaded, then send *ToughTymesText1* followed by a Control-D. To print the second article, send *ToughTymesTitle* again followed by *ToughTymesText2* and finally a Control-D.

These articles represent what I call **Hybrids**. A hybrid is a program that combines the best of *geoPublish* and pure PostScript coding. Within these articles, I was able to print text in a circle (actually an ellipse), use the **concat** command for shadowing, print from a non-resident font, and rotate graphics. My point is that with a little effort, we can master the basics of PostScript and thereby add enormous versatility to an otherwise stagnant DTP application.

Recently, with Roger Lawhorn's help, I was able to convert 650 PFB (PostScript Fonts Binary) to PFA (PostScript Fonts ASCII) so they could be used on my Commodore. Don't ask me why the PFB's would not print. I am still working on that one. However, we now have 650 additional public domain PS fonts for our Commodores which I hope to upload to Jim Brain's web site before the end of the year. And that is only the beginning!

I ordered a CLICKART product with six CD's containing 65,000 files. Most are EPS files (for Encapsulated PostScript). This is the way that PostScript handles the GEOS equivalent of photo scraps. Imagine, that I can buy CD's with thousands of these, including a catalog of illustrations, for only $30.00! It's true. Currently, Roger must upload the files I need to my ISP on the internet. From there, I can download them into the Commodore environment and use them in my newsletters or other GEOS documents! There is so much available in the world of PostScript, and the only thing separating us from this rich treasure is our 'lag' in understanding. Hopefully, this will soon be changing.

I would like to clarify where we stand now. By using a program called *geoPublasrDisk*, we can covert a geoPublish document into a TRUE ASCII PostScript file on disk. We can then convert it back into a geoWrite file using *WrongIsWrite81* and change ANYTHING! For instance, we could even make the document print in color, if we have a color PostScript printer. We thus create a *hybrid* program that uses all the features of *geoPublish* while allowing us to add special PostScript touches where GEOS is lacking. Thus we can draw from the best of both worlds!

The GEOS/PostScript connection has enriched Commodore computing for many years. Yet, in many ways, it is just beginning. Only now, as PostScript printers become cheaper and more accessible, do I see Commodore people really beginning to inquire about PostScript. Only now, as Maurice Randall's new *WHEELS* program promises huge gains in memory management, do we have the capacity to really handle the large files that PostScript might require. The time is right for us to move forward on this, and I hope the GEOS/PostScript connection will continue to provide us with challenge and excitement far into the future.

Hybrids Allow Us to Rotate Graphics to Any Degree!

0°   23°   130°   72°
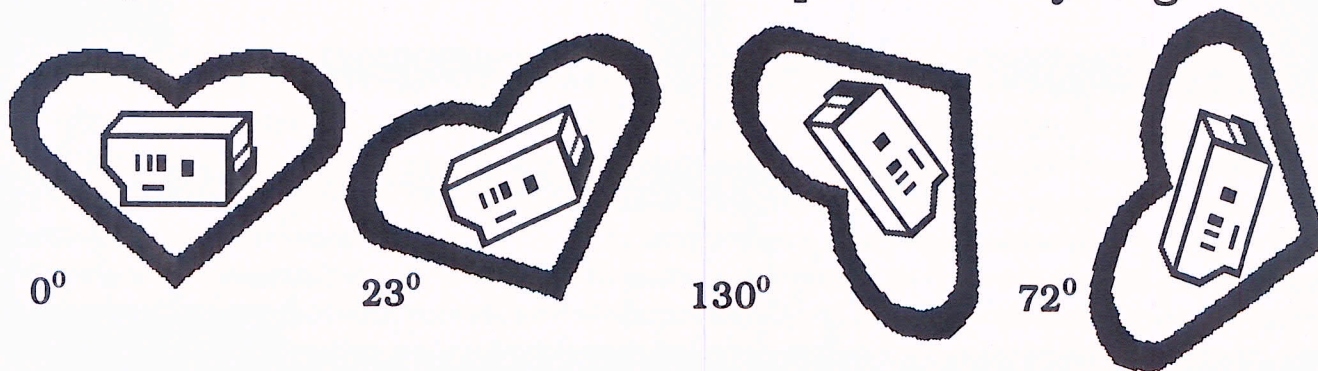
Maurice Randall, the geoMaster

