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.

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 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!

Notes: