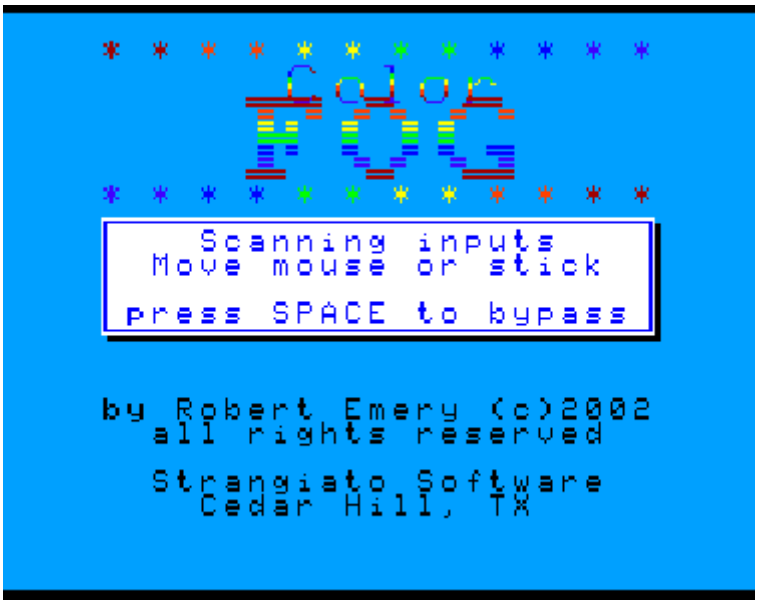
CoCoNutz! E-Zine.

www.coco3.com

Vol. 2 Issue 3, October 2006



An Interview with Brian “Briza” Palmer

CoCoNutz: Where do you live, have you always lived there?

Brian: I live in Broken Hill, NSW, Australia. I have been living here since 2000. Yeap, if you are curious about the name, it is the place that Mad Max 2(Road Warrior, Starring Mel Gibson), was filmed.
I haven't always lived here. I was Born and bred in Sydney Australia, then my family moved to Broken Hill around 1977. We then moved to Woodberry, Newcastle, NSW, in 1980, and lived there for 7 years. The longest place I have lived in, was Wollongong, NSW, I lived in The Gong from 1987-1999, then moved back to Broken Hill, and have been here ever since.

CCN: Are you married, kids, and pets?

Brian: Not married, But in a Defacto relationship. That is close to being married, except we don't have the Black/White paper to say we are. I have 3 Kids, Tate, Hamish, and Ada. Hamish and Ada are Twins. I have a step son Jamie, he is 18 now.
I have 2 pets, 1 Black cat called Blackie or Black Balls (Not my idea, was the Wife's), and 1 Major Mitchell. It's a sub species of Parrot. Charlie is a gorgeous bird, and he does have a curious mind. He learn't to whistle by listening to wild birds and people who just happened to be going past and doing a whistle.

CCN: What are your main hobbies?

Brian: I like playing around with Cars, Stamp collecting, and House Renovations.

CCN: What are you main interest in the CoCo?

Brian: I like software collecting. Anything coco related and I'll be there for sure. I'm that bad, I dream about my next big score in locating old Coco software. I am starting to learn Assembly. I was doing Graphics years ago, using the Super Extended

Basic commands. I never did like using the graphic designers, instead I liked to create the pictures using my mind, and plotting the points on the graphics screen.

CCN: What projects do you have going right now?

Brian: I'm currently working on around 4 projects. 1st major project is software archiving to the Pc. 2nd, is porting Dragon games to the coco computers, I have managed to get around 8 ML games over. About 3 work without any Rom patching, the other 5 will need to be patched, before they will be 100% compatible. 3rd I am helping other Coco websites with Forums and any other coco related topics. 4th Bob Devries and I are going to be scanning in Old Coco Magazines produced in Oz. Then we will put them up on a coco website for other coco users around the world.

CCN: Tell me about you and this game index of DSK files…….

Brian: I wouldn't call it an Index; I would call it a mess. Half the time, I don't know where I have put them in my hard drives. Guess I'll have to spend at least a couple days sorting it out. That means no internet, no chats, or anything else.

CCN: How many do you have now?

Brian: Well I have all up. So far I have imaged, scanned docs and so forth, around 3 1/2 Gigabytes worth of coco software. I still haven't got back to scanning the rest of my disks in yet, just been so flat-out Like a Lizard baking on a rock.

CCN: How can I get them?

Brian: When I and a few other CoConuts are finished, we'll be looking to put them up on someone's website. I can imagine how much fun you Guys and Gals will be having when you look thru the Dsk images we have compiled.

CCN: Do you have a list?

Brian: I have a List half done, but still have a lot to add to it yet.

CCN: You are not just porting over CoCo games, you have managed to do Dragon games as well, right?

Brian: Yeah, done about 8 ML games. The others are Dragon Basic games, Mixture of Arcade, Text Adventures, Graphic adventures and so forth.

CCN: How many altogether?

Brian: Can't tell you how many, but I would say around 30 games. Some still need to be patched to run fully on the Coco computers.

CCN: What games or programs?

Brian: Ok, In the ML games, I have got over Stone Raiders 2. (Which has already been ported, just had to know which website had it in the DSK images. Airball (This will be a great strategy game when patched to use coco keyboard). Cuthbert in the Cooler. This game seems to work 100%. I tried and tested it on a real coco 3.
Fire-Force, this works but the bottom half of the screen is scrambled. I found this was the same on the Dragon emulator. It could be an emulator problem. If so, it might be 100% on the CoCo. This game is similar to Rambo and Mission Russian assault.
Plus various other games I have managed to get loading into the emulator.

CCN: How do you do it?

Brian: When it comes to ML loading games, 1st I hope they are non-
auto executing games. If they ain't, I load them into the coco emulator, and at this point I don't EXEC them. Instead I mount a blank image into drive 0, then I use the Peeks to find the Start, End, Execute addresses. Then I resave the game to DSK image using the addresses I got from the peeks.
When it comes to the Dragon Basic software, I first load the basic program into the Dragon emulator. Then I resave the program in ASCII format to a new CAS file, and then I load the new CAS file into the Coco Emulator and resave the file to disk. Then I go thru the basic listing, looking for all commands dealing with Cassette usage, and change them to Disk drive commands.

CCN: You have recently started writing a column, tell me about it?

Brian: OK, You must be talking about my Aussie Musings Blogs in Militant Buddha's Website. http://www.militant-buddhist.com That all started when I got to chatting with him one night. I said I wouldn't mind writing up an Aussie coco blog or anything else that was Aussie. Militant Buddha said if I write anything send it to him and he will host it. So now I have Aussie Blogs to do as well. It has been a lot of fun; I reckon I have a pretty good knack for these Blog thingys!

CCN: You also help George, tell me about what you do there?

Brian: George has 2 Forums, 1 Forum is for My coco topics, and the other is for Windows. It is used for people who have problems with windows able to leave a post in the forum. So others who don't know what to do to fix something can get answers, without relying on Microsoft for answers.
I also thought it would be a great idea to have Coco based forums all over the web, that way old time coco users, who drop into George's website, might be curious enough to setup their coco's again.
George's website URL is, http://os9user.blogspot.com/
But the number 1 website for coco's would have to be Roger Taylor's www.coco3.com , Your Da man Roger!

CCN: You have recently been granted permission to scan old Australian magazines, which ones are they?

Brian: Australian Rainbow, Australian Coco, I'm hoping to get permission for another Magazine called The Coco-link Newsletter.
There was another Magazine, But long forgot the name of it now.

CCN: Will they be available on disk or for download? Where at?

Brian: Any website who is willing to host the scanned files, if there are any Website owners interested, drop me a line.

CCN: When will you start to do this?

Brian: Already have, Bob has started on some.

CCN: Do you have all those magazines?

Brian: No I don't, but hopefully other coco users in Oz still have copies in their Garages somewhere. By what I have found out, Bob has quite a good collection of Aussie coco publications in his shed.

CCN: If not where can I get a list of the ones you need?

Brian: That I'm not sure of, but I have a few ideas. I'll get back to you about that.

CCN: Is it true you would like to start up an Australian coco group?

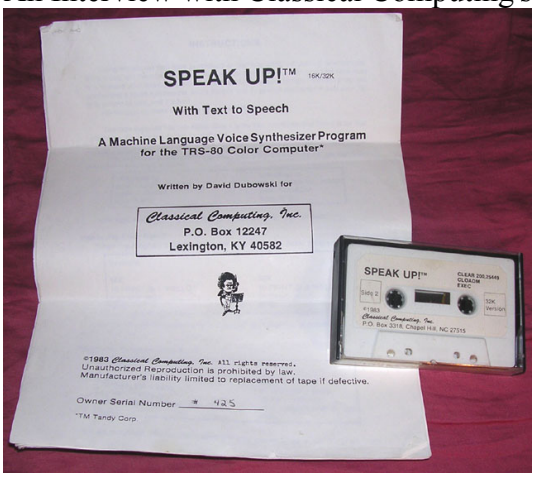
Brian: Yeappppppppy!!! Be Awesome to have Fellow Aussie coconuts in a major group again, I miss the days when we had The OS-9 user group, and the Coco-link group.

CCN: Tell me about that?

Brian: Not much to say, except we could really use a Coco group in Oz. Especially if you need hardware to be modified. That way Aussie coco users know that Bob Devries is the man for the job, Software Librarian would have to be me, and so forth. While in the Great USA, you have quite a few companies still dealing in Hardware and Software.

Speaking Up!

An Interview with Classical Computing's David Dubowski



In March of 1983 a half-page advertisement from Classical Computing appeared in *The Rainbow* magazine touting a completely software based voice synthesizer called *Speak Up!* What was so amazing about this debut was that it retailed for a mere US$29.95 and promised easy speech and integration into your very own BASIC programs. In an era when voice synthesizers were usually hardware based and ran up to a $100.00 or more, Classical Computing's little talker was a breakthrough. The author of *Speak Up!*, David Dubowski, recently consented to an email interview with CoCo Nutz! and was kind enough to share his memories of the program's development, the CoCo, and what he's been up to since.

CoCo Nutz!: I suppose we should start at the beginning. What got you interested in computers back then, when did you get your first Color Computer, what flavor was it, and what drew you to the machine?

David Dubowski: Well, I first learned BASIC in 1973, at the age of 12, when our junior high school had teletype connected to a mainframe somewhere, and I was immediately hooked. It had the well-known *Star Trek* game, you know the one; with <*> and >!< symbols for the Enterprise and star bases.

In 1981, when I was struggling through chemistry in college, the local Radio Shack got a CoCo, and I was playing with it every day after class, at the store. It was amazing that here was a little gray machine that could do BASIC just like the big mainframe, and much more!

By the summer of 1982, I had saved up enough money to buy a 16K CoCo with Extended Basic from one of those franchise Radio Shacks. Back then, lots of different types of stores had a Radio Shack section, with limited items. This one was in the back of a TV repair shop in rural Paris, Kentucky. The owner let me have it for $50.00 off, since it had been sitting in his store without much interest for months. But it was still $482 after tax, because it had the extended basic and 16k of memory. This was 3 months rent!

CCN!: A software based voice synthesizer for the CoCo, using just the machine itself, is pretty impressive. It's even more impressive when most synthesizers in 1983 required additional hardware and cost quite a bit more than that. Tell us about developing *Speak Up!*

DD: I just wanted the Star Trek program from the legendary *Basic Computer Games* to talk. I had gotten the *MegaBug* cartridge, and when it said "We Gotcha!" I knew that a voice could be recorded and played back in software. So looking around in

Byte magazine in late 1982, the answer appeared: it gave the addresses for the cassette input port.

I got the *EDTASM* editor assembler, and it was simple to record audio digitally into that 1 bit audio input port and play it back into a speaker click port, or something. I'm no expert, it was all done from trial and error, but the audio is 1 bit of resolution, and something like 3000 bits per second. The audio was pretty crappy, but it really did talk. 10 years later the bad audio solution flashed into my mind, too late of course.

In experimenting in 1982, the low volume noises in the background, like the refrigerator running, were playing back clear as a bell. My voice at the microphone, though, was loud but not clear. Had I just recorded the voice at really low volume, the intelligibility would have been drastically better, but it didn't occur to me that there may have been a way of boosting the playback volume. Had it been done that way, it would have been so clear as to be newsworthy.

I was a total nerd, working on *Speak Up!* for about 2 or 3 months in late 1982 to early 1983 to the exclusion of all else, waking up at sunset and working until the wee hours of the next noon, listening to Sally Jesse Raphael on talk radio advising the lovelorn! This was in a basement apartment, totally isolated, skipping college classes and flunking.

Figuring out and recording 37 phonemes only took a few days. Most of the programming time was devoted to making a decent text to speech parser and easy interface to BASIC. It had 37 phonemes of my voice recorded, and played them back according to the rules I just figured out by trial and error. Once in a while a flash of inspiration would just drop into my mind that would solve a problem, and I had this big flowchart on 4 pages of old sheet-feed printer paper taped together.