In June of 1990, a now deceased newsletter, *geoJournal*, carried a "Rumor Column." I was excited to read that someone *might* be working on *geoPublaser 128* to take advantage of the increased speed and 80-column display of a C128. I was editing my club's newsletter at the time, and I was very discouraged by the turtle-like processing of our current *geoPublish*. I yearned for an 80-column version that would enable me to complete each issue in half the time.

A couple of years later, I discovered that this person might be Maurice Randall. I contacted him by phone, and he said that he was working on such a project. He was very confident that it could be done, and I wanted to support him. So I offered to donate some money to the cause. I wanted to be involved, if only peripherally. I quickly learned that one of his idiosyncrasies is that he doesn't accept donations...period! He was willing to sell me anything that was 'finished', but he would not accept the obligation which donations imply.

I thought, "What the heck! Since I simply want to support his work, what do I care if he mails something back?"

In this unique and oblique fashion, I came into possession of a program called *geoSHELL*. One of my idiosyncasies is that I like to read the 'docs,' so I read his instructions from cover to cover. Then I thought, "This is a great program, but it's not for me!" and laid it away.

Then May of 1994 arrived, and I found myself floundering. I was trying to download a non-resident font to my laser printer and NOTHING in the world of Commodore could do it properly. I was desperate! Then I remembered reading the docs for *geoSHELL*. Maurice had created a GEOS-compatible operating system that was designed for easy modification. Could he modify *geoSHELL* to download my font? He answered, "No problem!"

Soon he sent me my geoSHELL upgrade and included in it was a special surprise. It was a COM [command] file called *laser*. As he did not yet have a laser printer of his own, would I be willing to test it? Is a frog willing to become a prince? Needless to say, his program enabled me to do what had previously been impossible!

During all this time, he had not given up on his dream to produce an 80-column DTP program for the Commodore. However, he needed more memory to work from. A 1581 partition (800K) was just to small. So he wrote drivers for *geoSHELL* which allowed him to use native-mode partitions. He could now access four drives and native-mode partitions while working from within GEOS 2.0. To me, this seemed miraculous!

Still not satisfied, he decided to rewrite the *CONFIGURE* file and all the drivers. Still not satisfied, he decided to redo the entire *DESKTOP!* Believe me, before long he was doing more than spinning his *WHEELS*.

During this time, I asked him to help me out with a special program which could send a PostScript file straight from a geoWrite document. He agreed to do it, although it was nearly two years before he found time to complete it. When Maurice sent it to me, he also offered to let me market it! So I began to think of everything I could include with it. I thank him for his program and his encouragement. They became the impetus for me to assemble this LASER LOVERS disk for all Commodore users who "care enough to print their very best!"

I like to think of Maurice as the geoMaster. I have spoken with many good (some great) programmers over the last fifteen years, and all have said to me at one time or another, "I don't know why it (the program or the computer) does this, but it does."

Maurice has always told me, "I don't know why it does this...yet...but I am going to find out."

He has a bulldog tenacity that refuses to accept anything less than success. With the CMD's SuperCPU, we may not need a *geoPublish 128* anymore, but look what he has accomplished along the way. Although our Commodores have had advanced hardware, thanks to CMD, until recently they had no advanced software! Single-handedly, Maurice has taken GEOS, the best operating system ever written for Commodore, and dramatically improved it. His *WHEELS* can fully integrate everything that CMD has made available to us. This is an unparalleled accomplishment, and I therefore see him to be the unparalleled master of this graphic environment.

