

UC Santa Cruz

UC Santa Cruz Electronic Theses and Dissertations

Title

The Stabilization, Exploration, and Expression of Computer Game History

Permalink

<https://escholarship.org/uc/item/4rn402db>

Author

Kaltman, Eric

Publication Date

2017

Copyright Information

This work is made available under the terms of a Creative Commons Attribution-NoDerivatives License, available at <https://creativecommons.org/licenses/by-nd/4.0/>

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA
SANTA CRUZ

**THE STABILIZATION, EXPLORATION, AND EXPRESSION OF
COMPUTER GAME HISTORY**

A dissertation submitted in partial satisfaction of the
requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

by

Eric Kaltman

September 2017

The Dissertation of Eric Kaltman
is approved:

Noah Wardrip-Fruin, Chair

Michael Mateas

Henry Lowood

Tyrus Miller
Vice Provost and Dean of Graduate Studies

Copyright © by

Eric Kaltman

2017

Table of Contents

List of Figures	vi
List of Tables	viii
Abstract	ix
Dedication	xi
Acknowledgments	xii
1 Introduction	1
1.1 On the history of technology	4
1.2 On the history of software	10
1.3 On the history of computer games in the history of software in the history of technology	12
1.4 On preservation	13
1.5 On knowledge accumulation, exploration and expression in the history of technology	15
1.6 On an intermediary perspective for the history of games as software	20
1.7 Stabilization	25
1.8 Exploration	27
1.9 Expression	29
2 Appraisal	31
2.1 Compiling the Record	31
2.2 Appraisal	33
2.2.1 Related Work	38
2.3 <i>Prom Week</i>	41
2.3.1 Choice of <i>Prom Week</i>	42
2.3.2 Process	43
2.3.3 Context	57
2.3.4 Documentary Enumeration	62

2.4	Conclusion	79
3	Description	82
3.1	Introduction	82
3.1.1	A Brief on Controlled Vocabularies	84
3.1.2	A Course Through the Thicket	86
3.2	Controlled Vocabularies	90
3.2.1	A Brief Record Example	92
3.2.2	Vocabulary and Ontology Best Practices	97
3.2.3	“Aboutness” of Platform	106
3.2.4	Levels of Abstraction	110
3.2.5	Derivation of Terms	114
3.2.6	Format, Conceptualization and Reasonable Compatibility	117
3.2.7	Semantic Web Integration	122
3.2.8	Vocabularies in Institutional Practice	127
3.3	Future Work	143
3.4	Conclusion	146
4	Citation	149
4.1	The Pivot	149
4.2	Citation	152
4.2.1	Citation in Use	153
4.2.2	Citation as Discourse	155
4.3	Bibliography and Citation in Game Studies	161
4.3.1	Presupposition of DOOM!	165
4.4	Reduction and Intertextual Expression	171
4.5	Types and Examples of Reduction	175
4.5.1	Video	176
4.5.2	Visualization	177
4.5.3	Emulation	180
4.5.4	Closing	186
4.6	Back to Citation and Archives	188
4.7	A Tool for Descriptive and Manifest Citation of Games	190
4.7.1	Game v Performance	191
4.7.2	Citation Tool	193
4.7.3	Command Line	195
4.7.4	Web Application	198
4.7.5	Future Work	206
4.8	Evaluation	207
4.8.1	Discussion	208
4.8.2	Improvements and Future Work	211
4.9	Conclusion	213

5	Discovery	215
5.1	Intro	215
5.2	Game Discovery	217
	5.2.1 Forms of Discovery and Their Limitations	219
5.3	A Goal for Discovery	224
5.4	A Model for Discovery	226
	5.4.1 Related Work in NLP for Games	227
	5.4.2 Latent Semantic Analysis	233
5.5	Tools for Discovery	235
	5.5.1 Related Discovery Work	235
	5.5.2 GameNet	237
	5.5.3 GameSage	238
5.6	Visualizations for Discovery	239
	5.6.1 GameGlobs	240
	5.6.2 Gamespace	241
	5.6.3 GameTree	241
5.7	Evaluation	245
	5.7.1 Expert Evaluation	246
	5.7.2 Novice Game Designer Evaluation	249
5.8	Future Work	253
5.9	Conclusion	254
6	A Model of <i>Doom</i>	256
6.1	Introduction	256
6.2	Fractal History	260
6.3	A Model of Game Software Historical Study	265
6.4	Doom in Fractal Coherence	275
	6.4.1 Historiographable Target	278
	6.4.2 Reified Object	295
	6.4.3 Technical Expression	310
	6.4.4 Enacted and Tacit Knowledge	318
6.5	Conclusion	329
7	Conclusion	333
7.1	The Conditions of a History for Games and Software	335
7.2	The Network Contingency and its Implications on Practice	338
	7.2.1 Problems of the Network Contingency	340

