KINEPHANOS
Revue d'études des médias et de culture populaire / Journal of media studies and popular culture

Special issue
Preserving Play
August 2018 34-50

Finding the invisible

An experience-based methodology for selecting retrogame archaeology "sites"

John Aycock

University of Calgary

Abstract: Part of understanding and preserving game history is the study of games' implementation, a technical approach that I refer to as "retrogame archaeology." The first problem in retrogame archaeology is simply choosing a game to study, because interesting implementation aspects are not necessarily known or visible even to a trained eye. This paper distills years of experience conducting retrogame archaeology and addressing this exact problem, categorizing the different methods I have used, their strengths and weaknesses, and begins to examine how these efforts might scale.

Keywords: Retrogame, Media archaeology, Game history methods, Game selection

It is hard to begin a paper with the phrase "in any field of human inquiry" and not sound like a pompous ass. Nevertheless: in any field of human inquiry, it is important not only to document and disseminate findings, but also the process through which those findings were arrived at. This is certainly a critical part of a scientific endeavor for purposes of reproducibility and

possible falsifiability. More generally, especially for developing fields of study, documenting the ways of knowing allows others to follow the same methods, or indeed, subject those methods to critique, improvement, or even rejection.

The field of study in question here is retrogame archaeology, a technical study of 'the tools, techniques, and technology used in old games' implementation' (Aycock, 2016, p. 206), and the connection of those ideas to a broader, modern technical context. Retrogame archaeology is both complementary to and overlapping with a number of extant areas. Game history is an obvious area, along with game studies; the latter is particularly noteworthy in that the technical analysis performed by retrogame archaeology is a conspicuous omission in some accounts of how to analyze games (e.g., Egenfeldt-Nielsen et al., 2015), and yet it underpins the very games they study. In the broader sense of software, critical code studies (e.g., Montfort et al., 2013) and software studies (Fuller, ed., 2008) are connected to retrogame archaeology. Media archaeology (e.g., Parikka, 2012) and media forensics (Kirschenbaum, 2008) may also be thrown into the mix. However, perhaps the largest overlap is in games with platform studies (e.g., Montfort and Bogost, 2009; Altice, 2015), although retrogame archaeology is more technical still. A parallel development comes from archaeology itself in the developing field of archaeogaming, 'the archaeology both in and of digital games' (Reinhard, forthcoming, p. 1), where retrogame archaeology fits nicely insofar as it is an "excavation" and study of old game artifacts, both digital and physical. The separation of retrogame archaeology from these other fields is revisited more thoroughly later in the Discussion section.

The high-level workflow of retrogame archaeology is shown in Figure 1. A game is chosen as an object of study, is acquired, run (if possible), analyzed using one or more methods, and finally contextualized. Depending on the game, any or all of these steps may be non-trivial undertakings. The analysis methods have been described already (Aycock and Reinhard, 2017), and in this paper I focus instead on the first step: choosing a game. In the over three years that I have been working on retrogame archaeology, I've observed that the methods I've used for choosing games fall into five categories that I present along with my experience with them and examples of their use.

