Loading… The Journal of the Canadian Game Studies Association
Vol 17(28): 71-86
http://loading.gamestudies.ca

Close Reading a Close Reading: Revisiting *Mystery House* in *Mechanisms*

John Aycock
University of Calgary
aycock@ucalgary.ca

Abstract

Mystery House is a game that was released in 1980 for the Apple II computer. It was the first text adventure game known to be illustrated and was the game design debut of celebrated game designer Roberta Williams. The landmark 2008 book *Mechanisms: New Media and the Forensic Imagination* performed a close reading of a *Mystery House* floppy disk, and in this paper I turn a critical eye to this previous work. My close reading of *Mystery House*'s earlier close reading reveals a number of problems, which I use as a vehicle to highlight a notable and important technical gap in game studies scholarship, and argue for an inclusive approach to advancing the field.

Author Keywords

Close reading, *Mystery House*, floppy disk, computer science, interdisciplinarity.

Introduction

Close readings of games are an accepted part of game studies. The technique features in introductory texts to the field (Egenfeldt-Nielsen et al., 2024; Fernández-Vara, 2024), and we can find any number of examples published in *Game Studies*, *Loading*, and other scholarly venues. The importance of close readings to marginalized scholars has been noted (Stang, 2022), and there are nuances to understanding a game as a text to be read (Carr, 2019). Bizzocchi and Tanenbaum (2011) offer an extensive discussion of the close reading technique with respect to games, underscoring the breadth of lenses that can be applied.

Juxtaposed with this, Frans Mäyrä's book, *An Introduction to Game Studies*, characterizes game studies as having "reached the point where it has become established both as a field of *scientific* inquiry and as a branch of knowledge that is formally taught" (2008, p. 4, emphasis mine). This is a bold assertion; as a scientific endeavour, Mäyrä's claim implies the use of certain evidentiary standards and methodologies within game studies. It would mean that results from previous close readings might be subject to independent verification and possibly even refutation. Even if we do

not accept Mäyrä's claim, however, there is a scholarly tradition of engaging critically with previous work.

In this article, I undertake such an exercise: I independently conduct a close reading of a lauded close reading that examines the early game *Mystery House* and its floppy disk medium. This leads to larger discussions about game studies scholarship, but first, some background information is necessary.

Background

In late 2007, Matthew Kirschenbaum's book *Mechanisms: New Media and the Forensic Imagination* was published (Kirschenbaum, 2008), to positive if not gushing reviews. Science-fiction author Bruce Sterling, writing for *Wired*, led his review with "Hey wow, another humanities professor who gets it about computers" (2009). Kirschenbaum's methodology was praised—Drucker (2009, paragraph 8) opined that "the importance of the book is in its demonstration of method as well as in the substance of its argument," a sentiment echoed elsewhere (Grigar, 2010).

By the time the book's 2012 paperback edition was published, the one we use here to allow time for errata fixes, the back cover had been adorned with a list of awards that the book had received from the Society for Textual Scholarship, the Society for the History of Authorship, and the Modern Language Association (Kirschenbaum, 2012). The book is now highly cited,[1] over a decade and a half after its first release, and its methodology has been taken up directly by others (e.g., Black, 2012; Hodges, 2019). Given its accolades and influence, it stands to reason that if any work could withstand scrutiny, it would be *Mechanisms*.

A central feature of *Mechanisms* is its three case studies, which Kirschenbaum characterizes as "close readings of individual electronic textual objects" (18).[2] The three objects he selected for these readings represent a range of works: William Gibson's *Agrippa*, Michael Joyce's *Afternoon: A Story*, and the 1980 Apple II game *Mystery House*. Of these, I examine the latter for three reasons. First, I have experience with games from this era and can therefore offer an informed critique. Second, the core of *Mechanisms*' *Mystery House* reading is only 19 pages (114–132), making a highly focused, robust close reading of Kirschenbaum's close reading possible. Third, Kirschenbaum deliberately puts forth *Mystery House* as an exemplar: "Pedagogy is also among my motives; is it my hope that from this example others may be led to undertake similar walkthroughs or forensically replete readings of new media objects" (115).

1 As of November 3, 2025, Semantic Scholar and Google Scholar reported 425 and 1,336 citations, respectively.
2 For brevity, unqualified instances of page numbers are references to Kirschenbaum (2012).

A brief background of the game is helpful for context. *Mystery House* is an entry in the text adventure game genre, where game input and output is purely textual; this was already an established form by the time of *Mystery House*'s release in 1980. On its own, this game genre could be, and has been, examined at book length (Montfort, 2003). What made *Mystery House* notable among computer users of the time was that it was the first known *illustrated* text adventure game, as shown in Figure 1.[3] To a modern audience, the crude line drawing illustrations may be deeply underwhelming, but this illustration was an innovative feature at the time and helped to make the game a bestseller (Nooney, 2023). The game fit onto one side of a 5¼-inch floppy disk, and it was an image of this disk—Mystery_House.dsk—that Kirschenbaum used for his close reading. More specifically, Kirschenbaum used an image containing the version of the game released into the public domain in 1987. The word "image" as used in this context means that the .dsk file contains a version of the binary data that was magnetically stored on the floppy disk, which can act as a substitute for the physical media for certain operations. With the data from this disk image, for example, the game can be played in an Apple II emulator as in Figure 1, or it can be analyzed using a variety of software tools. Kirschenbaum chose a tool called "FishWings" for his analysis of the .dsk image (115), and at this point, a brief digression is required to conduct a closer examination.