List of Figures

1.1	Intermediary Accumulation	16
1.2	Intermediary Exploration and Expression	18
1.3	Full Historical Process	19
1.4	Mapped Historical Process	21
3.1	Platform Abstraction Diagram	111
3.2	Basic Simple Knowledge Organization System (SKOS) Diagram .	124
3.3	SKOS Terminology Hierarchy	125
3.4	Region and Version Compatibility	126
3.5	Information Sheet Example	145
4.1	GISST components and pipeline.	194
4.2	Basic CLI Pipeline	197
4.3	Data Flow to GISST Indexer	198
4.4	Network Diagram for Loading a Game into the Indexer	201
4.5	Network Diagram for Loading a Save State into the Indexer . . .	202
4.6	Network Diagram for Saving a Save State.	202
4.7	Network Diagram for Video Recording	203
4.8	Indexer User Interface	204
5.1	GameNet Search and Results for Wall Street Kid	237
5.2	GameSage Query for Non-Corpus Game	238
5.3	GameGlobs Showing 20 Clusters	242
5.4	GameSpace Intro Screen and Main Space	243

5.5	GameSpace Game Selection with Wikipedia and Youtube	244
5.6	GameTree with Racing Game Branch Highlighted	245
6.1	Two Versions of Historical Analysis	265
6.2	Basic Software Historical Model in Four Layers	268
6.3	“T” of Omissions in Historical Model	273
6.4	Opening of <i>Doom</i>	279
6.5	Opening of <i>Hovertank 3D</i>	284
6.6	Opening of <i>Catacomb 3D</i>	286
6.7	Opening of <i>Wolfenstein 3D</i>	287
6.8	Version Tree of Doom	302
7.1	Growth of Network Contingent Games 1950 to Now	343
7.2	Growth of Network Contingent Games Into Future	343
7.3	Network Distribution 1950 to Now	344
7.4	Network Distribution Now Into Future	345

List of Tables

2.1	<i>Prom Week</i> File Formats and Dependent Programs	68
3.1	Star Raiders Entry from Cabrinety Collection Finding Aid	94
3.2	Aggregated Number of Potential Platforms and Formats Per Col- lection	116
3.3	Example Research Entry for a Platform (Before Disambiguation)	116
4.1	GISST Supported Resources	198
6.1	Comparison of id Games' File Structures	298
6.2	Comparison of Doom Image Contents 0.2 to 1.25	307
6.3	Comparison of Doom Image Contents 1.4 to The Ultimate Doom	308

Abstract

The Stabilization, Exploration, and Expression of Computer Game History

by

Eric Kaltman

Computer games are now a significant cultural phenomenon, and a significant artistic output of humanity. However, little effort and attention have been paid to how the medium of games and interactive software developed, and even less to the historical storage of software development documentation. This thesis borrows methodologies and practices from computer science, the history of science and technology, and information science, and brings them to bear on the historical study of computer games. It posits that in order to understand and reconcile the place and effects of cultural software in society, new means of stabilizing software outputs, exploring their contents, and expressing their histories must be created.

The thesis's contributions are tied to a model of scholarly process, in which software practitioners — in this case specifically game developers — produce documentation that is then explored by historians and expressed — through scholarly works — to other scholars and lay audiences. That is, the accumulation of historical records about games and software needs to be filtered through stabilizing processes before historians can make use of them. Furthermore, the subsequent expression of those records needs to take their digital, computational, and technically engineered nature into account.

The first three chapters focus on the appraisal of game development documentation, the description of game records in institutional archives, and the citation and reference of game resources. “Appraisal” analyzes the records of the game *Prom Week* and provides an appraisal and institutional ingestion its development documentation. “Description” presents semantic controlled vocabularies

and theoretical models for computer game platforms and media formats. “Citation” introduces the Game and Interactive Software Scholarship Toolkit (GISST), a system for the citation of emulated computer games, their computational state, and recordings of their game play. GISST allows for embedding emulated games into online documents to aid in historical expression.

The two latter chapters focus on record expression. “Discovery” presents a novel application of natural language processing to the visualization of computer game history. The final chapter focuses on the history of the computer game *Doom* and the need to devote more time to the organization and historicization of its documentation.

To Agi for keeping me sane, and Elliott for keeping it soft.

Acknowledgments

A lot of glorious people contributed to the success of this dissertation. Thanks to Noah Wardrip-Fruin, Michael Mateas and Henry Lowood for advising on the drafts and seemingly unending support for every question and issue — no matter how small — I had along the way. This work would not have been possible without the input of the Game Metadata and Citation Project team — Marcia Barrett, Christy Caldwell, Greta deGroat, and Glynn Edwards. Further support for GAMECIP efforts by Rachel Jaffe, Gloriana St. Clair, Simon Carless, Kari Kraus, James Newman, Jin Ha Lee, David Gibson, and Mark Nelson.

A significant amount of collaborative effort went into the creation of various contributions in this work. I'd like to thank Joe Osborn — for his work on GISST, James Ryan — for his work on the NLP models and ideas underpinning the visualizations, and Stacey Mason — for help in conceptualizing game citation practices. Additionally, thanks to Mitch Mastroni for significant efforts on the controlled vocabularies. Also my undergraduate researchers: Timothy Hong, Salvador Flamenco, Joshua Navarro, Malcolm Riley, Michael Harrold, Joseph Sandmeyer, Austin Yen, Sam Fields, Ryan Cori, Sergiy Ravnyago, Yasha Taylor, and Neeraj Mallampet. And thanks to graduate researchers Molly Jostock, Vanessa Nutter, and Nina Acosta. Input and feedback on this thesis was also solicited (and gratuitously provided by) Chaim Gingold, Jacob Garbe, William Huber, Miguel Sicart, Dylan Lederle-Ensign, Douglas Wilson, Nathan Altice, Katherine Isbister, Fox Harrell, and Warren Sack.

