Remembering the Stone Age

Mark Baldwin

Computer games were not always built with multi-million dollar budgets and huge development teams. Originally, games were designed, written, and produced by a single individual. The first games were amateur productions published by sticking a disk and photocopy manuals in a baggie and hawking them at the local computer stores. It was in this Stone Age that I got started writing computer games and would like to tell you a little about it.

My first computer game was written on a "personal computer" that probably most of you have never heard of. In 1976 Hewlett Packard started manufacturing the HP9825. Although for marketing purposes they called it a ‘programmable calculator’ it was a true computer with a fully functioning (although strange) interpretive language called HPL. It had a full keyboard, a paper tape printer and a 32-character LED display. The ‘computer’ didn’t belong to me, but I was using it at work and playing with it after hours.

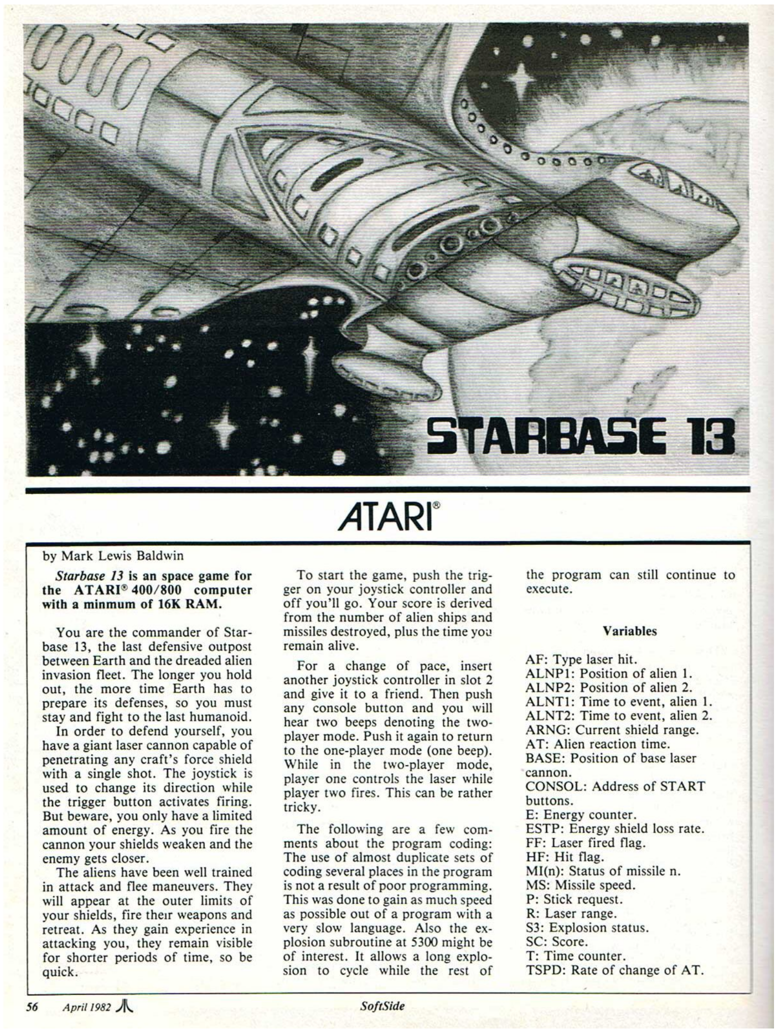
The game was my own version of the perennial ‘Trek’ that had been floating around main frames for years. This was a game that displayed a character map of space in which you flew a starship around destroying bad guys. You may ask how I could write a game with only a 32-character display? Well, the HP9825 also supported a printer. By combining the LED display for requesting input and using the printer to display the output (every time the game state would change, the ‘screen’ would be output to the printer), a playable version of the game was created. Simple and crude it may have been, but it was unique and impressive for those days, and gave me the computer game bug.

I moved to Houston, Texas, in 1979 to work on the Space Shuttle. Shortly thereafter, I bought my first personal computer, an Atari 800. Let me tell you about the Atari 800. This computer came out a little bit later than its two main competitors, the Apple II and the Commodore 64. All three used the 8-bit Motorola 6502 chip at 1.79 MHz. The chip could address 64k of memory, although all of it was not directly available to the programmer. The OS and supporting graphics and sound chips occupied the lowest 6k of memory and the upper 16k of memory was reserved for cartridges (basically ROMs that plugged into a cartridge slot) which contained programs and games. Graphics was 40 columns by 24 lines in text mode or 160x96 pixels in 128 colors (320x192 in 2 colors). Although by today’s standards this doesn’t sound like much, it was absolutely fantastic, especially for games by 1980 standards. In addition, there were specialized chips for both sound and sprites that allowed for games the Atari’s competitors couldn’t match. For that matter, the Atari was faster and better than the computers I was having to work with that were installed on the Space Shuttle!

