RESEARCH ARTICLE

Inspecting the Foundation of *Mystery House*

John Aycock
University of Calgary, Canada
aycock@ucalgary.ca

Katie Biittner
MacEwan University, Canada
biittnerk@macewan.ca

Abstract

Computer games are recent artifacts that have had, and continue to have, enormous cultural impact. In this interdisciplinary collaboration between computer science and archaeology, we closely examine one such artifact: the 1980 Apple II game *Mystery House*, the first graphical adventure. We focus on implementation rather than gameplay, treating the game as a digital artifact. What can we learn about the game and its development process through reverse engineering and analysis of the code, data, and game image? Our exploration includes a technical critique of the code, examining the heretofore uncritical legacy of Ken Williams as a programmer. As game development is a human activity, we place it in a theoretical framework from archaeology, to show how a field used to analyze physical artifacts might adapt to shed new light on digital games.

Introduction

In 1980, Roberta Williams and her husband Ken Williams created and released the adventure game *Mystery House* for the Apple II, a successful early personal computer. It was the first product of their new company, On-Line Systems – later to become Sierra On-Line – and reportedly sold 3000 copies in six months (*International Directory of Company Histories* 2001), and over 10,000 copies in total (Sierra Help Pages n.d.). Roberta was the game designer and illustrator, and Ken the programmer who implemented it. It

Keywords: archaeogaming; binary reverse engineering; *chaîne opératoire*; computer game; Ken Williams; Mystery House; On-Line Systems; Sierra On-Line

Journal of Contemporary Archaeology 6.2 (2019) 183–205
ISSN (print) 2051-3429 (online) 2051-3437 https://doi.org/10.1558/jca.36745

equinoxonline

FIGURE 1: ***Mystery House*'s opening screen.**

was at a time of transition away from games in arcades to computer games as products for homes and for mass consumption (Newman 2017).

What of the game? Montfort (2003, 169) described *Mystery House* as "a minimal, bizarre treasure hunt with a two-word parser so primitive it made *Adventure* seem as intelligent as HAL from *2001*". Loguidice and Barton (2009, 144) are equally as flattering, referring to the game's "crudely drawn monochrome graphics, poorly edited script, and one of the worst text parsers in the business" but conceding that for the time it was "tremendously progressive".

Mystery House was an entry in the text adventure canon, a type of game also referred to as interactive fiction. In text adventures, the player is given a textual description of their environment and events occurring within the game, and they issue commands by typing simple sentences in English.[1] Part of the experience of playing a text adventure, for better or worse, is guessing the vocabulary of words the game knows and expressing commands such that the game's parser understands them. One reason *Mystery House* is singled out as noteworthy is that the game was the first text adventure[2] to augment its prose with images, rough line graphics whose inclusion arguably makes it the first graphical adventure game. Figure 1 shows the start of the game; note the color aberrations are an artifact of Apple II graphics (Sather 1983) and aren't unique to *Mystery House*.

There is a mythos surrounding *Mystery House* and its development. Besides being the first graphical adventure, it was the first entry in Roberta Williams's *oeuvre*, the humble

1. Or some other human language.
2. At least the first one known as of this writing.

beginnings of someone who would become a renowned game designer. As the debut product of On-Line Systems, *Mystery House* is the start of a story that embodies the romanticized entrepreneurial dream: starting a home-based company on a shoestring, growing into an industry powerhouse, and retiring early.

As a foundational game in the computer game industry, *Mystery House* has more to offer as a case study besides its gameplay and aesthetics. The contributions of this paper, an interdisciplinary collaboration between a computer scientist and an archaeologist, are fourfold:

1. We provide an informed technical critique of game code as opposed to considering gameplay. As gameplay is determined by the game code, either directly or indirectly (i.e., via emergent behavior), code is clearly an important aspect of a game to study.
2. Our work acts as an example of what can be learned from a digital artifact, taking an archaeological point of view. Oral histories have limitations: there are already game developers whom we can no longer interview, and in any case memory of technical details is imprecise after so many years have elapsed. We need to develop methodology in preparation for the inevitable future where digital artifacts are all that remain of some human activity.
3. We place the process of game development in a theoretical framework used in archaeology, seeing how it must be adapted to handle the digital instead of the physical. This is a first step to drawing on archaeology as a way to understand the human activity of game development.
4. Through *Mystery House*, we critically examine the legacy of Ken Williams as a programmer. Characterizations of his skill over the last 30+ years have been largely praising and uncritical; we look at whether or not this view is warranted based on how the digital artifact represents his technological choices.

From an archaeological point of view, both the study of a video game and the collaborative approach we are undertaking here fall within the subdiscipline of archaeogaming. Archaeogaming is a portmanteau of *archaeology* and *gaming*, representing the intersection of gaming with archaeology (Reinhard 2013, 2015a). Dennis (2016) more specifically defines archaeogaming as "the utilization and treatment of immaterial space to study created culture, specifically through videogames [and requiring] treating a game world [as] bounded and defined by the limitations of its hardware, software and coding choices." As such, archaeogaming includes the real-world archaeology of video game hardware and software (Perry and Morgan 2015; Reinhard 2015c), analyses of videogame or virtual worlds as archaeological sites (Reinhard 2016b), the archaeology of particular game titles (e.g., K. M. 2018), the examination of the material culture associated with gaming, including both real-world and virtual artifacts, cosplay, and museums themselves (Reinhard 2016a), the ethnography of virtual worlds (Boellstorff *et al.* 2012), the participation of archaeologists in game design and implementation and the creation of games by archaeologists (Copplestone 2017), and the use of games in science communication, outreach, museums, education, and public archaeology programs (González-Tennant 2016; Watrall 2002).

Accumulating methodologies from both within (e.g., lithic analysis) and without archaeology (e.g., software version analysis), archaeogaming is a reasonable expansion of the discipline of archaeology when one considers what archaeology can be in practice and in application – our contemporary technologies challenge what it is archaeologists can and should study. While archaeogaming is currently a niche subdiscipline within archaeology, largely motivated by archaeologists who game (Mol *et al.* 2016), it has broader applications and should be considered distinct from media archaeology (see Huhtamo and Parikka 2011); this is because archaeogaming explicitly incorporates and challenges archaeological methods and theories (Reinhard 2015b). The particular approach to archaeogaming that is applied in the current analysis of *Mystery House* as an artifact is that of a preliminary *chaîne opératoire* analysis.

We begin with background on our computer science- and archaeology-based approach. We then take a look at some features of *Mystery House*'s implementation, and then critique the game's implementation, asking what a realistic assessment of the code is while contextualizing this work from an archaeological perspective. This is followed by a summary conclusion.

Background

This interdisciplinary research has related work in multiple areas: computer science, archaeogaming, and archaeology generally through the *chaîne opératoire*.

Computer Science: Games as Code

Mystery House routinely receives mention in computer game histories (e.g., Loguidice and Barton 2009; Donovan 2010) by virtue of its status as a "first". Only a few dig deeper: Kirschenbaum (2008), for instance, examines the contents of the *Mystery House* disk image. Technically, this is mostly a walkthrough of the filesystem structures described in *Beneath Apple DOS* (Worth and Lechner 1981); the only look at the internals of *Mystery House* is a cursory view of the MESSAGES file.

There is also a tradition of game postmortems in the game industry – the GDC Game Developers Conference, for instance, recently featured one on the classic game *Oregon Trail* (Rawitsch 2017). While useful and fascinating, they exist for only a handful of games, and are not unbiased sources. Similar presentations occur in the (very) occasional History of Programming Languages conferences (ACM Digital Library n.d.), although in general "the technical history of computer science is greatly understudied" (Haigh 2015, 43) – an observation that applies equally to the technical history of computer games.

Analysis and critique of code, in the form of peer code reviews, is commonplace to try and improve code quality (Wiegers 2002). There is also a tradition of code critique in computer security, specifically directed towards malicious software; Spafford's (1989) analysis of the Internet Worm is a notable early example. There, as in our examination of historical code, the code author is not a party to the analysis.