CCN!: You also did an Apple II version of the program. How did that come about?

DD: I went to work at the local company Intelligent Statements, now long defunct, which published *Pogo Joe* and others by the owner Mike Denman. They let me use their Apple II to convert *Speak Up!* to Apple, and they were going to use my system to make their programs talk and pay me a royalty. They didn't want it after the 2 months I was there, so I refunded their minimum wage pay and they let me keep the Apple version. The Apple version, released in early '84, never sold well. It only sold about 150 copies, just breaking even. Ads in the Apple magazines were really expensive, and they never reviewed it.

CCN!: What were some of your experiences with CoCo customers and selling the program though *The Rainbow* ads?

DD: The program sold about 20 copies the first month, which was great, but it got a huge boost from a review in *The Rainbow* in April 1983, page 132. Someone told me to incorporate, which I did, and Classical Computing was born, with only myself as the president, treasurer, secretary, and only employee.

When *The Rainbow* review issue hit, I got 22 orders the next Monday at $29.95 in the P.O. Box, including some from Canada and Brazil. What a thrill! (In 1983 that $630

day was 3 months rent money in one day!!) That never happened again, but I lived off of filling 3-7 orders a day until it died out in late '84. It only cost about $2.50 to make and mail a copy.

Each copy was made directly from a CoCo with its own serial number, written on the manual, to discourage copying. If anyone loaded it, changed the serial number and then made a copy of the program, the pirated copy would not work. *Speak Up!* checked the visible serial number with a hidden encrypted copy of the serial number and if they didn't match, it would erase itself in memory and not run - a primitive anti-piracy measure that was never revealed.

About 7% of buyers took advantage of the unconditional money back guarantee, but that was fine, I was happy to make enough money to pursue electric guitar. What really shocked me was that for the entire run of *Speak Up!*, there was always a small group of buyers who hated it, a small group who loved it and asked for more programs, a group who liked it OK, and a group who were lukewarm on it, and a group who were a little dissatisfied. At the time I had no idea that human opinions form a bell curve. I thought everyone would see it the same way, either good, fair or bad, - ha ha, I was so naive! After about 18 months, sales dropped off, ad rates had nearly tripled. Boy was that depressing when it faded away.

CCN!: What was it like trying to market a CoCo program in 1983?

DD: Having no experience probably helped! If I had talked to anyone other than one best friend about trying it, no doubt they would have discouraged me. But at the time, I was living alone in a basement apartment, and had plenty of free time outside of classes.

In late 1982 or early 1983, I saw an article in a newspaper or somewhere about a 17 year old guy who had sold 500 copies of a program he wrote for $50 each (25 grand!). I think it was called *Lock-It*, or something like that. It allowed Apple II software sellers to copy protect their software. It showed a picture of him in his parents' driveway with a new sports car he had bought. I figured this was a hot way to make some big money, so it helped inspire me to actually try selling *Speak Up*. *Speak Up* was originally just for my own use, to get that Star Trek game to talk.

I always loved buying things through the mail, especially the old Johnson Smith catalogs. When I first found *The Rainbow Magazine*, I noticed that it was filled with ads for programs and hardware parts, and many of the ads and programs looked homemade. I thought that I could do the same.

When I called *The Rainbow* for information on running an ad, it was pretty cheap. $175 for a half page ad. So, I gave it a shot. The only voice synthesizers around were over $100, so I figured if I could sell just 7 copies at $29.95, it would cover the cost of the ad. And if one person from each state ordered one, that would be some good money. Remember, in 1983 $29.95 was worth about $60 today.

The first month it sold probably 30 copies. But the next month, April of 1983, *The Rainbow* ran a review of it. The day after it hit the stands, there were 22 orders in the post office box, with several from overseas! That month it sold about 200 copies, and

I moved myself and the company back to Chapel Hill, NC where I had lived before. It sold fairly steadily for 12-18 months, but like any hit record or bestselling book, it had a finite lifespan, and sales stopped.

CCN!: Why the name Classical Computing?

DD: Classical Computing! Yes, I had the program all finished, and named it *Speak Up!*, but couldn't come up with a good company name. I wanted something general, so other products could be released, but unique enough to be remembered.

After a couple of days of going through lots of ideas, I finally gave up and went to the dictionary! After going through A's and B's, the word CLASSICAL jumped out. The definition said something about being well known or established, and I liked classical music. "Well," I thought, "it's not great, but it's taken too long to try to figure out a name, so that's going to have to do." It turned out to be very good, though. I liked it more and more as time went on.

CCN!: Where'd the "angry conductor" for the Classical Computing logo come from?

DD: The angry conductor is Beethoven, taken from a clip-art book. The original picture was clearer, he's not really angry; he's focused and listening intently. When the program was done, I went to a small typesetting company to make up the cassette labels, instruction manual and the ad for *The Rainbow Magazine*. Back then, there was no such thing as desktop publishing. So at the typesetting company, they had a big book of clip art, and there it was: Beethoven conducting, royalty-free, for anyone to use. It was perfect! A few years later I saw that same picture in another, non-computer ad somewhere. It was probably a large typesetting supply company that published those clip art books nationwide.

CCN!: Did you ever attend any of the CoCoFests? If so, what were your experiences?

DD: I did attend a Rainbowfest, it may have been the first one. It was in 1983, in Chicago, a few hours drive from Lexington, KY, where I was living at the time. It was amazing to see all those companies there with CoCo products.

I had sent 20 copies of *Speak Up!* to a guy who rented a big booth for the entire event to sell various companies' products. He was going to split the money 50/50 with me, and mail a check. I saw him there, and he hadn't even taken them out of the box!

I said, "Hey, let me just have those back," and he said "Oh, don't worry; I'll sell them all by tomorrow." So I agreed, since I was only going to stick around a couple of hours. You can guess the rest of the story, of course. I never heard from him again and got no check in the mail. Who knows if he even sold one *Speak Up!*, he was selling disk drives and high dollar items instead. But, since they only cost me about $1 to make each, it wasn't a biggie.

I didn't want to rent a booth or stay overnight; I just wanted to go see what was up. All the other companies had lots of really good stuff, full color packaging and all. My one product, on the other hand, was black and white in a ziplock bag! I didn't bother

to listen to the speakers; I just milled around and looked at all the cool programs, and left after a couple of hours.

CCN!: Were there any other CoCo folks you got to meet, correspond with or hang out with?

DD: Oh no, recluse was my middle name! I was living all by myself at age 22 in a basement apartment. I didn't know anybody else who had a home computer, except one friend who had a Commodore Pet or something similar, and he lived across town.

When the orders were coming in 5 or 10 a day, I hired a couple of friends now and then to fill orders that had backed up, or make *Speak Up!* copies. I'm not quite as much of a hermit now.

I wrote back to customers who asked questions about *Speak Up!*, all by snail mail. I never had any interest in the modem or Compuserve back then. I really hadn't heard of e-mail until the early 90's, because I completely lost interest in computers from about 1988 until 1994 when the Internet went public.

CCN!: Did you ever write any other programs for the CoCo?

DD: The only other program I wrote and sold was written in about 30 minutes. It was called *Adventure Cracker*, which was used to find the commands in adventure games. I had a really fun adventure game for the CoCo from Radio Shack called *Bedlam*. It got frustrating, because the player didn't know what all the commands were.

So, to find them, I just wrote a machine language program to look at what was in memory, displaying only ASCII characters, A-Z, that were in strings of length 2 or greater on the screen. It worked like a charm! There was the entire catalog of command words, right on the screen, with no garbage.

I saved it, and just for fun one day typed up a 1 page instruction sheet, priced it at $15 and sent a copy to several magazines. 6 months later, I had forgotten about it, but orders started coming in, maybe 3 or 4 a week for a couple of months. Apparently one of the magazines had written a little paragraph about it and loved it. So I found it, photocopied the paragraph and enclosed it with the *Speak Up!* orders, and that generated some sales.

I wrote lots of little programs here and there for my own use, like a graphing program and a calculus program to just try to understand Calculus III in college. It was a great help, I had gotten a D on the first try, but re-took it and by programming in the steps to calculate volumes and areas, the ideas sunk in and I got a B+ .

I used the CoCo to print mailing labels for orders, with a used dot matrix printer. It saved all the customer names, addresses and purchases on cassette tape (data, not voice)! That tape and everything else is long gone. The only thing I have left is a few copies of *The Rainbow Magazine*, *Color Computer Magazine*, *Hot Coco Magazine*, 2 copies of *Speak Up!* For Apple II+ , and now *Speak Up!* for CoCo thanks to Cris's generosity!

CCN!: Do you still own a CoCo?

DD: No, I don't own a CoCo. I got rid of all my CoCo stuff around 1988, during a move. There had been no ads running, no sales, and no mail at the post office box for several years by then.

In 1991, however, someone had a CoCo for sale in the newspaper for $25, so I went and got it. Twenty-five bucks, and it had the works! It had 2 disk drives, extended basic, 64 K memory, and lots of games. I played with it for a few weeks, the disk drives were fabulous, compared to cassette tape, but it just wasn't the same, it didn't hold my interest. So, after a month I donated it to the local thrift shop. Turning 30 can really bring on changes, probably at the neuronal level.

CCN!: So, what are you doing now?

DD: During the year and a half of *Speak Up!'s* commercial life, I put a lot of effort into learning and later performing electric guitar. In the years after Classical Computing, I did a lot of various jobs, from stereo salesman, corporate bookkeeping, truck driving, and 7 years of pizza delivery, which was actually a lot of fun. It turns out that I just do not enjoy doing "brain work" for others. I only enjoy it if it's my own idea and on the spur of the moment.

Right now, I work at home answering telephone calls for infomercials! This is a pretty enjoyable way to make the rent money, and leaves lots of free time. I do a lot of reading, a lot of web surfing, some gardening, and occasionally try other business ideas, in the hopes of catching another big money wave from home, but so far, the J-O-B's have been the best sources of income.

CCN!: Any fond memories, funny anecdotes, or thoughts about the Color Computer you can share?

DD: Well, it's always a nostalgic rush to see a CoCo anywhere, but that is now pretty much happening only on the Internet. Seeing Cris's list of other programs brought back a lot of cool memories of the old software games. It was a lot of fun to get that little computer to do so much, especially with all the sound and graphics capabilities. When friends would come over they always loved playing with *Speak Up!*

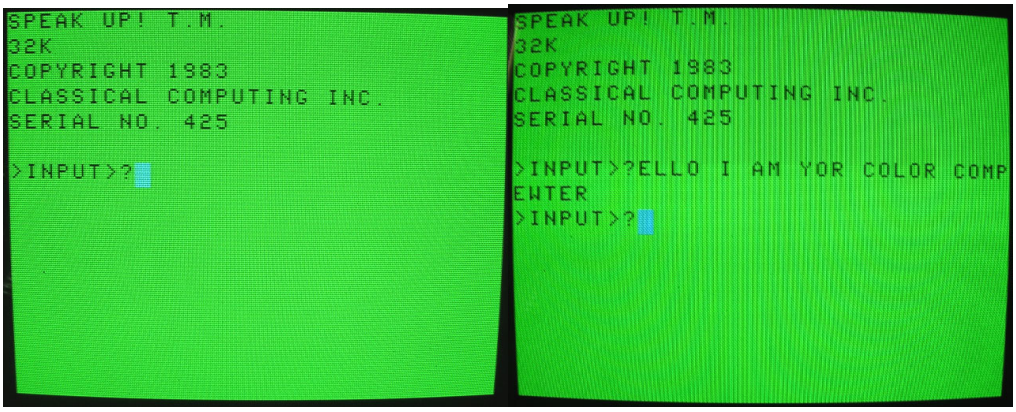
One amusing anecdote is when I had to buy more memory for it in back in 1982, just to get the Star Trek game to fit, typed in from the *Basic Computer Games* book. The really funny part was how you had to install the memory, you had to actually piggyback the 8 new chips on top of the ones on the motherboard, and solder on a jumper wire! Oh, and this upgrade, from 16K to 32K, cost $50!

Mr. Dubowski has graciously consented to release *Speak Up!*, free of charge, for non-commercial use, in .dsk image format for all *CoCo Nutz!* to play with and enjoy.

The 32K version runs fine on 64K Color Computer 1 and 2, as well as the Color Computer 3. Please note that the proper CLEAR statement is required before loading and that the program is position dependent and should not be shuffled in memory.

Mr. Dubowski invites CoCo users who remember and used his program, or anyone for that matter, to contact him at davidwhy@iwon.com.

- Cris Egger (captcpu@clubltdstudios.com)



Ahhhh Memories!!!! See if you can recognize any of these people!