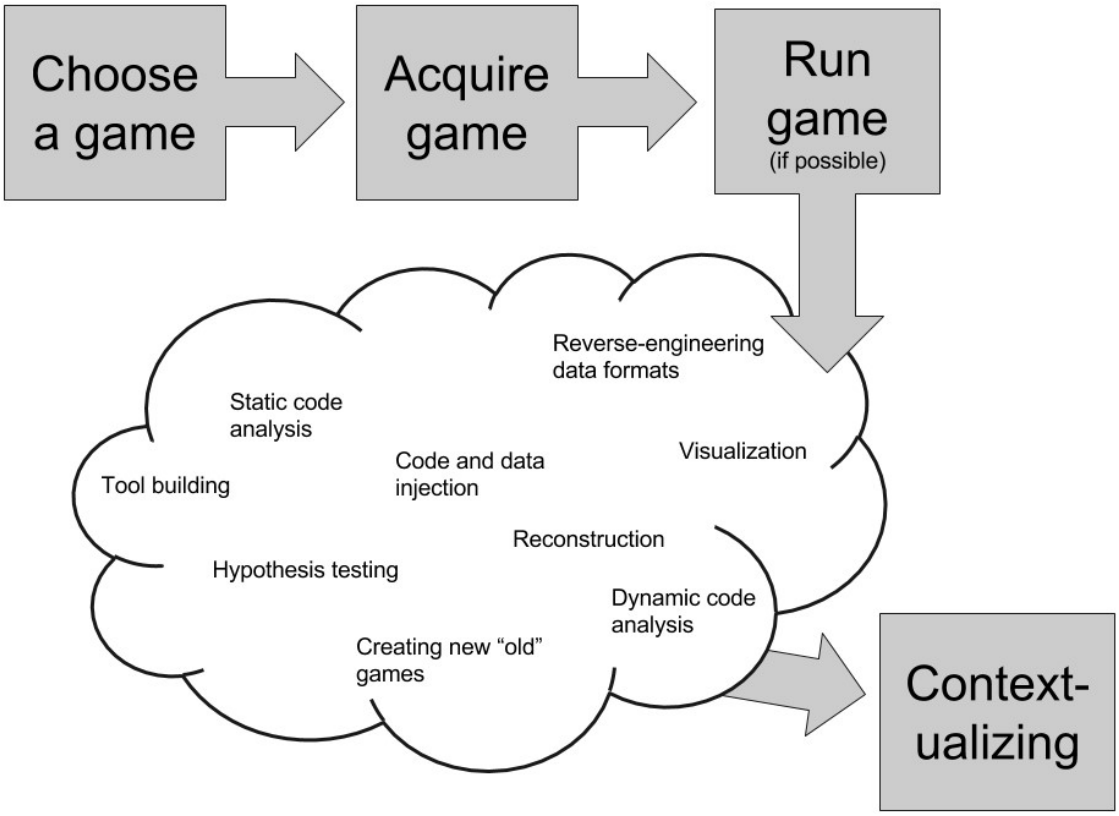

Figure 1. Retrogame archaeology workflow.

Moreover, it is important to note how this differs in some key aspects from how a game historian might approach choosing a game to study. Retrogame archaeology is *not* concerned with games that are the first in some way, or the best, or the most influential. Indeed, some examples of certain implementation techniques are rare, and to tightly constrain the set of possible games would condemn some searches to failure. This begins to hint at the problem of choosing a game in retrogame archaeology, though. For a game historian choosing a game, they would likely start with a reasonable first-order approximation of a game's impact and influence on later games and culture, an understanding of whether the game was important and why. By contrast, a retrogame archeologist is concerned with the game's implementation, that which is inside and often invisible through gameplay even to a trained eye. Plumbing the depths of a game's implementation takes time and effort, and a researcher's time is finite; where should their energies be devoted?

Choosing a game

The methods for choosing a game to study have fallen into five categories, based on my experience performing retrogame archaeology.

Traditional searching

In an academic setting, it might be expected that a search using "traditional" sources would include publications in academic venues. The study of game implementation has not yet reached the point of critical mass where such publications abound, however. In cases where detailed treatises exist, they may exist in hackeresque journals: for example, a technical bestiary of copy protection schemes for the Apple II (Ferrie, 2016), or reverse engineering of Atari's *Star Raiders* (Wiest, 2016). The name of the journal where these excellent technical articles appear? PoC||GTFO, the 'International Journal of Proof-of-Concept or Get The Fuck Out.'

Instead, the traditional sources I refer to here are ones not unfamiliar to any student, and in particular I mean Wikipedia and Google. There are lots of game enthusiasts who have devoted inordinate amounts of time to studying their favorite games, making web pages devoted to them, discussing them in forums, and blogging about them; there is no reason not to leverage their efforts.[1] Certainly there is a tradition in research of building on what has come before, and while there is a tendency to think of "what has come before" as scholarly work, there is really no reason that that must always be the case.

As an example, I wanted to find an example of a game using a type of data compression called run-length encoding. This is a technique whose use would not at all be apparent from the point of view of a game player; it is a method for fitting more data (e.g., an in-game image) into a tightly-constrained storage space. After a number of Google searches employing various keywords, I happened across a blog called 'Gaming After 40' that talked about the 1982 game *Spook House* on the TRS-80, mentioning in passing 'My own peek at the disk contents implies that the graphics were stored as raw blocks of screen data, perhaps with some sort of run-length encoding for compression' (Dobson, 2009).

Finding this was really only the beginning of my work on *Spook House*, though, and it would be fairer to characterize this - and in fact all - methods of choosing games to study as lead

[1] With appropriate acknowledgment, of course.

generation. The mention of run-length encoding in a blog is helpful, without a doubt, but it is not by any stretch of the imagination a primary source. Even a meticulous third-party disassembly of game code is not a primary source, because already it is subject to human interpretation and error. There are really only two primary sources for games in the retrogame archaeology sense: the original source code for the game, which is relatively rare to acquire, and the game code running on the target platform, which is typically in compiled or assembled binary form. Finding leads is good regardless of what form they come in, but their veracity *must* be properly ascertained.

Magazines

At the time of this writing, there are two English-language magazines devoted to retrogames (*Retro* and *Retro Gamer*), and at least two foreign-language ones (*Return* in German and *WarpZone* in Portuguese). While some of these magazines' content will seep out over time to be found with the traditional searches above, I have found enough interesting leads in these magazines to warrant separate mention.

Something these magazines feature are interviews with retrogame programmers, because we are at a fortunate time historically when many of them are still alive. While an interview as a whole is not of use, in my experience there may be one or two sentences mentioned in passing by the programmer that point to potentially interesting technical feats. For example, Greg Holmes wrote *Jack the Nipper* and was quoted in *Retro Gamer* as saying 'I had to use illegal Z80 instructions because I ran out of registers' (Milne, 2014, p. 46). Old computer CPUs would have an official, documented set of instructions they supported, but would often have unofficial, undocumented instructions that a programmer could also use, albeit with some attendant risks. Holmes is effectively saying that the language he used to express his game incorporated nonstandard commands, along with a rationale as to why he did this. In another example, a sidebar on an article in *Retro Gamer* about *Pac-Land* posed a handful of questions to Alan Ogg, the programmer who ported the arcade game to the Commodore 64. Talking about how constrained the computer's memory was with respect to the game, he said the memory was 'full to bursting' and required 'having to compress some of the sprite and level data and were even using the 6502 stack for running code in, which was really pushing our luck. Even the tape loader over-wrote its own code at the end of the load' (Bevan, 2014, p. 73). While the technical

meaning of this would take a while to unpack, suffice it to say that he was incorporating programming tricks out of necessity that would not be normally considered best practice.

The down side is that a programmer's recollections about game code they wrote decades ago do not constitute a primary source, a point I have argued elsewhere (Aycock and Reinhard, 2017). In my work, I have encountered retrogame programmers who enjoyed near-perfect memory of their old code, and at the other end of the spectrum, programmers who recalled almost none of it. The original programmers cannot be generally considered a reliable source for their own code; too much time has elapsed. As with traditional searches, anything the programmers mention must be treated tentatively until confirmed in a primary source. Here, I verified the technical claims for both *Jack the Nipper* and *Pac-Land* in the game code itself.[2]

Brute-force search

Programmer reminiscences are a good lead-in to the next method for choosing a game. David Crane, creator of *Pitfall!*, gave a game postmortem talk at the 2011 Game Developer's Conference. During the question period afterwards, he mentioned in passing that some games on the Atari 2600 would run game code from the 2600's RAM (Crane, 2011, 42:12). This is a surprising claim, because the Atari 2600 only had 128 bytes of RAM, making it a precious resource not to be consumed lightly. I contacted David, and he kindly replied but was unfortunately unable to recall which games might have done that (Crane, 2013).

Fortunately, that tip was sufficient. The conditions under which this occurs can be expressed as a binary condition; I set an Atari 2600 emulator to stop the game if this condition arose,[3] and started running a large corpus of 2600 games one by one. In computer science, this is referred to as a brute-force search. Although laborious, eventually I found some games that used this technique and, as with other brute-force searches, there are advantages to cleverly ordering the search space. The first game I found testing positive with my search happened to be the infamous *E.T.*, by Howard Scott Warshaw. Knowing how programmers are fond of reusing code, I looked in Warshaw's other games next and immediately found two more examples of its

[2] As a sidebar to Ogg's sidebar, the importance of serendipity in analyzing game code: *Pac-Land* contained a use of run-length encoding as well.

[3] The condition is where the CPU's program counter is referring to the 6507's zero page memory where the Atari 2600's RAM is located. In the Stella emulator I used, the stopping condition is expressed as breakif { pc < 256 }.

use.

In another instance of a brute-force search, I was not following a tip, but a hunch. I suspected that, due to the architecture of the Atari 400/800 platform, a particular game copy protection technique would be used whereby the game (stored in a non-writeable cartridge) would try to write to its own memory locations to thwart copies that weren't in a cartridge. Again this can be expressed as a binary condition for an emulator to test, although a more complex one.[4] Running this test during a brute-force search on a small assemblage of games revealed my hunch to be correct and helped winnow down the candidate games into a single good example.

Curiosity-driven questions

Other areas of research have curiosity-driven questions, and it is not surprising that these are found in retrogame archaeology too. Some questions derive from an understanding of a platform's limitations. For example, upon learning about *The 7th Guest*, a very early CD-ROM game with elaborate full-motion video and sound, it occurred to me that the game ran on machines with early (read: slow) CD-ROM drives and limited bandwidth between the drive and the computer; it seemed almost certain that some type of data compression was at play. But what was it? Similarly, I was playing *Super Mario Bros.* on the NES when, with Mario mid-jump, I wondered how the level data was stored - the cartridge's capacity was not large, yet the levels seemed long. Was some compression or clever encoding being used?

In other instances, the driver is a strong visual similarity between two games, one using already-discovered techniques and one employing unknown means. For example, I had already studied *Pitfall!* on the Atari 2600 when I noticed that the later *Pitfall!* port for the Intellivision appeared to produce a series of game screens that was suspiciously close to the Atari version; was it doing it the same way? In another case, I knew the technique used for the colorful 'neutral zone' and other visual effects in *Yar's Revenge*, and stumbled across *Laser Gates* that had force fields exhibiting the same apparent random colors. Again, was the game accomplishing this the same way?

More complicated processes in games have more ways in which they can be implemented, and

[4] Using the MESS emulator, wpset 8000,4000,w,pc>=8000 && pc<c000. In English, this is watching for writes to be made to a specific range of the computer's memory where the cartridge would reside, but only when the CPU's program counter is located within that same range.

the curiosity-driven questions need to be adjusted accordingly. *Amazing Maze*, *Entombed*, *3D Labyrinth*, and *Rogue* all produce mazes, yet learning their algorithms revealed that they all use different methods. Here the goal is comparative retrogame archaeology, where the question is not if two games implement something the same way - they probably won't - but how these two games differ in their approaches. As a final example, I already knew how *Berzerk* generated the walls in rooms, and I saw a faint visual likeness to the rooms in *Castle Wolfenstein*. After reverse engineering code for the latter game and writing a program to reconstruct the castle rooms from the game data, I can attest that the two games use wildly different approaches, and in some ways act as a counterpoint to one another.

It is interesting to note that in the examples I have given for curiosity-driven questions, all of these situations arose when I was not actively researching. Many cropped up when casually playing the games, like *Castle Wolfenstein*, *Pitfall!*, and *Super Mario Bros.*; the question about *The 7th Guest* came to me while sitting in on a lecture in a game history class; *Laser Gates* was the result of perusing eBay listings, not recognizing the title, and finding a gameplay video. There is value in being attuned to ask these kinds of questions.

Social media

The final method for choosing a game to study involves leveraging social media. One possibility would be to request examples from an online social network, but not all researchers enjoy that sort of following (I certainly don't) and this active social network use would not bear fruit for most.

Over a period of months, I had noticed that my traditional searching methods for game copy protection examples would often reveal a common moniker: 4am. He/she is a modern cracker of retro software, and leaves not only unprotected software behind, but a complete write-up about the copy protection scheme in minute detail. Clearly this was a valuable flow of research leads; the question was how to see them all and not just encounter them by happenstance.

The answer came through Twitter. I discovered that 4am (@a2_4am) announced all his/her cracks that way, and by following on Twitter I had a constant stream of information. As I expanded the set of people I followed judiciously, to receive more signal than noise and not miss anything important, more examples came rolling past. This included retrogame source code releases that I might otherwise have overlooked, but more importantly extended the scope

of examples to other areas of the world whose retrogames I was unfamiliar with. The *Jet Set Willy* copy protection I later studied (Aycock and Reinhard, 2017) was one of these, as was the start of my ongoing involvement with a system called the *Graphic Adventure Creator*.

Choosing all games: retrogame archaeology at scale

How is the transition made from choosing one game to choosing all games? For me, it began by accident. A 2015 tweet by a person I followed on Twitter offhandedly said 'Coding text adventure conditions like a pro' along with a picture of a few lines of code, all uppercase, in a language I was unfamiliar with (Vogt, 2015). When I inquired about the language, he told me it was an old tool called 'G.A.C.,' the *Graphic Adventure Creator*. *GAC* was a program that allowed users to create text adventures, optionally with accompanying graphic images, and was one of several similar tools that enjoyed most of its popularity on computers in the UK and Europe, like the ZX Spectrum. Not somewhere I have lived; not a computer I have used extensively.

Something on the Wikipedia description of *GAC* caught my eye ('Graphic Adventure Creator,' 2017): 'Over 117 titles were written using GAC.' The implication was that if I could analyze this one tool, I could potentially analyze over 100 games in one fell swoop. I found two manuals that described tolerably well - but not completely - the syntax and semantics of the *GAC* language (Incentive Software Ltd., 1985, 1986) and used that information plus some experimentation to create my own reimplementation. My version, implemented in Python, had a *GAC* interpreter that would read games expressed as *GAC* code that happened to be located inside Python modules. This gave me an implementation I could easily work with and experiment on.

Next, I manually reverse engineered the *GAC* game format on the ZX Spectrum. I then applied this knowledge to write a Python program that would automatically extract a *GAC* game from a ZX Spectrum memory image, and output it as a Python module that I could run with my Python *GAC* interpreter. Obviously the next step was to acquire *GAC* games, and I collected a corpus of all the ZX Spectrum games made with *GAC* that I could locate: 130 games and 152 images in total (some games have multiple parts). The games are primarily in English, with the occasional example in Spanish, Portuguese, and Polish.

