Tom Defanti Interview (January 16, 2019)

By Kevin Bunch

Version 1.0 - April 24, 2020

Thanks to Kevin Bunch, who on April 24, 2020, told me that "Tom Defanti gave me the green light to pass this [interview] along to you for posting on the [BallyAlley.com website] if you want. It's the notes from the interview I did with him last year. - Adam Trionfo

Defanti interview notes

*Tom Defanti developed the GRASS computer language and was involved in its Z80 version, ZGRASS, as well as the Datamax UV1 computer and the unreleased ZGRASS add-under for the Bally/Astrocade Professional Arcade game console. Text is not verbatim and was originally typed while he was talking, as my recorder was not handy; it has been cleaned up for readability. This interview was conducted January 16, 2019.*

**Kevin Bunch**: How did you get involved with Bally working on the ZGRASS expansion for their home system?

**Tom Defanti:** While I was teaching at the University of Chicago, one of my grad students - Larry Meske, he was one of my grad students - went off to work for bally midway in their skunkworks, Dave Nutting Associates. Their chief engineer was Jeff Frederiksen. Larry brought us in, and it was clear they were mainly interested in coin operated video games but then they were interested in developing the Bally Arcade home unit, and it was clear that thing had enough – the graphics chip in it was brilliant, NTSC video, low res and higher res mode, served both the arcade games you put quarters in and in the home market. It was constrained by memory costs but it was the same chip that ran both of those systems; it was an in-development effort for the home game that was being done by Dave Nutting Associates. They brought me in potentially rewrite the GRASS language into the Z80 chip, their idea was that there would be a home library computer for a little bit; they showed it at CES a couple of times, it was certainly shown before the Apple II hit the streets, but then Bally decided to get out of that market. They didn’t actually produce the keyboard version, called the add-under, for the home library computer [editors note: alternate name for the Bally Arcade]; Anyway that was the gestation of it, and then we decided – Bally jumped – this company called Astrocade bought the Bally, and had some ideas about doing the add under but it didn’t make it, and then another company formed called Datamax, and they used the commercial chassis. They built a few hundred of these boxes, they were real computers with blinking lights and toggle switches, and we actually had a pretty good time. Most of the people who bought them were video artists who could record what they were doing and put out legitimate NTSC recordable video, which the Apple IIs didn’t. It was quite ideal for the artists because it was easy to use, kind of a version of BASIC but much more sophisticated with what you could do with screen manipulation, and all sorts of time-based things that you could coordinate. The visuals were simple but nevertheless it was very important to them, easy to add new things and try things; we wanted to make it bulletproof and real-time enough to use in a performance context; we did a lot of performances, lots of live events that involved a lot of sophisticated things you could do. You could essentially reboot the machine and it would be up and running immediately, that was pretty critical. The ZGRASS thing came along and eventually we put it into RT/1 boards, which went for a few more years and then I gave up.

**KB**: Why did you opt for the Bally custom graphics chip for the Datamax UV1?

**TD**: The same graphics chip was used in those arcade games like Wizard of Wor that used that chip. That chassis that has that chip in it, and up until that point the arcade games they were mostly motherboards scraped together - no really exchangeable parts, you’d just get a newer version of something that came along, but then this particular generation of games came along. The card cage was something you could plug into for extra memory or other features, and that’s what we adopted for the Datamax machine. We chose it because that was available; these things take a lot of work and these video game chips were pretty sophisticated: it had a bitmap and not many games at the time used three line buffers, given the energy the computer had stuffing the buffer would satisfy what you were doing with the video footage. The competing home games had tiny buffers, this chip was from a graphics point of view giving us a lot of flexibility. We had some investors bring it to us. I’ve gotten a lot of friends and students who used this to teach classes; they’ve put some stuff up on Youtube. People went from there to work on other things and did other stuff;

**KB**: Could you talk more on the working environment with Bally and converting GRASS to Z80?

**TD**: Technically they had this thing called the “in-circuit emulator” to step through code to debug it, otherwise it’s not viable to stop the thing and look over the variables as it would take too long to compile. So this could disassemble code and could look at the hex; generally speaking you couldn’t make much sense of it while running through the program, so that was a very useful box to debug code. We didn’t have sophisticated debuggers yet, that was the technical environment.