Flash Backs from
Richard Crislip

Upon our arrival late Friday night, we met Chris Hawks, Mark Marlette, and Boisy Pitre in lobby. Unfortunately, after an 8 hour drive from Norton, Ohio at the end of a work day, we were too tired to party with them. We could not set up our area that night because the conference room was being used for a wedding reception. Therefore, we retired to our room to get some rest for the big day Saturday. Our room was nice. For seventy nine dollars a night, you will not get the bridal suite, but our room was very spacious and comfortable for the wife and me.

Saturday morning was used to setup the fest displays and vendor areas. The setup went well. Power was supplied via patched together industrial strength extension cords. Coffee, bagels, and doughnuts were supplied by the club for vendors and anyone else who bought a table.

We met Bob England, who is the author of BackUp. With this program, any CoCo disk can be backed up. This backup program is able to backup copy protected disks too.

Renting a table was, for me, the best thing I ever did at a fest. I wish that I had thought to do that much sooner. I was not a vendor, but having that table allowed me to set up my CoCo and PC based CoCo emulator without dominating the hotel room. It also made me feel like I was truly part of the fest rather than a customer. It allowed me to immediately try out my latest purchase or the latest

suggestion garnered from those who forgot more about the CoCo than I will ever know. It provided me a place to sit without leaving the fest area and to truly sit down and get to know the many fine folks who were the power behind the CoCo community. The table also allowed me to renew acquaintances.

One of the nice things about the fest was the ability to renew my GCCC membership. The Glenside Color Computer Club (GCCC) has been putting on these fests ever since Falsoft bowed out of active CoCo support. They have always done a marvelous job and this time was no exception. I wish I lived a little closer to the area so that I could attend their meetings.

This year, as he did last year, Dave Keil demoed his excellent CoCo emulator. Dave purchased quite a few older PCs and has converted them into CoCo emulators. The average price of this improved PCs was less than $30. Dave has written one of the best CoCo emulators available. His emulators can run most of Sockmaster's demo programs which will crash every other emulator as if this writing.

Over the previous year, Roy Justice finished developing his CoCo to SVGA video adapter. I received one of the first production units earlier this year. While at the fest, I met Roy and he replaced my RGB to SVGA converter for a more current model. This resolved an image problem I didn't realize I had been experiencing until he pointed it out. It was great to be able to switch between the PC based emulator and the real CoCo3 using one monitor

Chris Hawk awed us with his SVHS converter. It was marvelous seeing the CoCo displayed on a large-screen plasma TV/monitor. Chris was selling them at the Hawksoft table. This was another piece of hardware developed since last year's fest.

FWD was there again this year too. They helped us understand what the escalated gasoline prices were doing to the traveling vendors. Basically, it is not worth it for a vendor to travel to a show, especially for the vendors who need to show a profit after attending the show. Gasoline prices are becoming an attendance issue also. The conversations on the mail-list are attesting to that fact.

The WebCam was up sporadically. I wish I could remember who it was that tried to keep it up and running, but alas....

Bro Jeremy's Monk-o-Ware was also present. He had his usual CoCo museum on display. Brother Jeremy has been able acquire an impressive array of CoCo and CoCo hardware prototypes including proof of concept boards.

Cloud-9 had a reduced display this year due to their decision to all but shut down until the SuperBoard is finished. This project has dominated all of Mark's time and energy for at least two years now. Cloud-9 had hoped to have the boards ready in time for the fest, but it was not to be. In the meantime, as I understand it Boisy Pitre has agreed to sell the remaining Cloud-9 offerings until Mark can get the SuperBoard rolled out.

Lunch was catered in from the Benigan's Restaurant attached to the hotel. The gang met for supper in the hotel restaurant party room. Later that afternoon the GCCC no minimum bid auction was held. Anything and everything CoCo and non-CoCo was up for grabs. Most of these auctioned items were donated by members of the community as a way to support the club and to pay for the fest. Which leads me to one ominous fact; As I understand it, the year the club finishes a fest in the red, will be the year of the truly Last CoCo Fest.

Sunday, Bro Jeremy held his traditional Sunday morning service. We did not attend as we usually do. The announced focus of the service was continued prayer for the health of fellow CoCoNuts and for those who had health issues this year. Prayers of thanksgiving were also offered for those who had recovered from their health issues this year.

Again, as on Saturday morning, coffee, doughnuts, and bagels were provided. The no-minimum bid auction was held again. We had to leave by 2pm CST, so I'm not sure when the party truly ended. The drive home was exciting. The wife and I decided to avoid the toll booths by taking the shortcut through downtown Chicago. We did fine until we hit the road constriction.. err road construction. At that point we got to enjoy the sights and sounds of Chicago from the vantage of the multilane parking lot called I90. We named it The Shortcut That Wasn't. We arrived at home base around 11:30 EST.

Our closing thoughts are:

1.Buy a table next year. If, you intend to bring your CoCo equipment; you'll be glad you did.
2.Become part of the group by taking part.
3.The Fest can be more than just a computer flea-market.
4.Have a great time

One-Liner

Here's a short one originally submitted to The Rainbow by Greg H. Taylor of Naperville, IL. Kaleidobox was published in the May 1985 issue of the Rainbow as well. I like this one because it illustrates a creative way around a problem. Instead of trying to replicate to four identical quadrants, which would be slow, the author kind of sidesteps the whole issue. And he does it using one of the CoCo's built in commands, achieving a certain economy, and literally thinking outside the box!

```
0PMODE1,1:PCLS:SCREEN1,1:FORX=0TO65536:FORY=0TO50:R=RND(129)-
1:S=RND(97)-1:C=RND(4):COLORC:LINE(R,S)-(255-R,191-S),PSET,B:NEXTY:
PCLS:SOUND160,1:NEXTX
```

This wasn't necessarily one of the best one liners I've ever seen, but it's fun. It turns your CoCo into the ol' Magic 8-Ball. The program demonstrates an good technique for selecting random items using an array without multiple IF/THEN statements. Originally appearing in the May 1986 issue of The Rainbow and submitted by Bruce Gerst of Omaha, NE.

```
10 CLS0:PRINT@229,"PRESS <ENTER> FOR ANSWER";:EXEC44539:
A$(1)="ASK ME LATER":A$(2)="FORGET IT":A$(3)="YES":A$(4)="NO":
A$(5)="IT IS CERTAIN":A$(6)="IT IS DOUBTFUL": X=RND(-TIMER):
X=RND(6):CLS0:PRINT@233,A$(X);:SOUND150,3:FORT=1TO999:NEXT:GOTO
10
```

His ghost lives on!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

This is a program I found tucked inside one of Dads disks!! This is one he wrote.

```
1 '******************
2 '*  TIK-TAK-COCO   *
3 '*  BY:DALE KRAMER *
4 '*  COCO ECB       *
5 '******************
6 CLS
7 PRINT@42,"TIK-TAK-COCO"
8 PRINT@11,"BY"
9 PRINT@171,"DALE KRAMER"
10 PRINT@420,"PRESS ENTER":INPUTA$
15 TT=0:X=124:Y=94:CLS
16 '
17 '
20 PMODE3,1:SCREEN1,1:PCLS
30 LINE(80,0)-(84,188),PSET,BF
40 LINE(164,0)-(168,188),PSET,BF
50 LINE(0,60)-(248,64),PSET,BF
60 LINE(0,124)-(248,128),PSET,BF
62 '
63 '
70 GOSUB200:GOSUB100:IFTT=4THEN130
80 GOSUB200:GOSUB120
90 TT=TT+1:GOTO70
100 LINE(X-40,Y-30)-(X+40,Y+30),PSET
110 LINE (X+40,Y-30)-(X-40,Y+30),PSET:RETURN
120 CIRCLE(X,Y),25:RETURN
130 E$=INKEY$
```

```
140 IFE$<>CHR$(13)THEN130ELSERUN15
150 GOTO130
152 '
153 '
200 'CURSOR
210 K$=INKEY$
220 IFK$="1"THENGOSUB380
230 IFK$="2"THENGOSUB390
240 IFK$="3"THENGOSUB400
250 IFK$="4"THENGOSUB410
260 IFK$="5"THENGOSUB420
270 IFK$="6"THENGOSUB430
280 IFK$="7"THENGOSUB440
290 IFK$="8"THENGOSUB450
300 IFK$="9"THENGOSUB460
310 IFK$=CHR$(13)THENRUN15
320 IFK$=CHR$(32)THENRETURN
321 '
322 '
330 LINE(X-40,Y-30)-(X+40,Y+30),PSET,B
340 FORT=1TO600:NEXTT
350 LINE (X-40,Y-30)-(X+40,Y+30),PRESET,B
360 SOUND55,1:GOTO200
361 '
362 '
370 'SQ SUB
380 X=40:Y=30:RETURN
390 X=124:Y=30:RETURN
400 X=208:Y=30:RETURN
410 X=40:Y=94:RETURN
420 X=124:Y=94:RETURN
430 X=208:Y=94:RETURN
440 X=40:Y=158:RETURN
450 X=124:Y=158:RETURN
460 X=208:Y=158:RETURN
461 '
462 '
500 CLS:PRINT"PRESS ENTER TO START A NEW"
505 PRINT"GAME ANYTIME. USE NUMBERS"
510 PRINT"1 TO 9 TO SELECT YOUR"
515 PRINT"SQUARE. THEN PRESS SPACEBAR"
520 PRINT"TO MARK THE SPOT. X ALWAYS"
525 PRINT"GOES FIRST!"
530 RETURN
540 END
```

Precision Floating Point Numbers

What They Are
and
Why You Might Need Them

by
Robert Gault

Floating Point Numbers

There was a thread on the Maltedmedia Coco newsgroup about high precision floating point numbers. A poster requested code for narrow range ultra high precision without indicating why that amount of precision was needed. One reader suggested that the requested precision was extreme overkill and could serve no useful purpose. I suggested that perhaps it was to be used for something like Mandelbrot calculations. This engendered a response equivalent to what's a Mandelbrot when it grows up?

So, just what are floating point numbers and what does precision mean in this context? You are all familiar with decimal numbers when dealing with money; $2.99 per gallon of gas. There can be any number of digits before or after the decimal point; 1234567890.0987654321 . Numbers like this or larger can be cumbersome but they can be represented in a shorter format. For example 12,345,000,000 can be represented by 1.2345×10^{10} and 0.0000000001 by 1×10^{-10}. The decimal point has "floated" to a new position. The Coco replaces x10 with E so 1×10^{-10}=1E-10.

When there is no limit on the number of digits before or after the decimal point, the precision is infinite. Computers have limited precision which means numbers with more digits than permitted by the operating system get rounded off or truncated. For example if your hardware can only work with four digits, then 1.23456 will get either truncated to 1.234 or rounded to 1.235.

On the Coco, Basic stores floating point numbers in six bytes of memory. Four bytes are used for the mantissa, a fifth for the exponent, and the sixth for the sign.

This restricts the range of values to 1E-39<N<1E+39 with 9 digits of precision. Larger values are truncated which can be seen when you try to print them.

```
10 PRINT 0.1234567890123456789
20 PRINT 1234567890123456789
```

results in

```
 .123456789
1.23456789E+18
```

The smallest gap between numbers represented by four bytes or 32 bits is 2^{-31} or 0.00000000005 (5E-10). Basic09 uses the same range of numbers and will not let you enter more than 10 digits in the mantissa without generating an error. The tenth digit gets thrown away in any case. So if you enter a program line of LET a:= 1234567899 and list it, you will see LET a:= 123456789.

To get additional precision it is necessary to use a different language. On the Coco the C language offers both single (float) and double precision numbers. The float uses three mantissa bytes while the double uses seven, two more than Basic or Basic09. The range of the numbers has stayed the same 1E-39<N<1E+39 but for the double the smallest gap between numbers represented by seven bytes / fifty six bits is 2^{-55} or 3E-17. This is seven more digits of precision than provided by Basic or Basic09.

Why Does Precision Matter?