The geoPublaser Prologue

Nearly every PostScript 'job' starts with a prologue. It consists of all the terms and definitions that will be needed to complete the project. The geoPublaser prologue is one and a half pages long. If you want to read it, you can look at the start of any file sent to disk by the program, *geoPublasrDisk*. To show you how this is done, we must interrupt this section to bring you the following important announcement.

<div style="border:1px solid">

HOW TO COVERT DTP DOCUMENTS INTO GEOWRITE FILES

The term DTP is widely used to abbreviate desktop publishing. In Commodore, this refers to a geoPublish document. To convert such a *'DTPFILE'* into *geoWrite*, follow the outline below.

1. Open from Drive A or B, *geoPublsrDiskA* or *geoPublsrDiskB*, and select the *DTPFILE*. A new file, *PS.DTPFILE*, will appear on your disk or partition.
2. Generate or double-check the *Text Scrap* file, making sure the text is Times-Roman, plain text, 12 point font. You may want to take the precaution of 'grabbing' the text in the middle of a sentence to guarantee there is no font confusion.
3. Open *WrongIsWrite81* by Joe Buckley, which I usually refer to as simply *WW81*. Click on the word 'source' in the command bar above. Then select 'true ascii.' Next you must select which geoWrite format the new file will be converted to. I recommend 2.0 for 64 mode, and 2.1 for 128 mode. Then you can allow *WW81* to name the new file for you or input your own filename. If you allow *WW81* to name it, it will be called, *PS.DTPFILE+*.
4. STAY IN *WW81*! Before you leave it, you need to make one more conversion. Again, click on 'source' and then on the geoWrite format you selected in step #3. Then click on the 'font' option. This option will create a new file, *PS.DTPFILE++*, using a font identical to the one you stored in *Text Scrap*.

Why do I recommend the last step? For your answer, open each of them in turn now, *PS.DTPFILE+* and *PS.DTPFILE++*. Notice the second opens and reads easily. The first file generates one or more page breaks and gives you only the BSW font. You will learn to love *WW81*. Joe Buckley has created an extraordinary program, and you will find many uses for it. [*Now we return you to our regularly scheduled discourse.*]

</div>

I want you now to look at the "real" geoPublaser prologue. Please open *geoPublish* and create a file called *NAME*. Open the *NAME* file within *geoPublish*, and you will be in "page graphics" mode. Put your name anywhere on the page using the Times-Roman (50 point) font. Switch to *geoPublasr 1.8_GC* and send it to your laser (PostScript) printer.

Now open *geoPublasrDisk* and "print" the same file. Of course, the document will not go to the printer. Instead it will be sent to a disk or a partition. Follow the instuctions above to convert a file called *NAME* into a geoWrite file called *PS.NAME++*.

Now open it up within *geoWrite*. The first thing you will see at the top of the page is the start of the geoPublaser prologue. It starts with...**72 80 div 80 80 div scale 0.25 80 mul 18 translate**...and finishes in the middle of the next page on the line where SC stands alone. I have always wondered what SC stands for. Does it mean "Send Composition" or "Start Cooking"? I don't know, but I do know that you will not see your individual data containing (Your Name) in parentheses until immediately following this line.

Notice that right away, the prologue gets down to business. The first command is **72 80 div 80 80 div scale** translates to **72/80 1 scale**. I mentioned before that the normal PostScript model is 72 dpi horizontal and vertical. The first number (72/80) effectively takes a unit that is 72 points long (one inch in PostScript) and divides it into 80 equal parts. We have thus converted PostScript so that it will conform to the GEOS model of 80 dpi across. There is now a one-to-one correspondence between the pixels on a GEOS screen and the points on a PostScript page. What You See Is What You Get [WYSIWYG]!

Next we see the command **0.25 80 mul 18 translate**. Please remember my earlier statement that GEOS automatically clips the outer quarter-inch from every page. This command then translates the origin (0,0) up and over to the left a quarter of an inch so that it is in the lower-left corner of the printable portion of the page.