The last piece was an analysis framework I wrote in Python that allows me to automatically run

through the entire corpus and gather statistics, analyze certain features, and understand how programmers used the *GAC* language in practice. This in turn draws on my *GAC* interpreter for cases where dynamic analysis is needed. The full workflow is shown in Figure 2.

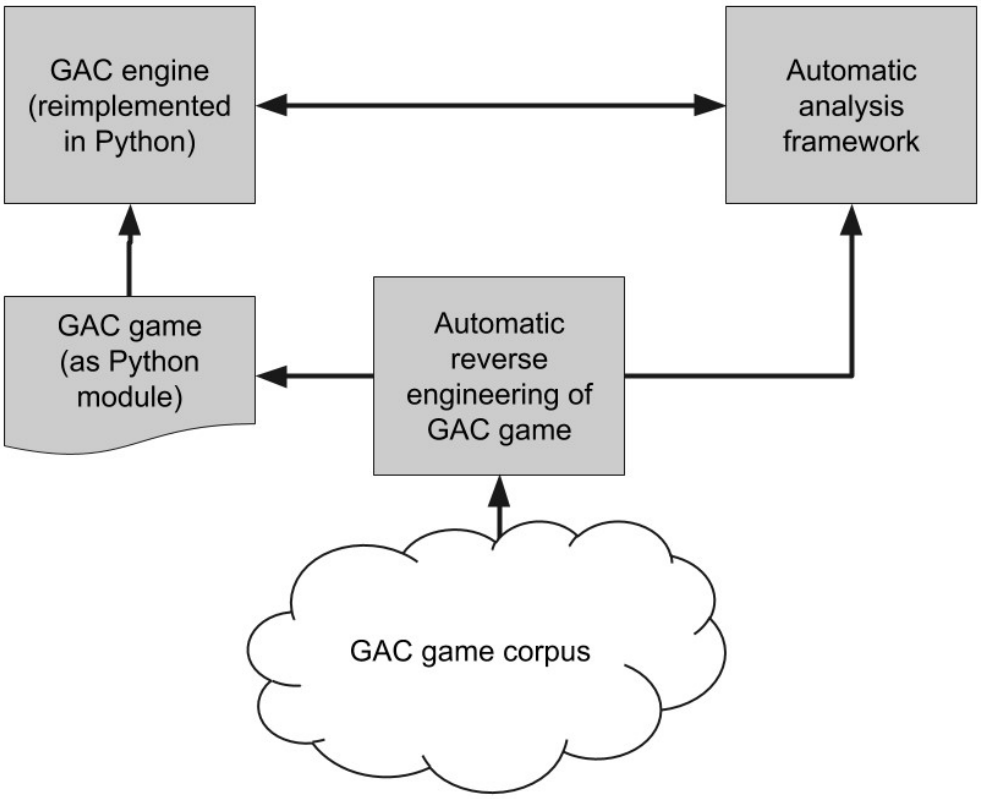

Figure 2. *Graphic Adventure Creator* game analysis workflow.

From a digital humanities point of view, this use of a game corpus may be construed as distant reading. I should point out, however, that this system allows me to see both the forest and the trees. I have analyses that gather aggregate statistics about, for instance, the number of verbs, rooms, and objects in the games. At the other extreme, I can ask very specific questions. In one game, I happened across a hidden message that could not be displayed during gameplay; it was a simple matter at that point to create a small analysis program using my framework to search for hidden messages in all other games in the corpus.

Once all this infrastructure is in place, other possibilities manifest themselves. I currently have used this as part of ongoing research to build a system that automatically looks for solutions to the games that, at present, has solved over 60 of them - including ones in languages I don't understand. This implies that situations exist whereby games can be experienced firsthand as a player even when a researcher is not personally adept at the game, and even when no walkthroughs or cheats are available.

Discussion

The most important consideration with any method for choosing games to study is the potential for selection bias, a concern also raised with respect to platform studies (Apperley and Parikka, 2015). There is unquestionably bias towards games and platforms that were part of one's own experience in growing up, through nostalgia or simply greater (technical) familiarity. Pragmatically, game emulators are not always straightforward to get installed and running, and consequently it is easier finding more examples for an already-installed emulator than a completely different platform. Traditional search methods will tend to favor games and platforms that were popular and have more information available about them, and the serendipitous nature of spontaneously composing curiosity-driven questions while playing games implies that the games were interesting and well-known enough to play.

Some of the described methods are more resistant to bias, however. Social media, despite valid criticism of it being an echo chamber, has proven exceptionally useful in expanding the scope of examples I have employed to other regions and platforms. Here I think there is unquestionably an echo chamber, but an echo chamber that can be curated to include a broad selection of people interested generally in retrocomputing and retrogames, as opposed to focusing on a more myopic view. Using a brute-force search is as biased or unbiased as the sample of games comprising the corpus being searched; for platforms like the Atari 2600, the small game size makes it easy to acquire archives containing almost all known games for the platform, allowing game selection bias on that platform to be reduced to near zero.