3 There is a fuller backstory to the game that is out of scope here; refer to Nooney (2023) for a more extensive discussion.

Vol 17(28): 71-86
http://loading.gamestudies.ca

Figure 1. Screenshot of *Mystery House*'s opening scene, running in the MAME 0.255 emulator.

When Fish Fly

Mechanisms concerns itself with computer forensics, part of forensic science (48), and a central tenet of science is reproducibility. We should be able to reproduce the results Kirschenbaum reported in his close reading based on the information that he provided. The book starts promisingly by revealing that he used a computer with the Windows XP operating system (22). Nevertheless, although *Mechanisms* emphasizes particulars in other sections (185, 195–96), its reading of *Mystery House* is less forthcoming. The version of FishWings that Kirschenbaum used is never specified. Only the address of the website where he found the program (115), which appears to be the personal Verizon website of the tool's author, Charlie Danemark, is mentioned, and this site is no longer reachable. The Internet Archive's Wayback Machine saves web-based content and happened to preserve four snapshots of the FishWings site prior to when *Mechanisms* was published, but Kirschenbaum did not provide the access dates for online resources (265–6), thereby denying us the information we need in order to decide which of the four FishWings snapshots to use. By sheer luck, FishWings was not subject to feverish

development, and the version of the software is the same throughout that time period: version 0.87.

Reproducing Kirschenbaum's results also requires using the same .dsk image that he did, but I was faced with a similar identification problem, since I only had the address of a volatile software repository (114) which held no filename that exactly matched the "Mystery_House.dsk" of *Mechanisms*. Ironically, a solution was readily at hand when the book was written, as Kirschenbaum mentions the forensic use of hashing (157)—a calculation computed on the .dsk image that could convey with exceedingly high accuracy the identity of the image he used, even if the filename were different—and this "hash value" could easily have been supplied to allow reproducibility. Instead, I needed to manually, painstakingly compare the data shown by my installation of FishWings in Windows XP against the figures in *Mechanisms*, at times resorting to a jeweler's loupe to magnify fine details on *Mechanisms*' printed pages. I succeeded in finding the correct .dsk image in the end, and the figures labeled "reproduced" in this paper are my own verified ones.[4]

A Close (Re)reading

We can now embark on the close reading at last. *Mechanisms*' tour of Mystery_House.dsk begins with an acknowledgment of some filenames found on the disk, specifically "various text files (HELLO, MYSTERY.HELLO, MESSAGES)" (115). This is an unfortunate start, because HELLO and MYSTERY.HELLO are not, in fact, text files; this might seem to be a picayune complaint, but it is vital to understanding both the depth of language at play in the disk image and how the computer used the disk image's contents.

Kirschenbaum relies heavily on the reference book *Beneath Apple DOS* (Worth & Lechner, 1981) for his low-level analysis of the disk image (120). I concur that this is an excellent choice; it is a superb, detailed technical treatise from that time period. By later drawing on *Beneath Apple DOS* (henceforth referred to as *BAD*) to interpret low-level filename information, Kirschenbaum not only demonstrates his awareness of the variety of file types but also provides a figure that, when read closely in conjunction with *BAD*, shows that HELLO and MYSTERY.HELLO are not text files (122). We do not need to resort to low-level machinations to reach this conclusion because the standard user view of an Apple II's disk contents, including filenames and file types, was provided by a command the user would type called CATALOG. As a self-professed Apple II user in his youth (109), Kirschenbaum would have been familiar with this command. Even if he forgot this bit of trivia, FishWings is able to provide the CATALOG command's output as shown in Figure 2.

4 For anyone performing a close reading of my close reading of Kirschenbaum's close reading, the .dsk's MD5 hash value is 7290a9e90291c37e8dce54f1b8d74a37.