There weren’t a lot of people working on the project, Jamie Fenton wasn’t that involved anymore by the Datamax time, so I had a bunch of students doing things: running code and helping with it. It went from being a Bally project to being a project run out of this small company Datamax, which basically became a collective of people at the university working on it, so it was a social environment. The original environment with Bally at that point in the early 80s was using more RAM than everyone else on the planet combined, taking in more quarters in the coin-op boxes than Hollywood was getting in movie receipts, so this is nothing to sneeze at; it was a huge enterprise. Going into Bally’s manufacturing location in Chicago and saying that “you want to use my operating system and language,” was like a young author walking into the RL Donnelly building in Chicago. It was pretty exciting in the beginning, we really tried to do something that would be suitable for home use and would allow people to arrange recipes and do forecasts and stuff like that. Just, Bally decided to go elsewhere. They were investing in health clubs and amusement parks and got out of the game business, so it was a business decision. Bally chose what to do based on what they thought their future was. Of course I don’t agree with it, but given the size of the company and what we represented this was just an experiment.

**KB**: What’s your background with video art?

**TD**: My PhD is in computer science and computer graphics; I’m really a computer scientist, but got into computer graphics because my hobby is photography and filmmaking, videomaking. But I surrounded myself with people who were much more skillful. I can record a videotape but there are people who are better, so I wanted to use visual technology to make things look better and get buy-in from different groups. I made lab equipment for research labs, educational technology, research station technology; You put your heart and soul into something, sooner or later it wears out and you move on to something else, sort of a five-year cycle.

Jamie Fenton did work on the add-under, I’m sure she has an excellent recollection that could back up what I said. I spoke with her recently in Chicago and she was in very good spirits about the past. We had a panel together on digital media, if she chooses to – she wasn’t involved much with the Datamax stuff, was doing other things at that point. But there are definitely ideas in ZGRASS that were much improved from the GRASS language that Jamie adapted from her knowledge of the Z80 systems. There’s another person who would know, Nola Donato. She was a grad student who did a lot of code. She’s disappeared, and I don’t know where to find her - I think she went off to Microsoft.

**KB**: What made ZGRASS different from the GRASS language?

**TD**: The GRASS language was really a thing that assembled instructions, mostly endpoints but matrices and controlling this analog vector generating machine to create images in 3d and rotate and scale them in real time. It was very much pointed towards control systems in that box, and was also used in a lot of performances. If you look up Spiral TTL, that’s an example of what we did around 1980-81. Fundamentally it’s control over this vector box, and its languages – it had some very sophisticated screen processing parts of the language, which is what I did with this Spiral TTL, but also it was a live piece: the music; the color was black and white because we had an analog video processor; the ZGRASS had a frame buffer so it could do some stuff in real time but couldn’t do real time rotations except for small objects, but there were things we could do with a frame buffer as those were too expensive earlier to have. So ZGRASS had some fairly sophisticated elements to it. They both had a compiler, which allowed you to speed up the athletic sections. Probably the biggest difference in ZGRASS was Jamie’s addition of a way to have the necro-structure subroutine of it more sophisticated so that the GRASS language would have more variables. With ZGRASS, it only operated in one plane, which made things way simpler. It really had to do with the commands and interpretive languages in each thing you typed, each thousand lines of code. If you’re looking at thousands of lines of code you can do things. RT/1 was another complete recoding done around mid-87. Each time you learn something, but they’re all pretty much the same conceptually. The whole idea was to make it available to artists so you could type and interact with the controls - tactile controls and other kinds of audio inputs and control things, parts you could use in real time. And back in those days with single frame video was prohibitively expensive, and now - the Star Wars thing we did with Vector General, we used a 35mm film camera, had to make everything fit within the video production workflows, and that was kind of revolutionary.