In the same way, performing retrogame archaeology research at scale can remove game selection bias, apart from the initial bias in selecting the platform (or, in the example above, game-creation system) to begin with. The obvious tradeoff is that substantial effort is involved. Gathering a corpus of all known game images and verifying their functionality takes time;

writing software to analyze them *en masse* takes time and technical expertise. Since a brute-force search is also labor-intensive, it is fair to say that while the effects of selection bias may be mitigated, it comes at considerable cost.

Moving away from the bias issue, it is instructive to understand the limitations of these game selection methods and when they fail. Brute force, for example, does not always succeed. It may be that the trigger condition the emulator is looking for does not happen to occur during the portion of a game that is played; in other words, the condition may indeed exist but is overlooked because a dynamic method of analysis is being used. Or, the condition may not exist at all. I spent time conducting a brute-force search looking for a particular code obfuscation used in copy protection on the Apple II, and never found a single instance of it. Other cases may be too complex to express with a binary condition in the emulator, and as a result a bespoke analysis program would need to be built; I posit the existence of a specific type of self-modifying code on the Texas Instruments TI-99/4A, but it falls into this non-binary category and I have not yet discovered it in a game as a result.

One limitation of magazine-based tips is that they may be wrong. Magazine interviews are subject to the vagaries of retrogame programmers' recollections, article writers' (mis)interpretations, and magazine editing. Claims may be overdramatized. In one recent instance, the VIC-20 game *Moons of Jupiter* was lauded as a 'game that used ingenious hardware tricks to expand the size of the screen' (Drury, 2016, p. 53). Upon acquiring the game, analyzing the code, and duplicating the 'ingenious hardware tricks' with test programs I wrote, I found that the most interesting hardware use is not in the game proper, but in the gradual reveal onscreen of the game instructions prior to playing, and was unrelated to screen size expansion.

Finally, I mentioned the need to verify technical claims made by retrogame programmers about their games. I want to emphasize *technical* in "technical claims" because what programmers are a primary source for, even now, are behind-the-scenes stories and knowledge about their development practices; this is information that is not captured in the game. For instance, Chris Cannon worked at Software Projects, the company behind *Jet Set Willy*. He told me (Cannon, 2016) that 'the Software Projects Z80 team hardly ever programmed the ZX Spectrum directly. All of the code was written on TRS-80 Model IIIs and ported to the Spectrum via an interface that Matthew [Smith] had designed. [...] An entire 48K of machine code could be uploaded to the host Spectrum in a matter of seconds rather than possibly several tens of minutes using the traditional way.' Chris also volunteered that 'when Matthew was writing Manic Miner, he and I

used to talk for hours on the phone brainstorming ideas for the game's baddies. When Matthew wanted to show me what graphics he had created I had the idea to connect our Spectrums directly to the phones. We simply sliced the ends off the tape leads and spliced them directly onto the phone microphone and earpiece wires.' In essence, Chris is saying that they sent the cassette port's audio signal across the phone lines as a makeshift, one-way modem. In neither of these two cases would we know about these practices from the game.[5]

One way to understand the scope in which the methodology described here is applicable is to delineate how retrogame archaeology differs from other fields. The case for retrogame archaeology as a form of archaeology, via archaeogaming, has already been made (Aycock and Reinhard, 2017). Intuitively, retrogames are both physical and digital artifacts resulting from human activity, and as a result can be seen to reside firmly in archaeology's wheelhouse.

However, the code implementing a retrogame can also be viewed as a text, one underlying and making possible the game-as-text perspective of game studies (Fernández-Vara, 2015). Retrogame archaeology thus can also fit comfortably within document-centric history, its methodology construed as historiography, if the definition of historiography as "methods used to study history" is employed (Cheng, 2012; Hoefferle, 2011). Halsall does draw a line between history and archaeology with a working definition of the latter that excludes written records (Halsall, 1997), but this distinction - while clear-cut - seems unnecessarily artificial and not worth adopting here. Perhaps the best characterization of retrogame archaeology with respect to history can be drawn from an article by Haigh (2015) talking about the state of the history of computing. He argues that 'the technical history of computer science is greatly understudied' (p. 43), and that 'historical work forming the backbone of a scholarly career and intended as a contribution to computer science' (p. 43) is 'Almost impossible to accomplish' (p. 43). In this regard, with its technical, computer science, historical focus, retrogame archaeology in the academy seeks to accomplish the impossible.