Casting the net more widely, publications within the area of platform studies can incorporate technical material (e.g., Altice 2015). Areas like software studies (Fuller 2008) and critical code studies (Marino 2006) engage code to varying degrees, although they side much more on the humanities.

Archaeogaming: Games as Digital Artifacts

Archaeology, through archaeogaming, has examined the technical aspects of computer games (Aycock 2016) as well as the methodology for undertaking such work (Aycock and Reinhard 2017). As an example of this, Aycock (2016) previously examined *Mystery House*'s graphics format, which was situated in terms of other graphics encodings. In contrast, the current study, by examining the game as a digital artifact, makes it possible to examine the format in the context of *Mystery House* itself and the game's development.

One of the challenges of archaeogaming is reconsidering and redefining commonly used and understood terminology. As such, some effort must be made to provide explicit descriptions and definitions for the terms we are adopting for use in our study of the game, our digital *artifact*. The most inclusive definition of an artifact from an archaeological perspective is any object that is made, used, or modified by humans. Following this definition, video games are artifacts and can thus be approached as material artifacts, as software or digital artifacts, and/or as cultural artifacts (Sotamaa 2014). This treatment of video games as artifacts is not new to those in the digital humanities or game studies. Greenfield (1994, 4) identifies video games as cultural artifacts – specifically, cognitive artifacts in which symbolic systems are encoded and in which players are socialized and trained to interact with technology. Styhre *et al.* (2018) acknowledge that this position of labeling video games as cultural artifacts is part of a process of legitimization, situating video games as more than entertainment and as digital objects with wide significance. While this appropriation of the term "artifact" from archaeology by those in the digital humanities or in game studies is not incorrect, it does result in games being analyzed without consideration of the broader methodological and theoretical practices of archaeology, such as considering how archaeologists analyze and interpret artifacts.

All archaeological analyses and interpretations of artifacts hinge on context. Following a classic textbook definition such as that of Renfrew and Bahn (2015), this means the matrix (the material surrounding the object), its provenience (its horizontal and vertical position within the matrix), and its association (occurrence with other archaeological remains). As archaeologists engage with artifacts once they've left their context of use and entered the archaeological context/record, discard studies have been incorporated into some of the work coming out of digital humanities. For example, Bailey (2015, 48–49), in studying various media objects such TVs and VHS recorders, argues that it is the context of "rejection and abandonment" that reveals the "changing relationships of people to media". However, these kinds of definitions for and application of context become challenging when considering digital artifacts from an archaeogaming perspective. Where are games located? When is the "when" of the game? When did it leave its context of use (if at all) and enter the archaeological record? How is a game "rejected and abandoned"?

There are several ways one can reimagine context for digital artifacts. First, one can treat the game not as an artifact but as an archaeological site. Reinhard (2017) uses this along with the concept of the Harris Matrix to visualize software versions, periodization, and phasing. This approach structures the code as artifact within a game; the game-as-site then serves as the context and is subject to changes and modifications much like

the archaeological record proper. Second, we could reconsider what is meant by "the matrix" of the artifact. For "dirt" archaeologists, the recorded field notes, photographs, maps, bags, tags, and labels about and resulting from an excavation all become part of the archaeological record, along with the artifacts recovered (Wexler *et al.* 2015). If we were to use this approach for digital artifacts, we could consider all the surrounding materials about the game created by its designers, reviewers, publishers, and players, including walkthroughs, print ads, posters, letters to the editors, etc. as the matrix. Or, and finally, one can situate the game within a narrative of implementation, examining the artifact using technological organization as its context – the approach taken here.

Digital Artifacts and the Chaîne Opératoire

In this paper, technological organization is addressed via *chaîne opératoire*. This "technological approach […] seeks to reconstruct the organization of a technological system at a given archaeological site" (Sellet 1993, 106). Used for the study of lithic artifacts (e.g. Bar-Yosef and Van Peer 2009) and ceramic manufacture and production (e.g. Roux 2016), the *chaîne opératoire* successfully addresses two fundamental kinds of research questions: first, those which identify the sequential technical operations by which natural resources were transformed into culturally meaningful and functional objects (i.e., lithic raw material into tools); and second, once these sequences are identified, those which infer something of abstract cognitive processes and underlying normative logic systems structuring those acts (i.e., the technological choices made by the person selecting the raw material, and shaping, using, and discarding it) (Dobres 2000).

As a conceptual model, the *chaîne opératoire* is a dynamic act of material and social transformation (Dobres 2000); it deals explicitly with people who were engaged in a decision-making process. Building on the theoretical foundations established by Lemonnier (1992), Sillar and Tite (2000, 2) frame these decisions as "technological choices", and they define five areas of analysis within a technology where choices exist (Sillar and Tite 2000, 4): (1) raw materials; (2) tools used to shape the raw materials; (3) energy sources used to transform the raw materials and power the tools; (4) techniques used to orchestrate the raw materials, tools, and energy to achieve a particular goal; and (5) the sequence, or *chaîne opératoire*, in which these acts are united to transform raw materials into consumable products. The choices that are made are influenced by properties and by performance characteristics (Sillar and Tite 2000, 4). Both aspects can be readily identified or inferred by archaeologists for artifacts from a real-world archaeological site, and they expand greatly upon a use-lives, life-cycles, and life-history approach to artifact analysis (see Tringham and Ashley 2015 as an example of a use-life approach in media archaeology), as well as upon a history of technology approach, tracking change and transformation of and in media technologies. This then is technological analysis involving three levels: the object itself, the series of technological sequences that led to the production of the object, and the specific technical knowledge that is shared by all involved in its production and use (Sellet 1993). Note that this application of *chaîne opératoire* adapts Sotamaa's (2014, 8) argument that the explicit use of artifact as a term for video games "helps us to conceive of the forms of technological agency invested in video games *and* their material manifestations" (emphasis added).

For the purposes of video games specifically, the value of an operational chain model lies in its being able to define the technological tradition of a video game in relation to raw material procurement while also providing a method for examining the context for the development of the game and the decisions, technological choices, constraints, and influences of its designers. As for technological choices, we are defining the raw material as the computer with the programming to do anything through use of code.[3] As will be subsequently presented, the properties and performance characteristics of the Apple II and its core programming shaped the choices made by Ken regarding *Mystery House* implementation. The assembler (LISA, discussed below) and the language (possibly domain-specific) are the tools that were used to shape the raw material, while the programming used to create the game (including BASIC) are the techniques that were used to orchestrate the raw material. The energy sources that were used are the intellectual energy, the time, and the effort of the Williamses. This sequence, wherein each technological choice is co-dependent on the other technological choices, is the *chaîne opératoire* of *Mystery House* – i.e., a particular game, our digital artifact, is the end product of the use of a computer, code, intellectual power, and programming tools and techniques.

Mystery House *Implementation*

A *Mystery House* disk image contains a number of files that are used by the game. The uses of some are immediately apparent: the MESSAGES file, for instance, contains lines of text representing (most of) the text messages the game displays. Understanding the contents of other files requires more effort. We reverse engineered[4] the binary code and data in *Mystery House*'s files to understand its implementation, and as a case study to see what a digital artifact can tell us.

Graphics

The graphics images are encoded and stored in files BLOCK1 through BLOCK19; each file contains several images.[5] Image data consist of a sequence of byte pairs representing absolute (*x, y*) coordinates, terminated by the value (255, 255). As a side effect of this design, images' coordinates could not specify the full 280-pixel width of the Apple's high-resolution screen, but could use the entire 160-pixel height. Each coordinate pair gave the endpoint of a line; the line's starting point was the last endpoint drawn, thus optimizing for the case of continuous line drawing. The coordinates (0,0) temporarily "lifted the pen" to permit motion without drawing a line. Figure 2 illustrates the image format by showing the incremental drawing of a window image from data in BLOCK15.

