ORPHANED COMPUTERS & GAME SYSTEMS

Volume II, Issue 9 April 1999

wELCOME TO ANOTHER FUN-FILLED ISSUE OF oRPHANED cOMPUTERS AND gAME sYSTEMS. oOPS. lET ME HIT MY CAPS-LOCK KEY AND WE'LL GET STARTED.

Much better. Sorry about that. It happens. It's one of the side-effects of this new age in which most every activity is computer-driven, including writing anything at all: an editorial, of course, but also a letter to a friend, a mail-order request or a resume. We have all learned to deal with trivialities like accidental caps-lock, slow Net servers, system lock-up and other things that have become newly accepted minor hindrances in life, right along with the old usuals: burnt-out light bulbs, missing laundry socks and freezers that need defrosting. In fact, that's the topic of an article in this issue - how computers have finally fulfilled all of the old predictions and worked their way into life's usual routines, and how we should be careful to discriminate between relevance and novelty.

There are plenty of other neat things, of course (even besides slick writing tricks like making it look like my caps-lock key was hit on purpose by paralleling it with the topic of an article inside), including an examination of Jeff Minter's creations for the Atari Jaguar and an incredible article Adam's written regarding his extensive work as he endeavors to learn how to program the Atari VCS. Sorry it's a little late, but we think you'll find this super-sized issue worth the wait! -- CF

Is it the simplistic game-play that keeps many people interested in the classics? Though several classic games hold the original play value today, many do not. However, I collect the dud games too. Now why would I do that? I do it because even the worst games must surpass enormous technical boundaries on the consoles of yesteryear to even work. This doesn't excuse the dud games, but it does allow me to appreciate what the programmer was trying to achieve. In this sense, the games that I do not find fun are like historic pieces of art that no one sees; I try to look past the rough edges of unfinished work to see the jewels that lay within. So, to me, even the bad games have a right to stand alongside the quality games. This doesn't seem to be an opinion that too many people share.

It is a common belief that some 2600 games are bad because of hardware limits. It is also a common belief that some 2600 games are good because programmers had to learn to work within these limitations. But what limitations? The answer lurks deep behind the scenes -- in the power of the VCS itself. I have challenged myself to learn what makes the 2600 tick. I believe that this will give me a better perspective of the games available for it. I don't expect to learn the machine quickly: I'm rather happy to move along at a slow pace toward an undefined goal.

The first step that I took in the direction of understanding VCS games was to change the graphics. Excellent PD programs available for MS-DOS machines turned this into a very easy task. This issue includes an article called "Changing Atari VCS Graphics -- The Easy Way" that takes the reader through the steps needed to change a graphic in *Space Invaders.* The article can be used as a springboard to change graphics in other games, and hopefully to give would-be programmers the seed to go on to create new VCS games.

This issue is packed with other great articles as well. Don't miss the sociological comments made by Chris in "When Does Efficiency Become Irrelevant?" In a world that people are claiming has grown small because of mass communication, Chris looks toward our future with an eye that questions the true distance between us all.

Chris looks at programming as ***fun*** -- imagine that! Read his insights into the subject in "Chris's Top Five Programming Tips." Not only that, but he'll discuss two of his favorite games for the Atari Jaguar in "Y2K: Yak 2000." If you are a dedicated Bally Professional Arcade fan, then you will be interested in "The Software CD Project For the Bally Professional Arcade."

Lastly, don't overlook the fixation that Chris has with gas...
-- AJT

Index of Volume II, Issue 9

Reprint of Volume II, Issue 9

This issue of *OC&GS* may be downloaded, in PDF of text format, from:
http://w3.tvi.cc.nm.us/~atrionfo

ORPHANED COMPUTERS & GAME SYSTEMS
4321 Montgomery NE, #406
Albuquerque, NM 87109

Founder/Writer/Editor
Adam Trionfo
(505) 875-1526
adam@thuntek.net

Writer/Editor
Chris Federico
chris@abq.com

Visit us online:
http://w3.tvi.cc.nm.us/~atrionfo

OC&GS is published bi-monthly. The newsletter is typeset on an Amiga 2500 and an Amiga 3000 using Pagestream v2.2. This issue also made use of Word 97.

One issue's $1.00.
A subscription comprises three issues ($3.00).

Give this to your friends! Spread the word! We appreciate it!

If you would like to exchange your 'zine or newsletter, send it! We'll send the next *OC&GS!*

If you received this issue free, think about sending a buck for the next one. Suggestions are always welcome. What subjects would you like covered?

If you want to contribute material, send us some stuff or e-mail text files to us. We love having other writers.

The Software CD Project For the Bally Professional Arcade

By Adam Trionfo

The CD Premise

This is important: There is no software CD project for the Bally Professional Arcade(BPA) -- but there will be. This article is a call to all of those Bally owners who would like to put their Bally BASIC cartridges to good use -- and are willing to get their hands a little dirty in the process. Why let the BASIC cart go to waste when it is the best piece of software that has been released for the system?

People that own a BPA know what a great prize it is. I am not going to go into detail about the system here: either you know what it is or you don't. If you have a fondness for the system, then you will appreciate the idea of owning a CD full of Bally software. But how will something like this work? For that answer, you may want a bit of a Bally BASIC history lesson.

A Very Condensed Bally BASIC History

From the very start, the system did not have much support from the parent company, Bally. There was never a large amount of software released on cartridge. On September 1, 1978, the first steps to alleviate this software problem were taken. This was the day on which Robert Fabris contacted other Bally owners who had responded to an ad he had placed in *Kilobaud* (which I presume is a magazine). It spoke about the forthcoming TBASIC that would allow the Bally to be programmed in, you guessed it, BASIC. Most people who really liked their machines soon purchased the Bally BASIC cartridge and bought a subscription to the newsletter that followed Robert's initial letter: the *Arcadian*. A few years later, another newsletter called *BASIC Express* (AKA >*Cursor*) was released. This publication often complained about the quality of the *Arcadian;* none of the whining is worth mentioning. Regardless, both of these newsletters are an invaluable addition to any Bally collection.

The version of BASIC originally released for the BPA consisted of a cartridge and a cable that connected to the joystick ports. This cable allowed an ordinary cassette player to be used as a storage device. The interface was rather slow (300 baud). It is known to be, at best, only slightly reliable when loading programs saved on a different cassette player.

When Astrovision bought the BPA, a new version of the cartridge was released: Astrovision BASIC. This is a slightly improved version, featuring a few more commands. More importantly, a built-in cassette interface is included that transfers data at a much faster 2000 baud. Not only does this eliminate the need for the special cable that was used with Bally BASIC, but the increase in speed allows any BASIC program to be loaded or saved in thirty seconds or less (compared to about 3 and a half minutes with the original Bally BASIC). Unfortunately, the new cassette interface is not backward- compatible with the older, 300- baud version. A program is included with the Astrovision BASIC book that allows older cassettes to be converted to the new 2000-baud standard. Although this conversion program is a welcome patch, it is not complete backward- compatibility.

CD Support For Astrovision BASIC Only!

The idea behind this project is to create an audio CD, as was done with the *Stella Gets a New Brain* CD for the Atari VCS. Both versions of BASIC for the Bally would allow programs to be saved as .wav files and then converted into audio tracks. However -- and this is very important -- this audio CD will only support Astrovision BASIC. This is because about six times as many programs can be stored on a CD this way. I'm not sure how rare the Astrovision version of BASIC is, but every single Bally machine I have purchased has come with it. It seems that the upgrade syndrome isn't limited to the new computer platforms: If you have Bally BASIC then there is no alternative but to upgrade to Astrovision BASIC for any planned use of this CD.