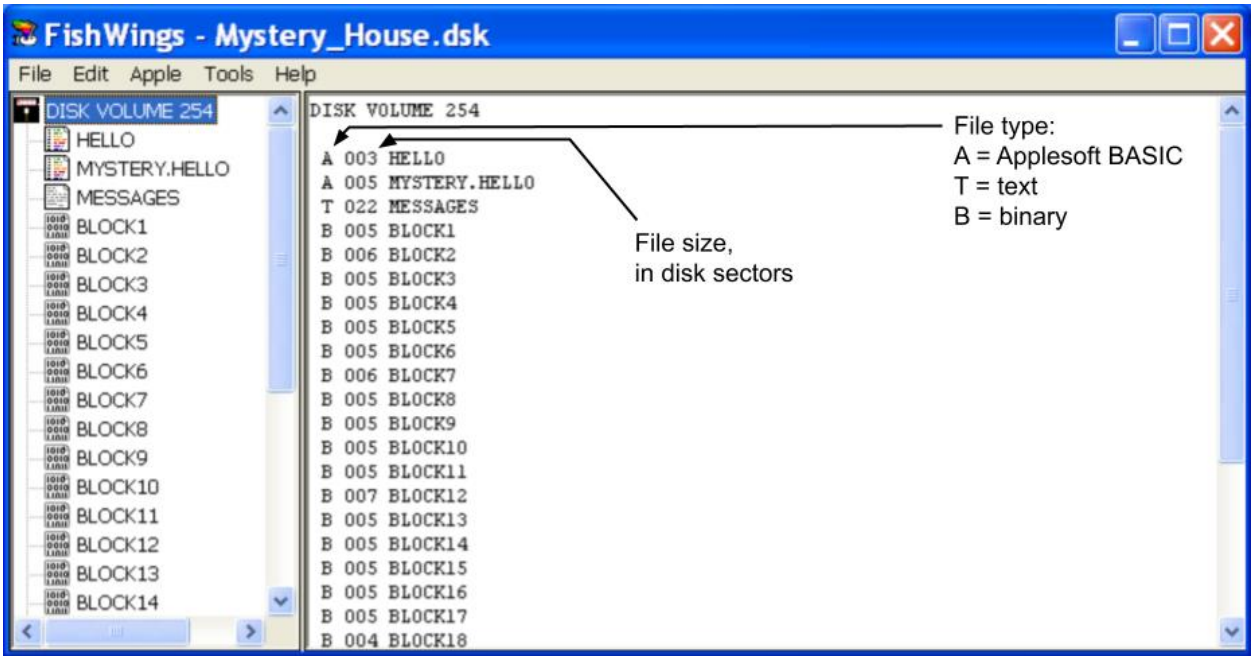

Figure 2. Mystery_House.dsk's disk catalog, annotated.

The CATALOG view in Figure 2 confirms Kirschenbaum's assertion about MESSAGES being a text file, with its "T" type on display, and HELLO and MYSTERY.HELLO are both shown having the type "A" indicating that those files contain computer programs written in the Applesoft BASIC computer language. In other words, there is another layer of language and text here. BASIC was a significant language from the computer user's point of view, because it was a programming language designed to be usable by novices and nonprofessionals (Rankin, 2018), and the Apple II had its dialect of BASIC immediately available to the user the instant the machine was powered on, providing a direct invitation to program the computer. Programs in the BASIC language were thus not just a form of text; they were a functional, empowering form of text for the computer user of the time. Viewing HELLO in FishWings (Figure 3) shows what the user view of this program text would have been, which does appear on the surface to be a text file, but FishWings has reconstituted this view for us. Applesoft BASIC programs, like numerous other BASICs of the time, were both stored and run in what is known as a tokenized form, an abbreviated version of the language used for both space and speed reasons. These tokens are mentioned in *BAD* (Worth & Lechner, 1981, p. 4-12) and are well-documented (Apple Computer, 1978, p. 121). Even without this information, it is easy to see that something is amiss using FishWings; Kirschenbaum teaches us to compare the high- and low-level representations of a file using MESSAGES (124–5), and applying this same technique to HELLO reveals that its low-level view is highly discrepant from that shown in Figure 3.[5]