3. While for this article we have focused on the implementation, it is possible to perform this kind of application of the *chaîne opératoire* wherein the computer is defined not as raw material but itself containing raw materials. The consideration of computer as a tangible artifact has been undertaken in media archaeologies (see articles in *Journal of Contemporary Archaeology* 2 [1]) but is out of the scope of this paper.
4. Although it is beyond the scope of this paper, Moshenska (2016) explores the relationship between reverse engineering and archaeology in depth.
5. The image format was reverse engineered directly from the BLOCK files and validated using a Python script we wrote to reconstruct the images from the file data. For reproduction and enhancement of this work, our scripts are available at https://github.com/aycock/mh.

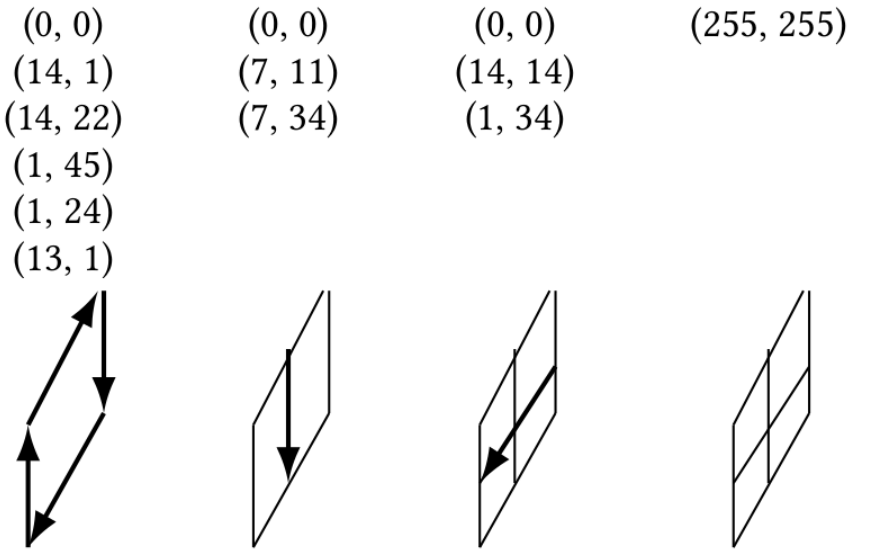

FIGURE 2: Graphics drawing from compressed image data format (origin at top left).

Interpreter

Mystery House required a variety of information to implement the game.[6] For instance, there was a list of verbs, and a list of nouns, each of which could have zero or more synonyms. The most interesting part, however, is the game logic. There were two sets of logic rules in the game, both of which were used for each player input: the first ruleset stopped being processed after a successful rule match, and all but one room-specific rule was here; the second ruleset was always processed completely. A rule was effectively a conditional statement; if the conditional test was true, then any instructions associated with the rule would be executed by the interpreter. The interpreter supported 27 different types of instruction.

Figure 3 contains an excerpt of the game logic used early in the game: room #2 is the porch. It is important to note that this form is how our script prints it. The actual representation of the code in the game is encoded as unreadable bytes. Below we discuss what the developer view might have been.

The design of the interpreted code's encoding reveals the importance placed on particular elements. One design would have made the encoding of the conditionals in rules consistent regardless of the type of conditional test specified, but that is not what we see in *Mystery House*. Instead, the room, verb, and noun are first-class entities, always represented in every encoded rule whether needed or not – a default value of 254 in the encoding acts as a wildcard value. Any of the additional five types of conditionals

6. Information in this section was obtained by reverse engineering the binary code. We validated our analysis in part by comparing afterwards with the ScummVM source code that emulates *Mystery House*, and wrote a Python script that dumps the game data: nouns, verbs, messages, objects, rooms, and the interpreted game logic code.

```
if ROOM = 2 and VERB = "OPEN" and NOUN = "DOOR" and VAR 12 = 1 then
        print "IT IS OPEN"
if ROOM = 2 and VERB = "OPEN" and NOUN = "DOOR" then
        set room images to 2
        VAR 12 = 1
if ROOM = 2 and VERB = "CLOSE" and NOUN = "DOOR" and VAR 12 = 1 then
        set room images to 1
        VAR 12 = 0
if ROOM = 2 and VERB = "CLOSE" and NOUN = "DOOR" and VAR 12 = 0 then
        print "THE DOOR IS CLOSED"
if ROOM = 2 and VERB = "GO" and NOUN = "DOOR" and VAR 12 = 1 then
        goto room 3
        VAR 1 = 1
        print "THE DOOR HAS BEEN CLOSED AND LOCKED"
if ROOM = 2 and VERB = "GO" and NOUN = "DOOR" then
        print "THE DOOR IS CLOSED"
if ROOM = 2 and VERB = "GO" and NOUN = "STAIRS" then
        goto room 1
```

FIGURE 3: Excerpt from interpreted game logic.

are encoded differently, and in a consistent and extensible manner: indeed, it would not have been a problem at all to encode room, noun, and verb tests the same way.

The conditional tests are shown explicitly ANDed together in Figure 3 for readability, but in fact that was implicit in the game's rule interpretation; the only way to express a logical OR was by adding a separate rule. While variables were used to store state, as shown in the excerpt, the current picture number could be and was also used for that purpose, suggesting that whoever wrote the logic rules was thinking of the game visually.

Source Code

Documentary evidence tells us that *Mystery House* was written in assembly language (Levy 1984), but in fact the game image itself tells us that and much more about the development process.

It was not uncommon for shipped games at that time to accidentally capture some non-game memory contents in the game image. Here, we discovered partial fragments of assembly language source code for *Mystery House* embedded, hence confirmation that assembly was used – what was visible was sufficient to start matching with pieces of the disassembled binary code. However, the fragments were not fully readable as assembly; we realized that they were in an encoded (tokenized) form, and we were able to reverse engineer the format, creating a Python script to find and fully decode all the embedded assembly code fragments. Furthermore, due to the unique signature the encoded fragments provided, we could identify the tool being used, meaning the assembler applied to translate the assembly source code into binary code: LISA.[7]

7. This assembler should not be confused with the later Apple Lisa computer. We verified our analysis by comparing with an independently created LISA file decoder as well as in-emulator experiments with LISA.

Given that LISA was later published by On-Line Systems (Hyde 1981), the use of LISA is consistent with the historical record, but we cannot know if LISA was originally used or whether the code was switched to LISA early on.

Although we look at this in much more detail in a subsequent section, it is worth observing that the assembly code expresses information that would otherwise be unknown from the binary image alone. Human-readable comments, for example, are lost during the assembly process, as are the names of variables. The latter can reveal how the programmer conceived of data in the game. As an example, the interpreter has instructions to add and subtract constant numeric values to/from a variable, and yet the recovered source code refers to a variable as a "SWITCH", implying that they were originally conceived only as Boolean flags.

Even before discovering the assembly code fragments, we saw indications of how the code was structured just from the binary code analysis. Specifically, there is evidence suggesting that the game code was spread over more than one file.

A set of nine routines exist in the *Mystery House* code that are very generic: copying bytes, comparing bytes, 16-bit arithmetic, and so on. What is notable is that these routines are accessed via a jump table, each entry of which is simply an instruction that jumps immediately to the corresponding routine. If that was in the same assembly code file as the rest of the game, there would be no need for this table; the assembler could figure out the start locations of the routines. Instead what this tells us is that that code was in a separate assembly file, and the jump table (which resided at a known memory location) was used to call the routines in a robust way. It also tells us more about the development tools – a linker would have been able to connect addresses in two separately assembled files without needing a jump table, thus we can conclude that a linker was not used.

Given that programmers are wont to reuse their code, it is definitely possible that this generic "utility" code might have been written prior to *Mystery House* and used by Ken to save development time. As if to underscore this point about code reuse, we found the nearly identical code in *Wizard and the Princess*, the follow-on to *Mystery House*.

Other Versions

While most of our work used the version of *Mystery House* that was released into the public domain in 1987, we were later able to locate additional versions that provide new information to our analysis.

The Original

Original copies of *Mystery House* are both rare and expensive, and it was sheer luck that an original disk image became available. Primarily the original allows us to confirm what was or was not present on the disk prior to the public domain release. It would otherwise be impossible to discern what had been added at some later date.

In particular, we were interested in PIC SIZES. This is a separate BASIC program present on the public domain disk and, we discovered, also on the original disk. This is not part of the game: it is a small program that loads each BLOCK file containing in-game pictures one at a time, printing the size of the file. It is a program that would be

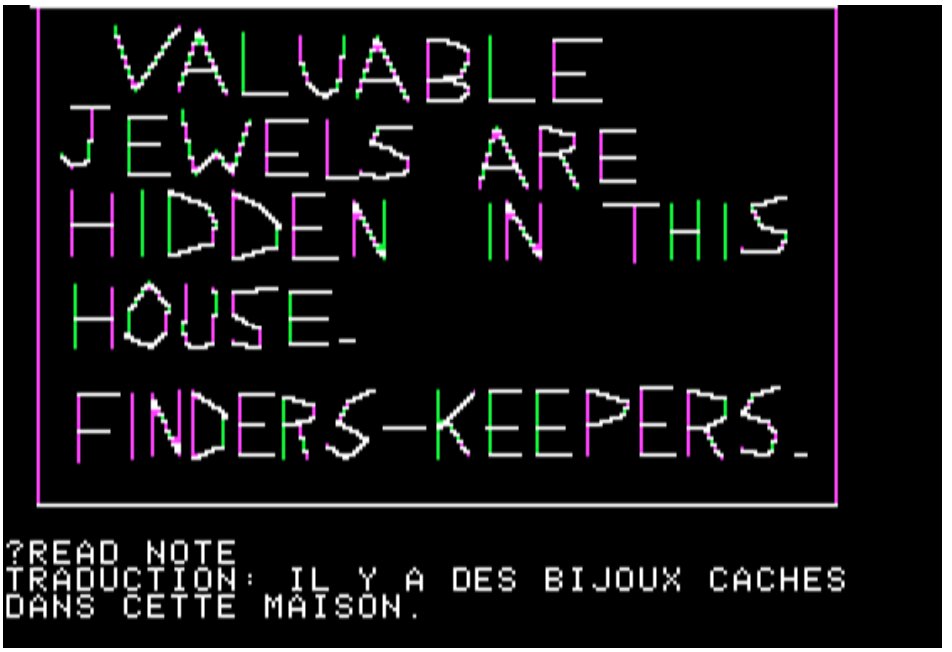

FIGURE 4: *Mystery House* in French, sort of.

used during development, likely to verify that picture sizes would not occupy too much memory space. Seemingly quickly written, it deliberately terminates via an error. This captures part of the game development process as well as confirms that BASIC "helper" programs would be used during development of a game written in assembly code.

The French Connection

A French version of *Mystery House* was mentioned in one of its earliest advertisements (On-Line Systems 1980a). We acquired a disk image of it, whose splash screen credits translation to Tom Nalevanko of Malibu Microcomputing.

The text in the MESSAGES file was translated into French, a fairly easy task. None of the graphics were translated: where the graphics conveyed important textual information, this was given to the player in plain text – this led to strange juxtapositions of French and English onscreen, as shown in Figure 4. This screenshot also reveals that the French version understood some English commands. Running our Python script that dumps the game information, we see that for both nouns and verbs, the primary form in English was retained, but French synonyms were supplied. The English NOTE is still recognized, for example, but LETTRE or PAPIER would also work. Interestingly, the complete original English verb table is still present in its entirety, albeit unused in the French version.

Different assembly source code fragments were captured on this disk image. Our script reconstructed fragments such as shown in Figure 5. This means that the translator was not working from the binary alone; Nalevanko must have had access to *Mystery House*'s source code to do his work.

```
.asc "TELLE DIRECTION NE FRAPPEZ PAS 'ALLER"
.hex 8D00
jsr PRINT
.asc "EST'; FRAPPEZ 'EST' OU SIMPLEMENT 'E','O',"
.hex 8D
.asc "'N','S','H','B'."
```

FIGURE 5: Assembly code, *en français.*

On Language

Nooney (2017, 85) reports on an interview with Ken Williams, where Ken describes a paper-based process he devised for Roberta to create the game logic: "A column for verbs, and a column for nouns, and then a column for what happens." This means we know both how the game logic was created, and from our work above, how it was stored and interpreted. What we *don't* know is what happened in between, and how Roberta's on-paper recording became the in-game encoded logic.

There are several possibilities. We have already seen how BASIC was used for a development program for *Mystery House*, and it would have been relatively quick for Ken to create a BASIC program that he could enter the game vocabulary and logic into, and that would produce the necessary encoded version as output. We know that Scott Adams took this approach for his text games, as well as what the menu-driven program looked like (Aycock 2016). Another option would have been for Ken to encode the data by hand in the assembly source code – theoretically possible but tedious and error-prone; a contemporaneous example of this is seen in the assembly source code for the 1981 Apple II text adventure *Adventure in Time* (Savetz 2016).

A third, intriguing possibility is that Ken created a domain-specific language (DSL) to describe the game logic. Roberta's hand-written "code" could then have been typed into the computer in this DSL, and Ken would have needed to write a translator from the DSL into the encoded logic. This is not far-fetched: Ken had the necessary skill set to implement a DSL, something we will return to in our discussion. Roberta, for her part, mentioned in an interview that "when Ken first sat down to write the code for Mystery House, he wanted to write a special language that just did graphic adventure games" (Byron 1990, 26).[8] The question is really *when*, not if, this DSL was first created, and whether it was early enough for use in *Mystery House*. Given the prevalence of BASIC at the time, and Sierra's later game language being BASIC-like (Trivette 1985), it is reasonable to hypothesize that the DSL would have resembled BASIC if it did exist.[9]

Is there evidence to support any of these three possibilities? Yes. At the end of the list of verbs and list of nouns in all the game images is the word DONE. The word is not

8. Levy (1984, 201) talks of an "Adventure Description Language" but says it was for the Williamses' second game. Levy also incorrectly conflates language and interpreter, so it is unclear exactly what he is referring to.
9. Or, taking Ken's Fortran background into account, we could also argue that Fortran would have been a language design influence.

used in any of the game logic and, crucially, is not provided with French synonyms in the French translation. Taken together with the meaning of the word "done", this strongly suggests that the word acted as a sentinel value indicating the end of the noun and verb lists. A sentinel like this would not have been necessary if the logic had been hand-coded in assembly, so we infer that there was processing of the nouns and verbs done by a program of some kind. While the nature of the tool – DSL translator or simple BASIC program – is unknown, we can see the marks made by the tool on this digital artifact.

Discussion

While Roberta Williams's role in the creation of *Mystery House* and Sierra's other games has been explored (Byron 1990; Nooney 2013), Ken's has seen much less close scrutiny. One of the aspects of archaeogaming and the examination of digital artifacts is that we can sometimes directly engage in discussion with the developer/coder/designer (a form of ethnoarchaeogaming if you will) or we can access other sources of information that talk about the individual(s) in question. As noted in the introduction, precise memories of technical detail are unreliable after so many years; for this reason, and to develop methodology going forward, we turn now to these "other sources of information".

Ken Williams's Technological Choices

Levy's *Hackers* (1984, 13) lavishes Ken Williams with praise, calling him an "[a]rrogant and brilliant young programmer". Further, he described the encoding of the pictures in *Mystery House* as "a dazzling program bum that characterized Ken's facility for top-level hacking" (Levy 1984, 298).[10] Maher (2011b) continues this narrative when speaking of Ken, saying that "[o]ther than Bill Gates, I don't know of another figure in the early PC world who combined such technical acumen with such an instinct for business." Nooney (2017, 86), who has extensively studied Sierra On-Line, recently called the game "a nimble bit of programming ingenuity". Tommervik led his 1981 profile of On-Line Systems thus: "In the world of the programming cognoscenti, the name of Ken Williams is much honored" (Tommervik 1981, 4); this profile elicited a letter in response: "I've been exposed to Ken's technical talents [...] and he unquestionably ranks with Bill Budge, Bob Bishop, Nasir [Gebelli], and just a very few others as one of the software greats" (Leff 1981, 8). Such gushing prose does not sound particularly objective, nor does any of it apparently originate from the computer science perspective. Of course, archaeologists too not only encounter but can produce glowing reviews of artifacts. Instead of challenging subjective accounts of skill and expertise explicitly, we do think it is necessary to critique them in light of what can be interpreted from an examination of Ken's technological choices at that point in time directly from the digital artifact he had a hand in creating.

First, breathless mentions of Williams using assembly language (Tommervik 1981; Levy 1984) celebrate something wholly uninteresting at the time; many programmers of that era used assembly, often for speed or space reasons.

10. Levy uses "bum" to describe a code optimization. This term has been unused for decades and was described in 2002 as "thoroughly obsolete" (Jargon File n.d.).

Second, we can take "speed" to mean the speed of code development, which various sources claim was around a month (Levy 1984; *International Directory of Company Histories* 2001). To be fair, *Mystery House* was a side project, something that had to compete for time with Ken's other jobs (Levy 1984; Ramsay 2012), but for comparison, let us consider an extreme case of development time spent on a game for a contemporary platform that was much more challenging to program, the Atari 2600. The Atari 2600 had real-time programming requirements and only a fraction of the Apple II's space, yet there are instances of Atari 2600 games like *Chase the Chuck Wagon* (1983) being developed in three days (Schwartz, email communication to author, 21 January, 2014). There is a clear difference between three days and thirty. It seems fair to say that Williams was not a superprogrammer (*Jargon File* 2003) exhibiting unnatural levels of programming productivity. Also, Williams's Apple II, besides having substantially more memory than an Atari 2600, had the BASIC language built-in along with a machine-language monitor for debugging (Espinosa 1979). That, plus a floppy disk drive and DOS, would have made Williams's development relatively painless for the time.

Third, now taking "speed" to mean the speed of *Mystery House*'s execution, it was not an arcade-style game needing high performance and correspondingly skilled coding. Instead, the relaxed time constraints are nicely summarized in an internal development document from Infocom, another producer of text adventure games back then: "The design goal also requires no more than a few seconds response time for a typical move" (Berez *et al.* 1989, 5).

Judging Ken's code quality is not as clear-cut a task. An initial assessment would start at the disassembled code, and there we find several instances of "brk" instructions used that would halt *Mystery House* and leave a bewildered player dumped out of the game and staring at the machine-language monitor prompt – if those instructions were ever executed. Why would instructions be left in the code that shouldn't be executed? In modern programming parlance, the brk instructions act as assertions, where the intent is to have the program fail (during development) if some condition is not met. Here, the brk instructions would be triggered only if an object being searched for wasn't found, likely indicating a bug in the interpreted program logic. These instructions suggest good programming practice, and were used very deliberately by Williams. By chance, the assembly code fragments happen to capture two brk instructions and the comments beside them: "BAD OBJECT" and "UH-OH, KABOOM!"

More generally, the fact that Williams bothered to comment his code at all indicates a certain level of discipline and training, again especially considering that *Mystery House* was a side project for him. However, surviving label names in the assembly code indicate some clear areas for improvement. Many label names in the main part of the code use a BASIC-esque numbering scheme as opposed to using names that convey information: AC3200, AC3300, AC3400, and so on. This allowed him to use the numbers in between for labels internal to a specific routine as would be the case in BASIC, admittedly, but this same concern is addressed in more readable ways by experienced assembly programmers (e.g., Apple Computer, Inc. 1978b; Budge n.d.).

At the same time, inline comments in the code fragments seem somewhat *too* prevalent, and in some cases give little additional information than an experienced programmer would be able to discern with a glance at the code. For instance, in the reconstructed code fragment

```
;
; ISSUE MESSAGE X
;
   ldy #X;GET X
   lda (ZWORK),Y
   tax
   jsr MESSLSTR;ISSUE MESSAGE
   ldx #$1
   jmp AC9000
```

the inline comment "ISSUE MESSAGE" is completely redundant, as it is literally the only place in those six lines of code where the message could be issued, and furthermore, the name of the routine being called there (MESSLSTR) already has mnemonic value. If we compare Williams's assembly code to that of Bill Budge, allegedly one of Williams's technical peers (Leff 1981, 8), we see that Budge's comments are much more spartan (Budge n.d.).

Lack of code optimizations, paradoxically, could indicate either a neophyte or a trained programmer in some cases. Studying the disassembled code reveals a number of inefficient code sequences, like the following four-instruction routine:

```
ldy #$00 — sty $089d — ldx #$00 — jmp $69ba
```

At first glance, it is puzzling why Williams would not prefer the shorter and more efficient three-instruction sequence

```
ldx #$00 — stx $089d — jmp $69ba
```

which additionally uses one fewer register. But this is not sloppy programming, as it happens. This corresponds to one of the recovered code fragments:

```
;
; TURN LIGHT ON
;
   ldy #ON
   sty LAMPSTAT
   ldx #ZERO
   jmp AC9000
```

We can now see that, even though ON and ZERO turned out to have the same value (0), they were representing different things semantically in the code; Williams was thus making his code more readable – if less efficient – by using the symbolic names. Unfortunately, only small portions of the assembly code are preserved, so we are forced to extrapolate this to other inefficient code in the disassembly. For example, in Williams's code to render images are several variants of

```
lda   memory-location
cmp   #$00
branch if (not) equal to 0
```

The comparison (cmp) in the middle is totally unnecessary, as the first instruction will set the processor status properly for the branch. We are forced to conservatively infer that the comparison must have involved a symbolic constant in the source code, at the cost of some inefficiencies in the implementation. Overall, our initial interpretation of the evidence is that Williams was well trained in general at this time, but perhaps not masterful at 6502 assembly programming.

Finally, we should look at the progression of technical ideas in *Mystery House* and some likely influences. One source of influences is Ken's own background, of course. It is well documented that he was originally intending to develop a Fortran compiler for the Apple II and had the commensurate skill set for doing so (Tommervik 1981; Levy 1984; Jong 2006). The other source of influences we draw upon are computer magazines. Levy characterizes Ken Williams as someone who would read to come up to speed on a subject (Levy 1984). Between that, the Williamses' physical location in Southern California, and the first advertisement for *Mystery House* being placed in a computer magazine (On-Line Systems 1980b), it seems reasonable to assume that they had access to a broad selection of computer magazines.

One key idea in *Mystery House*'s implementation is the use of an interpreter. An examination of compiler books from the 1970s shows that interpreters were mentioned there (e.g. Gries 1971; Aho and Ullman 1977; Calingaert 1979). Whether Williams read these books is immaterial; the point is that it definitively establishes that interpreters were part of the body of knowledge for compiler developers. The BASIC on the Apple II was interpreted too, and it is trivial to find references to interpreters in the computer magazines of that time.

One might argue that there is creativity required to transition from the idea of interpreters used for programming languages to interpreters used for games. There is an important precedent here, however. In mid-1979, Scott Adams (1979) published an article in *Creative Computing* magazine talking about his use of an "adventure interpreter" with a fairly detailed description of the "adventure data base" (i.e., the language being interpreted). This would have predated work on *Mystery House*. Further, the Williamses were familiar with, and had played, Adams's text adventure games (Byron 1990; Tommervik 1981; Williams n.d.) before they created their own.

If the interpreter idea was not novel, perhaps we could find something interesting in the implementation instead. One piece of code whose idea stood out as particularly clever in our reverse engineering was a multiplication routine; it is likely that this routine is what is referred to as MULTIPLY in one of the embedded assembly code fragments.