Is This For Real?

I have discussed the possibility of a Bally CD before. It was just wishful thinking, but now it's passed beyond that stage. This CD is going to be created even if I must do it alone (which would take a very long time). But it doesn't have to be that way. Indeed, a small group of Bally owners working closely together could make this project move along quickly. The final result would be a much better compilation.

The Preliminary Goals

For this to work, a solid foundation has to be laid so that it becomes easy for others to offer help. Some of the first goals that need to be reached are:

1) *Give universal access to the Astrovision BASIC manual*

Many people who own Astrovision BASIC do not own documentation for it. It is an excellent resource that should not be missed. The highest priority should be given to converting this manual to a text file for others to benefit from. An *Adobe Acrobat* file might even be preferable: It would be easier to create and read (but would be much larger).

2) *Gather together cassette program resources*

There were a number of games released on cassette for the Bally. The number of programs is small enough so that if we pooled our resources, we'd probably have the entire collection to convert to .wav files.

3) *Complete a collection of the Arcadian and BASIC Express*

These publications will have to be distributed among everyone who's helping with the project. Selected programs will need to be entered and saved as .wav files.

4) *Inclusion judgement calls*

What will be selected to go onto the CD? Although it would be easy to create more than one disc, it should be avoided. Why include mediocre software? Programs could possibly be judged on use-value. Would a program that just creates random pictures, a precursor to today's screen-saver, be allowed on the CD?

5*) Split responsibility for entering BASIC programs*

The *Arcadian* and *BASIC Express* both have many programs that need to be typed-in after the selection process. This chore is not as large or difficult as it might seem. Astrovision BASIC only allows access to 1.8K of the total 4K system memory. None of the programs take long to enter, even taking into account the use of the calculator-style keypad. With careful consideration for distributing program listings, this step will be simplified.

6*) The creation of the CD itself*

Will all copies be gold CDs? Will there be enough interest to consider making a few hundred regular CDs? Should the first track on the CD-ROM be, say, a PC-formatted track?

Are You Willing?

There will probably be few people willing to devote time to enter programs. But when the CD is complete, there will be an archive for all time that gives testament to one of the finest game machines ever created. Are you willing to contribute to a part of history?

Even if you have no desire to enter programs, why not spend just a few minutes to flaunt your collection? Maybe you could send along that one miscellaneous Bally cassette you have. How about sending a few comments on anything unusual that you know about the Bally? Perhaps you know where one of the original developers might be contacted. No contribution is too small, and all contributors will be given credit.

Who and How to Contact

My name is Adam Trionfo, a co-editor of *OC&GS.* The best way to reach me is through e-mail. Choose either address: atrionfo@tvi.cc.nm.us or adam@thuntek.net. It will be difficult to reach me by phone right now. If the desire strikes you to talk to me, send me an e-mail with your phone number and I will contact you. I'm looking forward to hearing from you!

Y2K: Yak 2000

Examining Jeff Minter's Jaguar Games in a Somewhat Timely Fashion

by Chris Federico

It's easy to blame the Jaguar's sadly short shelf life on the Tramiel marketing motif ("We don't have to support it. Whatever we put out there will sell. Besides, the word MULTIMEDIA is on the box."); but, as Adam pointed out to me during a coffee-booth discussion about the system, Atari was broke by the late '80s anyway, which best explains why the company's eligible early-'90s entry into the 32-bit arena (with a partially 64-bit box, at that) didn't get the push it needed to buy it a ride down the mainstream. Its games were superior among its peers, but they were too few sticks of dynamite to make a big enough explosion to rupture the outer layer of the gaming world.

When I say "superior," I mean it. The most intense Jag contests illustrate this beyond argument: *Iron Soldier, Cybermorph, Alien vs. Predator* and two enhanced and expanded old arcade coin-op titles, *Tempest 2000* and *Defender 2000,* are all among my absolutely favorite video games of all time -- and I've played hundreds and hundreds. Those last two were conceived and programmed by a Mr. Jeff Minter, a.k.a. Yak, a British coder and garage-entrepreneur who'd founded the famous yet erratic Llamasoft, a one-man company that leased 8-bit games to bigger corporations. His love of hoofed mammals resulted in a somewhat demented, humorous twist to most of his games -- animals figured into them in one capacity or another, best exemplified by the name of one of his early-'80s games: *Attack of the Mutant Camels.* He was also responsible for a tremendous public-domain Atari ST and Commodore Amiga version of *Robotron* called *Llamatron.*

How do Yak's games hold up a few years after their appearances, now that we're upon the actual pinnacle of the century? Did they really improve upon their original arcade counterparts? Grab that large but comfortable Jag controller and let's have a play. (I'm warning you: I'll kick your ass.)

Tempest 2000 was released in 1994 to much acclaim. In fact, it's probably more responsible for moving the Jaguar units that did sell than all of the other available games combined. There hadn't been a single home console version of Atari's Vector classic *Tempest* available (I'm not counting the IBM) since its fanatical player base first assimilated in 1981, and attempts at emulating it in the 8-bit universe, like Electronic Arts's *Axis Assassin,* fell far short; add to this the fact that the revamp really was incredible, and compound it with bizarre, dramatic magazine ads and a handful of raving reviews, and you can understand why most early Jaguar patrons picked this cart up along with their actual units.

Like with *Defender 2000,* a comparison between this package and the original game is rendered moot by the provision of the actual "classic" *Tempest* on the main menu as an alternative to the *2000* version. The coin-op is perfectly rendered in every way, but *Classic Tempest* suffers a little in one area where *Classic Defender* does not: Coin-op *Tempest* featured a rotary knob that delivered quite a large part of the game's stimulating feel and addictive player-power appeal. The Jaguar controller is among the best ever made for any system, but in a game like *Tempest,* one has to admit that arrow buttons just don't feel as cool as a little wheel that you can spin violently.

Except for this, though, *Classic Tempest* brings back the arcade game vividly and without flaw, and it's quite a thrill; and *Tempest 2000* itself is utterly over-the-top, excessive action that engages players at any difficulty level and even presents a rigorous "Beastly Mode" after you've completed all of the levels (a nearly impossible feat in itself -- thank you, thank you) which toughens the enemies and maliciously slows your blasts down a bit.

Besides its fresh flair for spectacle, extravagant explosions and more numerous colors, *Tempest 2000* adds two main beneficial elements to the original idea: imaginative new baddies and ass-saving power-ups. The Super Zapper was always that one bailout on the threshold of panic, the monumentally relieving button-slap that turned the player from a trapped mouse to an omnipotent god -- but just once per wave. This Smart-Bomb equivalent remains useful in *Tempest 2000,* but it now serves as a meantime refuge while the player waits for one of the enemies he's vaporizing to turn into a power-up. The most useful two are a (greatly) enhanced laser -- usually the first available power-up, thankfully -- and the ability to jump. This latter supplement has saved me countless times from being snatched by one of the invariably abundant Flippers that reach the perimeter and clink around it with no good intentions.