5 The start of HELLO is on track 33, sector 14.

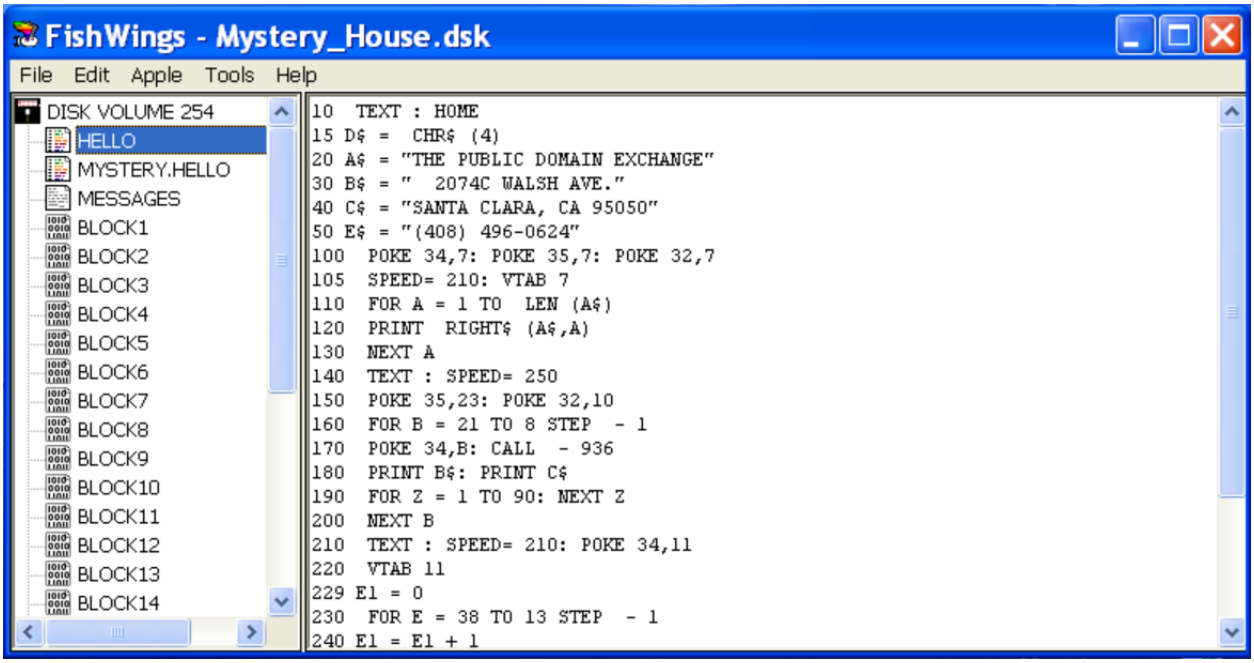

Figure 3. The beginning of the HELLO Applesoft BASIC program as reconstructed by FishWings.

There is additional significance to the filename "HELLO" in that, of all the files on the disk, the BASIC code that HELLO contains is run first when the Mystery House disk is booted; the BASIC code for HELLO, in turn, runs the BASIC code in MYSTERY.HELLO. When an Apple II disk was initialized and prepared for use, the user was required to supply the name of a BASIC "greeting program" that in theory could have had any name, but was conventionally called "HELLO" (Apple Computer, 1981, p. 13). This convention was so strongly established that *BAD* refers to this file simply as the "HELLO program," including the part of *BAD* where it explicitly describes the role of HELLO in the boot process (Worth & Lechner, 1981, p. 5-8). By dismissing HELLO and MYSTERY.HELLO as text files, two layers of textuality and their importance to the mechanism of starting *Mystery House* have been ignored.

Mechanisms immediately follows its mischaracterization of the two BASIC program files with an observation about the numbered "BLOCK" filenames: "there are numerous 'BLOCKS' of binary data, a block being equivalent to two sectors" (115–6). This too is mistaken. While Kirschenbaum is correct in that the term "block" can be used in this way, he has apparently misunderstood what FishWings was displaying, conflating the idea of disk blocks with the fact that the filenames happen to have the prefix "BLOCK." As was the case previously, *Mechanisms* itself supplies evidence to the contrary in some FishWings data (122) and, also as before, we can see this clearly in Figure 2's CATALOG output. The CATALOG shows the size of each file, and none of the numbered BLOCK files occupy only two sectors; they are not "blocks" in that sense of the word. And, although it would take a deeper analysis than *Mechanisms* provides, the

BLOCK files are essential to what made *Mystery House* unique, as they contain the binary data for the line-drawn illustrations in the game (Aycock & Biittner 2019).

Kirschenbaum relies extensively on low-level views of disk data for his close reading, and since they provide key evidence for his conclusions, it is worth examining them in detail. A sample of this low-level view, a reproduction from *Mechanisms* (128), is shown in Figure 4. This is what is referred to as a hex dump,[6] in this case displaying the 256 bytes' worth of data on a disk sector, where a sector is a discrete division of space on the disk. Each of these 256 data values is expressed in hex(adecimal), or base 16, whereas the numbers we would normally use in everyday life are in base 10. All this means is that the numbers appear differently: in base 10, there are 10 possible digits (0–9); in base 16, there are 16 possible digits, but lacking conventional digits beyond 9, base 16 repurposes the alphabetic characters A–F to represent the digits 10–15. Every pair of base 16 (hex) digits in the hex dump expresses the value of a single byte of data, which may be a value from 0–255. For example, in Figure 4, "A0" is the value of a single byte, as is "D5", and in base 10 they are the numbers 160 and 213, respectively.