It may surprise people used to modern computers that CPUs in old microcomputers did not commonly have a multiply instruction, meaning that any multiplication had to be implemented in software. The multiplication technique used by this particular routine in *Mystery House* used what is called "Russian peasant multiplication", a technique that only requires the ability to double and halve numbers, something easy to accomplish with

shift operators that even old CPUs supported. As with most of the other ideas in *Mystery House*'s implementation, this was not new; an implementation of the technique can be found published in a 1977 *Byte* magazine, albeit for a different CPU (Glaeser 1977).

If the idea stood out for being clever, the implementation of the multiplication stands out for being very naive assembly code. For example, an in-memory counter is decremented using the three-instruction code sequence

```
ldy $087d — dey — sty $087d
```

This loads the counter value in the CPU's "Y" register, decrements the register value, and stores it back to memory. This sequence takes 10 machine cycles and occupies seven bytes, and the register's value is never re-used. It would be clearly better to select the single instruction dec $087d, which takes fewer cycles (i.e., is faster to execute), less than half the number of bytes, and doesn't require one of the precious 6502 registers. This is very much a rookie error for a 6502 assembly programmer.

However, let us say for the sake of argument that Williams somehow overlooked this 6502 instruction. What is more telling is the structure of the loop containing the counter decrement. The loop is written as a straightforward translation of a while loop into assembly: there is a conditional test at the top of the loop, and an unconditional branch at the bottom. Because the loop must always be executed at least once, experienced assembly programmers would code it as a repeat loop. This would move the conditional test to the bottom of the loop, which uses fewer instructions and takes advantage of the fall-through when the loop completes. This casts doubt on an interpretation of Williams as a skilled assembly programmer. One possibility is that this multiplication routine was not written by Ken, but that itself would challenge the traditional Ken-as-programmer narrative in a different way.

Perhaps the most visible idea in *Mystery House* is the graphics. The idea to include them was apparently Roberta's (Levy 1984), which would align with what is known about her game design process (Nooney 2017). What Ken would have been responsible for is the *how*: how could a large number of pictures be fit onto a low-capacity floppy disk?

The fact that "hires" pictures on the Apple II could be stored in a reduced amount of space was no secret. In fact, a 1979 issue of *MICRO* magazine contained an article entitled "Apple II Hires Picture Compression" (Bishop 1979). This is the same magazine that *Mystery House* would be advertised in a few months later (On-Line Systems 1980b), and the article was penned by none other than Bob Bishop, one of the programming luminaries that Ken Williams was compared with (Leff 1981, 8). While Bishop's focus was on digitized images rather than line drawings, he prophetically wrote that "it is clearly possible to store an 8-K HIRES picture in considerably less than 8-K bytes, if you are willing to accept a little loss in the image quality" (Bishop 1979, 18).

Maher (2011a) attributes the internal representation of *Mystery House*'s lines to an idea Ken got from the Williamses' use of a VersaWriter for digitizing Roberta's images. However, we think the full story of the lines is more complex. Arcade games using line-based vector graphics were in production in the late 1970s, including both *Asteroids* and *Lunar Lander* in 1979. These games admittedly used a different display technology than the raster graphics of the Apple II, but even on that machine there were line-oriented

FIGURE 6: Reconstructed image from shape table.

graphics. Applesoft BASIC supported "shape tables", line-based programmer-defined shapes that could be drawn in the same high-resolution graphics mode that *Mystery House* used (Apple Computer, Inc. 1978a).

We can prove that Williams was aware of shape tables, as it happens, because they are used to draw some small elements of *Mystery House*'s pictures. Figure 6 shows one of the shapes we reconstructed from *Mystery House*'s data. These were used for images requiring small, precise text that the VersaWriter was probably not well suited for: "WELCOME" on the doormat, the "DOORWAY" label, a note in the room, and the matchbook. The shape representation was not compact enough to use for all the pictures, however.[11]

Williams was not rendering these shapes using the shape table code in Applesoft BASIC, though. We discovered that the shape table shapes – indeed, all the line graphics – are drawn using code from a library supplied by Apple. In other words, Ken did not write that part of the code; Apple co-founder Steve Wozniak did it in 1977. That graphics code in *Mystery House* is in fact (barring address relocation and two very minor patches) identical to the code contained in Apple's Programmer's Aid #1 ROM,[12] whose manual even included the assembly source code (Apple Computer, Inc. 1978b).

Beyond shape tables, there are many references in computer magazines to plotting data and, of course, hardware plotters (a type of printer) also existed at that time. One article on plotting states that "basic plotting software" includes "the capability of chain plotting. That is, plotting a vector from the ending point of the last vector move to the new position on the plotting field without explicitly defining the beginning point every time" (Lerseth 1977, 300). As described above, this is exactly what *Mystery House*'s picture representation was doing.

Clearly, a number of influences existed that could have contributed to the design choices for *Mystery House*'s graphics. Taking context into account, they seem less than novel. The real graphics insight Williams had was simply forgoing some horizontal screen real estate, and representing *x* coordinates using a single byte in the representation – this was really the key to their compactness.

11. This reconstruction was done using a Python script we wrote. We performed a full game walk-through, monitoring shape table usage, but did not detect any used beyond the ones we describe here. We *did* find another set of shapes elsewhere in the game's memory, reconstruction of which yields random-looking doodles. It is possible that this accidentally captured the results of an early experiment using the VersaWriter to create shape tables, but this is highly speculative.
12. Verified using a Python script we wrote to compare the two.

Technological Choices and *Chaîne Opératoire*

For video games, there are technological, player, and developer constraints that frame the technological choices made (Aycock 2016). The technological constraints are those of the CPU itself including the machine, I/O, storage, memory, instruction set, and cost. The CPU is also a player constraint – access to an appropriate system to run the game could be limited in terms of expense but also time and user thresholds. Finally, while developers faced a large assortment of platforms to choose from, few were mutually compatible, and the number of programming tools was limited.

An important consideration of video games as artifacts is that the individual(s) responsible for creating them can be known by name. In a *chaîne opératoire*, humans are active, knowledgeable agents; the Williamses are centered throughout this paper as agents, who made the technological choices and performed the technical acts to produce *Mystery House*. *Mystery House* represents the Williamses' "constellations of knowledge" – the materials used, the techniques employed, and the desired end-point of use and manufacture "depend upon the knowledge that an individual has acquired of them" (Sinclair 2000, 200). Ken Williams did have a certain level of discipline, training, and creativity; and as demonstrated, there are clear examples of not just the choices Ken made through the implementation of *Mystery House* but also some that reflect choices he did not make. We can identify the latter because of our knowledge that there was more than one way for Ken to create the same product. We have framed Ken consistently as an agent in considering his motivations and the broader cultural arena of programming at the time of *Mystery House*'s implementation; as such we can argue that the culture of programming during the 1970s is also reflected in the technological choices that were made by the Williamses throughout the operational sequence.

Space precludes us doing a full *chaîne opératoire* analysis, but it is in fact more important for the development of archaeogaming methodology to illustrate the challenges. Foremost is the abstraction required. We've discussed how terms such as "context" must be redefined, and that we must reframe the areas of analysis and components of technique of the *chaîne opératoire*; we must also challenge the archaeological understanding of what an attribute is. The attributes one would normally use in the analysis of a digital artifact "tool" are beneath the surface in the code and data, and what is measurable and observable to the analyst is in fact only a side effect of the digital artifact running on the computer. In this paper we argue the computer is the raw material and the game is the product made from that raw material. But what is "the computer" exactly? It has hardware attributes but also software attributes. The computer's hardware components (e.g., monitor, keyboard, chips, boards, cords) allow the user to utilize the computer, but only because the ROM on the motherboard contains software. Software, including BASIC and LISA, serves as tools that run on the computer-as-tool; this means the computer is both raw material and tool. The game on the disk image is what is directly created using the computer-as-tool to transform the computer-as-raw material. Further, this product, "the game", itself can only be used using the computer-as-tool. Finally, we must consider that what the people experience as "the game": its attributes, including size, colors, shapes, images, commands, and game messages,

only occur through running the contents of the disk image. This player experience is one step away, in a sense, from what was actually created. Computers clearly introduce a level of abstraction that challenges the approaches of traditional fields like archaeology, and the rise of archaeogaming as a subdiscipline reflects the need to address these provocations to method and theory.