Finally I'd like to thank my partner Agnieszka, for providing much needed love, support and encouragement in getting this thing out the door; the soon-to-arrive Algernon, for providing significant motivation without even knowing it; and you, whoever you are, for choosing to spend a small part of your sacred and

limited intellectual life with me and my ideas. Thank you all.

Chapter 1

Introduction

In Henry Lowood’s and Raiford Guins’s edited volume *Debugging Game History*, the pair lay out the current state of the history of computer games and the needs for its continued development.¹ They quote Jeffrey Yost — editor of the IEEE Annals of the History of Computing — from his preface to a special issue on games, “Little critical historical analysis has been written on computer games to date. Much of the existing literature is blindly celebratory, or merely descriptive rather than scholarly or analytical. Only a small number of scholars have undertaken rigorous analysis of computer games.”² Lowood and Guins remark that “fives years later [in 2016], the situation remains the same,” and further enlist Erkki Huhtamo who describes game historical study as stuck in “a mode of writing history consumed with the ‘when’ and ‘what’ to the detriment of the ‘why’ and ‘how’.”³ Apparently, there is much work to be done to ground out the history of games through the more rigorous practices of critical historical studies.

Lowood and Guins call for more work in a number of areas, but this thesis is

¹[86] Guins, Raiford, and Henry Lowood, eds. *Debugging Game History: A Critical Lexicon*. Game Histories. Cambridge, Massachusetts: The MIT Press, 2016.

²[86] pg.xiii

³[86] pg. xiii

primarily concerned with three:⁴

1. “Multidisciplinary methodological and theoretical approaches to the historical study of games.”
2. “Game preservation, exhibition, and documentation, including the place of museums, libraries, and collectors in preparing game history.”
3. “Material histories of game artifacts and ephemera.”

In calling up these topics — each a staggering vessel that well exceeds the volume of this humble thesis — we limit our scope directly to software-based games. This focus gives us a methodological toehold in the history of software, and allows us a more limited object of study. Therefore, each entry above is slightly altered, replacing “games” generally with “game software” specifically. While this may seem trivial, “games” and software that manifest “games” are two very different things and ones that appear conflated in many texts in game studies.⁵ But game software and its development deserve a deeper and more critical look, one that falls in line with and augments the history of software in general.

Above, two specific phrases passed without question. The first, in Jeffery Yost’s, about the existing literature on games being “merely descriptive,” and the second, in Huhtamo’s assertion that the “when” and “what” of games is detrimental to the “why” and “how”. Both statements implicitly assume that the

⁴[86] pg. xii

⁵Part of this linkage can be traced back to the birth of game studies as a field, which connected the field’s criticality to early studies of physical games and play. Salen and Zimmerman’s *Rules of Play* [205] and *Game Design Reader* [206], are an early example of the linkage between earlier board games and play to the practices of computer and software based game play. This lineage usually begins with the work of Johan Huizinga [93] and works through other play theorists — like Roger Caillois [47] and Brian Sutton-Smith [201] — before arriving at computer games. For a significant overview of play theorists and how that influences the playful design of games, [79] Gingold, Chaim Ophir. “Play Design.” eScholarship, January 1, 2016. <http://escholarship.org/uc/item/8qr533m2>.

description of games, their “what”, are good and solved, and that consideration of the “what” somehow limits history. This thesis stands as an argument that, for game software at least, the “what” has not really been attended to, and that to pass off work as “merely descriptive” is to gloss over fundamental material concerns that influence and, in some cases, inhibit the study of game software. In order to move forward with the history of games software, we need a more stable base of “what” and “when” to articulate a “how” in search of a “why”.

This thesis is then a call for a deeper consideration of the stabilization of historical material records of computer games, and the exploration and expression of their histories. With stabilization, we address the organization and linking of historical documentary accumulations that allow for coherent historical studies of games. With exploration and expression, we create new historical models, tools and visualizations that aim to reveal computer games — as constructed technical objects — in new ways. Stabilizing records — with the help of archival and library science — and exploring them through computational mediatic interventions then point toward new means of historical expression, which could advance the fields of game studies and software studies, along with the more general history of software. Our focus on computer games also benefits from the significant attention that they have received, both historically and intellectually as a class of software. Computer games as an expressive medium have received the lion’s share of scholarly attention afforded to software in general. However, this attention has resulted in a partial erasure of the foundational, computational and technical nature of computer games, as they are most usually discussed in abstraction, as a medium, writ large, to be reckoned with or defined in totality. This work aims to directly account for the material conditions of the history of game software, and explores how to leverage computer games as objects of constructed technology in

future historical investigations.

The remainder of this introduction lays out a deeper elaboration of concepts and substantial recommendations from the fields of the history of technology, the history of software, and record preservation as they relate to a unified model for the stabilization, exploration and expression of computer game history. We seek to point out how the work of this thesis addresses foundational issues in the larger fields of science and technology studies, and to show how our contributions light up some methodological crevices not often addressed in historical work. This introductory section concludes with a summary of the chapters of this thesis and a brief description of each of their contributions and goals.