One could go so far as to say that humanities fields incorporating technical ideas from computer science in their work commit not cultural, but technical appropriation, especially when the treatment of technical ideas evidences an imperfect understanding of them. I do not take such an extreme view, and indeed it is instructive to examine how retrogame archaeology differs from

[5] Even though it would not be reflected in the published game, some stories may be at least partially verified through other means: for example, an experiment could be conducted to transmit data between two computers in this fashion.

some closely-overlapping areas: platform studies and media archaeology. Of the areas mentioned in the Introduction, examining this pair is sufficient. Others that are farther-flung in terms of technical detail than platform studies (critical code studies, software studies) already are separated from retrogame archaeology on that basis; for its part, media archaeology makes claim to media forensics by taking Kirschenbaum's work into its purview, a point made by Apperley and Parikka (2015) with reference to Parikka (2012).

To contrast retrogame archaeology with platform studies, I follow the method Apperley and Parikka (2015) use as a starting point in their critique of platform studies, namely a close examination of the three principles laid out in the platform studies books' series foreword (quoted here from Montfort and Bogost, 2009, pp. vii-viii). First, platform studies has 'a focus on a single platform or a closely related family of platforms.' The unit of study in retrogame archaeology is the game, not the platform, although it is often the case that a good technical understanding of a platform is necessary for analysis. However, as Aycock (2016) shows by example, the games whose implementation is studied may be drawn from a wide range of platforms; retrogame archaeology has narrowed its scope not by platform, but by its focus on retrogames. It is also not the case that a platform in platform studies need be associated with games at all, as the existence of Patterson (2015) attests. Second is platform studies' 'discussion of how computing platforms exist in a context of culture and society,' a perspective that retrogame archaeology eschews in favor of situating retrogame implementation techniques in a modern technical context, answering the question 'where are these old ideas useful or in use today?' (Aycock, 2016, p. 206). Third, platform studies advocates 'technical rigor and in-depth investigation of how computing technologies work,' a value shared with retrogame archaeology.

Distinguishing retrogame archaeology from media archaeology is more difficult because media archaeology itself has purposely resisted clear boundaries. Media archaeology is not archaeology, to begin with, according to Huhtamo and Parikka (2011, p. 3): 'Media archaeology should not be confused with archaeology as a discipline.' They go on to say (p. 3) that it 'move[s] fluidly between disciplines, [...] allowing it to roam across the landscape of the humanities and social sciences and occasionally to leap into the arts.' Computer science is conspicuously absent. Elsaesser distills the criticism of media archaeology by saying that it has 'no discernable methodology and no common objective' (2016, p. 182). There is already much to separate retrogame archaeology from media archaeology in the latter's choice of what it is

not (archaeology) and where its explorations may go (not computer science). The methodology of retrogame archaeology is being explicated, as an additional contrast, as shown by this paper along with Aycock and Reinhard (2017). Where the two do find common ground is in the willingness to study the unsuccessful. Media archaeology does this purposefully (Huhtamo and Parikka, 2011), whereas retrogame archaeology is simply indifferent to a game's success, as mentioned earlier.

Another, better, way to understand the scope in which the methodology described here is applicable is to look at the methods themselves. To paint these methods with the label "retrogame archaeology" is to miss the point. Insofar as a researcher is interested in finding interesting implementation aspects of software - not just games, not just retrogame archaeology - then these methods are applicable. The only limitation is that some methods for selecting an object of study require the ability to emulate a platform, which provides a useful segue to one lingering question: what is "retro?" There are a number of problems that arise when trying to precisely define the term (Aycock, 2016), but from the point of view of determining when retrogame archaeology methods are applicable, it doesn't matter. Considering the methods described here along with the analysis methods in Aycock and Reinhard (2017), emulatability is the key factor. Our current capacity to emulate neatly subsumes anything that might reasonably be considered "retro," and this implies that the applicability of retrogame archaeology methods can only grow over time.

Conclusion

There are many ways to find treasures inside old games' implementation. However, the first problem is knowing in which games to look, making the methodology for choosing a game to study as important as the methodology for studying a chosen game. This experience report has drawn on my years performing retrogame archaeology to distill the process into five categories. In terms of lead generation, it would not be fair to conclude that some are superior to others; they all have potential for meaningfully identifying games to study, and are perhaps best thought of as different tools that can be employed to address the selection problem. Some methods do have more potential bias than others, which is important to be cognisant of when performing research. And, with appropriate software infrastructure, retrogame archaeology can scale, where a broad range of games can be selected and analyzed *en masse*. It is important to

note that these methods are generally applicable to any work that seeks to find interesting curios inside the implementation of software.

What of retrogame archaeology's future? More examples of retrogame archaeology being done are needed, both to populate the field as well as to evolve its methodology. As well, extant tools are not nearly as useful as they could be for retrogame analysis tasks; we need better software to analyze software.