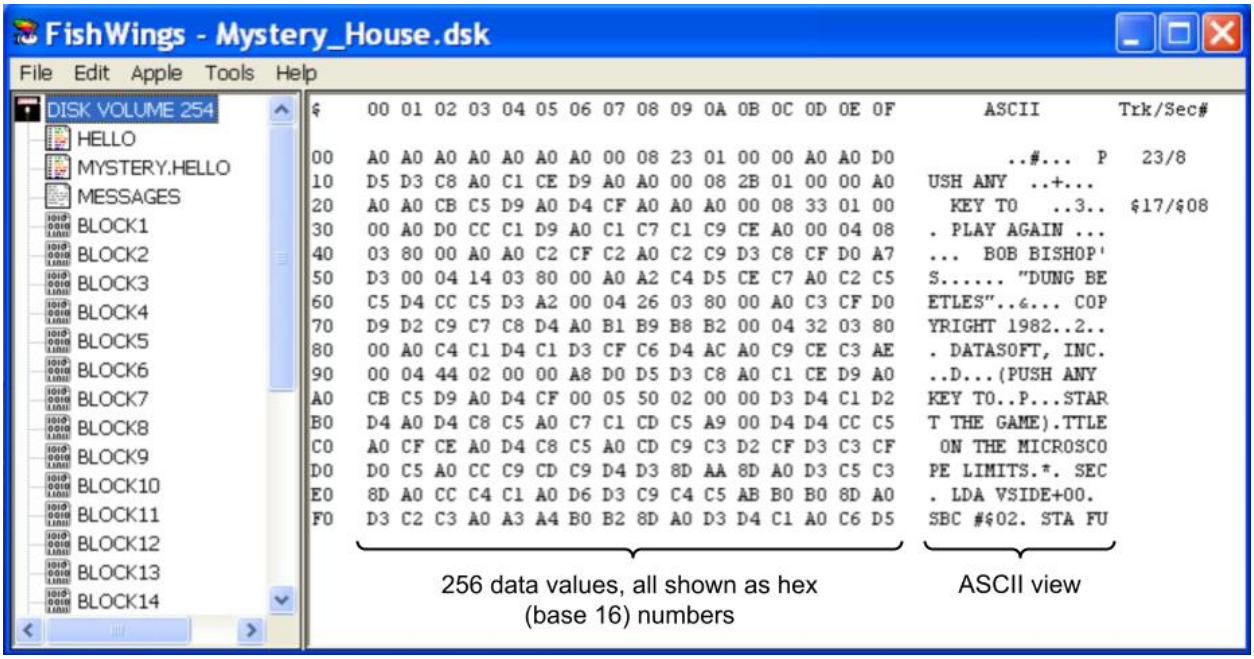

Figure 4. An annotated hex dump of some sector data from Mystery_House.dsk (independently reproduced from *Mechanisms'* Figure 3.9).

A typical hex dump also shows a different view of the data values: an ASCII view, and each character in a line of the ASCII view corresponds one-to-one with the 16 data values on the same line. As Kirschenbaum explains, "wherever the byte matches a particular value in the ASCII

6 *Mechanisms* prefers the term "hex editor," by contrast. While hex editors do exist, FishWings is not one of them because the data shown cannot be edited. In fact, the FishWings help information (Danemark, 2004) explicitly says "Note that this is a display only, not an editor. Changing the display does not effect [sic] the disk image."

character set [an encoding of character data], that ASCII character is displayed. A period indicates that the binary value falls outside the range of standard ASCII (0–127 decimal)" (117). He then offers three examples, none of which coincide with the definition he provides. The problem can be seen in Figure 4, where some values between 0–127 are nonetheless represented by periods in the ASCII view, like 00, 01, and 04; a hex dump only shows *printable* ASCII characters, a small but significant difference in terms of understanding what the tool is and is not displaying. Furthermore, of the alphabetic characters shown in Figure 4's ASCII view, one hundred and thirty of them have values greater than 127, which all serve as counterexamples to Kirschenbaum's definition. There is not just one value corresponding to the letter "A" in the Apple II (118): for example, in that computer's screen memory, one value creates an "A" with a flashing background, a different value creates an "A" with foreground and background inverted, a different value again creates a normal-looking "A" (Apple Computer, 1985, p. 20). FishWings, as a tool designed to work with Apple II disks, has done us the favour of showing us text outside the standard ASCII range for this reason. While one can argue that the different manifestations of "A" are "allographically indistinguishable" (134), what this illustrates is that there are nuances to understanding the text of a computing platform and its tools that were not captured in *Mechanisms*.

Hex dump format aside, *Mechanisms* uses the data reproduced in Figure 4 as evidence that the disk containing *Mystery House* was re-used (127), and I do not dispute this. The name of one of the disk's previous occupants, the early game *Dung Beetles* by the late Bob Bishop, is visible in the ASCII view of the hex dump. What is equally remarkable about this data, however, is something that is immediately apparent to an experienced observer, something that *Mechanisms* omits mention of completely. At the end of the sector data is the text:

```
SEC
LDA VSIDE+00
SBC #$02
STA FU…
```

This is a fragment of assembly-language source code for the Apple II, a (slightly more) human-readable form of the machine language of the computer. The presence of this assembly language code was not needed for a game to run; it would have been captured here accidentally through a series of fortunate events. Yet again, we are seeing a layer of text that is overlooked by *Mechanisms*' close reading.

On page 123, the close reading takes a substantial detour: only three lines of the main text are included, with the rest of the page dominated by a footnote containing a gripping Odyssean tale of detective work. Kirschenbaum's contention is that Mystery_House.dsk's Volume Table of Contents (VTOC)—the data structure through which the Apple II's operating system is able to

locate the remaining data on the disk—is wrong. The MESSAGES file is located on the disk at tracks 33–34, yet his reading of the VTOC, reproduced in Figure 5, is that "no part of the program should have been found past track 22" (123), eventually deciding that an errant program must not have updated the VTOC after placing MESSAGES in that location.