Programming the Atari 800 was an interesting problem. Originally, there were two options, either Assembly or BASIC. I hated both. Assembly was fast but was obscure and time consuming to write. The BASIC we had at that time was slow to execute, limited, and cumbersome to write. How I desired to have a compiled language like FORTRAN that I used at work. Then one day I was looking through the Sunday paper and spotted a big ad by a department store chain selling the Atari. And in one corner, it said that they had not one but FOUR Computer Language Programs for the computer! I thought this a little odd since I had not heard anything about new languages being released, and even stranger that it was a department store, not a computer store that was selling them. You have probably already guessed my mistake, but at the time, it just didn't occur to me. I rushed down to the store salivating at the idea of getting a real computer language for my computer, and being dumfounded. The languages were Spanish, French, Italian and German. These were programs to teach languages, not computer languages, something the ad writers had no inkling or understanding of.

As a side story to these early days, I use to be in a medieval reenactment organization called the Society for Creative Anachronism. Being an active leader in the organization, I had a number of teenagers that hung around my house. One of these teenagers was trying to write games for the Apple computer. One of his first games he put together in baggies and started selling them at local computer stores. The name of the game was Akalabeth; the author was named Richard Garriott, later to be famous as Lord British for his Ultima games. I have always found it interesting that the two of us crossed paths before either of us made it in the game industry. Perhaps it was the same thing in both of us that led us both to sword fighting and computer games.

It was with my Atari 800 that I was finally in a position to start making a little money on computer games. Early on, there were a number of magazines on the market that published programs for the early personal computers. These were not programs on disk or downloadable from the internet (which didn’t exist), but instead programs printed on the pages that you then typed into your computer as source code. Generally these games and programs were in BASIC. But one problem was that games written in BASIC were very slow, being interpreted, which made arcade type games almost impossible. I took up the challenge of trying to write an arcade game in BASIC. This resulted in my game Starbase 13 which involved defending a starbase against alien attackers. I designed the game specifically to use minimum processing time so the result was a very simple game. But it worked and paid me $50 when it was published in Softside magazine.



STARBASE 13

ATARI®

by Mark Lewis Baldwin

Starbase 13 **is an space game for the ATARI® 400/800 computer with a minmum of 16K RAM.**

You are the commander of Starbase 13, the last defensive outpost between Earth and the dreaded alien invasion fleet. The longer you hold out, the more time Earth has to prepare its defenses, so you must stay and fight to the last humanoid.

In order to defend yourself, you have a giant laser cannon capable of penetrating any craft's force shield with a single shot. The joystick is used to change its direction while the trigger button activates firing. But beware, you only have a limited amount of energy. As you fire the cannon your shields weaken and the enemy gets closer.

The aliens have been well trained in attack and flee maneuvers. They will appear at the outer limits of your shields, fire their weapons and retreat. As they gain experience in attacking you, they remain visible for shorter periods of time, so be quick.

To start the game, push the trigger on your joystick controller and off you'll go. Your score is derived from the number of alien ships and missiles destroyed, plus the time you remain alive.

For a change of pace, insert another joystick controller in slot 2 and give it to a friend. Then push any console button and you will hear two beeps denoting the two-player mode. Push it again to return to the one-player mode (one beep). While in the two-player mode, player one controls the laser while player two fires. This can be rather tricky.