An interesting type of power-up is a bonus-round toggle: It only appears after you've gotten most all of the other possible enhancements. Grabbing three of these warp power-ups enables you to play the bonus round after you complete the current screen. There are three of these rounds, depending on what screen you've been playing, and they each offer a five-wave jump and multitudinous points if you reach the end. All three are quite beautiful graphically, but sluggish and lackluster game-play wise. I usually deliberately miss the next marker or bend in any of these stay-on-the-trail rounds to get back to the actual, infinitely more arresting game itself.

All of the old enemies are here, but there are interesting new ones: Mutant Flippers are a lot more aggressive then their older relatives; Mirrors reflect your blasts back at you, necessitating a shoot-and-dodge strategy, and take several hits to wipe out; Demon Heads explode and shoot horns at your blaster when you kill them; and UFOs fly ***above*** your gun instead of approaching from the distance like everything else. These late-game nasties require the jump power-up to obliterate, a condition that highly successfully works the jump feature into the new game at a level of innovation beyond even that power-up's very useful early-wave appearances.

So *Tempest 2000* is mostly an improvement, and it's quite addictive and never lets the player take a breath (my kind of game!). But there are a couple of things that make it a more tedious game than it could be, and these aspects concern the flair for spectacle that I mentioned earlier. The demonstrations of graphical savvy go a bit overboard and wind up being detrimental to the player's enjoyment of the game (or my enjoyment of it, anyway).

First of all, there's this thing that you're supposed to look forward to acquiring called the AI (Artificial Intelligence) Droid. When you grab the power-up that makes it appear, it flies around far above the perimeter of the playfield and shoots everything for you. I don't want something playing my game for me. All it really does is make it hard to see what's coming at you; once you discern your next target, the Droid's already on it and your game has changed to a dodging test.

Visibility is also the mark of my next gripe: Whenever you get an extra life or grab a power-up, huge congratulatory words fly out from center-screen and grow to fill the entire view for a couple seconds. As anyone who's played *Tempest* (and especially *2000*) knows, a couple seconds of obstructed sight can determine whether you live or die. Being a video game veteran from way back, I can certainly handle a lot of things going on at once, but to have my vision completely blocked by unnecessary showiness robs me of the raw arcade quality of any game no matter how good it is. Throw in the stupid '80s slang samples -- "Excellent!" "Yes! Yes! Yes" -- and the annoying "Super Zapper recharge!" voice that concludes each wave, and you have a game that, while in this manner certainly reminiscent of the era in which *Tempest* first flourished, consists in part of unfortunate aggravating moments that would have strengthened the game with their omittance.

1995's *Defender 2000* has two utterly debilitating problems, one of them being a returned nuisance from *Tempest 2000:* Easily 90% of the time, your ship is assisted by a smaller vessel (yep, another AI Droid) that plays most of the game for you. If you shoot a Lander that's kidnapping a Humanoid, your little buddy flies ahead and rescues the captive for you. He even deposits it back on the planet surface. He helps you kill bad guys as well, and basically does everything except turn the game off and tuck you into bed. It takes away from what ***would*** be a fun *Defender* enhancement with neat new enemies -- if it weren't for the player's perspective on the playing area.

For *Tempest 2000,* Yak decided to make the "camera" follow the player's blaster around the playfield. This isn't too inhibiting, because it's not really a close-up; the corridors simply shift around a bit to follow the player's movement, even though it's understood that they're actually stationary. During some waves, however, the bumpy angles of the battle area prevent seeing very far to the left or right, which results in death more often than it should. There's an option to quickly pull the view back and keep the corridors stationary, but for some reason it makes them really small, and it's hard to see anything at that distance. I would've liked this option to keep the playfield large and just quit moving the "camera," however slightly. For some reason, this trait is also found in *Classic Tempest,* an addition that definitely decrements its faithfulness to the original.

Why do I bring this up here? Because this "follow-the-player camera" is what completely ruins *Defender 2000.* It's even more impairing than the intrusive AI ship: You can't see most of the sky and still keep the planet surface in sight. The screen scrolls slightly up and down in addition to the usual side-scrolling. This means that your Humanoids and any aliens creeping below you are lost to view while you're taking care of business up in the air. This also means that it's very easy to run into things at any time, since the scanner isn't very precise -- and keeping yourself from flying too fast and colliding with enemies takes up most of your time, forestalling any strategies that might be put to use instead. I don't even play *Defender 2000.* I think I've tried it a total of five times in the year or so that I've owned this cartridge. It's a shame, because one of the new enemies, the Lander-Launcher -- its function obvious -- adds an interesting twist to the concept of wave-clearing.

But *Classic Defender* is as exhilarating as *Defender 2000* is cumbersome. This adaptation of the old coin-op alone was worth getting the cart. Everything's perfect, and you get the same adrenaline attacks and alternating feelings of power and panic when you play this translation. The graphics look exactly the same, the mechanics feel exactly the same, and although they seem a little faster than before, the explosions are all just as glorious.

I recently bought the Williams collection *Arcade's Greatest Hits* for the Playstation. The games, as you know, are emulated directly from the original coin-op motherboards. I highly recommend this one, by the way; it contains many interview clips that feature game designers, including Eugene Jarvis himself (writer of *Defender*). Anyway, playing the original version really emphasizes how well Yak did in programming it from scratch and endeavoring to make it look and feel the same. He really did a great job. Differences are minor: The bad guys in his version take longer to speed up than in the original, and the Landers in particular start off a bit slower, ascending with hostages more slowly at first than their predecessors. But it should be pointed out that neither the Jag version nor the Sony one provide a controller that comes close to matching the arcade panel. The utter mastery over the ship that's possible with the coin-op controls just can't be achieved with a joypad.

Both *2000* titles have extra games between *Classic* and *2000:* You can opt for *Tempest Plus* or *Defender Plus*. The former never gets played on my Jag. It's not that it's a bad game, it's just that it's *Classic Tempest* with a couple minor attributes from *2000* thrown in. It feels schizophrenic, and I'd rather go to one extreme or the other, y'know? It features a simultaneous-two-player mode: The screen is split vertically and you see your buddy down at the distant end of the tubes. Death-match does not work with *Tempest.* I don't know how much more clearly I can put it.

But *Defender Plus!* Now, ***there's*** a video game! ***This*** option should have been called *Defender 2000* instead of the sad (but admittedly pretty) dodging contest that was actually given that title. I don't know if you can really improve upon the original *Defender*, but if you could, this game would hit pretty close to the mark.

For one thing, Metropolis Digital's graphics are absolutely stunning, the centerpieces being the throbbing, translucent mountain range and the multidimensional star scrolling. Your ship maxes-out at a blinding speed and your laser is now rapid-fire (i.e. it keeps firing as the button's held down), allowing you to race along and clear out baddies like you're ice-skating through soap suds (glancing often at the scanner, of course).

A wonderful new firing feature also figures into the terrific mechanics: If you hold down the button to rapid-fire, you can move your ship backward. Pushing opposite the direction you're facing usually reverses your ship, of course, but as long as the button's down in this installment, you can backtrack, circle around and do all sorts of neat stuff, since you won't turn around until you let go of the button. This is especially fun when you've just demolished a Pod and a flock of Swarmers emerges; you can take out the whole group with a couple up-down sweeps of your overwhelming laser.