Acknowledgments

My work is supported in part by a grant from the Natural Sciences and Engineering Research Council of Canada. Thanks to Daniel de Castro and Claudia Maurer for verifying the nature of the Portuguese and German magazines, respectively.

References

ALTICE N. (2015), *I Am Error: The Nintendo Family Computer*, Cambridge, MIT Press.

APPERLEY T., and PARIKKA J. (2015), "Platform studies' epistemic threshold," *Games and Culture*, <https://doi.org/10.1177/1555412015616509>.

AYCOCK J. (2016), *Retrogame Archeology: Exploring Old Computer Games*, New York, Springer.

AYCOCK J., and REINHARD A. (2017), "Copy protection in *Jet Set Willy*: Developing methodology for retrogame archaeology," *Internet Archaeology*, vol. 45, <https://doi.org/10.11141/ia.45.2>.

BEVAN M. (2014), "Ultimate guide: Pac-Land," *Retro Gamer*, no. 127, pp. 68-73.

CANNON C. (2016, 31 May), email communication.

CHENG E.K.-M. (2012), *Historiography: An Introductory Guide*, London, Continuum.

CRANE D. (2011), "Classic game postmortem: Pitfall!," Game Developer's Conference, <http://www.gdcvault.com/play/1014632/Classic-Game-Postmortem-PITFALL>.

CRANE D. (2013, 2 August), email communication.

DOBSON D. (2009, 29 September), "Adventure of the week: Spook House (1982)", *Gaming After 40* blog, <http://gamingafter40.blogspot.ca/2009/09/adventure-of-week-spook-house-1982.html>,accessed 8 January 2017.

DRURY P. (2016), "The making of: Myriad," *Retro Gamer*, no. 162, pp. 52-53.

EGENFELDT-NIELSEN S., SMITH J.H., and TOSCA S. P. (2015), *Understanding Video Games: The Essential Introduction*, 3rd edition, New York, Routledge.

ELSAESSER T. (2016), "Media archaeology as symptom," *New Review of Film and Television*

Studies, vol. 14, no. 2, pp. 181-215.

FERNÁNDEZ-VARA C. (2015), *Introduction to Game Analysis*, New York, Routledge.

FERRIE P. (2016), "A brief description of some popular copy-protection techniques on the Apple][platform," *PoC||GTFO*, vol. 0x10, pp. 39-74.

FULLER M., ed. (2008), *Software Studies: A Lexicon*, Cambridge, MIT Press.

GRAPHIC ADVENTURE CREATOR (2017), Wikipedia, <https://en.wikipedia.org/w/index.php?title=Graphic_Adventure_Creator&oldid=719391655>.

HAIGH T. (2015), "The Tears of Donald Knuth," *Communications of the ACM*, vol. 58, no. 1, pp. 40-44.

HALSALL G. (1997), "Archaeology and historiography," in M. Bentley (ed.), *Companion to Historiography*, New York, Routledge, pp. 805-827.

HOEFFERLE C. (2011), *The Essential Historiography Reader*, Boston, Prentice Hall.

HUHTAMO E. and PARIKKA J. (2011), "Introduction: An archaeology of media archaeology," in E. Huhmato and J. Parikka (eds.), *Media Archaeology: Approaches, Applications, and Implications*, Berkeley, University of California Press.

INCENTIVE SOFTWARE LTD. (1985), *The GAC Adventure Writers [sic] Handbook*. Manual.

INCENTIVE SOFTWARE LTD. (1986), *The Graphic Adventure Creator (Commodore 64)*. Manual.

KIRSCHENBAUM M.G. (2008), *Mechanisms: New Media and the Forensic Imagination*, Cambridge, MIT Press.

MILNE R. (2014), "Jack the Nipper," Retro Gamer, no. 127, pp. 44-47.

MONTFORT N., BAUDOIN P., BELL J., BOGOST I., DOUGLASS J., MARINO, M.C., MATEAS M., REAS C., SAMPLE M., VAWTER N. (2013), *10 PRINT CHR$(205.5+RND(1)); : GOTO 10*, Cambridge, MIT Press.

MONTFORT N., and BOGOST I. (2009), *Racing the Beam: The Atari Video Computer System*, Cambridge, MIT Press.

PARIKKA J. (2012), *What is Media Archaeology?*, Cambridge, Polity Press.

PATTERSON Z. (2015), *Peripheral Vision: Bell Labs, the S-C 4020, and the Origins of Computer Art*, Cambridge, MIT Press.

REINHARD A. (2018), *Archaeogaming: An Introduction to Archaeology In and Of Video Games*, New York, Berghahn Books.

VOGT S. (2015, 22 August), tweet, <https://twitter.com/8bit_era/status/635141339563335680>.

WIEST L. (2016), "Reverse engineering Star Raiders," *PoC||GTFO*, vol. 0x13, pp. 5-20.