1.1 On the history of technology

In his 1996 presidential address to the Society of the History of Technology (SHOT) Alex Roland reflected on the history of technology in light of the passing of Mel Kranzberg, co-founder of the society.⁶ The address focused on the validity of the field, and its relation to the perceived “black box of technology.”

Does the history of technology matter? If so, how? What is it that we as a community do with the black box of technology? Do we really unpack its contents? Can we claim to have produced any special insight into the nature of black boxes, how they work, whence they come, and how they interact with their environment?⁷

The “black box” here is the terminus of a long process of engineering and design work that results in a technological object.⁸ This object is stable enough to be

⁶[178] Roland, Alex. “What Hath Kranzberg Wrought? Or, Does the History of Technology Matter?” *Technology and Culture* 38, no. 3 (July 1997): 697. doi:10.2307/3106860.

⁷[178] pg. 701

⁸This reference to “black boxes” is focused on technical objects and what their inner workings reveal about their place in society and culture in addition to revealing the practices of their creators. Others use the metaphor of “black boxes” to deal with technocratic phenomenon at

used without knowledge or consideration of its inner workings. A common trope (or concern) in the history of technology is to what degree the inner workings of the object matter. “Do we really unpack its contents?” is it really worth the time and effort to disassemble the black box? What do we gain when we do? Roland hopes for some “special insight” into the nature of technology, some way to point out how the inquiry revealed some new piece of the puzzle of reality.

His address grapples with the extent to which one should even bother with the insides of technology, and splits historians of technology into two rough camps,

Some historians treat the black box as a *machina ex deus*. It appears in history as a given. The important question for them is... what it does, how it influences its environment. To look inside is to invite confusion and distraction... But others, staring into the black box, become transfixed and beguiled. No locknut is too mundane, no gear too trivial. All are extracted from the box, and paraded in loving detail before the reader, in the historian’s equivalent of an exploded drawing. Failing to explain what it all means, these historians simply explain how it all works. While the former type lacks trees, the latter type lacks a forest. Neither is satisfactory.⁹

This tension is present in his work, he admits, and in the work of most historians of technology. On the one hand, technology functions in culture, interacting with and influencing people. On the other, technology is the result of people’s

different levels of abstraction. See Bruno Latour and Steve Woolgar’s *Laboratory Life*, and Latour’s *Science in Action* for discussions of “black boxes” of scientific belief and how they function in networks of societal influences (this ties to the larger project of Actor-Network Theory as developed by Callon, Latour, Law, Rip, and others). Nathan Rosenberg also uses the metaphor of the “black boxes” of macroeconomic theory, and how belief in those models structures economic reality. In all cases, the basic premise is the same, that closure over technical models and beliefs serves to partially obscure the foundational assumptions supporting modern society. See, [48] Callon, Michel, John Law, and Arie Rip. *Mapping the Dynamics of Science and Technology: Sociology of Science in the Real World*. Basingstoke: Macmillan, 1986. [118] Latour, Bruno. *Science in Action: How to Follow Scientists and Engineers through Society*. Cambridge, Mass.: Harvard University Press, 1987. [119] Latour, Bruno, and Steve Woolgar. *Laboratory Life: The Construction of Scientific Facts*. Princeton University Press, 2013. [182] Rosenberg, Nathan. *Exploring the Black Box: Technology, Economics, and History*. Cambridge [England]; New York: Cambridge University Press, 1994.

⁹[178] pg. 703-704

effort, and it is necessary to understand how those efforts lead to the object of discussion. A history of a technology without some discussion of technology itself would not present a coherent picture.¹⁰ Different histories of technology seemingly require a modulation between the desire to fully explain how something worked and the contexts within which it functioned.

This concern extends to the history of software technology as well. James Tomayko, directly echoing Roland’s remarks above, highlights the different approaches he used in two complementary works on software engineering, *Computers in Space Flight: the NASA Experience* and *Computers Take Flight: A History of NASA’s Pioneering Fly-by-Wire Project*.¹¹ Tomayko refers to two different “levels of sharing”, which are roughly equivalent to Roland’s “tree” of technical detail and “forest” of technical context. In the earlier work, *Computers in Space Flight*, Tomayko went into more technical detail than his editors believed relevant or necessary. They wanted a more institutional history and as a result, “relegated the exposition of technical details of the engineering history to sidebars and appendices.”¹² In *Computers Take Flight*, Tomayko’s editor again asked for “judicious” use of technical details, and in this case Tomayko agreed stating that, “understanding the technology led to (hopefully) a better story, but its development is not the dominant part, even though it is the central reason for the project.”¹³

Tomayko maintains that the important consideration is not whether there is

¹⁰See the chapter “A Model of *Doom*” below for a discussion of historical narrative “coherence” and where technological records fit within the structure of computer game software history.

¹¹[208] Tomayko, J. E. *Computers Take Flight: A History of NASA’s Pioneering Digital Fly-by-Wire Project*. NASA History Series 2000-4224. Washington, D.C: National Aeronautics and Space Administration, NASA Office of Policy and Plans, NASA History Office, 2000. [209] Tomayko, J. E, United States, National Aeronautics and Space Administration, and Scientific and Technical Information Division. *Computers in Spaceflight: The NASA Experience*. Washington, D.C.: National Aeronautics and Space Administration, Scientific and Technical Information Division, 1988.