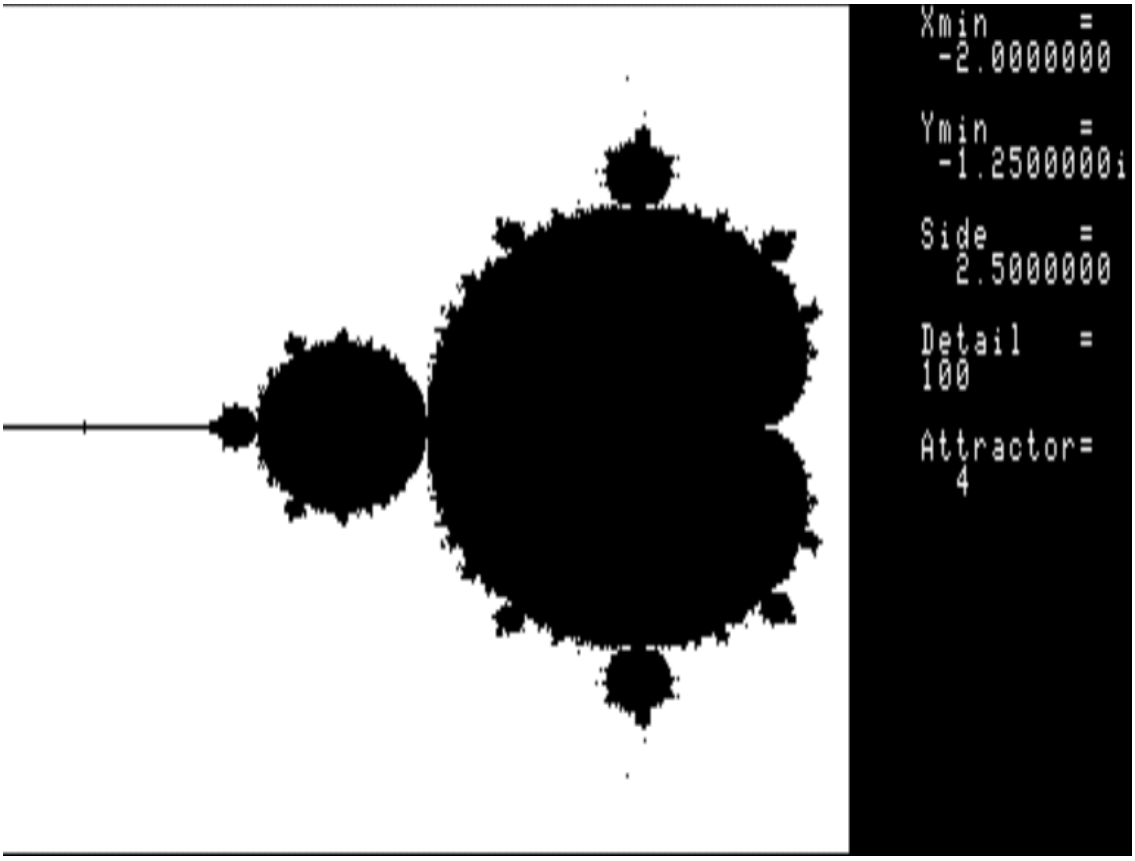
Suppose you were assigned a bookkeeping job but had a calculator that could not use fractional numbers. That is there was no way to represent fractions of a dollar. If you had to add several million entries, you could be off by almost one million dollars. For example $1.99 x 1,000,000 = $1,990,000 but your calculator would show $1,000,000. The error would be $990,000 caused by truncation of the 99¢.

Complex strings of mathematical operations can lead to extremely large errors unless great pains are taken to eliminate them. When the errors in data become large enough, no compensation can help. In similar fashion, limited precision is equivalent to having large errors in data.

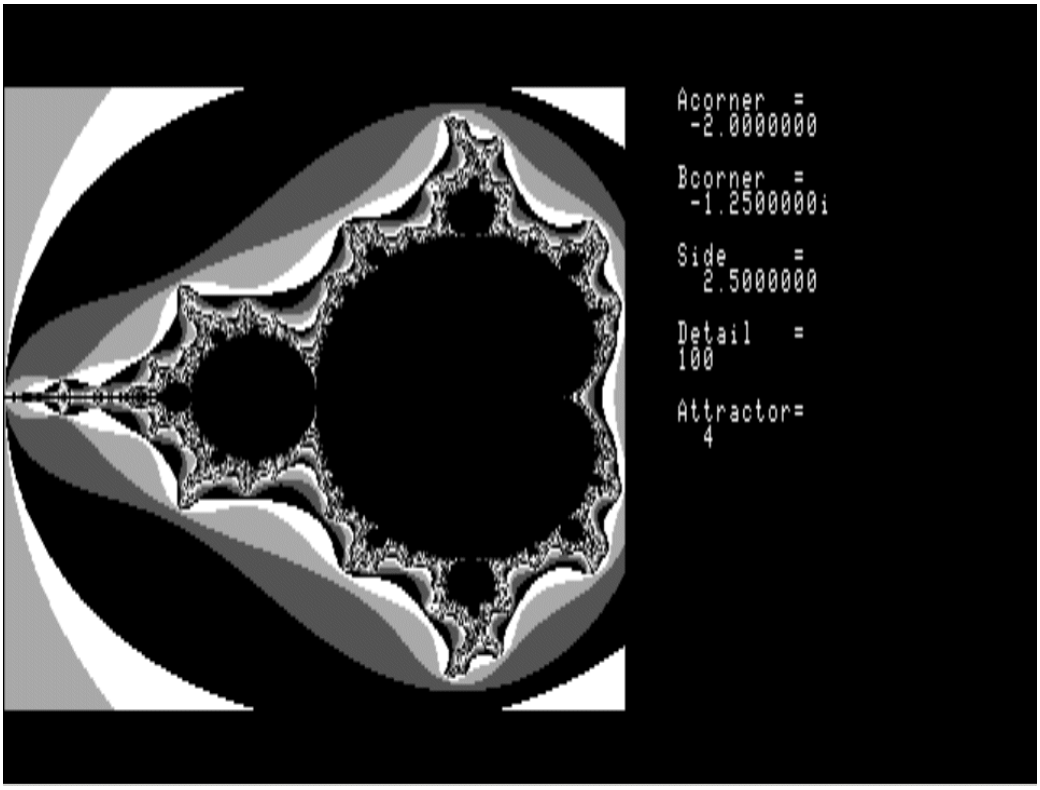
The Mandelbrot Set

Mandelbrot is a mathematician who, among other things, studied a special function; $Z_1=Z_0+C$. The formula is iterative, exists in the complex number plane, and starts with $Z_0=0+0i$. For any particular value of C, if Z_n is finite after an infinite number of iterations, then C is part of the Mandelbrot set of numbers.

I doubt that very many people would care about this if Mandelbrot had not decided to plot the set of numbers on a computer as a graphics image.

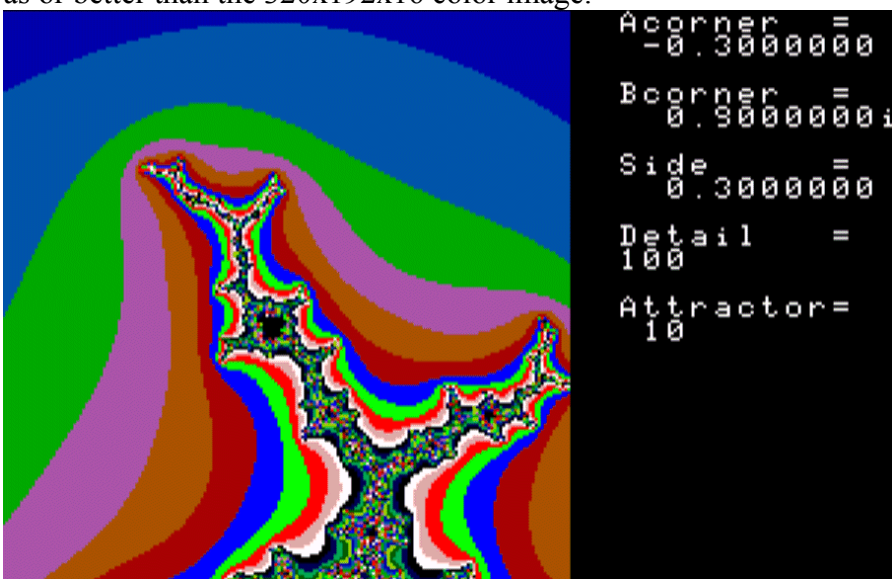


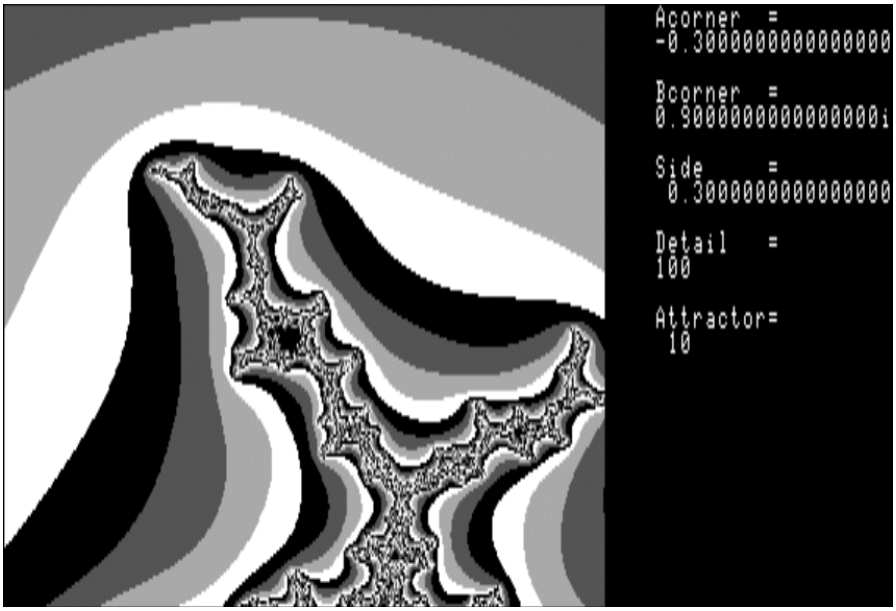
As if this image were not interesting enough, as you zoom in on smaller and smaller areas at the edge of the set, the shapes become ever more complex. Change the computer program so that values approaching the edge of the above pattern are given colors depending on distance from the set and the image becomes a work of art.



When the above images and a method for creating them were published in Scientific American[1], Mandelbrot became famous for his set of numbers and a household name. Look through support magazines for the Coco and you will find articles on generating Mandelbrot images. This also spawned general interest in fractals (the above image is one) and Coco programs to create fractal patterns were also published.

What makes this relevant to high precision floating point numbers is that attempts to zoom in on interesting areas of the above image, rapidly run into truncation errors using low precision numbers that are easily seen in the image. Basic or Basic09 numbers are quite limited in this fashion. Unfortunately the extremely low vertical screen resolution versus horizontal on the Coco just makes matters worse; as does the sixteen color palette. I think the 640x192x4 gray scale image looks as good as or better than the 320x192x16 color image.





The precision of the C double is good enough to produce very nice images with high zoom factors. Generating these images is a good example of requiring very high precision over a limited range, -2.0 to +0.5 and –1.5i to +1.5i. In fact, the speed of the Coco in running the calculations with double precision numbers will then limit what you can look at. It most definitely helps to run the following programs on an emulator set for maximum speed. Even on an emulator, expect several hours per image.

PROGRAMS

Three programs were used to generate the above images on MESS running Coco3 emulation. MESS made it possible to capture the images and the use of a PC shortened the creation time by many hours. The following source code (which will run under OS-9/NitrOS-9) combines all programs into one but needs to be recompiled with a few line changes to recreate the three programs.

```
/* MANDELBROT GRAPHICS GENERATOR by Robert Gault  (c) 1988
 C language source code
 formulas used:
  z(k+1)=z(k)^2 + c where z is a complex number and z^2 is defined
  z=x+iy c=p+iq    x(k+1)=x(k)^2 - y(k)^2 + p
                   y(k+1)=2*x(k)*y(k)+q
  gapx=side/384  gapy=side/192 rc=acorner+x*gapx ic=bcorner+y*gapy
  the math is coded as: realx=z1=x(k+1) rc=p ic=q imagy=y(k+1)
  x(0)=y(0)=0; for maximum accuracy both count and attractor >>0
*/

#include <stdio.h>
direct double acorner, bcorner, side, gapx, gapy, rc, ic, realx,
imagy;
direct double sqrx, sqry, attractor,dist;
char setcolor [] = "\x1b\x31\0\0";
char curoff [] = "\5\x20";
/* The next two lines are filled in by the program and sent to a
window
as a block of data. */
direct char forcolor [] = "\x1b\x32\0";
direct char point [] ="\x1b\x42\0\0\0\0";
direct int *xpoint,*ypoint;
direct int x,y,color,detail,total,path,i,j;
direct char ii,jj;

brkpt (sig)
char sig;
 {
  write(1,"\x1b\x21\n",3);   /* return to standard output */
  exit (0); }

main()
{
/* RGB color codes
Select the correct line for black/white, 4 shades gray, or 16 colors.
*/
/*
 static char palette [] ={63,0};
 static char palette [] ={7,56,63,0};
*/
```