Conclusion

A *chaîne opératoire* approach considers *Mystery House* as not just a digital artifact but one that can be analyzed using the principles and terminology derived from an archaeological understanding of artifact analysis and an anthropological perspective on the organization of technology (see Ingold 1997). This results in an explicitly anthropological incarnation of archaeogaming, and, in terms of how we approached our digital artifact, an inspection of the foundations of our digital artifact beyond a technical description. Additionally, this means that the reverse engineering of *Mystery House* serves not to produce a compendium highlighting Ken Williams's procedural correctness or faults, but as a means to say something more broadly about the culture of game design, and of the human process of creating games.

Acknowledgments

This work is supported in part by the Natural Sciences and Engineering Council of Canada, grant RGPIN-2015-06359. Thanks to D. Finnigan, K. Sherlock, and A. Vignau for helping identify LISA as the assembler used; P. Hagstrom and 4am for help getting the French version into a runnable and analyzable format; and Brieal M.-T. for contributions to #anth498 archaeogaming.

References

ACM Digital Library. n.d. "HOPL: History of Programming Languages". Online: https://dl.acm.org/event.cfm?id=RE352

Adams, S. 1979. "An Adventure in Small Computer Game Simulation." *Creative Computing* 5 (8): 90–97.

Aho, A. V. and J. D. Ullman. 1977. *Principles of Compiler Design*. Boston: Addison-Wesley.

Altice, N. 2015. *I AM ERROR: The Nintendo Family Computer / Entertainment System Platform*. Cambridge, MA: MIT Press.

Apple Computer, Inc. 1978a. *Applesoft II BASIC Programming Reference Manual*. Cupertino: CA: Apple Computer, Inc.

____. 1978b. *Programmer's Aid #1: Installation and Operating Manual* (Apple utility programs). Cupertino, CA: Apple Computer, Inc.

Aycock, J. 2016. *Retrogame Archeology: Exploring Old Computer Games*. New York: Springer. https://doi.org/10.1007/978-3-319-30004-7

____. and A. Reinhard. 2017. "Copy Protection in *Jet Set Willy*: Developing Methodology for Retrogame Archaeology." *Internet Archaeology* 45. https://doi.org/10.11141/ia.45.2

Bailey, G. 2015. "Symmetrical Media Archaeology: Boundary and Context." *Journal of Contemporary Archaeology* 2 (1): 41–52. https://doi.org/10.1558/jca.v2i1.27114

Bar-Yosef, O. and P. Van Peer. 2009. "The *Chaîne Opératoire* Approach in Middle Paleolithic Archaeology." *Current Anthropology* 50 (1): 103–131. https://doi.org/10.1086/592234

Berez, J. M., M. S. Blank and P. D. Lebling. 1989. *ZIP: Z-Language Interpreter Program.* Internal document.

Bishop, B. 1979. "APPLE II Hires Picture Compression." *MICRO – The 6502 Journal* 18: 17–24.

Boellstorff, T., B. Nardi, C. Pearce and T. Taylor. 2012. *Ethnography and Virtual Worlds: A Handbook of Method.* Princeton, NJ: Princeton University Press. https://doi.org/10.2307/j.cttq9s20

Budge, B. n.d. *Pinball Construction Set Source Code for Apple II*. GitHub. Online: https://github.com/billbudge/PCS_AppleII

Byron, T. 1990. "Roberta's Bequest." *STart: The ST Monthly* 4 (8): 22–26.
Calingaert, P. 1979. *Assemblers, Compilers, and Program Translation.* Rockville, MD: Computer Science Press.
Copplestone, T. 2017. "Adventures in Archaeological Game Creation." *SAA Archaeological Record* 17 (2): 33–39.
Dennis, M. 2016. "Archaeogaming?". Online: http://gingerygamer.com/index.php/archaeogaming
Dobres, M.-A. 2000. *Technology and Social Agency: Outlining a Practice Framework for Archaeology*. Malden, MA: Blackwell.
Donovan, T. 2010. *Replay: The History of Video Games*. Hove, UK: Yellow Ant.
Espinosa, C. 1979. *Apple II Reference Manual.* Cupertino, CA: Apple Computer, Inc.
Fuller, M., ed. 2008. *Software Studies: A Lexicon.* Cambridge, MA: MIT Press. https://doi.org/10.7551/mitpress/9780262062749.001.0001
Glaeser, C. D. 1977. "Novel 8 Bit Multiplication." *Byte* 2 (7): 142.
González-Tennant, E. 2016. "Archaeological Walking Simulators." *SAA Archaeological Record* 16 (5): 23–28.
Greenfield, P. 1994. "Video Games as Cultural Artifacts." *Journal of Applied Developmental Psychology* 15 (1): 3–12. https://doi.org/10.1016/0193-3973(94)90003-5
Gries, D. 1971. *Compiler Construction for Digital Computers.* New York: Wiley.
Haigh, T. 2015. "The Tears of Donald Knuth." *Communications of the ACM* 58 (1): 40–44. https://doi.org/10.1145/2688497
Huhtamo, E. and J. Parikka. 2011. "Introduction." In *Media Archaeology: Approaches, Applications, and Implications*, 1–15. Berkeley, CA: University of California Press.
Hyde, R. 1981. *LISA: A Professional Assembly Language Development System for Apple Computers* (Version 2.5). [Oakhurst, CA]: On-Line Systems.
Ingold, T. 1997. "Eight Themes in the Anthropology of Technology." *Social Analysis* 41 (1): 106–138.
International Directory of Company Histories, 2001. "Sierra On-Line, Inc." In *International Directory of Company Histories* (41st edition), edited by T. Grant, 361–364. Detroit, MI: St. James Press.
Jargon File. 2003. "Superprogrammer." In *The Jargon File*, edited by E. S. Raymond, version 4.4.7. Online: http://www.catb.org/jargon/html/S/superprogrammer.html
____. n.d. "Jargon Chaff File." In *The Jargon File*, edited by E. S. Raymond, version 4.4.7. Online: http://www.catb.org/jargon/chaff.html#bum
Jong, P. 2006. "Ken Williams." [Interview] *Adventure Classic Gaming*. Online: http://www.adventureclassicgaming.com/index.php/site/interviews/197/
K. M. 2018. *The Archaeology of Tomb Raider*. Online: https://tombraiderarchaeology.com/
Kirschenbaum, M. G. 2008. *Mechanisms: New Media and the Forensic Imagination*. Cambridge, MA: MIT Press.
Leff, B. 1981. "Distribution Does Not a Publisher Make." [Letter to the editor] *Softalk* 1 (8): 8.
Lemonnier, P. 1992. *Elements for an Anthropology of Technology*. Anthropological Papers, Museum of Anthropology, University of Michigan 88. Ann Arbor, MI: Museum of Anthropology, University of Michigan.
Lerseth, R. J. 1977. "A Plot is Incomplete Without Characters." In *The Best of Byte, Volume 1*, edited by D. H. Ahl and C. T. Helmers Jr, 300–308. Morristown, NJ: Creative Computing Press.
Levy, S. 1984. *Hackers: Heroes of the Computer Revolution*. New York: Dell.
Loguidice, B. and M. Barton. 2009. *Vintage Games: An Insider Look at the History of* Grand Theft Auto*,* Super Mario*, and the Most Influential Games of All Time*. Amsterdam: Focal Press.
Maher, J. 2011a. "Mystery House, Part 1." *The Digital Antiquarian*, 8 October. Online: https://www.filfre.net/2011/10/mystery-house-part-1/
____. 2011b. "On-Line Systems is Born." *The Digital Antiquarian*, 17 October. Online: https://www.filfre.net/2011/10/on-line-systems-is-born/
Marino, M. C. 2006. "Critical Code Studies." *Electronic Book Review*, 4 December. Online: http://electronicbookreview.com/thread/electropoetics/codology
Mol, A., C. Ariese-Vandemeulebroucke, K. Boom, A. Politopoulos and V. Vandemeulebroucke. 2016. "Video Games in Archaeology: Enjoyable but Trivial?" *SAA Archaeological Record* 16 (5): 11–15.
Montfort, N. 2003. *Twisty Little Passages: An Approach to Interactive Fiction*. Cambridge, MA: MIT Press. https://doi.org/10.7551/mitpress/6936.001.0001
Moshenska, G. 2016. "Reverse Engineering and the Archaeology of the Modern World." *Forum Kritische Archäologie* 5: 16–28.
Newman, M. Z. 2017. *Atari Age: The Emergence of Video Games in America*. Cambridge, MA: MIT Press. https://doi.org/10.7551/mitpress/10021.001.0001
Nooney, L. 2013. "A Pedestal, a Table, a Love Letter: Archaeologies of Gender in Videogame History." *Game Studies* 13 (2). Online: http://gamestudies.org/ 1302/articles/nooney
____. 2017. "Let's Begin Again: Sierra On-Line and the Origins of the Graphical Adventure Game." *American Journal of Play* 10 (1): 71–98.Online: http://www.journalofplay.org/sites/www.journalofplay.org/files/pdfarticles/10-1-Article-3-Lets-begin-again.pdf