¹²[210] Tomayko, James E. “Software as Engineering.” In *History of Computing: Software Issues*, 65-76. Berlin; Heidelberg; New York: Springer-Verlag, 2002. pg. 75

¹³[210] pg. 75

too much or too little technical detail, but whether the history constructed around those details is “interesting and useful to the practitioner and layman alike.” He also notes that although, “practitioners have produced much of the history of computing since they easily understand the technology and are motivated to set down their own stories. They make prevalent the overly-technical history. What is needed are more trained historians to enter this area of the history of technology.”¹⁴ Tomayko highlights two major considerations for the history of software in the history of technology. First, a purpose for these histories is to appeal to both people inside their construction (practitioners) and those outside who experience their effects (laymen). And second, that the construction of histories of software needs dedicated consideration by historians, or more specifically, historical methodologies and perspectives.

Briefly setting aside the first point of appeal to practitioners and laymen, Albert Endres in his response to Tomayko’s call for more history notes, “most private knowledge will eventually be documented not by the designers and developers themselves, but by people who are willing to study designs and compare them.”¹⁵ Endres appears at odds with Tomayko’s statement about practitioners as a source for the history of computing, but Endres’s point embeds a more subtle distinction. Recall that Tomayko highlights practitioner’s desire “to set down their own stories,” as a major motivation for self-documentation practice. As such, documentary details do not focus on greater historical trends nor on comparative structures, but a bit too tightly on the practitioners’ minutiae and areas of expertise (which are most likely not historical disciplines). Endres is acknowledging that practitioner directed history might not display “private knowledge” in ways intelligible to future historians. Practitioners understand their practice intimately, but

¹⁴[210] pg. 76

¹⁵[71] Endres, Albert. “Commentary on James E. Tomayko.” In *History of Computing: Software Issues*, 77-82. Berlin; Heidelberg; New York: Springer-Verlag, 2002. pg. 81

may not know how to distinguish which parts of that practice are approachable for outsiders as opposed to ones steeped in complex networks of tacit knowledge.¹⁶ Therefore, work needs to be done to document software systems and their development to allow for more comparison between engineering techniques and their resultant software objects. And though practitioner self-documentation practice is valuable,¹⁷ Endres is correct in that a majority of the documentary record will be collected by those “willing to study designs and compare them.” Documentary collection is implicit here since you obviously need records of a design to study it. Documentation lends to comparison of forms and designs, which leads to a particular mode of historical construction. Given that many practitioners are not also practicing archivists or historians, documentation of software design practice is rarely readily available or well-organized. A major point of the work below is to fill some of the methodological lacunae in the documentary record of computer game and software history.

Interestingly, Endres further notes that “there are not nearly enough researchers doing this type of work [documentation and comparison] in the field of software engineering,” and that “they will be welcomed by both practitioners and historians.”¹⁸ This implies an intermediary position between practice and history for individuals interested in documentation and comparison. We believe that this intermediary functionary need not be an individual (or even human), but that consideration must be made for how the construction of the history of a technology is built on the collection, exploration and expression of its documentary

¹⁶For more on tacit knowledge and its influence on technological and scientific knowledge sharing, see [54] Collins, Harry. *Changing Order: Replication and Induction in Scientific Practice*. Chicago: University Of Chicago Press, 1992. and [55] Collins, Harry. *Tacit and Explicit Knowledge*. Chicago; London: University Of Chicago Press, 2012.

¹⁷See Chapter 2, “Appraisal,” below for in-depth discussion of development team’s self-documentary material, and Chapter 6, “A Model of *Doom*,” for descriptions of game community developers and their documentation practices.

¹⁸[71] pg. 81

record. Endres believes that more intermediary research is needed so that we “do not lump all software together and treat it as a single phenomenon.”¹⁹ Consideration of software objects needs to be more critical about the nature of their technical ontology and construction. Otherwise you get “people who have written one type of program [declaring] themselves experts on all types.”²⁰ The history of software must then admit to specializations based on the constitution and construction of software objects. One way to bring those specializations to the fore is through the intermediary work of documentation and comparison, which supplies the “stuff” for later, more specific historical methodologies.

Tomayko’s call for more historians is then based on availability of Endres’s intermediary class of documentarians. These intermediaries are most likely also historians, but they have an awareness of the needs for a better documentary record for the history of software. In order to make the history of software apparent to a more diverse and larger audience, more accounting needs to be made of the collection and dissemination of its material record. This notion of a thorough accumulation of material history loops us back to Roland’s address on the purpose of the history of technology more generally.

Roland began his inquiry into the worth of the history of technology with a lamentation of the confusion inherent in its practice. Opening up black boxes and tying their contents to larger revelations about the nature of technology is difficult, in no small part because of the constant pressure to explain how a technology works instead of what it does for history. Roland admits that there is no truly consistent methodology in the history of technology — a point that might bother Tomayko and his call for more method in relation to software study — but asserts that the strength of the discipline lies in its ability to challenge common

¹⁹[71] pg. 81

²⁰[71] pg. 81

conceptions of the nature of technological change. This strength “was achieved by building up a knowledge base about the history of technological change and sharing that knowledge base with one another.”²¹ The accumulation and sharing of knowledge inherent in the historical study of technology is then both its purpose and legitimation. This point mirrors those made about the history of software by Endres and Tomayko in calling for more historical resources and argumentation about software systems.