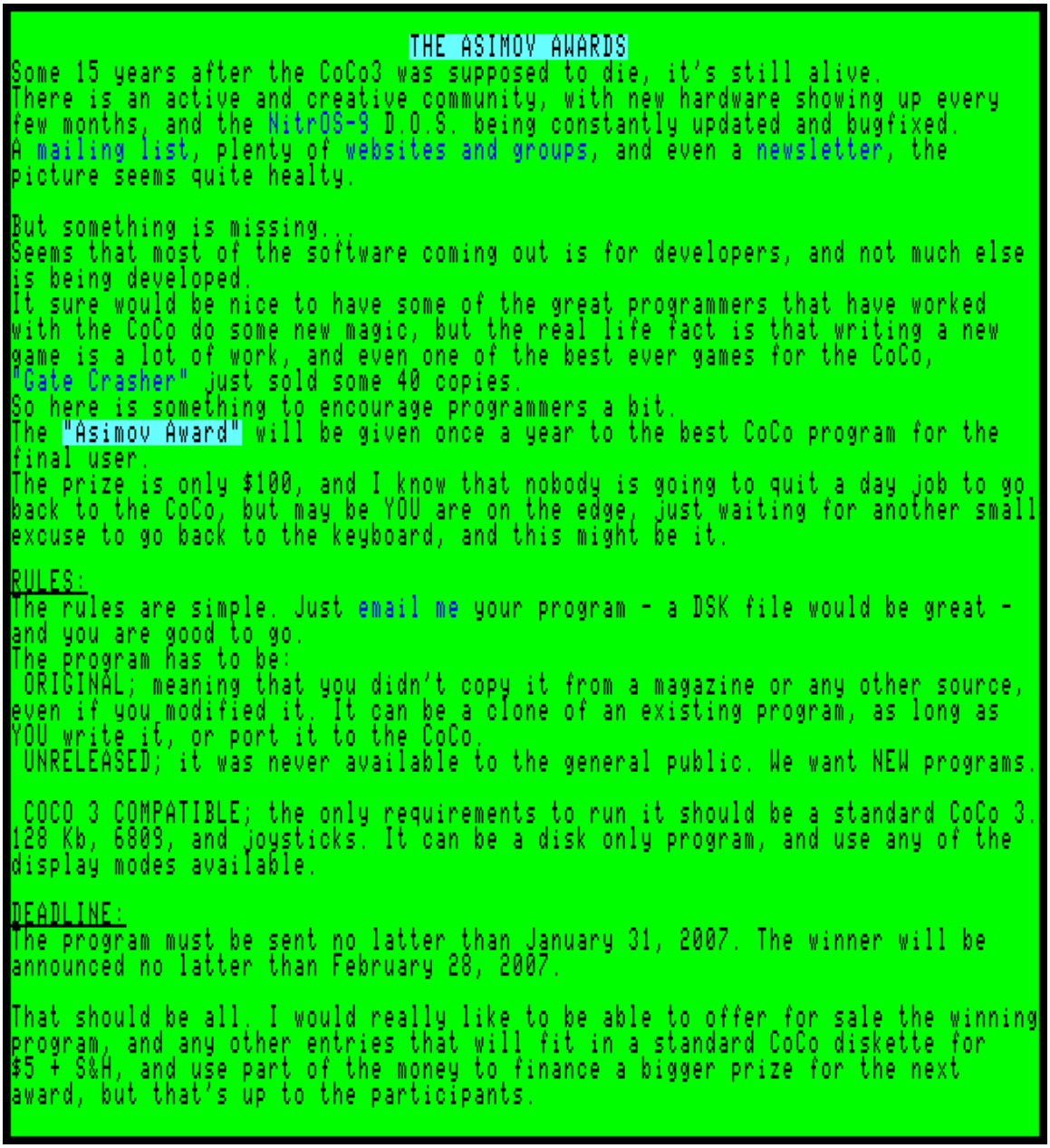
```
 static char palette []
={1,17,2,50,8,10,16,42,34,32,9,18,36,60,63,0};
 register int count;  /* reg U inner loop counter */
 char *window ="/w2";
 pffinit();  /* allow double floating point printing */

     intercept(brkpt);
     path = open (window,3);
     if (system("xmode /w2 -pause\n\3")) {
        printf ("\xcXmode must be loaded or in the CMDS
directory.\n");
        exit(0); }
     printf ("\xcRecommended values ()");
     printf ("\nThe value for LeftEdge (-2) = ");
     scanf ("%F",&acorner);   /* F = double floating point */
     printf ("\nThe value for LeftBottom (-1.25) = ");
     scanf ("%F",&bcorner);
     printf ("\nThe value for Side (2.5) = ");
     scanf ("%F",&side);
     do
      {printf ("\nThe value for Detail\n>=10 & <=1000 (100)\n\nvalue
= ");
       scanf ("%d",&detail);}
     while (detail<10 || detail>1000);
     do
      {printf ("\nThe value for xsqr+ysqr = Limit\n>=4 & <=100
(4)\n\nLimit = ");
       scanf ("%F",&attractor);}
     while (attractor>100.0  || attractor<4.0);

/*
   640x192x2 window
     if (10!=write(path,"\x1b\x20\x5\x0\x0\x50\x18\x1\x1\x1",10)) {
   640x192x4 window
     if (10!=write(path,"\x1b\x20\x7\x0\x0\x50\x18\x3\x3\x3",10)) {
*/
/* 320x192 16 color window */
     if (10!=write(path,"\x1b\x20\x8\x0\x0\x28\x18\xf\xf\xf",10)) {
         printf ("Window /W2 is not available at this time.\n");
         exit(); }
     write (path,curoff,2);

 /* redirect standard output to window for printf; write uses path #
*/
     window = freopen(window,"r+",stdout);

/* Set constants for the aspect ratio of your monitor.
You can draw a square on a 640x192 window and see if it truly is
square.
The ratio of the sides is used below to determine the value in gapx.
*/
     gapy = side/192.;   /* set for 1:1   height:width aspect */
     gapx = side/384.;

/* The next line uses 2, 4, or 16 depending on the size of the
palette for respectively black/white, gray scale, or colors. */
     for (ii = 0; ii <16; ++ii) {
         setcolor [2] = ii;
         setcolor [3] = palette[ii];
         write (path,setcolor,4); }
```

```
/* adjust forground */
     write (path,"\x1b\x32\xe",3);

/* For 640H screen, change all x3a to x54. Also change all %11.10 to
%17.16 */
     printf ("\xc\2\x3a\x20Acorner  =\x2\x3a\x21%11.10f\n",acorner);
     printf    ("\2\x3a\x23Bcorner  =\x2\x3a\x24%11.10fi\n",bcorner);
     printf    ("\2\x3a\x26Side     =\x2\x3a\x27%11.10f\n",side);
     printf    ("\2\x3a\x29Detail   =\x2\x3a\x2a%d\n",detail);
     printf    ("\2\x3a\x2cAttractor=\x2\x3a\x2d%3.0f\n",attractor);


/* switch to graphics screen at path # */
     write (path,"\x1b\x21\xd",3);


/* enter main program: Mandelbrot Set with contour lines color coded
*/
     xpoint=&point[2];
     ypoint=&point[4];

/* For 640H screen change the x+=2 to x+=1. */
     for (x = 0; x <384; x+=2) {
         rc = x * gapx + acorner;
         *xpoint=x;
         for (y = 0; y <192; ++y) {
/* Screen 0,0 is upper left. Convert to normal graph coords at lower
left.*/
             *ypoint=191-y;
             realx = imagy = sqrx = sqry = 0;
             ic = y * gapy + bcorner;

             for (count = 1; count <= detail; ++count) {

                  imagy = imagy * realx * 2 + ic;
                  realx = sqrx - sqry + rc;
                  dist  = (sqrx=realx*realx)+(sqry=imagy*imagy);

                   if(dist>attractor){

/* When running 640x192x2 remove the next 7 lines through forcolor.
*/
                     if(count>255)
                       i*=.01;
                       else if(count>50)
                         i*=.1;
                         else
                           i = count;
                     forcolor [2] = i&15;

                      write (path,forcolor,10);
                      break;}

                  }
             }
         }
    pause(); /* keep picture until seen; may need save routine */
    write(1,"\x1b\x21",2); /* return to stdout */
    }
```

The above program has limited error trapping so don't deliberately make illegal entries. If you want to enhance the program, think about ways to make data entries by outlining an area of the screen with the joystick.

For those readers that don't have C, you should be able to convert this program to Basic or Basic09 without too much effort. That will however, reduce the ability to zoom in on details of the image.

1. Dewdney, A.K. ``Computer Recreations.'' Scientific American, August 1985, December 1986, July 1987, November 1987, February 1989, May 1990



A Basic09 Tutorial
by Bob Devries

Another problem area for programmers recently converted to Basic09 is the TYPE statement. This is used when a programmer needs to lump together several variables to be referred to as one unit. Let me give you an example. Say I want to write a little database programme (as my earlier version in C) to keep names, addresses, and phone numbers. Here's what the start of the programme would look like:

```
PROCEDURE Program
TYPE record=surname:STRING[20]; firstname:STRING[20]; street:STRING [20];
city:STRING[20]; state:STRING[3]; postcode:INTEGER; area:STRING [3];
phone:STRING[7]
DIM address:record
```

So here I have the same database record as I have used previously (a series of articles starting March 1989). A database entry is a complex variable called address of TYPE record. That is, the variable record has in it all the variables referred to in record. So to refer to the city field in the database entry, I would call it address.city, easy see?

If I want to fill each of the variables of the complex variable address, I could do this:

```
address.surname = "DEVRIES"
address.firstname = "BOB"
address.street = "PO Box 319"
address.city = "DALBY"
address.state = "Qld"
address.postcode = 4405
address.area = "07"
address.phone = "46696412"
```

If I want to write a database entry to a disk file, I would merely do this:

```
PUT #file, address
```

This will put all the variables which make up the complex variable address into the diskfile one after the other. Of course the diskfile must have been opened first.

Similarly, to read an existing entry from a diskfile, I would use this line:

```
GET #file, address
```

If the disk file was 20 entries long, and I wanted to get the fifteenth one, I would first seek to the fourteenth (all records start at the zeroeth) record like this:

```
SEEK #file,14 * SIZE(address)
```

Then I would read the entry as before.

Here is a sample piece of programme which sets up the database record in memory using TYPE, and fills it, and then displays it in an overlay window. One thing you should be aware of, you MUST initialise variables in Basic09, because all variables are filled with garbage after being dimensioned.

```
PROCEDURE Program
TYPE record=surname:STRING[20]; firstname:STRING[20]; street:STRING [20];
city:STRING[20]; state:STRING[3]; postcode:INTEGER; area:STRING [3];
phone:STRING[7]
DIM address:record
DIM file:INTEGER
DIM a:STRING[1]

PRINT CHR$(12)

address.surname="Bentzen"
address.firstname="Gordon"
address.street="8 Odin Street"
address.city="Sunnybank"
address.state="Qld"
address.postcode=4109
address.area="07"
address.phone="3443881"

RUN gfx2("OWSet",1,9,4,32,11,1,0)
RUN gfx2("OWSet",0,10,5,30,9,0,1)

PRINT "Surname:";
PRINT address.surname
PRINT "Firstname:";
PRINT address.firstname
PRINT "Street:";
PRINT address.street
PRINT "City:";
PRINT address.city
PRINT "State:";
PRINT address.state
PRINT "Postcode:";
PRINT USING "i5",address.postcode
PRINT "Area code:";
PRINT address.area
PRINT "Phone:";
PRINT address.phone
a=""
WHILE a="" DO
RUN inkey(a)
ENDWHILE
```

```
OPEN #file,"DATABASE":UPDATE
SEEK #file,0
PUT #file,address
CLOSE #file

RUN gfx2("OWEnd")
RUN gfx2("OWEnd")
```

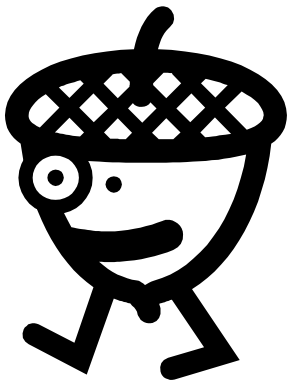
I'll give you a run-down on what is in this programme.

First, we look at the TYPE command; setting up the memory image of the database record. Next, dimension the complex variable, as well as some other useful variables. Then I filled the various parts of the complex variable with data for one record of the database with Gordon's name, address etc. No doubt you'll understand now that to access each part of the database record, its identifier is 'address.xxxxxxx' where the x's are the various sections of the record, e.g. city.

Now to display the record, I open an overlay window big enough to display the fields of the record, and print them. Notice the use of PRINT USING for the postcode field. For this, you must use a format length of one more than the length of the variable, hence, 'i5'. Next I wait for a keypress before closing the overlay, and writing the record, and quitting. In this example, the record is always written to position zero of the file.

Next month, I'll show you how to convert programmes from other BASIC languages, including RSBasic, GWBasic etc. I'll include a working example, in both the original format, and the converted Basic09 programme.

Regards, Bob Devries



“Wipeout Revisited”

Written by Mark McDougall

Back in Volume 1, Issue 2 Richard Kelly presented “Wipeout” – a program to wipe a screen randomly pixel-by-pixel. He presented a technique to ensure pixels are only chosen once, so that the program does not stall waiting to randomly choose the last few pixels on the screen.