You definitely need this quality for the toughest new enemy, the Big Space-Station-Looking Thing. That's my name for it, anyway; the manual for this cartridge absolutely sucks, and *Defender Plus* is reduced to two or three paragraphs. None of the enemies are explained, nor is the extra firing-control feature I disclosed. The book's skimpiness probably has something to do with the fact that Atari saw fit to squeeze three languages into most of their Jaguar game manuals. At least other titles come with complete directions; this handbook may as well not even exist.

Anyway, the BSSLT is really frustrating at first, because as you fly over it unawares, it launches itself upward from its hiding place below the planet surface, giving you absolutely no reaction time before your ship bursts in an impressive cloud of color. But what initially seems like cop-out bad-guy addition (one of my pet peeves, as you know) turns out to be a quite gripping, quite beatable super-enemy. All you have to do is slink along near the bottom of the playfield with the fire button down, careful not to kill Humanoids of course. No hiding BSSLT can escape your wrath if you hunt like this, and it adds an unexpected pensive quality to the formerly purely vicious *Defender* concept while adding to the panic felt when you hear the cry of an abducted Humanoid: You can't just race recklessly to the scene of the crime anymore (at least in wave 4 and above). The BSSLT also launches little hopping things at you that can take you right out if you're not careful to avoid them; they boink back and forth at sweeping angles and have a tendency to evade sight.

Another fascinating new enemy appears after you beat wave 15 (or 16; I can't remember which, but I've done it a few times): A giant guy in a jet-propelled spacesuit, even bigger than your huge ship, zooms ruinously over the planet, ready to ram your ship into pieces. Like the BSSLT, this guy requires several blasts to take out, but you can have a lot of fun obliterating him; and, like the BSSLT, his destruction scores very high.

Another fantastic element in *Defender Plus* is the hyperspace portal called the Stargate, revived from an arcade game called, oddly enough, *Stargate* -- the original *Defender's* sequel. Flying into the portal either warps your spaceship halfway around the planet or brings you to the site of an abduction if any are taking place. Flying into it with at least a few Humanoids in tow warps you ahead as many waves as Humanoids you're carrying.

Speaking of neat tricks, entering "Ovine" as your name in the *Defender Plus* high-score table and then starting a game by pushing the A button instead of B will transform your ship into a giant sheep! You're protecting little llamas instead of Humanoids, and deploying a Smart Bomb renders a moo-like sound. While Flossie's thrusting (for this is surely a graphical depiction of Yak's favorite sheep that he owns), the exhaust egresses humorously from under her lifted tail. Flatulence Fuel? Do I detect a possible breakthrough for NASA here?

Also, entering "Nolan" (meaning Bushnell, of course) in any of the cart's high-score tables will give you a further option besides the three *Defender* games: *Plasma Pong!*

Overall, Yak and his graphic teams did an amazing and enthralling job of updating two of the gaming world's most beloved entities, and as long as he never comes out with, say, *Congo Bongo 2000*, his reputation remains optimum.
-- CF

Doesn't Anyone Have Gas *Yet*?

Three issues ago now, we posted a blurb about a new contest we were having, spawned by a reader letter mentioning the fart-like sound of a strike in the Fairchild *Bowling* game. We asked that readers send in any **farting moments** they could remember in the extensive library of classic video or computer games. In the past two issues we've printed reminders, since we'd only gotten two responses. We still have yet to receive a third!

Come on, professional gaming colleagues! E-mail or send us brief descriptions of any farting sounds you'd like to enlighten us on. We're hoping to write a whole article about them (with mention of the contributors, of course).

By now, of course, it might seem like I'm pushing this a bit much. I mean, if all of our readers lack the sense of humor (or the ears for flatulence) required of a monumental project like this, then it might be foolish to hope for any more responses. But we have a rare chance, you and I. We have the opportunity to unfurl a brand-new flag that can be planted with pride amongst the electronic community. We can show the world that we are ***not*** a bunch of lifeless geeks who sit at home all day and take our machines more seriously than our hearts! We can prove that we're ***not*** lost in a world of bits and bytes that prevents us from displaying secure, self-assured senses of humor, or enough self-esteem to lighten up about our hobby once in a while! WE CAN MAKE A DIFFERENCE! *(Braap)* -- CF

ATTENTION:
Address Change

Once again, we put this here instead of near the address bar on page 2 so that it would be more noticeable. The apartment number in our address has changed (as has Adam's e-mail address). Thanks for looking at this box. This box hopes to serve you with more life-easing information in the future.
-BOX ENDS-

Changing Atari VCS Graphics- The Easy Way

By Adam Trionfo

You, The Reader

You think of yourself as a real VCS fan-- a true collector. Perhaps, instead, you are just a casual VCS player. You have all the best games, and even most of the worst ones. Maybe you just have a handful of your favorites. When you heard of the *Stella Gets a New Brain* compilation for the Starpath Supercharger, you had to have one. Possibly you have no idea what *the Stella Gets a New Brain* CD even is. You have an Atari VCS emulator installed on your computer along with every ROM image that you can get your hands on. Or, you *despise* playing emulators, preferring the real thing every time.

Whichever category you place yourself in doesn't matter. This article is aimed squarely at the player (not the programmer) who is interested in making a VCS game a little more of their own by changing the graphics that have become so familiar to us all.

A devout scouring of the Net for VCS information will pull up enough to keep anyone occupied for a lifetime. So if you are just a little bit curious, where do you start? That was the proposition that lay before me last summer. I looked around for information that was aimed at a non-assembly-programmer: I never found any.

On the Internet, there are a number of games that have had the graphics changed. One of the more interesting graphic conversions of an Atari VCS game is *Space Invaders* by Yak (of *Tempest 2000* fame). The game is infested with- what else- Llamas. It was this graphic conversion that first intrigued me to look beyond the mere playing of VCS games. After a little work I was able to change *Space Invaders* myself.

I wish to encourage tinkering with VCS games. I have tried to make this a step-by-step approach to changing the graphics in a 2600 game- specifically *Space Invaders*. This article is a guideline to the programs and a few of the techniques that I used last summer to convert the graphics in *Space Invaders* to OC&GS Invaders.

Basically, if you have the skill to use *Windows* and a bit of *DOS*, then you should have no problem with the programs required to change the graphics in a VCS game. Perhaps, after trying this out, someone reading this will say, "Wow, this is so easy. I'm going to make a new VCS game too." I hope so!

And People Do This *How…*?

How do people make new games for the VCS? How do they change the graphics in a classic game like *Space Invaders*? Do you (like me) look with awe at anybody who has the ability and devotion to learn to program the VCS? Creating an Atari VCS game is beyond me at this point. That fact did not stop me from wondering how to change in-game graphics.

If you ever thought that it would be neat to replace some Atari game graphics with some of your own, then you will be thrilled to know that it is rather easy. The best part of all is that it requires no programming skill.

No Assembly Required

I shall discuss two ways to change the graphics in an Atari VCS game; neither requires any assembly programming skills. Surprised? The reason for this is that there are now tools available that allow anyone, even a casual computer user, to look at what makes a VCS game tick. Devoted programmers have been kind enough to donate the tools they create to the public domain.

To change graphics in a VCS game, one only needs to know addition and conversion from decimal to hexadecimal. Uh-uh- you don't know how to change 'regular' numbers to the hex numbering system? No problem- most modern calculators can do it for you- or use the scientific calculator in any version of *Windows*.

Does that still sound like too much work? Well then you will be pleased to know that there is an even easier way that requires no math at all. This is for the person who wants to stay away from the code as much as possible.