Roland ends his address with an appeal for those present to work toward more awareness and representation of the field. Again, reflecting Tomayko’s sentiments towards software history, Roland rallies, “we must step back from our own shared understanding that technology matters and produce scholarship to convince unbelievers that technology matters.”²² There is then nothing wrong, inherently, with the historical inquiry into technology and its functions, but there is much work to be done expressing that history and its importance to others.

1.2 On the history of software

The work of software creation is still mostly invisible or unaccounted for in many works in the history of games. When games are based on software and development processes, those functions are generally swept under the rug. Sometimes because the scholar does not have the technical skills to recover the object’s development record and in others because the game as a technical object is not considered germane to its existence as a “game,” the subject of historical analysis. Many times, however, it’s because no evidence of development, of the systems underlying the game or of the existence of the software object remain. These issues

²¹[178] pg. 711

²²[178] pg. 713

in game software studies are, in fact, inherited from the history of software.

Michael Mahoney, twenty years ago, in his presentation at the second History of Programming Languages Conference, directly addressed the need for more evidence of practice in the history of computing. A major problem with recovering practice is that “it is a skill, a body of techniques, and the habits of thought that go with them, that constitutes effective knowledge of a subject. Because all practitioners of a subject share that skill, they do not talk about it.”²³ Those working with software, or working on games, develop a large, inter and intra-personal corpus of implicit and tacit knowledge. They draw on it in application to new problems and rarely write it down or critically reflect on it. Mahoney states that as a result of this deficit of engineering knowledge, “historians are learning to do what engineers often take for granted: to read the products of practice in critical ways.” They must “learn to read the artifacts as critically as they do the [written] records,” and by extension learn to explore those records in ways outside the methods of traditional historiography. Mahoney continues the discussion by looping in the need for access to the artifacts of computation. One needs to be able to make the object available in order for others — or even themselves — to study it. This is not a trivial consideration for historical software since “what programs do and what the documentation says they do are not always the same things.” Historians, in addition to inheriting the engineering knowledge needed to comprehend software, must “inherit . . . the problems of software maintenance,” as well. For “the farther the program lies from its creators, the more difficult it is to discern in its architecture and the design decisions that inform it.”²⁴

Therefore, in understanding the history of game software, we need to gain an intimate understanding of its practice, recover the software artifacts that consti-

²³[133] Mahoney, Michael S. “Issues in the History of Computing.” In *History of Programming languages-II*, 772-781. ACM, 1996, pg. 774

²⁴[133] pg. 775

tute it, maintain them in a way that they can inform future historical work, and get out alive afterward. Mahoney illuminates these core issues in the history of computing — though directed at the history of software — and then leaves them lying there. He hopes, we guess, that others will pick them up and do the rigorous work necessary to support the critical exercise of software history.

1.3 On the history of computer games in the history of software in the history of technology

This thesis is mainly directed toward mapping the larger issues in the history of technology — documentary stabilization, expression and relevance — to the history of computer games as software technology. The consideration of computer games as a unique technical sub-discipline in the history of software is currently in its infancy. This is partly due to the general issues of acceptance of computer games as a legitimate form of cultural expression, but more significantly due to the fact that computer games are software, and that software itself is significantly under-researched. As discussed in the previous section, the ability to conduct historical investigation of a set of technologies is intimately tied to one's access to the material traces of its development. This thesis is a step towards a larger discussion of a methodology for the history of games as software, and how the exploration and expression of that history is a unique specialization in the history of software technology.

Locating the history of computer games in this way — as explicitly software-based objects — is intentionally narrow. Much of the discussion of games as phenomena includes detailed accounts of how rules systems express play, and how that play links up to larger social processes and structures. Almost all of the

discussion of the technical history of computer game software is written by practitioners reflecting on their practice. Thus we are in a similar position to that of the history of software engineering discussed by Tomayko above. Practitioners focus on the dissemination of their specific knowledge, and rarely pair their exposition with comparative historical practice or methodology. In many cases, they also ignore how their exposition could help the construction of a future history of games. This is because much of what is written about technique and design is directed at other practitioners and not larger historical themes, context, or lay understanding and introduction.

There needs to be a greater accumulation of records of design and implementation, and a greater awareness of the potential of intermediary work in the collection of documentation and comparative frameworks for games. To form a history of computer games as software technology we need to take stock of what is available for study, what should be available for study, and ways to expose that material to historical analysis.

1.4 On preservation

One major implication of intermediary documentary research into game software is the complementary considerations it creates for that research's preservation. Absent from much of the discussion in the history of technology is the actual process of maintaining and organizing the records of technological process — what we have referred to as stabilization. Documentation of technical artifacts requires explicit description of their ontological organization, and how their existence is tied to the material evidence of their creation. Once description and positioning are solved, then comes the desire to recall that evidence for historical study, to locate it within the documentary accumulation and get it to speak for a historical

argument. These preservation processes are, in fact, common to most types of historical records. Our work here is again a mapping of the issues of other fields onto the history of computer game software. In the last case that field was the history of technology, in this case archival preservation.