And so it goes. Everything in this prologue can have a profound effect upon our final product. By controlling the prologue, we can take advantage of everything *geoPublish* offers, which is considerable, while enabling us to add much, much more.

I will inform you now that this chapter is the one that needs most to be updated in the future. Eventually, we will want to know what every single command in the **geoPublish** *prologue will do and how it can be tweaked to our advantage. For this initial version of* **PSPFTC**, *I have only touched upon some of the more prominent features and then challenged you to discover more on your own.*

For this reason, I have included my own version of the prologue. It is exactly the same as generated by *geoPublasrDisk* except that it has been reformatted for clarity, extending over five pages. All terms, commands, and procedures are exactly the same as the one you will find in *PS.NAME*. My file, *GP.Prologue*, gives us all a common point of reference. Thus, if you were to call me with a question, I could refer you to page 3, line 7, and you would know exactly where I was. Keep this file "as is"! Always duplicate it and work from a copy.

Now I want you to go to your *PS.NAME++* file which you have converted to a geoWrite document. Open it up to page 2, and scan down to the line where you will find SC all by itself. Your individual data should begin just below that. Highlight the text starting on the line below SC and continuing to the end of the document. Make a text scrap of this material which contains all your individual data.

Now click once on the file called *GP.Prologue* and make a duplicate of it. The shortcut is [C=] [H]. Name your new file *GP.PrologueName*. Now open this file, go to page four, and paste the text scrap you have just created at the end of the file, below the letters SC. Be sure that there is a space or a carriage return to act as a separator or delimiter after SC. Your text scrap should be visible immediately after it.

Now open up *PostPrint* and select *GP.PrologueName*. Click on "OK" and then send a Control-D. Now your name should print out and look exactly like the one you just printed earlier using *geoPublasr 1.8_GC*. **If this is not the case, review these steps and find out why there is a problem. If you cannot locate the problem, call me because you cannot go forward unless this is working!**

If your name printed out properly, then I can breathe more easily because you are now in control. To reaffirm that you are in control, open *GP.PrologueName* again and do a search-and-replace. In the 'search' line, fill in Times-Roman and in the 'replace' line, type in Tymes-Elfin. Now open *PostPrint* again and download the *Tymes-Elfin* font file followed by a Control-D. Then download the program, *GP.PrologueName*. You should now be able to see your name printed out in a non-resident PostScript font! If you failed to download *Tymes-Elfin*, the default font will be Courier.

Open up the *GP.PrologueName* file again with *geoWrite*. Using the 'search' feature again, locate the word, "Tymes-Elfin." You should find it on page four. It will be part of a font array which will look like this:

/fonts[/Tymes-Elfin/Times-Bold/Times-Italic/Times-BoldItalic/Helvetica/Helvetica-Bold/Helvetica-Oblique/Helvetica-BoldOblique/Symbol/Symbol/Symbol/Symbol/Courier/Courier-Bold/Courier-Oblique/Courier-BoldOblique/Avant Garde-Book/AvantGarde-Demi/AvantGarde-BookOblique/AvantGarde-DemiOblique/Bookman-Light/Bookman-Demi /Bookman-LightItalic/Bookman-DemiItalic/Helvetica-Narrow/Helvetica-Narrow-Bold/Helvetica-Narrow-Oblique/ Helvetica-Narrow-BoldOblique/NewCenturySchlbk-Roman/NewCenturySchlbk-Bold/NewCenturySchlbk-Italic/New CenturySchlbk-BoldItalic/Palatino-Roman/Palatino-Bold/Palatino-Italic/Palatino-BoldItalic/ZapfChancery-MediumItalic /ZapfChancery-MediumItalic/ZapfChancery-MediumItalic/ZapfChancery-MediumItalic/ZapfDingbats/ZapfDingbats /ZapfDingbats/ZapfDingbats]def

The font array is how GEOS communicates with the printer to access the laser fonts. Each font family is listed four times. The first indicates the plain font, the second is **bold**, the third is *italics*, and the fourth is ***bold-italics***. Notice that some fonts are the same regardless of style. On the PostScript printer, 'Symbol' will be printed the same way in any style. Check out line two of the font array and see that 'Symbol' is listed identically four times. As we learn more about the geoPublaser prologue, we will be able to add special touches to our documents more easily.

