Interview with Steven Boykey Sidley re: *Entombed*

John Aycock
Department of Computer Science
University of Calgary
2500 University Drive N.W.
Calgary, AB, Canada T2N 1N4
aycock@ucalgary.ca

TR 2016-1090-09, October 2016

Preamble

This is an interview with Steven Boykey Sidley, conducted via email on October 1–2, 2016. He programmed the Atari 2600 game *Entombed* while working at Western Technologies, which was published by US Games in 1982, and also worked on a number of other games for different platforms.

Interview

(Interview questions appear in *italics*.)

Tell me the story about how you got involved with Entombed*.*

I notice that my name was not mentioned in the Wikipedia article[2] – just Tom and Jeff, who were game designers at Western Technologies where I developed the game. In any event, there is an interesting story about this game. I am now an award-winning

[1] https://creativecommons.org/licenses/by-sa/3.0/

[2] https://en.wikipedia.org/w/index.php?title=Entombed_(1982_video_game)&oldid=738614519

novelist, so you are going to hear it, if for no other reason than I never thought I would have reason to repeat it.

It was around 1980, and I had recently graduated with an MS in Computer Science from UCLA. Not really wanting to go to IBM or equivalent, I applied for, and was accepted for a position at Western Technologies in Santa Monica. Video games had just started – Bushnell was the legend – everyone knew the story of Pong and the jammed coin slot in the bar.

So Western Technologies, who had been a toy company, decided to climb on the video games bandwagon. When I arrived, there were a number of realisations that came to me quickly.

1. The company was peopled with weirdos, awkward eccentrics, nerds, dopers and misfits. They were a strange menagerie for this South African Jew, recently arrived in the US. Incredible fun.
2. Atari, in their attempts to hold onto a game monopoly, had provided no manuals at all. The only way to understand how the 2600 worked was to build an emulator out of the innards of the videogame machine, wire it up to a very early MS-DOS PC (maybe it was an Apple II), and literally probe the registers of the 2600's 6502 and other chips to see what happened on screen. This was reverse engineering at its most pure.
3. I didn't have a fucking clue what I was doing. I had never seen an emulator, I had never programmed in assembly, didn't know what a chip was. I was armed with deep knowledge of PL/I and database schemas and mainframe architecture from my UCLA Masters. All entirely useless.
4. WT was expanding into a new building. No one had moved in yet. They directed me to a desk in the empty building with the PC/2600 Atari emulator. No humans, no manual. I nearly quit on the first day.
5. The basic maze generating routine had been partially written by a stoner who had left. I contacted him to try and understand what the maze generating algorithm did. He told me it came upon him when he was drunk and whacked out of his brain, he coded it up in assembly overnight before he passed out, but now could not for the life of him remember how the algorithm worked. So not only did I have to reverse engineer the chip, I also had to reverse engineer the algorithm – there were no comments in the codeset.

I went on to write a few more games for the Atari, TI-99/4A, and Commodore and then moved off into what seems like 10 or 20 different careers.

How long did it take you to develop Entombed*?*

It was about three months to first version, and then a month or two for attract screen and autorun, and score save, and packaging, etc.

What was your development environment like? Obviously you worked in assembly, but

what computer (dumb terminal?), tools, and editor did you use? Was it a line editor or was it full-screen?

I do not think there there was a full screen editor. I think the master PC was MS-DOS and a line editor was used. The emulator was built to drop into one of the expansion slots.

How did you debug your code?

There were no tools except single-step trace. That was it. But assembly is much easier to debug than high level languages, because as you step through, you can see registers change. However, real-time bugs that were a function of a given joystick/button combination at a given instant were not possible to debug by any other means than thinking about it.

Take me through a day in the life of a 2600 programmer at WT.

I had a fairly conservative work ethic (9–5) and sometimes I schlepped the kit home to program at night. Others in the group were more, um, eccentric, pulling all-nighters, getting fucked up, disappearing for days, etc. And yes, code printouts were *de rigeur*. Because each game was a lone wolf effort, there was little collaboration, except an occasional 'hey, anyone know how to transfer some ROM code to RAM', yelled across the room.

How far along was the reverse-engineering of the 2600 when you started work at WT?

I came in very close to the start – they had just finished the emulator and all of the 'basic' functions were emulated. We still did quite a lot of reverse engineering, at least on *Entombed*. We had it pretty much down after that.

Was there some internal "bible" where everything known about the 2600 was written down?

Not a thing. We were not disciplined enough. I am sure that they did it later for new programmers, though.

It looks like the row-generation algorithm in Entombed *takes the last two bits generated in the row, plus the three bits immediately adjacent in the row above, and uses that to index into a 32-byte table. The table's value determines whether to produce a 1, a 0, or a random bit. The question is, where did the values in the table come from? There doesn't seem to be an obvious pattern – were these values manually chosen?*

Heh heh... this was the piece that the other guy did when drunk and stoned. It was a mystery to me too, I couldn't unscramble it. I just used it to generate the new row at the bottom of the screen.

The code for the pseudo-random number generator in Entombed *appears to be the same as the one in* Towering Inferno. *Would that have been in the maze code you*

started with, or was there a standard pool of code at WT to draw upon? Or perhaps the sharing went the other direction, since the exact release dates of Entombed *and* Towering Inferno *are both simply given as 1982?*

Different code sets, and *Towering Inferno* was another programmer. We did not really reuse each other's code, so I don't know where he got the random number generator from. I seem to remember there were a few that were knocking about.

How did programming the 2600 make you feel?

Most fun I have ever had, when I finally understood how to do it. I felt in possession of secret knowledge.

Looking back, what would you have done to improve Entombed?

It pretty much pushed the edge of the technology of the time. I am not sure there was much more that could have been done, at least not technically.

You clearly had many challenges developing this game. What was the part of Entombed *that gave you the* most *difficulty during its development?*

Keeping the code tight enough to execute in 1/30 sec during vertical refresh. This required continual tightening of logic.

Does the source code for Entombed *still exist in any form?*

Unlikely.

For a 2600 game, a surprising number of people seemed to have been involved in one way or another with Entombed*. You did the programming, of course. Thinking of the other people credited (Jeff Corsiglia and Tom Sloper), what roles did they play in development of* Entombed*?*

They did the original design and sketches and game logic concept, I believe. I think they also did artwork for packaging.

I unfortunately haven't been able to find any programming credits for you. What were the names of the other games you worked on?

If you look up Sidley Mckay on Google, you will find a couple of games I and my partner, Graham Mckay, developed for a distributor named Tronix. There was *Suicide Strike* and *Slalom* (I developed the latter for Commodore 64 – there are a few screen grabs on the web somewhere). I cannot remember the names of the others – I think there were four in total. I also did one for the TI-99/4A, whose name also escapes me.

Do you have anything to add?

Those of us who programmed 6502 for those games have a special bond. Like climbing Everest without oxygen.