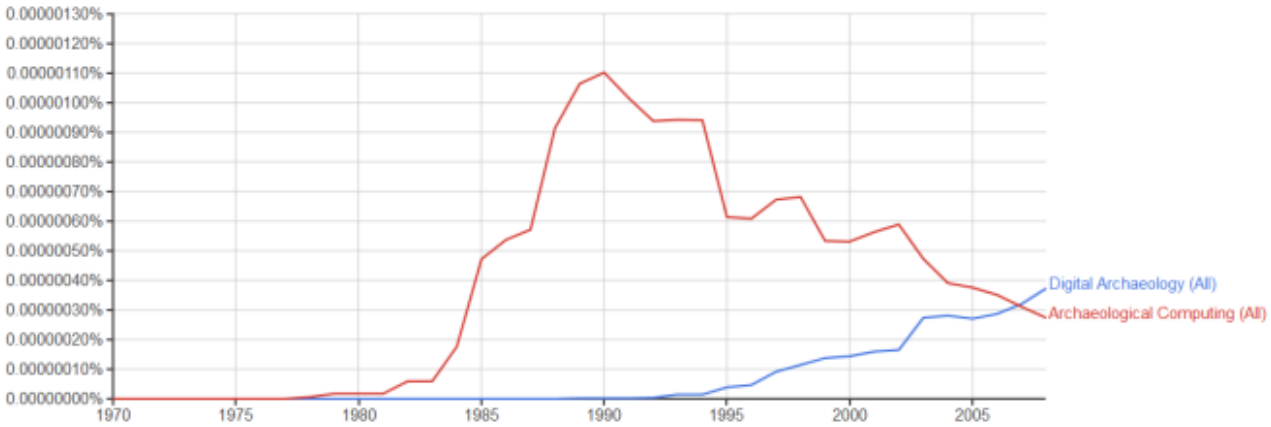
Mastering Mystery

written by Jeremy Huggett | 06/03/2015

A couple of articles have appeared in recent days which in different ways suggest that digital technology is too easy. Samuel Arbesman writes about how it is important to be able to see under the hood of the tools we use; Brian Millar suggests that simplicity of design takes away the satisfaction and confidence gained through mastery of the technology. In different ways, both link understanding with our ability to see past glossy interfaces designed to keep us at arms length from the guts of the process. Arbesman's reminiscences about typing games programs from magazines into a VIC-20 and gaining an appreciation of coding and debugging certainly makes me nostalgic – in my case it was working with my younger brother's Commodore-64 which led directly to working as a research student on machines ranging from ICL and VAX mainframes, North Star Horizons and Sirius PCs running CP/M, to the original IBM PCs and MS-DOS, and using them (and more recent models!) to construct archaeological programs in FORTRAN, C and C++.

Look across the collection of proceedings from the Computer Applications in Archaeology conference since 1973 and the change is evident – numerous discussions of purpose-built programs shift through time to discussions of largely off-the-peg software solutions. Is there a connection with the relative shift in terminology from 'archaeological computing' (with its connotations of computing science, coding, etc.) to 'digital archaeology'?



Google Ngram showing the trending of terms in Google's English corpus 1950-2008

This mirrors a shift from 'humanities computing' to 'digital humanities' seen as part of claiming a broader scope and disciplinification (as discussed in Svensson 2009, for example) and carries with it anxiety about the place of building things – coding – as a scholarly activity (for instance, Ramsay and Rockwell 2012).

This is not to decry the value and benefits of GIS, CAD, DBMS, and the like – indeed, many have their APIs and the ability to incorporate plugins coded in anything from C to Lisp to Python, although few archaeologists have seized the opportunity. And why should we, if there is no need? In the past, the need was more obvious. For instance, twenty years or more ago teaching archaeology students

to develop and use tables in a semi-relational way in dBase III required delving beneath the (barely) user-friendly interface, not just to work at the command line but to actually learn elements of the programming language – indeed, there was a real sense of mastering a mystery (and who remembers what Ctrl-K-S does/did? [*]). Have we lost something in the shift to off-the-peg solutions and zero coding as Arbesman and Millar suggest?

This isn't something that is much discussed these days, more so in the past. For instance, Huggett (2004) and Daly and Evans (2006) drew attention to the 'black box' effect of software in archaeology and expressed concern about the importance of not just knowing about inputs and outputs, but also understanding what it is they want to do, and how the tools they use do it (Daly and Evans 2006, 254). Such knowledge goes beyond which menu option to apply or button to press, but does it need an in-depth knowledge of the software itself (arguably an impossible requirement in the case of commercial packages)? Or is it more a case of acquiring an appreciation of how software works through, for instance, knowing how to extend it by writing code snippets to perform bespoke calculations or reformat data? As Arbesman writes,

> "We have tools and interfaces to prevent our lives from being miserable, even if it means we lose some of the mystique of our machines. But we should be able to see under the hood a little. If we see our tablets and phones as mere polished slabs of glass and metal, performing veritable feats of magic, and have little clue what is happening beneath the surface or in the digital sinews, something is lost."

Rather than preventing misery, these tools and interfaces have opened up access to a wider audience, perhaps. But what is actually lost? David Millar suggests:

> "We live in a world where almost everything is designed to be simple. When I work with designers, I often come across a kind of religious belief summed up in the mantra: don't make me think."

If something is difficult to do, then it requires understanding, practice, skill to achieve it. If it is too easy, it reduces the need to think, to reflect, and encourages a state of powerless acceptance. This isn't an argument based on the nostalgic premise that "it was much harder in my day" because it wasn't – the parameters and possibilities have changed so that while much of what was hard in the past is now easy(ish), it means we can focus on new, hard, and more interesting things instead. The question is whether we should be content with devolving responsibility to programmers remote in time, space, and intention, and to the black boxes they create for us? Or should we be concerned to ensure that we have a clear understanding of the tools we use, rather than mindlessly or unknowingly clicking away?

References

Arbesman, S. 2015 'Get under the hood', *Aeon Magazine*.

Daley, P and Evans, T. 2006 'Afterword', in T. Evans and P. Daley (eds.) *Digital Archaeology:*

*Bridging Method and Theory (*Routledge, Abingdon), pp. 253-255.

Huggett, J. 2004 'The Past in Bits: Towards an Archaeology of Information Technology?', *Internet Archaeology* 15.

Millar, B. 2015 'Why we should design things to be difficult to use', *The Guardian* (4th March 2015).

Ramsay, S.and Rockwell, G. 2012 'Developing Things: Notes toward an Epistemology of Building in the Digital Humanities', In M. Gold (ed.) *Debates in the Digital Humanities* (University of Minnesota Press, Minneapolis), pp. 75-84.

Svensson, P. 2009 'Humanities computing as digital humanities', *Digital Humanities Quarterly* 3 (3).

---

[*] Ctrl-K-S saved your work and continued editing in CP/M and MS-DOS Wordstar in the 1980s – it was the near-instinctive 'chord' pressed while you thought about the next phrase … </nostalgia> ^