For the historical record of computer games to become a stable basis for the history of computer games, certain preservation processes must be suitably re-oriented. We will address these processes in roughly chronological order. When collecting the records of game software and its development (this includes records that constitute the object itself) appraisal and description come to the fore. Appraisal is the process of deciding which records are relevant to future researchers, and is the first screen through which documentation must pass on its way to the historical record. Description is necessary to simply label the parts of each record in a way that is acceptable to the field in question, and that comports with standard understandings of the ontology of each object. Description is important because it allows for the future location of records in an archive. Additionally, it is hoped that the appraisal decisions are correct in anticipating future scholarship, and that the description conforms to an ontology stable enough as to be understood in the future (see the Appraisal and Description chapters for further discussion).

Following description is the act of storage that places the record in a stable location for future reclamation. This storage involves a complimentary set of descriptions (all descriptions mentioned are in effect metadata) that label a record for inclusion into whatever repository schema is used by a particular institution. After storage, comes access or the means provided to actually get the record back out of the archive. Access involves both the location of relevant materials — the process of “discovery” — and their linking to scholarly expressions — reference

and citation.

These processes: appraisal, description, storage, and access, are fundamental to the preservation of any history, and are underdeveloped not only for computer game software (this thesis being one of first examples of research in this area) but also for records of the history of software and, surprisingly, even technology in general. As will become apparent, issues relating to the history of computer software as technological objects are in most cases directly derivative of — and therefore can become a direct commentary on — issues in the history of technology. Most of the preservation practice discussed in the initial sections of this work is directly linked to practice involving all engineered technology, albeit with more focus on issues of software preservation.

The processes of the material archive then invite both a discussion of the kind of records needed for the construction of a technical game history, and of how the ontology of those artifacts is germane to their storage and retrieval. They also call for a consideration of what “access” means in relation to a software record (especially the software object itself), and how technical considerations of software as a unique phenomena shape capabilities and expectations of the archive. Lastly, these processes also influence the history of games as software because development and technical records are the basis for arguments about games as technology.

1.5 On knowledge accumulation, exploration and expression in the history of technology

From the above we have gathered that a purpose for the history of technology is to negotiate the revelations of its inner workings (the amount of the box to

open) towards a better conception of technological change and how that change has influenced people’s lives. The positive output of this research, as summarized by Roland in his address to SHOT, is first, the accumulation of knowledge about technology and the ability to share and build upon that knowledge with others and, second, to express that knowledge in a way that changes the perceptions of technology for both practitioners and lay people alike. After a further bit of elaboration, we will tie both outputs — knowledge accumulation and sharing, and historical exploration and expression — to the more limited domain of computer games as software.

The basis of knowledge accumulation is not only the practitioners’ work in creating a to-be-studied technology. It is also the result of the historian’s efforts in reconciling the material traces of the technology into revelatory discourse, and crucially, on the intermediary action of organizing, collecting, and maintaining those traces. These knowledge entities: practitioners, intermediaries, and historians, may frequently overlap in function or subsume one another. As mentioned above, a practitioner can engage in historical writing, or a historian can be the source of intermediary collection and management. However, we break them apart into three distinct entities in Figure 1.1 to elaborate on their relative positioning in the creation of knowledge about technology.

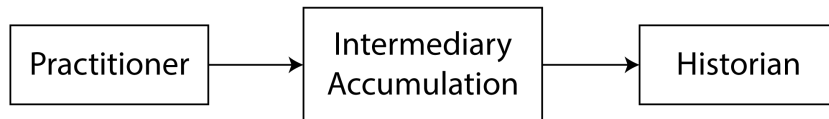


Figure 1.1: Intermediary Accumulation

The basic network in Figure 1.1 draws out the notion, described above by

Endres, that consideration must be given to the actual recording and management of technological production. Although some practitioners do take meticulous notes and organize their own personal archives, there are often times little understanding of how their production is tied to the creation of a historical discourse about the technology to which they devote their time and energy. Conversely, in the organization of technical historical discourse, much of the effort is dedicated to the use of sources (either found by the historian, or traced from others) and not on their organization, creation or navigation. That is, the methodology behind their management and means for its exploration. Roland even admits that the history of technology functions “not with any accepted methodology or focus.” Part of our work is to highlight how the intermediary actions involved in the creation of records of technology also force a discussion of methodologies for their management and use. Perhaps some discussion of methodology could illuminate ways to help direct intermediary knowledge accumulation towards places able to support new types of historical inquiries?

Conveniently, one large component of “intermediary knowledge accumulation” could refer to the organization of an archive. In that case, our discussions of preservation methodologies provide an opening to begin mapping the space of knowledge accumulation and to probe for potential holes and points of opportunity. There is a difference, however, between historians’ and practitioners’ tacit knowledge and its material traces. Our preservation discussions have to focus on the materials, since, sadly, an actual archive can only store material records. The intersection of previous archival practice with the constraints of software development and software objects provides the basis for the first sections of this thesis, Appraisal and Description. Appraisal of the types and kinds of traces implicated in the history of software, and the ontological conditions and resulting

commitments of their descriptions.