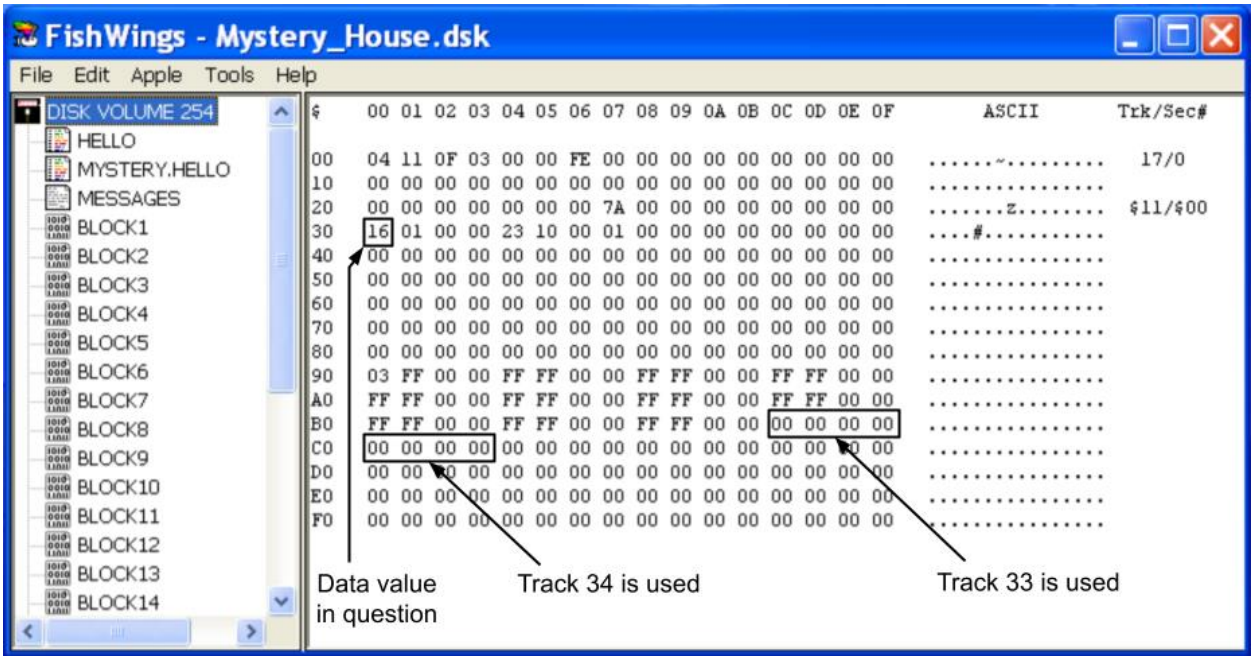

Figure 5. An annotated hex dump of the Volume Table of Contents from Mystery_House.dsk (independently reproduced from *Mechanisms*' Figure 3.2).

Kirschenbaum's confusion likely stems from a misinterpretation of his *BAD* reference, when he writes that the value identified in Figure 5 denotes "the last track with sectors allocated to it" (123).[7] *BAD* describes this particular value as the "Last track where sectors were allocated" (Worth & Lechner, 1981, p. 4-3), and this description admits two interpretations: first, that the value is the *maximum* track where sectors were allocated (the route Kirschenbaum takes), or second, that it is the *most recent* track where sectors were allocated. Reading further in *BAD* suggests that the latter interpretation is correct (Worth & Lechner, 1981, pp. 8-31–8-32), and from the VTOC data in Figure 5, it is apparent that tracks 33–34 are marked as occupied, that is, the VTOC was in fact updated counter to Kirschenbaum's claim. Finding this bookkeeping information in the VTOC is straightforward, because *BAD* has an annotated VTOC figure that helpfully highlights where track 34's information is located (Worth & Lechner, 1981, p. 4-4).

While Kirschenbaum can run *Mystery House* in an emulator (124), *Mechanisms* adopts a passive approach to studying the game, even though experimentation is central to the scientific

7 Kirschenbaum calls this the "thirtieth byte" (123) which can result from erroneously reading the hex dump's row and column labels, which are in base 16. It is actually the byte at location 30 in base 16, which is 48 in base 10.

framework it emphasizes. The VTOC situation can easily be induced by typing a handful of standard Apple DOS commands in an Apple II emulator, combined with an understanding of how data is placed on the disk that is explained on the page opposite the extensive footnote (122). While Kirschenbaum correctly concludes that DOS had not "simply made a mistake in the VTOC" (123), the phenomenon could be explained without resorting to untested theories of phantom software.

It is worth remembering that *Mechanisms* is not second-rate scholarship; it is an acclaimed book that is highly cited and influential. I am also not casting aspersions on Kirschenbaum or suggesting that he acted in an improprietous way. And yet, it seems sufficient at this point to say that *Mechanisms*' treatment of *Mystery House* differs at times from the evidence presented in the book, and also overlooks evidence both in its cited references and directly on the disk image. To understand how this might have occurred, and the broader ramifications of this to game studies, we must step back and look at Kirschenbaum's *Mystery House* reading in the context of *Mechanisms*' content and other research on *Mystery House*.

Discussion

Mechanisms suffers from a dichotomy in that, on the one hand, the book delves into the storage of bits on magnetic media to a microscopic level (62). On the other hand, Mystery_House.dsk is not a copy of the floppy disk's contents to that same degree, although one would be forgiven for thinking otherwise, since the book says that "a disk image is a literal representation of every *bit* of information on some original instance of source media, in this case some actual, physical, specific 5¼-inch floppy disk" (115, emphasis in original). While a .dsk file includes the decoded data from the disk's sectors, it completely excludes the additional data that surrounds those sectors on the floppy as well as the low-level bitstream read by the disk drive. A .dsk is thus several levels of abstraction away from the magnetic media's surface, a conceptual gap that *Mechanisms* fails to justify. The absent data and their encodings can themselves be seen as languages, and as *BAD* and other sources document, they were languages evolving through versions of the operating system that yielded dramatically different amounts of storage even though the physical medium was the same (Worth & Lechner, 1981; Sather, 1983).