I have a preference for the method that requires the use of addition and hexadecimal conversion. This is because I would like to eventually learn a little 6502 assembly. Of the two methods, you can choose the one that suits you best, as I shall discuss both.

The Essential Hardware

A VCS game can be changed in a number of different ways: I will only be discussing how to do this using an MS-DOS machine. Though these programs might work on slow machines (386 or less)-- I would recommend at least a 486 (for VCS emulation). If you plan to run your game on a real VCS, then you will also need a VCS, a Starpath Supercharger and a soundcard for your PC.

If you prefer another hardware or software platform, don't get discouraged. Even though this article will be covering programs for MS-DOS machines, all that is going to be described here can be done, in one fashion or another, on a Mac, an Amiga, a PC running Linux or even a Commodore 64.

The Essential Programs

You do not have to be running *Windows* to use any of these programs. However, the 'multitasking' capability of *Windows* will make the task a little easier for you. The best place to get the software described here is from Dan Boris's Atari 2600 Tech Info Page at:

```
http://atarihq.com/danb/vcstech.htm
```

This web site is crammed with additional Atari technical information as well as commented VCS source code.

The essential programs needed to change the graphics in a VCS game are:

PC Atari Emulator 2.1a (By John Dullea)

This program will be used to run your modified game ROM during the test phase. Make sure that you get at least the version noted here because older versions have some timing problems with *Windows 95* and *98*. Don't let me steer you away from other VCS emulators. Try them all- use the one that works best for you.

DiStella 2.1 (By Dan Boris and Bob Colbert)

This program is used to take an Atari VCS ROM image, as used with an emulator, and change it into source code. When a program is assembled, all documentation is left out of the executable file; so when the disassembly is complete there is no documentation. There are some novel ways that *DiStella* allows you to track graphics down without any knowledge of assembly language required.

DASM 2.02 (By Matthew Dillon)

This is a high level assembler that can be used for the 6502. It uses a text file as input and generates a file that is executable on the VCS. It is completely CLI driven, but don't let that scare you away. Note: I had to read the documentation file dasm.doc using the *MS-DOS* editor, otherwise it looked garbled.

In conjunction with *Distella*, *DASM* is the method I prefer to change graphics in a game. It gives me the feeling of additional control that I don't get while using *Showgfx* and *Editgfx*.

Showgfx/Editgfx (Rob Colbert & Dan Boris)

Showgfx is used to create an ASCII listing that can be examined for graphics very easily. It requires no knowledge of the hexadecimal numbering system. *Editgfx* takes the ASCII file that *ShowGfx* creates and changes it back into a binary file. Folks, it doesn't get any easier to change VCS graphics than this!

Makewav 3.1 (By Bob Colbert)

There is no rule that says you ever need to run a modified game ROM on a real VCS. It just seems much more satisfying watching a wood-grained Atari Video Computer System run a modified game.

Makewav will convert a VCS binary file into a wav file that can then be loaded into the Starpath Supercharger on a real 2600. This means that you must limit the size of the ROM to what the Supercharger can handle- 6K.

As a side note: I have been able to run the Amiga version of *Makewav* 3.1 on an Amiga 2000 with a 68000 and three MBs of RAM. I place the wav files into RAM and play them to the Supercharger using a separate wav player. It works great, and is rather speedier than I expected.

Space Invaders Binary ROM Image

In order to change *Space Invaders*, you will need to search the Internet for the spaceinv.bin file. This is the file that contains a copy of the VCS code. There is no way to follow these instructions without this ROM image.

Getting Set Up

Follow these steps to get all the programs you need into one place, which I will refer to as the programming directory. Later, when you are asked to type in a command, you must be in the programming directory.

1) Decompress all the zip files into separate directories.

2) Place the following files from the separate directories into another separate directory (I called mine Atariprg).
   a. makewav.exe
   b. distella.exe
   c. dasm.exe
   d. Showgfx and Editgfx
   e. PCAE (plus all the additional files it needs to run. If you don't get all the right files, the emulator will let you know what file is missing.)
   f. spaceinv.bin

3) Play *Space Invaders* using PCAE to assure that the emulator has been installed correctly. Type this:

```
pcae spaceinv.bin
```

4) If the emulator works but is running slow, then your computer is just too slow to run the game full speed. The game may also play too fast. For either of these cases, read the PCAE documentation for suggested fixes.

Method I: No Hex Involved
An Easy Way to Change VCS Graphics

Using this method is the easiest way to change the graphics in a VCS game. However, if you ever intend to change more than just a game's graphics, then you will need to use the second method. Remember that all the commands in these steps must be entered from your programming directory.

1) Create a text file that displays the contents of each byte (which includes graphics) in *Space Invaders*. It is important to know the size of the game ROM. *Space Invaders* is 4K, or 4096 bytes. To create the text file, enter the following command:

```
showgfx spaceinv.bin 0 4096 > spaceinv.txt
```

2) Now, using a text editor, you must look through spaceinv.txt for the graphics. *Showgfx* displays the contents of every single byte. Here are a few tips to find graphics:
   a. Graphics are going to be stored upside-down most of the time (this is certainly the case with *Space Invaders*).
   b. Sometimes game graphics are split into pieces. Train your eyes to see a character in less than whole parts.
   c. Not all the graphics are in one place. In *Space Invaders*, for instance, you will find that game graphics are separated from the game score characters.
   d. Some of the graphics will be obvious, while others will not. So, if you can't find what you are looking for, keep looking!

3) This is what the player's ship looks like in spaceinv.txt. We will change it to a happy face.

```
0c0a |XXXXXXX |
0c0b |XXXXXXX |
0c0c | XXXXX  |
0c0d |XXXXXXX |
0c0e |  XXX   |
0c0f |  XXX   |
0c10 | XXXXX  |
0c11 |  XXX   |
0c12 |  XXX   |
0c13 |   X    |
```

4) Unlike method II, changing the graphics with this method requires *no* math at all. Just change the X's to a happy face, like this:

```
0c0a | XXXXX  |
0c0b |X     X |
0c0c |X XXX X |
0c0d |X X X X |
0c0e |X     X |
0c0f |X     X |
0c10 |X X X X |
0c11 |X     X |
0c12 |X     X |
0c13 | XXXXX  |
```

5) After you have saved the changes to spaceinv.txt (making sure that it has been saved as ASCII), it is time to create a binary file for the VCS emulator. Call the new file something else besides spaceinv.bin (here I call it testspac.bin):

```
editgfx spaceinv.txt testspac.bin
```

6) Now run PCAE to test the changes. Type this:

```
pcae testspac.bin
```

7) The player's ship should now look like a happy face.

Method II: Some Hex Involved
Another Easy Way to Change VCS Graphics

The second method used to change a VCS game's graphics is only a little more complicated than the first. Here is an overview of the steps involved: disassemble the ROM, find the graphics, change the hex data statements, and assemble the ROM again. This method is a little more involved, but the effort it takes is well worth it. If, at another time, you intend to modify a game's gameplay, or even make a game yourself, then these steps are mandatory knowledge.

1) Disassemble *Space Invaders*. This step creates the source file needed to change the graphics.

   a. The following command will create the source file for *Space Invaders*:

```
distella -paf spaceinv.bin > spaceinv.src
```

2) Just to make sure that the disassembly worked okay, assemble the source file using DASM. Type the following:

```
dasm spaceinv.src -f3 -otest1.bin
```

3) Using a directory listing, check the size of the file test1.bin. It should be 4096 bytes.