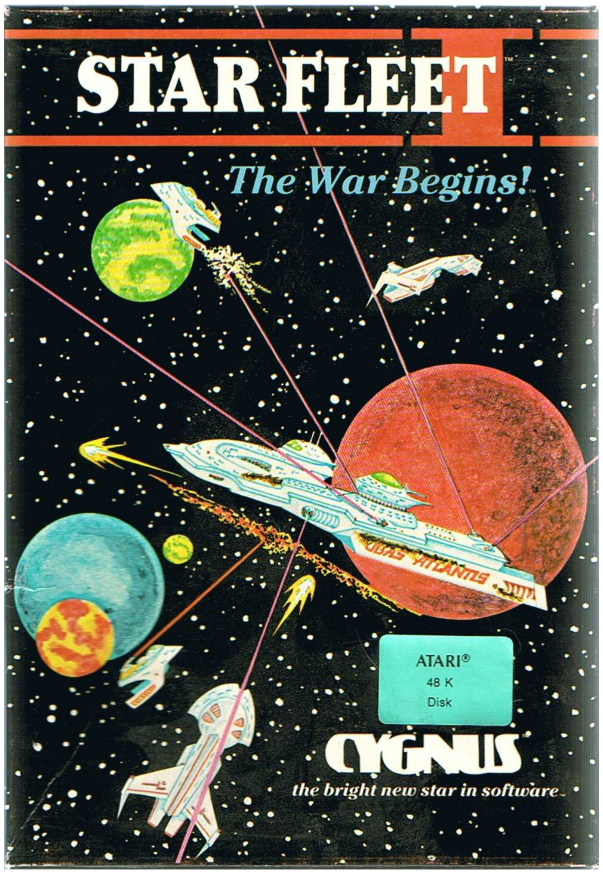
The following are a few comments about the program coding: The use of almost duplicate sets of coding several places in the program is not a result of poor programming. This was done to gain as much speed as possible out of a program with a very slow language. Also the explosion subroutine at 5300 might be of interest. It allows a long explosion to cycle while the rest of the program can still continue to execute.

Variables

AF: Type laser hit.
ALNP1: Position of alien 1.
ALNP2: Position of alien 2.
ALNT1: Time to event, alien 1.
ALNT2: Time to event, alien 2.
ARNG: Current shield range.
AT: Alien reaction time.
BASE: Position of base laser cannon.
CONSOL: Address of START buttons.
E: Energy counter.
ESTP: Energy shield loss rate.
FF: Laser fired flag.
HF: Hit flag.
MI(n): Status of missile n.
MS: Missile speed.
P: Stick request.
R: Laser range.
S3: Explosion status.
SC: Score.
T: Time counter.
TSPD: Rate of change of AT.

56 April 1982 SoftSide

These were leading me slowly towards professional games. In the mid 80's a few friends of mine who worked at NASA decided to form a computer game company. Originally, called Cygnus, and later changed to Interstel, the company specialized in strategy and role-playing games. Their first game was their own version of the old mainframe game Trek (similar to the one I wrote for the HP 9825) called Star Fleet I. Star Fleet I was first written for the new IBM PC, but Interstel needed it converted to the Atari 800 and asked me to do the conversion. BASIC was definitely not a solution, and I really didn't want to do the game in assembly. But luckily some genius had developed a new language for the Atari called ACTION just in time. This was a compiled language similar to C, came on a ROM cartridge and was a god send. Because of both the differences in language and hardware, I had to rewrite the game from scratch. And since I was rewriting the game anyway, I added new features and improvements. Note though that this was not like today's programming. My editor was a simple text editor on a 24 line by 40 column screen. There were no debuggers other than print statements I might put in the game. The compile time was something like 15 minutes; so much of the debugging was done on paper. For that matter, because the program was actually larger than the computers 42k of available RAM, I had to write my own memory overlay system.

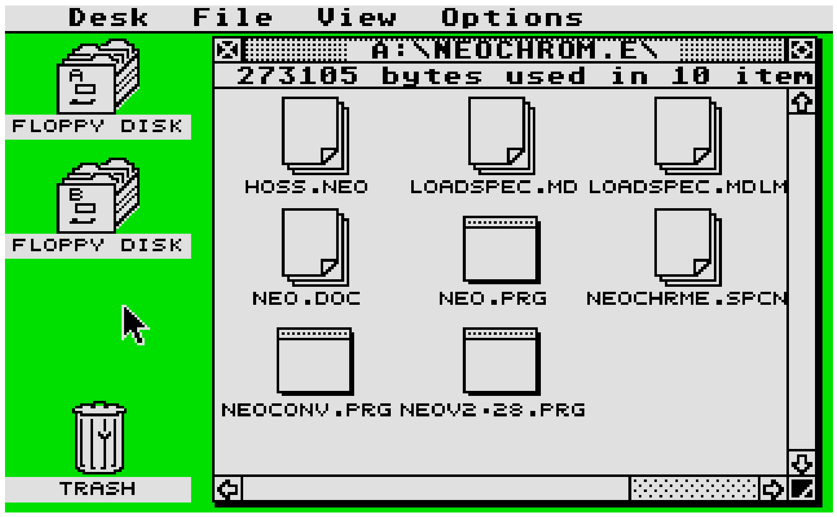


But it worked, and actually worked well. On the market, it was quite successful, and many of the innovations I developed for my version of Star Fleet I was then incorporated back into a new release of the PC version. This then became my first published box game. It was an absolute kick to go to a store and see my name on a game box.