Above the .dsk level, *Mystery House* was a game with multiple pieces working together. *Mechanisms* identifies the low-hanging fruit of the MESSAGES file, an easy find thanks to its centralized, all-textual format, and I have already noted how the BLOCK files that *Mechanisms* glossed over contained the prominent graphics data for the game. It has been known for some time that *Mystery House* was programmed in assembly language; Steven Levy, for instance, mentioned this in his book *Hackers*, published over two decades before *Mechanisms* (Levy, 1984). Assembly language has a nearly one-to-one correspondence with the machine language that Kirschenbaum mentions in his close reading, and that means that *Mystery House*'s machine language can be considered text suitable for consumption by the computer, as well as a text that

was crafted more or less directly by the human programmer (Ken Williams) who wrote the game—we can see the programmer's authorial skill on display by analyzing its machine code.

Levy's *Hackers* also discusses the follow-up game to *Mystery House*: "Ken [Williams] had developed a *whole new* assembly-language interpreter for writing adventure games" (1984, p. 301, emphasis mine). This implies that *Mystery House* also had an interpreter, with a layer of code that could not be run directly by the computer, which is consequently another distinct level of text that *Mechanisms*' close reading does not acknowledge. Indeed, a thorough analysis of *Mystery House* has found all this and more, including encoded fragments of the game's lost source code text (Aycock & Biittner 2019).

It is reasonable to wonder how a scholar trained in textual criticism would be able to discern all of the technical details above, *which is precisely the point*. The close reading of *Mystery House* in *Mechanisms* does succeed in providing a pedagogical example, but the example is closer to the very hubris the *Mystery House* chapter begins by dismissing (113). Kirschenbaum later writes "those of us who study artifacts like books may one day also want or need to study software—first learning some of the terminology and technical issues—out of the conviction that our intellectual perspectives are unique and valuable" (206). I agree that textual scholarship can contribute to the study of software, and in this case the study of games, but there is a danger in assuming that knowledge in a specialized technical field is so trivial as to go it alone.[8] Ultimately, this article is not about the failings of Kirschenbaum's analysis, but a sign that we must discard the romantic notion of the lone scholar toiling away in a vacuum and work with others across disciplinary boundaries.

A Call to Action

Is a technical understanding truly necessary for the study of games? The answer to this question was provided early on, barely one year after Aarseth's declaration of Year One of game studies (Aarseth 2001). Lars Konzack (2002, p. 91) published a paper on game analysis methodology in which he wrote "Every computer game depends on code. Therefore program code is essential to the understanding of computer games." Konzack's observation is both concise and correct. And yet, there is very little work of this nature in game studies; only a handful of examples exist, like Willumsen tackling the source code of the game *Passage* (Willumsen, 2017). Some possible reasons are outlined in a disarmingly humble excerpt from Konzack's paper (2002, p. 92), as he concedes that he did not follow his own methodology with respect to the code of the game he was studying, "partly because I did not know how to access the code, and partly because I would have [a] very difficult time figuring out how the code worked."

8 I acknowledge that many people in, say, the humanities now write programs, but viewing a technical field like computer science solely in terms of writing programs is a reductive view of the discipline.

Game studies seems to have addressed the technical gap by foisting responsibility for technical matters to other areas that intersect with game studies, specifically platform studies. Fernández-Vara, for example, does this very overtly when talking about the technical context of a game (2024, p. 88). This is not a problem *prima facie*, but the reality is that platform studies work varies widely in terms of the level of technical detail provided, and even then the peritext can reveal limitations.

Consider the platform studies book on the Nintendo Entertainment System (Altice, 2015), one of the platform studies books that does not shy away from technical detail. In one chapter, the endnotes mention that the reference used was not the original low-level code and data of the game in question, but a third-party analysis of it; a later endnote expressed gratitude to the author of that third-party analysis for emails that "explained in exhaustive detail how the byte encoding worked" (Altice, 2015, p. 369). In other words, not only was a secondary source used as reference, but the technical aspects of it required explanation to the author. A similar theme appears in the acknowledgments of the platform studies book on the Intellivision, where four undergraduate students were thanked whose "knowledge of coding and assembly languages helped us develop the discussions of videogame code in this book" (Boellstorff & Soderman 2024, p. xiv). One could argue that this need to rely on external expertise had no effect on the eventual product, but without engaging with the primary sources, without being *able* to engage with the primary sources, how could this be known? I also note that despite the essential nature of code to games (per Konzack), the people with this type of technical knowledge are clearly subordinate in these arrangements.