4) Start PCAE and play test1.bin. If the game plays okay, then continue.

5) If you are at this step, then you have disassembled a ROM, assembled that same ROM, and tested it to make sure that it works. This step creates a configuration file that will help us view our source.
   a. Uisng a text editor, view the source created by *Distella*.
   b. Search for .byte.
   c. Mark the beginning address and the ending address. For *Space Invaders* there are two places where these exist (FBFE-FD66 and FF4C-FFFF).
   d. If you have never used hex before then all these addresses make little sense to you. The fastest way to get the correct addresses (actually just the most likely to be correct) from any ROM is to look at the first and last address where .byte exists. In the case of *Space Invaders*, to get the addresses FBFE and FD66, look at this excerpt from spaceinv.src:

```
LFBFE: .byte $00, $00, $00, $00, etc
   More .byte filled data
LFD61: .byte $03, $17, $2B, $23, $75, $B4
              ^    ^    ^    ^    ^    ^
              FD61 FD62 FD63 FD64 FD65 FD66
```

   e. Using a text editor, create an ASCII configuration file called spaceinv.cfg with the following information (this file is case sensitive). Save it to the programming directory.

```
GFX FBFE FD66
GFX FF4C FFFF
```

6) In step 5, see how the first line contains the first and last .byte address? The same is true of the second line of the configuration file that we created above.
   a. If you don't understand, don't worry. Later, when you are doing this for another game's ROM image, just pick the first address and one near the end- for instance in this case choose FD61. No matter what you choose it is okay- it will just display what is in non-graphic bytes.

7) After the configuration file is made, it is time to disassemble the binary file again, this time revealing code that is a little easier to find graphics in. Type:

```
distella -paf -cspaceinv.cfg spaceinv.bin >
spaceinv.src
```

8) View the new spaceinv.src file. Beside each .byte there will be a graphic representation of what each byte contains, using commented X's. Here is an example taken from the area that represents the player's ship (upside down):

```
.byte $FE ; |XXXXXXX | $FC0A
.byte $FE ; |XXXXXXX | $FC0B
.byte $7C ; | XXXXX  | $FC0C
.byte $FE ; |XXXXXXX | $FC0D
.byte $38 ; |  XXX   | $FC0E
.byte $38 ; |  XXX   | $FC0F
.byte $7C ; | XXXXX  | $FC10
.byte $38 ; |  XXX   | $FC11
.byte $38 ; |  XXX   | $FC12
.byte $10 ; |   X    | $FC13
```

9) We are going to change the player's ship into a happy face that will look like the representation below. Do this by changing each line's hex value after .byte.

```
.byte $7C ; | XXXXX  | $FC0A
.byte $82 ; |X     X | $FC0B
.byte $BA ; |X XXX X | $FC0C
.byte $AA ; |X X X X | $FC0D
.byte $82 ; |X     X | $FC0E
.byte $82 ; |X     X | $FC0F
.byte $AA ; |X X X X | $FC10
.byte $82 ; |X     X | $FC11
.byte $82 ; |X     X | $FC12
.byte $7C ; | XXXXX  | $FC13
```

10) For those of you who don't know hex, here is a quick and simple explanation. Each 'X' graphic represents a binary digit. As you probably know, there are 8

bits in a byte. Each row of X's represents one byte. In order to get the byte's value, we have to add up each X and the placeholder it represents. For example, in step 10, address $FC0A looks like this:

```
                                  Total bits
128  64  32  16  8  4  2  1= 255
     X   X   X   X  X      = 124 decimal
                             7C in hex
```

11) Each byte must be translated to hexadecimal in this manner. Use a calculator to help translate between decimal and hex digits. Save the file (without changing the name) when you are done. Note that you only have to change the hex numbers-- there is no need to change the X representations (as was the case in method I), as these are just comments.

12) It is time to assemble the source and see your new graphic. Type this:

```
dasm spaceinv.src -f3 -otest1.bin
```

13) Play the new version of *Space Invaders* using PCAE.

```
pcae test1.bin
```

14) If everything went well, your ship should now look like a happy face!

15) If you would like to play your new *Space Invaders* on a real Atari 2600 using the Starpath Supercharger, then use Makewav to create a wav file on the computer. Depending on how fussy your Atari and Supercharger are, try ONE of the following, listed in order of preference:

```
a. makewav -f spaceinv.bin spaceinv.wav
b. makewav spaceinv.bin spaceinv.wav
c. makewav -k spaceinv.bin spaceinv.wav
```

16) Hook up the Supercharger to the computer's speakers. Play the spaceinv.wav file using any wav player. Adjust the volume as needed until the game loads. If the file doesn't load, then try either b or c in the above step.

It Worked! - Now What?

You now have a happy face instead of a space ship to do battle with the evil alien invaders. It didn't take long to do, so why not change the alien invaders as well. Make that terrible UFO into your initials. Make the shields into Pac-Man. Be creative.

Using these same methods, it is possible to change the graphics in most any Atari VCS game. Eventually, though, no matter how much you enjoy it, changing the graphics will become rather boring. The problem is that there is a real boundary to how creative you can be.

The solution to this boredom is to modify the actual code of a VCS game. I have not attempted this yet, but it is the next step that I will be taking. There is plenty of documentation on the 6502 and the VCS. There is also an abundance of commented source code available. So when you are ready for that "next step", don't be afraid to take it.

The logical progression after modifying a game's code is to create a game from scratch. I am not at that stage yet, hopefully someday I will be. Until that time, as I learn more about the VCS, you can be sure to read about it in upcoming issues of OC&GS!

Good luck with all your efforts! – AJT

VCS Programs That Don't Exist – But Should!

By Adam Trionfo and Chris Federico

An enormous amount of games are available for the Atari VCS, but most of these games are quite dated in terms of game-play. I don't mind playing a round or two of *Super Breakout*, but it can wear thin rather quickly. Numerous games have evolved over the years that take the idea of *Super Breakout* and expand upon it. Not all of these games are better than the original, but many are. This isn't a bad reflection on the VCS or the original game, but it does confirm that improvements of VCS games can be made; they may even be needed.

There is already activity in the VCS programming circles. However, only a very small amount of this is work on new games. Mainly what I have seen is proof-of-concept demos. These introduce new ideas to the VCS. The programmers who create these are having loads of fun; they are also contributing a wealth of resources to the VCS community. But why not work these efforts into a program — a game of some sort?

There is already proof that this can be done, for new VCS games have appeared over the last few years. The Internet newsgroups are doing a fine job at getting people in contact with one another. What I am going to do is offer my ideas about games that the VCS could have—but doesn't.

Every game genre for the VCS has already been visited in one way or another. But some of these games only scratch the surface of what can be

Continued On Page 16

Chris's Top Five Programming Tips

by Chris Federico

Whether it's emulating the benefits of Commodore 64 machine language with *Garry Kitchen's Gamemaker,* contriving a text adventure on the Amiga with HiSoft BASIC 2, trying to concoct speedy character-graphic games in good ol' 64 BASIC or trying to figure out how to make Atari 800 LOGO successfully persuade a value from the joystick to an onscreen character, I've always been mesmerized by the half-science/half-art of coding. Adam likes to torture himself by digging a little deeper into his machines: re-tailoring Atari 2600 games or studying 8-bit machine language.