The purpose of this article is to introduce an alternate technique that avoids the original problem altogether, and at the same time requiring absolutely no storage (arrays) at all. Without having to access arrays, generation is much faster - when implemented purely in BASIC, the program was benchmarked at 26s to generate the random sequence, whilst in pure assembler, it runs in about 1/10s.

Of course the generation of such random sequences need not be limited to wiping the screen – they also find utility in all sorts of game programming. One example would be shuffling a deck of cards, where you can’t deal the same card again until you’ve been through the entire deck.

Linear Feedback Shift Registers

The Linear Feedback Shift Register (LFSR) has long been used in the generation of pseudo-random sequences. Plenty of information on LFSRs can be found on the internet so I’ll only delve into them briefly here. Basically an LFSR is simply a shift-register with a feedback term that is calculated from the current round and fed into the input of the shift register in the next round. The feedback term is the exclusive-OR of a number of specific bit-positions (called ‘taps’) of a given LFSR.

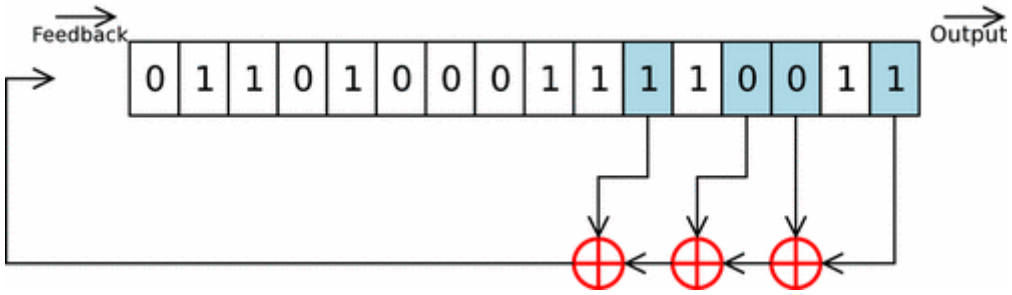


Figure 1 – LFSR with taps in bit positions 11, 13, 14 & 16

The upshot of all this is that an LFSR seeded with a random value will produce a ‘pseudo-random’ output each time it is ‘clocked’. Note however that eventually it will repeat the sequence ad-infinitum.

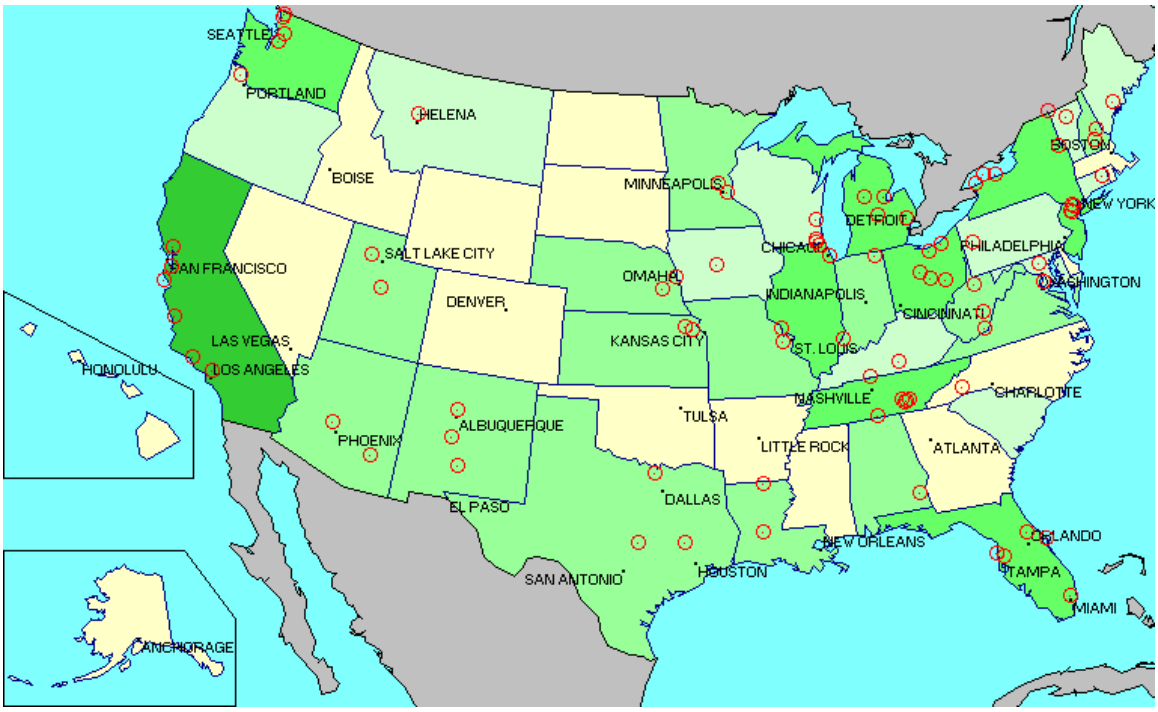
For our purposes we’re actually interested in a specific type of LFSR called a Maximal Length LFSR. A Maximal Length LFSR of N bits produces a unique sequence of 2^{N-1} numbers (the complete range of numbers, excluding zero) before repeating. In contrast, some LFSRs repeat well before they reach 2^{N-1} numbers.

Wipeout

In order to clear 512 screen locations, we need a 9-bit Maximal Length LFSR, which repeats after generating 511 unique numbers (1-511). It turns out that there are many different 9-bit Maximal Length LFSRs (people have compiled lists of them on the net) – but the simplest (and hence quickest to calculate) only has 2 taps in the feedback term – bit positions 9 and 5 (bits 8 and 4).

LFSRs are trivial to implement in hardware. In assembler, they're not too painful to implement either. Unfortunately, as I discovered soon after deciding to research this article, Coco BASIC is about the **worst** possible language you could choose to implement them! There's absolutely no bit-manipulation functions whatsoever, and we really could use bit-shift and bit-wise XOR.

However all is not lost, and although there's some terribly messy and inefficient maths to do, it's still several times faster than having to access arrays.



Where are all the CoCoNuts??????????????????????

The code to the right takes about 26s to 'wipe' the screen. Note that position 0 must be explicitly wiped because an LFSR can't produce the value 0. For simplicity's sake, I decided to wipe position 0 at the start of the procedure (line **15**). Also, I've chosen the value 66 to seed the LFSR (simply because it's '42' in HEX). Ideally, you would choose a random seed using RND() for example.

```
10 L=66
15 POKE 1024,128
20 FORI=1TO511
30 C1=INT(L/256)
40 C2=INT((L-INT(L/32)*32)/16)
50 L=L*2
60 IF C1<>C2 THEN L=L+1
65 IF L>511 THEN L=L-512
100 POKE 1024+L,128
200 NEXT I
210 IF INKEY$="" THEN 210
```

Listing 1 – Pure BASIC implementation

Lines **30** & **40** look pretty cryptic, but all they're doing is extracting bits 8 and 4 from the current LFSR value (L). Even the MOD() function would've been useful here! Line **50** does the shift whilst line **60** calculates the feedback term by in effect XORing the taps (bits 8, 4) and then feeds it back into the input of the shift register.

TIP: Just cut-and-paste the listing into MESS!

Speeding things up

Curiosity got the better of me and I decided to see how much improvement I could get by implementing the LFSR (lines **30-65** in Listing 1) in assembler.

```
Intcnv: equ   $b3ed       ; convert FPA0 to int (D)
givabf: equ   $b4f4       ; convert int (D) to FPA0

start:  equ   $2000
        org   start

        lbsr  intcnv      ; convert USR arg to int
        tfr   d,x         ; save LFSR
        aslb
        rola              ; left-shift
        exg   d,x
        lsrb
        lsrb
        lsrb
        lsrb              ; get bit 4
        stb   carry,pcr   ; temp store for XOR
        eora  carry,pcr   ; XOR with bit 8
        bita  #1          ; feedback term
        tfr   x,d         ; restore LFSR
        beq   skip
        addd  #1          ; add feedback term
skip:   anda  #$1         ; discard overflow bits
        jmp   $b4f4       ; store (FP) return value

carry:  rmb   1

        end   start
```

Listing 2A – Assembler Implementation of the LFSR

The shift register itself is implemented in only a handful of lines of assembler – that should run rings around the multiple floating point operations in the BASIC code. Disappointingly, this code wipes the screen in about 8s.

Turns out that for each USR0() call the routine must convert the argument

from floating point to integer, then back again to pass the value back to BASIC. Technically, the current LFSR value could be stored internally and not be passed in from BASIC, but that had no perceptible affect on the performance. It would seem that the limiting factor is how quickly the basic interpreter can execute the remainder of the code.

Listing 2B is a BASIC program that runs the assembler implementation of the LFSR.

```
10 A=8192
20 I=0
30 READ D:IF D=256 THEN 100
40 POKE A+I,D
50 I=I+1
60 GOTO 30
100 DEFUSR0=A
110 L=66
120 CLS:POKE 1024,128
130 FOR I=0 TO 511:L=USR0(L):POKE
1024+L,128:NEXT I
998 IF INKEY$=""THEN 998
999 END
1000 DATA 23,147,234
1001 DATA 31,1
1002 DATA 88
1003 DATA 73
1004 DATA 30,1
1005 DATA 84
1006 DATA 84
1007 DATA 84
1008 DATA 84
1009 DATA 231,141,0,18
1010 DATA 168,141,0,14
1011 DATA 133,1
1012 DATA 31,16
1013 DATA 39,3
1014 DATA 195,0,1
1015 DATA 132,1
1016 DATA 126,180,244
2000 DATA 256
```

Listing 2B – Run Assembler LFSR

Finally, I implemented the entire 'wipe' routine in assembler. Without any delay loop, it runs in no time at all – my estimate is about 1/10 of a second. I had to add a delay to get a visible 'wiping' effect.

```
start:  equ   $2000
        org   start

        ldd   #511
        std   count,pcr   ; loop counter
        ldx   #$0042      ; LFSR seed

loop:   equ   *
        tfr   x,d         ; get LFSR
        aslb
        rola              ; left-shift
        exg   d,x
        lsrb
        lsrb
        lsrb
        lsrb              ; get bit 4
        stb   carry,pcr   ; temp store for XOR
        eora  carry,pcr   ; XOR with bit 8
        bita  #1
        tfr   x,d         ; get LFSR
        beq   nofb
        addd  #1          ; add feedback term
nofb:   anda  #1          ; discard overflow bits
        tfr   d,x         ; save new LFSR value
        anda  #$1
        ora   #$4         ; construct screen address
        tfr   d,y
```

```
        lda   #$80
        sta   ,y          ; poke black square

        lda   #$80        ; delay value
delay:  deca
        cmpa  #$00
        bne   delay

        ldd   count,pcr   ; decrement loop counter
        subb  #1
        stb   count+1,pcr
        bcc   loop
        suba  #1
        sta   count,pcr
        bcc   loop
        rts

count:  fdb   0
carry:  rmb   1

        end   start
```

Listing 3A – Pure Assembler Implementation of 'WIPEOUT'

Here's a BASIC listing that runs the above code.

```
10 P=8192
20 READ D:IF D=256 THEN 40
30 POKE P,D:P=P+1
35 GOTO 20
40 DEFUSR0=8192
45 CLS:POKE 1024,128
50 D=USR0(0)
98 IF INKEY$="" THEN 98
99 END
1000 data 204,1,255
1001 data 237,141,0,72
1002 data 142,0,66
1003 data 31,16
1004 data 88
1005 data 73
1006 data 30,1
1007 data 84
1008 data 84
1009 data 84
1010 data 84
1011 data 231,141,0,57
1012 data 168,141,0,53
1013 data 133,1
1014 data 31,16
1015 data 39,3
1016 data 195,0,1
1017 data 132,1
1018 data 31,1
1019 data 132,1
1020 data 138,4
1021 data 31,2
1022 data 134,128
1023 data 167,164
1024 data 134,128
1025 data 74
1026 data 129,0
1027 data 38,251
1028 data 236,141,0,17
1029 data 192,1
1030 data 231,141,0,12
1031 data 36,196
1032 data 128,1
1033 data 167,141,0,3
1034 data 36,188
1035 data 57
1036 data 0,0
2000 DATA 256
```

Listing 3B – Run Assembler 'WIPEOUT'

BTW all my assembler in the above examples should be position-independent code, which means one could use the same DATA statements and relocate the code to another address, and reserve memory for it properly, for example.

Other uses of the LFSR

As I mentioned earlier, pseudo-random sequences generated by LFSRs have many other uses in programming. Whilst maximal length LFSR are particularly useful when generating sequences of unique numbers - it is important to recognise their limitations as well. I mentioned using a maximal length LFSR to shuffle (deal) a deck of cards. One would simply choose a random seed for the LFSR and deal the deck from there. However, it should be obvious that this would result in a rather limited number of scenarios (<64 in fact) because given a particular seed, the sequence never varies. How to deal with such a problem is outside the scope of this article.