On-Line Systems. 1980a. "Apple II Software from On-Line Systems. Advertisement." *MICRO – The 6502 Journal* 26: 1.

____. 1980b. "New Apple II / Apple II Plus Software from On-Line Systems. Advertisement." *MICRO – The 6502 Journal* 24: 80.

Perry, S. and C. Morgan. 2015. "Materializing Media Archaeologies: The MAD-P Hard Drive Excavation." *Journal of Contemporary Archaeology* 2 (1): 94–104. https://doi.org/10.1558/jca.v2i1.27083

Ramsay, M. 2012. *Gamers at Work: Stories Behind the Games People Play*. New York: Apress. https://doi.org/10.1007/978-1-4302-3352-7

Rawitsch, D. 2017. "Classic Game Postmortem: 'Oregon Trail'." Paper presented at the Game Developer's Conference, San Francisco, CA, 27 February to 3 March.

Renfrew, C. and P. Bahn (2015). *Archaeology Essentials* (3rd edition). London: Thames and Hudson.

Reinhard, A. 2013. "What is Archaeogaming?" *Archaeogaming*, 9 June. Online: https://archaeogaming.com/2013/06/09/what-is-archaeogaming/

____. 2015a. "Archaeogaming Map (Revised)." *Archaeogaming*, 18 December. Online: https://archaeogaming.com/2015/12/18/archaeogaming-map-revised/

____. 2015b. "Archaeogaming: Tools and Methods." *Archaeogaming*, 18 September. Online: https://archaeogaming.com/2015/09/18/archaeogaming-tools-and-methods/

____. 2015c. "Excavating Atari: Where the Media was the Archaeology." *Journal of Contemporary Archaeology* 2 (1): 86–93. https://doi.org/10.1558/jca.v2i1.27108

____. 2016a. "Materialization of the Immaterial." *Archaeogaming*, 9 March. Online: https://archaeogaming.com/2016/03/09/materialization-of-the immaterial/

____. 2016b. "Towards Archaeological Tools and Methods for Excavating Virtual Spaces." *SAA Archaeological Record* 16 (5): 19–22.

____. 2017. "The (Harris) Matrix, Part 1: Visualizing Software Stratigraphy." *Archaeogaming*, 2 April. Online: https://archaeogaming.com/2017/04/02/the-harris-matrix-parti-visualizing-software-stratigraphy/

Roux, V. 2016. "Ceramic Manufacture: The *chaîne opératoire* Approach." In *The Oxford Handbook of Archaeological Ceramic Analysis*, edited by A. Hunt, 1–17. Oxford: Oxford University Press.

Sather, J. 1983. *Understanding the Apple II.* Chatsworth, CA: Quality Software.

Savetz, K. 2016. "Phoenix Software Source Code." *AtariAge*, 6 November. Online: http://atariage.com/forums/topic/258786-phoenix-software-sourcecode/

Sellet, F. 1993. "Chaîne opératoire; The Concept and its Applications." *Lithic Technology* 18 (1–2): 106–112. https://doi.org/10.1080/01977261.1993.11720900

Sierra Help Pages. n.d. "Mystery House Help." Online: http://www.sierrahelp.com/Games/MysteryHouseHelp.html

Sillar, B. and M. Tite. 2000. "The Challenge of 'Technological Choices' for Material Science Approaches in Archaeology." *Archaeometry* 42 (1): 2–20. https://doi.org/10.1111/j.1475-4754.2000.tb00863.x

Sinclair, A. 2000. "Constellations of Knowledge: Human Agency and Material Affordance in Lithic Technology." In *Agency in Archaeology*, edited by M.-A. Dobres and J. E. Robb, 196–212. London: Routledge.

Sotamaa, O. 2014. "Artifact." In *The Routledge Companion to Video Game Studies*, edited by M. J. P. Wolf and B. Perrone, 3–9. New York: Routledge.

Spafford, E. H. 1989. "The Internet Worm Program: An Analysis." *ACM SIGCOMM Computer Communications Review* 19 (1): 17–57. https://doi.org/10.1145/66093.66095

Styhre, A., A. M. Sazczepanska, and B. Remneland-Wihkamn. 2018. "Consecrating Video Games as Cultural Artifacts: Intellectual Legitimization as a Source of Industry Renewal." *Scandinavian Journal of Management* 34 (1): 22–28. https://doi.org/10.1016/j.scaman.2017.11.003

Tommervik, A. 1981. "Exec On-Line Systems: Adventures in Programming." *Softalk* 1 (6): 4–6.

Tringham, R. and M. Ashley. 2015. "Becoming Archaeological." *Journal of Contemporary Archaeology* 2 (1): 29–41. https://doi.org/10.1558/jca.v2i1.27089

Trivette, D. B. 1985. "Inside *King's Quest*." *Compute!* 7 (2): 136–138.

Watrall, E. 2002. "Interactive Entertainment as Public Archaeology." *SAA Archaeological Record* 2 (2): 37–39.

Wexler, J., A. Bevan, C. Bonacchi, A. Keinan-Schoonbaert, D. Pett and N. Wilkin. 2015. "Collective Re-Excavation and Lost Media from the Last Century of British Prehistoric Studies." *Journal of Contemporary Archaeology* 2 (1): 126–142. https://doi.org/10.1558/jca.v2i1.27124

Wiegers, K. E. 2002. *Peer Reviews in Software: A Practical Guide.* Boston: Addison-Wesley.

Williams, K. n.d. "Introduction to the Roberta Williams Anthology." *Sierra Help Pages*. Online: http://www.sierrahelp.com/Misc/IntroductionToRWAnth.html

Worth, D. and P. Lechner. 1981. *Beneath Apple DOS*. Chatsworth, CA: Quality Software.

John Aycock is an Associate Professor in the Department of Computer Science at the University of Calgary. His research interests include exploring the implementation of old computer games. Address for correspondence: Department of Computer Science, University of Calgary, 2500 University Drive NW, Calgary, Alberta, Canada T2N 1N4.

Katie Biittner is an Assistant Professor of Anthropology at MacEwan University. Dr. Biittner is an anthropological archaeologist whose work in Iringa Region, Tanzania critically investigates the construction of cultural heritage using contemporary and archaeological material culture. She has also conducted archaeological field work in British Columbia, Alberta, Ontario, and Idaho. Her teaching and research interests include lithic analysis, gender archaeology, anthropology of the body, and archaeogaming. Address for correspondence: Department of Anthropology, MacEwan University, 10700 - 104 Avenue NW, Edmonton, Alberta, Canada T5J 4S2.

Journal of Contemporary Archaeology 6.2 (2019) 183–205
ISSN (print) 2051-3429 (online) 2051-3437 https://doi.org/10.1558/jca.36745