Again, this *Mystery House* close reading of a close reading points to a solution. I do not advocate that game studies scholars must gain deep technical expertise in addition to the knowledge they already have. However, game studies is known for its interdisciplinarity, and this must extend beyond the humanities: one can engage researchers from other disciplines who already possess technical knowledge, like computer science, and work together as equals on the complex subject matter of games. Game studies scholarship cannot simply recognize the diverse perspective of technology, game studies must be inclusive of it.

Acknowledgments

Thanks to Patrick Finn for valuable discussions on this material, and Greg Hagen and Carl Therrien for commenting on various drafts of this paper.

References

Aarseth, E. (2001). Computer Game Studies, Year One. *Game Studies*, *1*(1), https://gamestudies.org/0101/editorial.html

Altice, N. (2015). *I Am Error: The Nintendo Family Computer/Entertainment System Platform*. Cambridge, MA: MIT Press.

Apple Computer. (1978). *Applesoft BASIC Programming Reference Manual*. Cupertino, CA: Apple Computer, Inc.

Apple Computer. (1981). *Apple II, The DOS Manual: Disk Operating System*. Cupertino, CA: Apple Computer, Inc.

Apple Computer. (1985). *Apple IIe Technical Reference Manual*. Reading, MA: Addison-Wesley.

Aycock, J., & Biittner, K. 2019. Inspecting the Foundation of *Mystery House*. *Journal of Contemporary Archaeology* 6(2), pp. 183–205.

Boellstorff, T., & Soderman, B. (2024). *Intellivision: How A Videogame System Battled Atari and Almost Bankrupted Barbie*. Cambridge, MA: MIT Press.

Bizzocchi, J., & Tanenbaum, J. (2011). Well Read: Applying Close Reading Techniques to Gameplay Experiences. In D. Davidson (Ed.), *Well Played 3.0: Video Games, Value and Meaning* (pp. 289–315), ETC Press.

Black, M. L. (2012). Narrative and Spatial Form in Digital Media: A Platform Study of the SCUMM Engine and Ron Gilbert's The Secret of Monkey Island. *Games and Culture*, *7*(3), 209–237.

Carr, D. (2019). Methodology, Representation, and Games. *Games and Culture 14*(7–8), 707–723.

Danemark, C. (2004). Display Sector. [Help information for FishWings 0.87]. Available via the Internet Archive's Wayback Machine: https://web.archive.org/web/20061205172803/http://mysite.verizon.net/charlie.d/fishwings.htm

Drucker, J. (2009). A Review of Matthew Kirschenbaum, *Mechanisms: New Media and the Forensic Imagination* Cambridge, MA and London, UK: MIT University Press, 2008. *Digital Humanities Quarterly*, *3*(2). http://digitalhumanities.org/dhq/vol/3/2/000048/000048.html.

Egenfeldt-Nielsen, S., Smith, J. H., & Tosca, S. P. (2024). *Understanding Video Games: The Essential Introduction* (5th ed.). New York, NY: Routledge.

Fernández-Vara, C. (2024). *Introduction to Game Analysis* (3rd ed.). New York, NY: Routledge.

Grigar, D. (2010). Matthew Kirschenbaum, *Mechanisms*. *Hyperrhiz* 7. doi:10.20415/hyp/007.r01

Hodges, J. A. (2019). Comparing born-digital artefacts using bibliographic archeology: a survey of Timothy Leary's published software (1985–1996). *Information Research*, *24*(2). https://informationr.net/ir/24-2/paper818.html

Kirschenbaum, M. (2008, January 14). Published! [Web log post]. Retrieved 6 March 2025 from http://mechanisms-book.blogspot.com/2008/01/testing.html

Kirschenbaum, M. G. (2012). *Mechanisms: New Media and the Forensic Imagination* (paperback edition). Cambridge, MA: MIT Press.

Konzack, L. (2002). Computer Game Criticism: A Method for Computer Game Analysis. In F. Mäyrä (Ed.), *Proceedings of Computer Games and Digital Cultures Conference* (pp. 89–100).

Levy, S. (1984). *Hackers: Heroes of the Computer Revolution*. New York: Dell.

Mäyrä, F. (2008). *An Introduction to Game Studies: Games in Culture*. Los Angeles, CA: SAGE.

Montfort, N. (2003). *Twisty Little Passages: An Approach to Interactive Fiction*. Cambridge, MA: MIT Press.

Nooney, L. (2023). *The Apple II Age: How the Computer Became Personal*. Chicago: University of Chicago Press.

Rankin, J. L. (2018). *A People's History of Computing in the United States*. Cambridge, MA: Harvard University Press.

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

Stang, S. (2022). Too close, too intimate, and too vulnerable: close reading methodology and the future of feminist game studies. *Critical Studies in Media Communication 39*(3), 230–238.

Sterling, B. (2009, January 31). Dead Media Beat: Kirschenbaum's Mechanisms book. Retrieved 5 March 2025 from https://www.wired.com/2009/01/dead-media-be-2-11/

Willumsen, E. C. (2017). Source Code and Formal Analysis: A Reading of Passage. *Transactions of the Digital Games Research Association*, *3*(2), 213–235.

Worth, D., & Lechner, P. (1981). *Beneath Apple DOS*. Reseda, CA: Quality Software.