Now let's discuss non-resident fonts. To use non-resident fonts, we currently have two options. The first way has just been shown to you. We can format the text into a geoPublish document by substituting a regular GEOS laser font. You should pick a GEOS font which approximates the width of the non-resident PostScript font which will replace it. For instance, if it is wide, use 'Bookman,' **bold** style. If it is thin, then select 'Helvetica Narrow.'

In most cases, you will only want to use non-resident fonts for "special text." I suggest that "paragraphed text" is best reserved for the Times-Roman and Helvetica fonts. To replace the GEOS font with your non-resident font, you must convert your geoPublish document to a PostScript geoWrite file using *geoPublasrDisk* and *WW81*. The resultant file will have the "font array" on the second page, and you can substitute the name of the non-resident PostScript font for that of the laser GEOS font it will replace. This can also be done easily with the "search-and-replace" feature.

The second way to put non-resident fonts in your documents is to insert the text yourself. The exercises on pages 7-11 show you how to do this. Remember, if you are working from a GEOS document, then an inch across the page will be 80 pixels (points) instead of 72!!! Therefore, you will need to adjust your 'inches' to suit the GEOS environment. Horizontal and vertical inches must be *defined differently,* like this:

/inx (80 mul) def /iny (72 mul) def % Inches along Y-axis remain the same, but the X-axis in GEOS changes.

To print special text in a non-resident PostScript font into any geoPublish document, you must leave open the areas in which you intend to insert "special text." Then send the document to disk with *geoPublasrDisk*. Convert it to *geoWrite* with *WW81* and add your special text. I suggest that while you are learning to place text, define inches as in the example above to assist in your horizontal and vertical placement. "Special text" may be placed like this:

/Tymes-Elfin findfont 25 scalefont setfont   % Remember, GEOS cuts off the outer quarter-inch of a page.
.75 inx 7.75 iny moveto (SPECIAL TEXT) show   % Text will start 1" from the left and 8" from the bottom.

The 'showpage' command is already in your geoPublish document. Notice that you must specify which font and point size you want to use before you can "*show*" the text.

Text-in-a-Circle must be inserted into your documents in much the same way as the above example. Make a text scrap of the file *CircleText(Cond)*. This condensed version is only a paragraph in size. It starts with **/oct** and ends with **/pi 3.1415923 def end**. No, I do not fully understand the code either, but that doesn't mean that we can't use it. Paste this text in your document to define the terms before using them. All you need to know is that **oct** stands for "outside-circle-text" and that **ict** stands for "inside-circle-text." These commands are preceded by your text in parentheses and three numerical values...the font size, the angle at the center of your text, and the diameter of the circle. If you are printing "Merry Christmas and Happy New Year," you might use the sample below.

% Precede this example with the Text-in-a-Circle PostScript code from the *CircleText(Cond)* file.
gsave 80 72 div 1 scale 306 448 translate   % When used within the *GP.Prologue*, rescale the X axis
/Times-Bold findfont 50 scalefont setfont   % back to 72 dpi so the circle will not be distorted.
(Merry Christmas) 50 90 135 oct   % fontsize=50, angle=90 degrees, and diameter=135
(and Happy New Year) 50 270 160 ict   % fontsize=50, angle=270 degrees, and diameter=160
grestore showpage

We will next cover rotating graphics. Print out the article on "GraphicRotation" and read it along with the next two pages. It contains data you will need to make text scraps out of. This will be the last stop on our journey through PSPFTC.