Devising new universes by communicating with our machines has been a fascination of Adam's and mine since we were kids. After years of alternating frustration and success, what can I come up with for the would-be inventor who isn't satisfied with just ***using,*** but is leaning rather toward ***creating?***

This is directed toward experienced programmers as well -- it's good to take a step back once in a while and ponder what can be done to make things easier. (Trust me on this. I could use plenty of steps back myself.)

These are not necessarily in order.

Begin a project as freely as possible. Don't structure things right away. Improvise! I'm all for flow-charts and easy-to-follow structure, but sheesh, talk about giving yourself a tedious task right at the beginning of something that should represent a fun creative outlet! Think of that excited feeling you get when you're ready to sit down and start typing-up the recipe for your latest brainstorm. There's nothing that can kill that enthusiasm like turning it into a job from the outset. Start numbering lines or naming procedures, make them do what you want them to -- no matter where in the final execution they're actually going to be called -- and pour your brain onto the screen! You can always go back and re-structure everything, which you're going to wind up doing anyway.

Use comments freely. Whether it's using the REM statement, an apostrophe, a semicolon or some other symbol denoting that the computer should ignore what follows on the current line, you'll love yourself later for making it clear what every section (or line, for that matter) of your code is supposed to do. If you've ever stopped working on a program for a couple weeks and then tried to go back to it, you know what I'm talking about. If you're worried about execution speed (this mainly goes for you fellow 8-bit programmers), remember that you can always take comments out before saving the final product.

This tip goes hand-in-hand with the next one:

Study your code. Whether you're returning to a program you postponed a few days (or weeks) ago or you just want a breather to find out where you are, looking at a printout or scrolling screen to take-in the big picture can do worlds of good. There's nothing that can deflate enthusiasm like the need, every time you want to add or change something, to go back and re-learn where in your code you've put everything and what the ramifications of your changes might be. Keep up on your general structure -- another advantage is that debugging gets ***so*** much easier.

Don't think paper's obsolete. I find that it's more fun to start coding if I know ***what*** I'm coding. I write out everything that I'll need to know, so that I don't turn it into a chore by having to rely on memorization.

Do yourself a favor and list every variable in your program and what it does. Pre-designating them in this way -- even open (changing, temporary) variables and flags that you might need -- gets one whole task out of the way, ***leaving you free to improvise code without hangups,*** since you can just refer to your notes whenever you need to remember stuff. Sometimes, in addition to commenting in the code itself, I'll write down the line numbers or procedure names and add a brief description that explains what each section's function is. Sprites, sound file names, etc. should all be disclosed. It just eliminates so much drudgerous memory-work later.

Don't be afraid of failure. Failure is part of life. That's all there is to it. Without the possibility of not succeeding at something, how could the actual accomplishments be special? Don't talk yourself out of trying something in your program just because it might not work, or because it could crash everything. That's what backup files are for.

I hope this stuff helps. And I implore you, if you write anything on an orphaned computer (this includes the Amiga), by all means, SEND IT TO US! -- CF

When Does Efficiency Become Irrelevant?

The Internet, Portability and Their Implications

by Chris Federico

The Internet is, if you think about it, quite an amazing thing. First of all, it's the first multimillion-patron entity that's built by the whole world, people of all interests and skills and from all walks of life. You can help construct this growing, metamorphosing mammoth no matter who you are. All you need is the equipment: some rather inexpensive machinery in fact, if one were to compare a computer and modem to, say, a parts factory in the backyard if everyone were contributing to the building of a car. Secondly, it is, as I've stated before, making the world smaller and smaller. This is what humans do with their technology, and it's simultaneously marvelous and frightening: We make facts, materials and media instantly accessible and we can meet new people halfway around the world, drawing ourselves together like an unexpected globewide peace summit.

The third amazing thing about the Net, especially bewildering when taken into context with the previous two points, is that IT DOESN'T ACTUALLY EXIST. What we "surf" around (***groan***) is actually just a bunch of zeroes and ones whizzing through telephone lines; there's nothing static, no huge computer somewhere representing THE INTERNET BASE or anything. What we call the Internet is actually more of a widely accepted series of telecommunication protocols than a tangible body. It's therefore ironic that it's the most organic, shifting, consistently growing object to either 1. become so mind-bogglingly popular or 2. not actually exist.

Sure, there are major ways of accessing all of these zeroes and ones, this worldwide network. But this prominence is based on shrewd business, foresighted marketing and smart promotion. America Online or the Microsoft Network didn't invent the web any more than CompuServe or Prodigy invented the idea of special-interest bulletin boards. They're just two of the manifold ways of diving into the Internet, much like *Explorer* and *Navigator* are just two of the nearly infinite windows through which to see and use it. Even the Worldwide Web doesn't encompass everything on the Internet; only, at the moment, a big chunk of it.

Your medium doesn't matter, then. Returning to the car analogy, your local freeway doesn't offer different roads depending on the type of car you're driving. We can all get to the same zeroes and ones, and we can all add to them, and what this enormous, breathing, interconnected, nonexistent web of people's creative and industrial spills represents no matter how you reach it is a shrinking not only of the distance between you and a bit of information or you and another part of the world, but also a diminishing of the effort required of previously manual functions.

This makes the Net an excellent model for the entire electronics industry as it has now come to exist: Previously non-electronic tasks have been made much easier and more expedient by crossing over into bits and bytes. Telecommunicating is only part of it; writing something, designing something and even learning something have become computer-based activities for millions of people, making them much more at-hand accomplishments. Like the shrinking world, this is both fantastic and disconcerting. Add a laptop computer into the equation and we have ***portable*** ease with which to do all of these things.

For the first few years during which the Net was a household topic, you needed an actual computer to access it; modems weren't common purchases along with laptops yet. When things like the portable IBM 286 (and its clones, of course) and Apple's PowerBook (wieldy Macintosh) were made available, it became possible to dial-up your local server and check your e-mail or connect to the Net from restaurants, airplanes and even the damn beach if you wanted to.

All you need is a portable phone: Soon we'll all be plugged into each other, immediately available from anywhere. Laptops get smaller and smaller (ever see the palmtop computer the Psion?) and will eventually replace pagers and whatnot.

This constant-availability likelihood is both an attractive and an annoying prospect. I mean, sure, it will make things easier, but it will take the mystery and (for lack of a better word) romance out of meeting new people, or at least give it an entirely different kind of excitement.

I used to sit down and write letters to friends. I like writing. I like fooling with language. It's fun for me. But I realized that to get most people to write back, I was going to have to allow my recipients to bypass the physicality of replying. I was going to have to purge them of the chores of handwriting, finding an envelope, sticking a stamp on it and walking it to the mailbox. So I obtained an e-mail address, which of course also came in handy for readers' comments and questions once I discovered the Net's potentially huge audience and started my various sites.

After a couple years of these facilities for instantaneous communication, I myself have found it increasingly arduous to write using my hand. This is coming from a guy who (voluntarily) spent his childhood with pens and notebook paper. Most letters I type nowadays, even to people to whom I'm just going to hand the pages, are typed on a word processor. Ditto for ideas for new projects that I could just take a few minutes to jot down. This is mainly because it's possible to lose a piece of paper, whereas a file saved on a hard disk and a backup disk can always be re-summoned.