It was also my most lucrative contract. In those days we were still trying to find the right balance between developer and publisher, and the contract we worked out was no advances (not common then) but I did receive 35% of wholesale receipts. At the time it looked reasonable if not low, but these days that is extraordinarily high, especially for a conversion.

My next project was another conversion of Star Fleet I. Technology, as always, had been changing and Atari had a new computer, the Atari ST. The Atari ST was Atari's answer to the Apple Macintosh. It used the same 16-bit Motorola 68000 chip, had 512k of memory, had a full keyboard including number pad and would support either a color or black and white monitor. The operating system was GEM, a graphical OS similar to the Mac and a predecessor to Windows.

So after having written Star Fleet I for the Atari 800, doing it for the ST should have been easy, right? But there were a number of problems. Again, there was the problem of languages, especially compiled languages. ACTION was not available, but PASCAL was. So I had to learn another computer language. Actually, I've always been glad I learned PASCAL, it taught me programming habits that I have found invaluable with the more modern object oriented languages. So PASCAL was the language for Star Fleet I on the Atari ST.



There was another problem. People had either Color monitors or Black and White monitors and the two were mutually exclusive. We had to design separate interface and graphics for each. And this is important when one wants to maximize sales and reach as large of an audience as possible. This meant two interface/screen/graphics designs.

Let me touch on an interesting problem in early game design. We were always restricted both in the number of pixels we could have on a screen and the number of colors. For the Atari ST I had 640x400 pixels for monochrome, 640 x 200 with 4 colors or 320x200 with 16 colors. The colors were from a 256 color palette, but were limited to only that many colors at one time. This was still a lot better than the IBM PC which then had what was known as CGA graphics that gave 320x200 in 4 of the most ugly fixed colors one could imagine (black, white, cyan and magenta). The point of this is that because of both the limited number of pixels and colors, if one wanted a good game that communicated to the user, it became extremely important to design interface and graphics down to the individual pixel. There was an ‘art’ to designing each icon and sprite, trying to fool the user’s eye into seeing a rich, complex and interesting image with a very limited number of pixels. What was even worse was this was normally done by the programmer/designer. There were no artist who specialized in this arcane knowledge; it was done by the programmer/writer/designer/artist, i.e., yours truly. (I also wrote the music and it was bad!) Even later on when I finally could afford an artist for games, this was still a problem. Most artists could not think in the constraining environment of communicative visuals with a limited number of pixels and colors. I went through many artists before I ever found one who could think and enjoy doing art with those limitations.

There was one other problem in creating my Atari ST version of Star Fleet I – a problem that still exists today. It was the increasing expectations of the audience. The ST was a generation newer machine than the Atari 800 and therefore the people who bought the software expected a much better game. This meant redesigning the game again to add more bells and whistles to meet those expectations. Cool animations and new weapons and tactics and even improved AI were necessary. So again, it was basically a rewrite and redesign from scratch.

Note that through these projects, I was not working full time on them. This was part time evening work while I held a full time job, first at Johnson Space Center on the Space Shuttle, and later at Martin Marietta in Denver. But I was now making money from my games. As much, if not more than I was earning as a ‘rocket scientist’. And it was fun too. The Shuttle was fun but it was rife with politics and frustrations. The frustrations included seeing a shuttle disaster coming and not being able to do anything about it. (I left NASA just three days before my fears were proven in the Challenger disaster, but that’s another story for another time.)

The combination of frustration with the aerospace industry combined with fun in the game industry made a hard decision easy. In February of 1987, I left the security of aerospace to the scary world of working for myself on games full time. It’s a decision I have never regretted. In the 17 years since then, I have done almost everything possible in the game industry. I have had successes in the game industry – including Game of the Year from Computer Gaming World – and I’m even proud to say I have had spectacular failures including one of the "Worst 25 games of All Time" according to Computer Gaming World.

Unfortunately, those days are gone, the industry is different now. But there is something to be learned from these Stone Age beginnings. Back then as a single individual I needed to solve problems and craft work that were both technical and creative; left brain and right brain. I needed to draw from all of my skill sets. Today, all of the jobs I was forced to assume have been broken down into separate job descriptions. But even with all of the possible specialized jobs that exist in the industry now, there is still the exciting need to draw from your total skill set, from your technical and creative, something you may not find in any other career path. And that makes for both a challenging and exciting career, and I am looking forward to the next 20 years with the same gleam in my eye I have had for the last 20.