We must control the geoPublaser prologue to rotate graphics. Yes, *geoPublish* will allow you to translate (move) and scale (resize) graphics, and that it all very good. But wouldn't you like to be able to rotate them as well? Well, it is easier than you think.

Below is a familiar symbol to Commodore users. How difficult would this be to rotate?
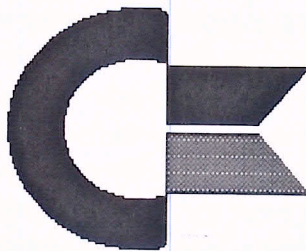
You will find a duplication of this data in the PostScript sample called, "GraphicRotation." Make a text scrap of the code illustrated below and paste it at the end of a copy of *GP.Prologue*. (Never use the original copy of *GP.Prologue*. Always use a duplicate so you can make a "clean" copy whenever you need it.)

136 96 136 96 261 95 261 95 615 191 PP

```
7D00830FFFF80D008403FFFFFE0D00843FFFFFFF0C00810104FF0C00810F04FF0C00813F04FF0C0005FF0B00810305FF0B00810F
05FF0B00813F05FF0B00817F05FF0A00810106FF0A00810306FF0A00810706FF0A00810F06FF0A00811F06FF0A00813F06FF0A0081
7F06FF0A0007FF0900810104FF83FC001F0900810304FF83C0000109008807FFFFFFFE00000107FF8AE0000FFFFFFFF000000107FF
8AC0000FFFFFFFC000000107FF8A80001FFFFFFF8000000107FF8A00003FFFFFFE0000000106FF8BFE00003FFFFFFC0000000106
FF8BFC00007FFFFFF80000000106FF8BF80007FFFFFF00000000106FF8BF00000FFFFFFE00000000106FF8BE00000FFFFFFC0000
0000106FF8BC00001FFFFFF800000000106FF8B800001FFFFFF800000000106FF86000001FFFFFF0400810105FF87FE000003FFFFF
E0400810105FF87FC000003FFFFFE0400810105FF87F8000003FFFFFE0400810105FF87F0000003FFFFFC0400810105FF87E0000003
FFFFFC0400810105FF87C0000007FFFFFC0400810105FF8780000007FFFFF80400810105FF8700000007FFFFF80400810104FF88FE
00000007FFFFF80400810104FF88FC00000007FFFFF80D008407FFFFF80D008407FFFFF80D008407FFFFF80400810104FF88FC000
00007FFFFF80400810105AA8700000007FFFFF80400810105558700000007FFFFFC0400810105AA8780000003FFFFFC040081010555
8740000003FFFFFC0400810105AA87A0000003FFFFFE0400810105558750000003FFFFFE0400810105AA87A8000003FFFFFE040081
0105558754000001FFFFFF0400810106AA8B00001FFFFFF800000000106558B000001FFFFFF800000000106AA8B800000FFFFFFC0
0000000106558B400000FFFFFFE00000000106AA8BA000007FFFFFF00000000106558B5000007FFFFFF80000000106AA8BA800003F
FFFFFC0000000106558B5400003FFFFFFE0000000107AA8A00001FFFFFFF8000000107558A00000FFFFFFFC000000107AA8A80000
FFFFFFFF000000107558A400007FFFFFFFE00000107FF83E0000304FF83C000010900810104FF83FC001F0A0007FF0A00817F06FF0
A00813F06FF0A00811F06FF0A00810F06FF0A00810706FF0A00810306FF0A00810106FF0B00817F05FF0B00813F05FF0B00810F05F
F0B00810305FF0C0005FF0C00813F04FF0C00810F04FF0C00810104FF0D00843FFFFFFF0D008403FFFFFE0E00830FFFF808007FB
F4DBF
```
saveobj restore showpage