There's been secured a fusion between my brain and fingers that makes the thoughts flow without the supposed burden of hand movement, especially at a fast typing speed. There's less time for reflection between written words, less time for rhetorical and structural planning, but then the word processor has a DELETE key.

Someone outfitted with a laptop computer containing a modem and a word processor has taken the incidental part of his work out of tons of things. A really positive angle on this is that it might encourage kids to write. It's not as much of a chore now, especially if you figure-in the "playing with a toy" nature still redolent in electronics for most of us. The task of physically turning thoughts into words will be further alleviated by the upcoming mass advent of these new voice-recognition programs (such as Dragon's *Naturally Speaking* for minimum 32 MB/133 MHz PCs). All of this human-effort-truncation is definitely where we're headed on a widespread basis. The pro side of this has been made obvious; but what about the down side?

I'll start on that cheerful prospect with an illustrative example concerning this newsletter. I'm typing this article in *Excellence!*, a word-processing application that runs quickly and smoothly on my Amiga. Once I open the program that I use to typeset this newsletter, *Pagestream*, I can import this text into the preset columns because *Pagestream* recognizes *Excellence!* files. I don't even have to convert the text to ASCII, the every-computer-knows-it code for the alphabet etc. Text saved as an ASCII file (i.e. with .TXT as the extension on its name) can be read by any computer no matter which one it was written on, provided the Disk Operating Systems -- DOSs -- are friendly with each other, or have been forced into a friendship with extra utilities.

Now for all I know, Adam's writing an article himself at the moment. If he is, he's most likely using his laptop IBM, typing it in *Microsoft Word*. If this is the case, he'll save his work as an ASCII file on a 3 1/2" diskette, which he'll simply hand over to me so I can load it into *Excellence!*. I'll then save it as an *Excellence!* file instead of an ASCII file so I can import it into *Pagestream* as usual.

I could name several other examples, but the point is that our creative outgrowths are composed of text that is fully transportable, no matter what application it's typed in or what's used to read it. Likewise for e-mails that your friends send you wishing you a happy birthday, or for that matter, images, downloaded off a website, that you think your girlfriend might like; you could view them using any paint program if they're .JPG files, .GIFs or even .TIFFs. It brings up the question of treasuring each other's thoughts and sentiments. The words we put together become less valuable -- disposable, even.

It also poses the risk of making us perceive each other, in the abstract anyway, as files or addresses. Being able to communicate with another person was, at one time, very special: a long-distance call, a car trip, a long letter with maybe a poem. Now we can access each other just by knowing e-mail addresses. Someone sitting on a bus on her way to work can e-mail her husband, or if he's planning on stopping at the store, she can call him on a cellular phone and immediately be with him, not necessarily to be sweet but rather to get incidental contact out of the way. I'm not saying that taking trips to visit people will be a thing of the past, but with the arrival of video phones, it won't be quite as necessary or special.

We can save each other's communiques alongside puzzle games and spreadsheets; we just assign each other to drawers and file-name extensions.

We're already numbers to the government, folks. We're already file names to the corporations we patronize. Let's try to retain some semblance of endearment and personality, huh? Let's be happy about what we're achieving, but let's be careful as well. As long as we're aware of every technological step we're taking, staying skeptical of advertising and remaining detached enough (and reliant enough on our own built-in computers) to know the difference between advantageous and thoroughly desirable inventions, and cold, superfluous ones that render us subhuman. Are we smart enough to remain aloof to our own devices? Well, how about this: Are ***you?*** -- CF

Continued From Page 12

done on the console. Some games could use updates, others deserve sequels and some titles from other platforms could be ported; but best of all would be entirely new games.

There is other work that can be done with the VCS besides creating new games. What about programming languages? BASIC already exists for the VCS, but in such a crippled form so as to be unusable. It's possible to create a version of compiled BASIC that doesn't run on the VCS at all, but creates executable VCS files. The files could then be transported to the VCS via the Supercharger or an EPROM (programmable ROM) cartridge. BASIC would not be as flexible as Assembly, but it would allow non-Assembly programmers to dabble with the VCS. If this could be done with BASIC, than it could also be done with other languages, like C. For anyone wondering how this would be possible, it is the same idea as a cross-assembler (such as *DASM*), which is an assembler that runs on one platform but creates executables for another.

There are utilities that are also needed which could help with the creation of VCS titles. One useful utility is called *Stella-Graph.* Created using *Visual BASIC 4,* it is much like the old-school sprite generators on the C-64 or Atari 8-bit. *Stella-Graph* is quite useful now, but it is very limited and could use an improved user interface. For instance, as it now runs, every bit must be turned on using the mouse to click a check- box. This takes completely too much time and is quite annoying. Besides an improved sprite generator, a host of other utilities could be created. How about any one of these: color manipulator, fast assembly keypad routine, musical-tone generator (like *Sound X*, but for a non-VCS machine) or a VCS *Linux* kernel (kidding!)?

The reason the 2600 was such a successful machine in the first place, of course, had nothing to do with utilities that helped users create games for it. It concerned the games themselves. So what new games could we old-style fans come up with to retain excitement for our beloved machine? To be on par with Atari's best late releases (or most of Activision's, Imagic's or Parker Brothers' titles), the ideas would have to be pretty remarkable, while at the same time comprising realistic limits so that they could shine within the restricted capabilities of the VCS. How about *Kill the Bad Pac-Man?* I'm serious: It would offer a humorous perspective on a universal 2600 complaint while providing an entertaining contest. The player wanders around a *Robotron*-style area, and he can't shoot any Pac-Men that actually look like the hero of the arcade original. He can only shoot the very few Pac-Men that look like the protagonist in the 1982 home version. I suppose this concept draws a bit from *Demons to Diamonds,* in which you can only shoot certain characters unless you want to inadvertently create more bad guys, so it could conceivably be called *Demons to Diamonds II: Kill the Guy With the Eye.* If you shoot a non-2600 Pac-Man, he could turn into an even worse foe who kills you immediately: say, an Intellivision controller. The children's variation could make it easier to spot the 2600 Pac-Men: They'd flicker outrageously. Killing the correct Pac-Men, of course, would result in that sharp DUNT sound you get for eating a dot (line?) in the '82 home game.

On the more serious side, I think *Adventure II* is long overdue (20 years in January, to be exact). All that we hardcore *Adventure* fans want is the same game, with the same graphics, mechanics and characters, but in a kingdom that's, say, ten times the size of the original. Could you imagine? Well, it's possible! If advancements in the understanding of 2600 coding loopholes haven't created the accommodation for something like this in a 4K space, then bank-switching would easily house an *Adventure* sequel of this magnitude. Imagine additional castles: maybe a purple one, a chartreuse one and a teel-green one. And two extra dragons: Hansel and Gretel.

Rob Fulop could have a seat at his old desk and add power-ups and other such *Galaga*-type extras to his original concept to create *Demon Attack II*, and Parker Brothers could license the rights to the game version of *Independence Day*. Or am I getting too caught-up in the possibilities? Well, why not? Caught-up is fun! Try it and you just might find the motivation to help start another widespread VCS era!

Any one of these games or utilities could be created. The question is -- are there any VCS programmers out there who are willing to take up that task? I will continue to use my beloved VCS either way. I do hope, though, that there is a group of people willing to help carry the VCS torch. Are you one of them? – AJT and CF