When using this technique as we have to wipe a screen however, the limited variability is not so important – it's not even perceptible in this case.

I last came across an LFSR when studying the circuit schematics of the arcade game Galaxian. If you fire up MAME and have a close look at the scrolling star field, you'll see that it is far from random – the stars scroll off the bottom of the screen and back onto the top, shifted across by 1 pixel.

Galaxian uses a maximal length LFSR (with taps at bits 4 & 16), implemented in hardware, to generate the star field pattern. The LFSR is clocked for each pixel drawn during the video raster scan. However, rather than using the LFSR as the screen location as we did in WIPEOUT, instead the LFSR value determines the pixel colour. The exact logic used is designed to paint a **black** pixel most of the time, otherwise choose a random colour for a star (I won't go into the 'blinking effect' here).

Furthermore, the scrolling effect comes 'for free' and is a direct by-product of the fact that the period of the LFSR is **one less** than the number of pixels on the screen.

Finally, I've implemented a crude 'Star Field Scroller' to illustrate the effect. Note that due to the orientation of the Coco monitor, the field scrolls right-to-left rather than top-to-bottom. Nonetheless, it gives an idea of how it works in Galaxian.

```
start:  equ   $2000
        org   start

        ldx   #$0042      ; LFSR seed
        ldy   #$0400      ; screen offset

loop:   equ   *
        tfr   x,d         ; get LFSR
        aslb
        rola              ; left-shift
        exg   d,x
        lsrb
        lsrb
        lsrb
        lsrb              ; get bit 4
        stb   carry,pcr   ; temp store for XOR
        eora  carry,pcr   ; XOR with bit 8
        bita  #1
```

```
        tfr   x,d          ; get LFSR
        beq   nofb
        addd  #1           ; add feedback term
nofb:   anda  #1           ; discard overflow bits
        tfr   d,x          ; save new LFSR value
        lda   #$80         ; black square
        bitb  #$2E         ; should we draw a star?
        bne   skip         ; no, skip
        addb  #5
        comb
        orb   #$8F         ; come up with a pretty colour
        tfr   b,a
skip:   equ   *
        sta   ,y+          ; poke black/coloured square
        tfr   y,d
        anda  #$05         ; wrap screen address at $05FF
        tfr   d,y
        bra   loop

carry:  rmb   1

        end   start
```

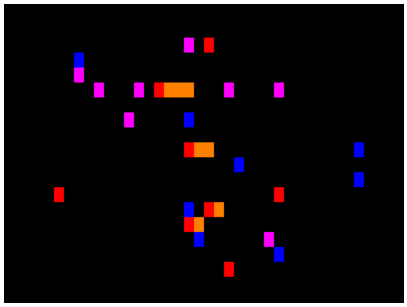
Listing 4A – Assembler source for Scrolling Star Field

And, so you can see it running in a matter of seconds:

```
5 I=8192
10 READ X
15 IF X=256 THEN 100
20 POKE I,X
30 I=I+1
40 GOTO 10
100 DEFUSR0=8192
110 U=USR0(0)
200 END
1000 DATA 142,0,66
1001 DATA 16,142,4,0
1002 DATA 31,16
1003 DATA 88
1004 DATA 73
1005 DATA 30,1
1006 DATA 84
1007 DATA 84
1008 DATA 84
1009 DATA 84
1010 DATA 231,141,0,40
1011 DATA 168,141,0,36
1012 DATA 133,1
1013 DATA 31,16
1014 DATA 39,3
1015 DATA 195,0,1
1016 DATA 132,1
1017 DATA 31,1
1018 DATA 134,128
1019 DATA 197,46
1020 DATA 38,7
1021 DATA 203,5
1022 DATA 83
1023 DATA 202,143
1024 DATA 31,152
1025 DATA 167,160
1026 DATA 31,32
1027 DATA 132,5
1028 DATA 31,2
1029 DATA 32,202
2000 DATA 256
```

Listing 4B – Run 'Scrolling Star Field'

BTW, this routine will never return to BASIC so you'll have to RESET your Coco/emulator to get out of it. You should see something like this…



Some tips

Here's a few tips I learnt when writing this article. I haven't written a Coco program in – quite literally – 20 years (as is probably evident by my 6809 code), so I was loathe to ramp up on EDTASM etc.

- I found a free cross-assembler (A09) and wrote my machine code on the PC and assembled under A09 to produce a listing. I then wrote a very small C program to parse the .LST output and display numbered lines of BASIC DATA statements. I then fired up MESS, loaded the most recent version of my BASIC test program, and then simply cut-and-pasted the lines from my command prompt into ECB – overwriting the old DATA statements from the last version.
- You can use the same method to cut-and-paste the entire BASIC listings from this article into MESS. No typing, no messing with virtual disk utilities.
- My debugging was all done on the DEBUG build of MESS.

Enjoy!

Lonnie Falk Reading Center
By Mary

We all know how much Lonnie Falk loved his books and magazines, so it pleases me to read on the Prospect Kentucky website about a Reading center he opened up for his community. They of course named the city's reading center after him and in memory of him. He was presented with a proclamation honoring the city of Prospect upon the opening of the Lawrence Falk Reading Center from a representative of the County Judge Rebecca Jackson. The reading center was officially opened March 25 2002.

The Lawrence Falk Reading Center is always taking donations of books to fill out the shelves. If any of you have any old books you would like to donate to them, I will be taking the donations and ship them out in one shipment each month. I will pay for the cost of shipping if you cannot afford to send them to me. I would like to see all of you sending at least one book to donate to them in memory of Lonnie Falk and his

passion for learning and reading. He would have continued to stock the shelves if he were still alive. I would also like to donate any extra copies anyone may have of the Rainbow to the library so that other can see what his life before he became Mayor was like.

Special Backup Utility for a Coco3 Using Full 512K RAM

by

Robert Gault

Disks should be backed up frequently to prevent loss of data and as a means of saving multiple versions/revisions of critical projects as a history. However, backups with a single disk drive are actually painful. Even with multiple drive systems, copying disks using the Disk Basic command or the OS-9 command could be a more pleasant procedure.

This utility requires a 512K Coco3, uses the full 512K RAM for the backup, and makes single drive backups simple. The utility has been optimized so that both Disk Basic and OS-9 disk copying is faster than using the stock commands. This speed increase is the result of reading sectors in the order that they exist on the disk rather than in sequential order from 1-18. If you do not use default skip factors when formatting disks, you should modify the sector tables at the end of the source to match your disks. If you do not, the speed of this program can decrease significantly.

This program is not intended for use with emulators but can be made to work with a change to one line in the source code in RDWRDK. It is not intended for use with drive numbers higher than 3 and will need patching to be used with RGBDOS or HDBDOS drives 4 or higher. It would be very easy to extend the program usage to drives from 0-9 by changing one value in KEYTST. More extensive changes would be needed for drive numbers higher than 9.

USAGE

LOADM"BACKUP"
EXEC

You will be asked if your drives are 6ms or 30ms, if the disk is 35 or 40 track, and if it was formatted for Disk Basic or OS-9. You answer with a single key press or BREAK to restart or quit.

You are then asked which drive contains the original disk. Once inserted, the full disk is read into memory and you are then asked which drive contains a blank disk. Once indicated, you are asked to change disks and the backup copy is written. You will then be asked if you want to make more copies of this disk. You can keep inserting new disks to get additional copies without rereading the original.

The program leaves Disk Basic set to the selected stepping speed. This is a side benefit which will speed up all subsequent disk access. While the program is running, you will be in 40 column high res. text mode but the program when finished will return to the screen width found at startup.

SOURCE CODE

The following source code is fully commented and can be assembled with most of the available programs for the Coco. It will be necessary to change some of the labels, if you are using Disk EDTASM. EDTASM does not like A@ local labels although my patch EDTASM6309 does use them.

The program is a good demonstration of how you can use routines already in the ROMs to write text on the high resolution text screens, run disk I/O, sample the keyboard, determine if a Coco is a model 3, how much memory it has, and make use of the MMU to access memory beyond the range of 0-64K.

SKIP FACTORS

Skip factors are described and shown for Disk Basic in the "Color Computer Disk System Owners Manual & Programming Guide" on page 58. In essense, the Disk Basic skip factor of 4 indicates "that the computer should skip 4 'physical' sectors between each 'logical' sector. OS-9 uses a default skip factor of 2.

The purpose of a skip factor is to attempt to coordinate sector locations with the speed of the operating system's disk access so that as soon as the OS needs a sector, it is just reaching the drive head.

Machine language programs being faster than Basic programs typically benefit from smaller skip factors. Backup obtains maximum speed by matching expected default skip factors.