After placing the above code (copied from "GraphicRotation") at the end of a copy of *GP.Prologue*, you should be able to print the Commodore logo pictured above. Why? Because this data is an exact copy of the PostScript code I created with *geoPublsrDisk*. Use *PostPrint* and confirm this for yourself. Now...how do we rotate it? You can open your file and put this code in front of the graphical data.

```
/PPR{gsave pop pop pop pop newpath gG translate/PPdy exch def/PPdx exch def/PPsy exch def/PPsx exch def /degrees exch def
    PPdx 2 div PPdy 2 div neg translate degrees rotate PPdx 2 div neg PPdy 2 div neg translate /PPs 0 def/PPp 0 def PPdx PPdy
    scale PPsx PPsy true[PPsx 0 0 PPsy neg 0 PPsy]{PPD}imagemask grestore{currentfile read{32 eq{exit}if}{exit}ifelse}
    loop}def
```

Let's pretend that **PP** stands for "Print-Photo." We want to change it to **PPR**, "Print-Photo-Rotate." Simply insert the degrees of rotation at the very beginning and then place a capital "**R**" on the end of that line. For instance, if we want to rotate the Commodore logo by 33^0, just replace the boldly printed line ending in 'PP' above with the one below.

33 136 96 136 96 261 95 261 95 615 191 PPR

I am especially proud of this code because is minimizes displacement. It saves the angle of degrees and then translates the origin (starting point) from the upper left corner (where GEOS starts) to the center of the graphic. Then the graphic is rotated on its center. After rotation, the origin of the graphic is moved to the graphic's lower left corner where PostScript requires it to be. This makes it much easier for you to place it. Normally, it would be rotated either on the upper left (GEOS) corner or lower left (PostScript) corner, making placement more difficult. However, nothing will erase the warping effect that is caused by different dots-per-inch along the X and Y axes.

When I am creating my own PostScript code, I often return the X-axis to 72 dpi. The GEOS standard is good for text, but bad for graphics. Why? Because having 72 dpi along both axes make your circles round and your squares square. It can be a real convenience, as well as being more pleasing to the eye.

If you want to temporarily return the GEOS environment to the PostScript model, use the code below:

gsave 80 72 div 1 scale   % This temporarily changes the X axis from 80 dpi to 72 dpi until you use **grestore**.

Remember, you can rotate ANY graphic in a geoPublish document if you first use *geoPublsrDisk* and *WW81* to convert it into a geoWrite document. You can then insert the **/PPR** code in front of the graphic. Place the desired "degrees of angle" up front and change 'PP' to '**PPR.**'

You are now on your own. If you have questions about this material, call or email me. However, if you have any questions about PostScript, you really ought to first buy Ross Smith's book, *Learning PostScript, a Visual Approach*, or the *PostScript Reference Manual*. Only then we will have something to refer to for specific information.

Good luck upon your new journey. I hope that PSPFTC will be a wonderfully enjoyable challenge for you, as it always has been for me.

Resources

Learning PostScript, a Visual Approach by Ross Smith
Call Peachpit Press, Inc. at 1-800-283-9444.
The cost is $22.95 plus $4.00 shipping.   20% discount if you are a member of a user group.
Because it is so popular, this book can still be purchased at local bookstores in my area.
It is also available on the internet from AMAZON.COM.

Maurice Randall	1-517-543-5202
P. O. Box 606	WHEELS for $36.00
Charlotte MI 48813-0606	geoSHELL for $24.95
email: arca93@delphi.com	$4.00 charge for shipping regardless of size of order
web: people.delphi.com/arca93/	

Joe Buckley's **Storm System Disk** only $20.00	
464 Beale Street	Includes many rare programs, like CIRCE,
W. Quincy MA 02169	a great game similar to Risk.

Creative Micro Designs (CMD)	Orders only 1-800-638-3263
P. O. Box 646	Information 1-415-525-0023
East Longmeadow MA 01028	Visit their website: http://www.cmdweb.com/