==========

```
            TITLE    BACKUP

************************************************
* This version of Backup is for a 512K Coco3 only!
* An entire disk is read in one pass.
* Both RSDOS and OS-9 skip factors suported for
* maximum speed of disk access.
* Multiple copies can be made with single read.
*
************************************************
* DEFINITIONS
CURSOR      EQU      $FE00    hres cursor
CR          EQU      $0D      screen carriage return
SCREEN      EQU      $2000    mapped location of screen
LINES       EQU      80       bytes per screen line
CHAR        EQU      2        bytes per character
* BASIC entry locations
GETKEY      EQU      $A1B1    get key into regA
PUTCHR      EQU      $A30A    print to screen
* DSKCON variables
DCOPC       EQU      $EA      operation code
DCDRV       EQU      $EB      drive number
DCTRK       EQU      $EC      track number
DCSEC       EQU      $ED      sector number
DCBPTR      EQU      $EE      buffer pointer
DCSTAT      EQU      $EF      error status
HRMODE      EQU      $E6      high resolution graphics mode value
SCRTYP      EQU      $E7      high resolution text mode
* Disk Controller I/O registers
CONTRL      EQU      $FF40
CMDREG      EQU      $FF48
```

```
****************************
* TEMP SPACE
TEMP        EQU     $4A
SCRIMG      EQU     $4B     scrtyp image
LOGSEC      EQU     $4C     logical sector
****************************
            ORG     $E00
****************************

            LBRA    EXEC    execution address

***************

COPYRT      JSR     $F6E0
            LDU     #SCREEN+2*LINES+15*CHAR
            LBSR    HPRINT
            FCC     "BACKUP"
            FCB     0
            LDU     #SCREEN+4*LINES+6*CHAR
            LBSR    HPRINT
            FCC     "Copyright (c) 1990, 2006"
            FCB     0
            LDU     #SCREEN+6*LINES+10*CHAR
            LBSR    HPRINT
            FCC     "By Robert Gault"
            FCB     0
            RTS

*****************************************************************
* GETDRIVE find drive number
* ENTRY: none
* EXIT:  A=drive #; or return to DOS
GETDRV      JSR     GETKEY
            BSR     KEYTST
            BCS     COLD    BREAK key
            BEQ     GETDRV
            JSR     PUTCHR
            SUBA    #'0
            STA     <DCDRV
            LDU     #SCREEN+12*LINES+1*CHAR
            LBSR    HPRINT
            FCC     "Insert disk then press any key."
            FCB     0
            JSR     GETKEY
            BSR     KEYTST
            BCS     COLD
            RTS

*****************************************************************
COLD        LDS     <$DA    get master return
            LDD     #$403A
            STA     $F80F   restore cursor
            STA     $F84F
            STA     $F7A3
            STB     $FFA2
            CLR     <HRMODE
            LDB     <SCRIMG
            DECB
            BEQ     A@
            BPL     B@
            JMP     $F652   WIDTH32
```

```
A@          JMP      $F65C    WIDTH40
B@          JMP      $F679    WIDTH80

*******************************************************
* KEYTEST
* ENTRY: A contains value
* EXIT:  carry set on BREAK  ; A=NA
*        equ   set on illegal; A=0
*        carry clear on legal; A=value
KEYTST      CMPA     #3       BREAK KEY
            BNE      A@
            COMA              set carry
            RTS
A@          CMPA     #'0
            BCS      B@
            CMPA     #'4
            BHS      B@
            ANDCC    #$FE     clear carry bit
            RTS
B@          CLRA              set equals
            RTS

*****************************************************
MMU         PSHS     CC
            ORCC     #$50
            STB      $FFA2    set block at $4000-$5FFF
            LDU      #$4000
            PULS     CC,PC

*****************************************************
* ERROR test error type; print message
* ENTRY: A=error code
* EXIT: message
ERROR       TSTA
            BPL      A@
NTREDY      CLR      CONTRL
            LDU      #SCREEN+18*LINES+5*CHAR
            LBSR     HPRINT
            FCC      "Drive not ready."
            FCB      0
L@          LDU      #SCREEN+20*LINES+5*CHAR
            LBSR     HPRINT
            FCC      "Press any key."
            FCB      0
            JSR      GETKEY
            LDS      <$DA     reset the stack
            LDB      <TEMP
            CMPB     #2
            LBEQ     SDRIVE
            LBRA     DDRIVE
A@          CLR      CONTRL
            LBSR     CLEAR2
            BITA     #$40
            BEQ      B@
            LDU      #SCREEN+18*LINES+5*CHAR
            LBSR     HPRINT
            FCC      "Write protected disk!"
            FCB      0
            BRA      L@
B@          LDU      #SCREEN+18*LINES+5*CHAR
            LBSR     HPRINT
```

```
            FCC     "I/O Error."
            FCB     0
            BRA     L@

*****************************************************************
*
CLEAR       LDU     #SCREEN+10*LINES+1*CHAR
            LBSR    HPRINT
            FCC     /                                       / 40
SPACES
            FCB     0
            LDU     #SCREEN+11*LINES+1*CHAR
            BSR     HPRINT
            FCC     /                                       / 40
SPACES
            FCB     0
            RTS

CLEAR2      PSHS    A
            LDU     #SCREEN+18*LINES+1*CHAR
            BSR     HPRINT
            FCC     /                                       / 40
SPACES
            FCB     0
            PULS    A,PC

*************************************************************
HPRINT      STU     CURSOR
            PULS    X       pull text location into X
A@          LDA     ,X+
            BEQ     B@
            JSR     PUTCHR
            BRA     A@
B@          TFR     X,PC    jump to next opcode

*************************************************************

EXEC        CLR     CONTRL  stop drives
            STS     <$DA    save stack pointer
            LDX     $FFFE
            CMPX    #$8C1B  Reset on Coco3
            BEQ     A@
            LEAX    <M@-1,PCR
            JMP     $B99C   Print string to screen
M@          FCC     "SORRY! THIS IS NOT A COCO3."
            FCB     CR
            FCC     "THIS PROGRAM WON'T RUN!!"
            FCB     CR,0
* test for 512K memory
A@          ORCC    #$50
            LDD     #$3A30  MMU blocks
            STB     $FFA2   $4000-$5FFF; 128k 0 page
            CLR     $5000
            CLR     $FFA2   512k $00000
            STB     $5000   will change data in a 128K coco
            STB     $FFA2
            LDB     $5000   test for ghosting
            STA     $FFA2   reset MMU
            ANDCC   #$AF
            TSTB            0=512K
            BEQ     B@
```

```
            LDU     #SCREEN+18*LINES+10*CHAR
            BSR     HPRINT
            FCC     "Sorry! You need 512K"
            FCB     CR
            FCC     "to run this program."
            FCB     CR,0
            RTS
B@          ANDCC   #$AF
            LDB     <SCRTYP
            STB     <SCRIMG
            CLRA
            STA     $F80F    remove underline
            STA     $F84F       "      "
            STA     $F7A3       "      "
            JSR     $F65C    start 40 column screen
            JSR     $F6E0    clear screen
            LDU     #SCREEN+10*LINES+1*CHAR
            LBSR    HPRINT
            FCC     /Select <F>ast 6ms or <S>low 30ms drives/
            FCB     0
            LDU     #SCREEN+11*LINES+15*CHAR
            LBSR    HPRINT
            FCC     /Speed ?/
            FCB     0
L@          JSR     GETKEY
            CMPA    #'F
            BNE     C@
            CLRB
            BRA     D@
C@          CMPA    #'S
            BNE     L@
            LDB     #3
D@          LDY     $C004
            PSHS    B
            STB     <$61,Y   restore
            LDA     $B7,Y    seek
            ANDA    #$FC
            ORA     ,S+
            STA     $B7,Y

L@          JSR     $F6E0    clear screen
            LDU     #SCREEN+10*LINES+1*CHAR
            LBSR    HPRINT
            FCC     "Select <3>5 or <4>0 track drives."
            FCB     0
            LDU     #SCREEN+11*LINES+11*CHAR
            LBSR    HPRINT
            FCC     "# of tracks ?"
            FCB     0
A@          JSR     GETKEY
            CMPA    #3
            LBEQ    COLD     return to Basic
            CMPA    #'3
            BEQ     B@
            CMPA    #'4
            BNE     A@
            LDB     #40
            STB     TRACK+1,PCR
B@          JSR     PUTCHR
            LDU     #SCREEN+14*LINES+1*CHAR
            LBSR    HPRINT
```

```
            FCC     "OS-9 Disk? <Y> or <N> -->"
            FCB     0
            LEAX    LGSEC1,PCR
C@          JSR     GETKEY
            ANDA    #$DF
            CMPA    #3
            LBEQ    L@
            CMPA    #'N
            BEQ     D@
            CMPA    #'Y
            BNE     C@
            LEAX    LGSEC2,PCR
D@          STX     <LOGSEC

*****************************************************************
* SDRIVE get source drive and read data
SDRIVE      LBSR    COPYRT
            LDU     #SCREEN+10*LINES+1*CHAR
            LBSR    HPRINT
            FCC     "Indicate Source drive number -->"
            FCB     0
            LBSR    GETDRV
            LDB     #2              read sector
            STB     <TEMP
            LBSR    RDWRDK

*****************************************************************
* DDRIVE get destination drive and send data
DDRIVE      LBSR    COPYRT
            LDU     #SCREEN+10*LINES+1*CHAR
            LBSR    HPRINT
            FCC     "Indicate Destination drive number -->"
            FCB     0
            LBSR    GETDRV
            LDB     #3
            STB     <TEMP
            BSR     RDWRDK

* ask for duplicate copy on extra disks
            LBSR    COPYRT
            LDU     #SCREEN+10*LINES+1*CHAR
            LBSR    HPRINT
            FCC     "For duplicates insert fresh disk."
            FCB     0
            LDU     #SCREEN+11*LINES+12*CHAR
            LBSR    HPRINT
            FCC     "Hit any key."
            FCB     0
            LDU     #SCREEN+13*LINES+7*CHAR
            LBSR    HPRINT
            FCC     "<<Hit BREAK to quit!>>"
            FCB     0
            JSR     GETKEY
            LBSR    KEYTST
            LBCS    COLD
            LBRA    DDRIVE

*****************************************************************
* RDWRDK read or write to disk
* ENTRY: COMMAND must be set for read or write
RDWRDK      CLR     <DCTRK   clear track
```

```
            CLR     <DCOPC   0=restore to track#0
            LDY     $C004    get DSKCON address
            JSR     ,Y       restore to track#0
* Use with real Coco.
            LDA     #$C0     read head address on disk
* Use with emulator.
* Does not really tell if disk is ready and formatted but that does
not matter
* with most emulators.
*           LDA     #$D0     forced interrupt
            STA     CMDREG
            LDX     #$4000
A@          LEAX    -1,X     wait for drive to read address
            LBEQ    NTREDY
            LDA     CMDREG   check status
            BITA    #1       ready?
            BNE     A@       loop if not ready
            BITA    #$10     record not found
            LBNE    ERROR
            LDD     #$100    initialize the sector counter, track 0
L1          PSHS    D
            LDX     #$4000
L2          STX     <DCBPTR  buffer address
            STA     <DCSEC   sector
            STB     <DCTRK   track
            LDB     <TEMP
            STB     <DCOPC   read or write
            JSR     ,Y       access disk
            ORCC    #$50     kill interrupts
            LDA     <DCSTAT  check for errors
            BEQ     C@       go if no errors
            BITA    #$40     write protected
            LBNE    ERROR
            LDU     #SCREEN+18*LINES+1*CHAR
            LBSR    HPRINT
            FCC     "Format/copy protect error; continuing."
            FCB     0
C@          INC     ,S       next sector
            LDX     <DCBPTR  point to buffer
            LEAX    $100,X   next sector in buffer
            LDD     ,S       recover sector counter and track#
            CMPA    #19      normal sector max=18 total
            BEQ     D@
            PSHS    B,X      save track and buffer pointer
            LDX     <LOGSEC  get table
            TFR     A,B      put sector counter in regB
            ABX              point to actual sector number
            LDA     ,X       regA = sector to access
            PULS    B,X      recover track; buffer loc.
            BRA     L2
D@          LBSR    CLEAR2
            PULS    D
            LDA     #1       reinitialize sector counter
            INCB             update track value
            LBSR    MMU      get fresh memory
TRACK       CMPB    #35      max track; default 35 can be modified
by program
            BNE     L1
            CLR     <DCOPC   restore head to track 0
            CLR     <DCTRK   track 0
            JSR     ,Y       return head to track 0
```

```
            CLR     CONTRL   turn off drive
            RTS

*****************************************************************
* format skip factor equals 4 for Disk Basic
* LGSEC1 Lists sectors as they actually are placed on the disk.
* Fastest backup must access sectors in this order.
* FCB 0 is a place holder and not a sector.
LGSEC1      FCB     0,1,12,5,16,9,2,13,6,17,10,3
            FCB     14,7,18,11,4,15,8

* format skip factor equals 2 for OS-9
* LGSEC2 Lists sectors as they actually are placed on the disk.
* Fastest backup must access sectors in this order.
* FCB 0 is a place holder and not a sector.
LGSEC2      FCB     0,1,7,13,2,8,14,3,9,15,4,10
            FCB     16,5,11,17,6,12,18

            END     EXEC
```

Graph

submitted by Roy R Justus

The following program will automatically generate a graph of any two variable function of the form y=f(x). Simply place your own X-Y function in place of the example function in the line 1000 subroutine.
Works on coco 1/2/3.

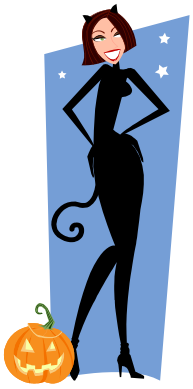
Enjoy

```
10 'WIDTH 32 :'OPTIONAL STATEMENT FOR COCO3
20 CLS :PRINT"ENTER DOMAIN (LEFT THE RIGHT)" :INPUT D,D2 :IF D>D2
THEN A=D :D=D2 :D2=A
25 DM=(D2-D) :SX=255/DM
30 N=255 :'RESOLUTION
40 DIM NY(N) :NX=DM/N
50 CLS :PRINT"PROCESSING...PLEASE WAIT"
60 FOR C=0 TO N :X=C*NX+D :GOSUB 1000 :NY(C)=Y :IF C=0 THEN YL=Y
:YH=Y
70 IF Y<YL THEN YL=Y
80 IF Y>YH THEN YH=Y
90 NEXT C
100 R=YH-YL :SY=191/R :'RANGE CALCULATION
105 PMODE 4,1 :COLOR 0,1 :PCLS :SCREEN 1,1
110 FOR C=0 TO N :XB=INT(C*NX*SX) :YB=191-INT((NY(C)-YL)*SY)
:LINE(XB,YB)-(XE,YE),PSET :XE=XB :YE=YB :NEXT C
120 SOUND 100,10
130 GOTO 130
1000 ' ===== ENTER YOUR FORMULA HERE =====
1010 IF X=0 THEN RETURN :'TEST FOR ERROR CONDITIONS
1020 Y=X*SIN(1/X) :'EXAMPLE FUNCTION
1030 RETURN
```

...

One liner by Richard Kelley

```
It's this little black box, and it's come to wipe away
everything on the text screen, then disappear just as
quickly as it came.  And the oddest part - the cursor
of ECB is right where you left it when you started the
program.  Enjoy.  :-)

10X=1535:POKEX,138:POKEX+0,138:POKEX,128:POKEX+0,128:FORX=1534TO1024S
TEP1:POKEX,138:POKEX+1,133:POKEX,128:POKEX+1,143:NEXT:X=1024:POKEX,13
3:POKEX+0,133:POKEX,143:?
```



Well another newsletter has come out for your viewing. I just barely made the deadline on this one. This month we have seen a lot of people come out of the woodwork and I would love to see more. I like the idea of Diego's award, and Rogers CoCo DVD, and all other things that keep the CoCo new and alive. I especially love all the preservation projects going on like Brian Palmer and his Dsk archiving.

I also love how all of us come together in prayer or in best wishes to help others. Like when Bob's wife Lesley had a heart attack many of us came together in prayer, and even for non-Christians they came together in hoping for the best. That is what this is all about; Family, community, and friendship. That is why I love this newsletter/e-zine. Look at what it has accomplished, more people are telling me how they got a coco or dusted theirs off. People are helping other with theirs, and new people are emerging and starting to dabble in them all over again. By the way did you notice the file size on this issue! It is getting bigger every time. I just love reading all the articles weather or not I have a clue what you are talking about. I still learn a little bit. I have achieved what I set out for and that was to find out why those boxes of disks were so important to him. It was because in his darkest times of his life when he lost his family he had a whole new family to keep

him company. It is like all you guys took over and kept him sane. I love every single one of you guys that I have met. And it is sad to know I can't ever spend Christmas with my dad again or that he will never meet his only grandson. But because of those disks, and all of you, my son will be able to understand my dad better when he is older. One day when I can learn how to use different programs to view all his graphics on I will see more of what he was like in his own mind. I think this one says a lot though.