Returning to Roland's second point, regarding the expression of the history of technology to others, we can construct a symmetric intermediary diagram (shown in Figure 1.2). The expression hinges on the historian (or any entity concerned with some analytical frame for the past), their ability to locate and explore records, and their desire to present the history of a technology to an audience. Complementarily, after the historian encounters the accumulation, the act of exploring it and expressing their results also becomes a significant function. We carefully note that while expression is a general framing supplemented by an exploration of accumulation, it is also an intermediary process sitting between the historian and audience. Much like the intermediary work required collecting and interpreting the material traces between practitioner and historian, the work of expressing history to an audience is also an intermediary action involving the exploration of material and its representation.

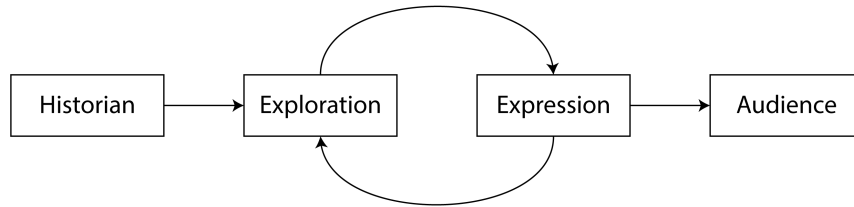


Figure 1.2: Intermediary Exploration and Expression

As with knowledge accumulation, intermediary exploration and expression are not usually discussed in the final technical historical output. It is mostly assumed that to communicate to an idea about the history of technology, one must write down some assertions, perhaps include a diagram or two, and with them construct a written discourse. This focus on the intermediary expression is not simply historiographical. There is plenty of discussion in the field about the ways to construct a historical discourse, and how those approaches are influenced by

the ideologies and subjective position of a particular historian.²⁵ There are historiographic points to be made, but intermediary exploration and expression are a bit more fundamental and material. Perhaps discursive practice is not the only way to communicate a historical argument. A consideration of new means of expression — and new ways to explore resources — might help with the appeal of technical history to wider audience and help make the specific history of software more approachable and comprehensible. We extend to the history of software specifically because of its proximity to our ultimate conversation about computer games. Additionally, we feel that consideration of the intermediary expression of software systems in particular is valuable.

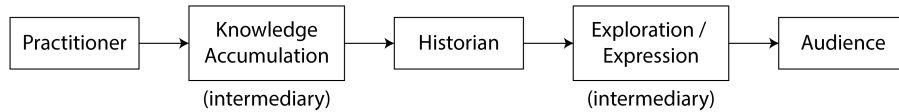


Figure 1.3: Full Historical Process

If we align both diagrams (Figure 1.3), we create a track for the creation and expression of knowledge about the history of technology. Beginning with the creative act of the practitioners, we proceed to the intermediary knowledge accumulation used by a historian to provide expression to a particular audience. While this track appears to be neat and tidy, it is also a bit of a lie. Rarely can one delineate all the intermediary actions and entities involved in a historical expression with such brutal clarity. Regardless, the basic arrangement is designed to point out two locations ripe for investigation and experimentation. Both intermediary nodes — knowledge accumulation, and exploration / expression — could benefit from active consideration by practitioners and historians.

²⁵See Chapter 6, “A Model of *Doom*,” for a deeper description of Hayden White and John Law’s work in this vein. Furthermore, Lowood and Guins call for game historical work at the beginning of this introduction is based, in part, on White’s historiographic analysis.

By looking more deeply into knowledge accumulation, we can identify the ways that practitioners, historians, and the archive of technical knowledge could be made to engage and interact in a more productive manner. Be it methodologies imported from other disciplines (information and library science), tools created to enable new forms of knowledge sharing and communication (computational media) or simply the contribution of a perspective on history that includes consideration of intermediary accumulation.

Complementarily, in examining the processes and objects of knowledge accumulation, we can then question how the historian organizes and explores them in the expression of historical argumentation. Our perspective therefore extends to intermediary expression, and engages with similar questions regarding the use of new disciplinary methods and computational tools. Hopefully, the result is a more thorough consideration of the intermediaries' effects on the history of technology and how embracing those effects can lead to improved scholarship and improved communication between practitioners, historians and audiences.

1.6 On an intermediary perspective for the history of games as software

Now that we have organized a basic perspective for the work of this thesis, we proceed to “ground it out” in an application of that perspective to computer games as software. The fundamental research contributions, to be elaborated below, concern a detailed look at how the intermediary actions of knowledge accumulation and expression function for the current state of the technical history of games, and what imported methodologies from preservation practice, and software engineering (primarily analysis tools) can provide to reveal and augment

those actions to the benefit of the field. Below in Figure 1.4, we map the model in Figure 1.3 of practitioners, historians, audience and intermediaries to the sub-domain of computer games. We then outline the tri-part structure of this thesis, of knowledge accumulation, expression and argument (not forgetting to illuminate specific research contributions). Finally, we conclude with brief summaries of the individual chapters and their goals.

Computer games as technical historical objects and their study, map directly onto the basic model above.



Figure 1.4: Mapped Historical Process

Missing from our general presentation of the model are detailed descriptions of the individuals, processes, or other expressive / knowledge entities implicated at each node. The general “practitioner” in the history of technology maps here to game designers and developers. Because we are focusing on computer games as a technology, there will be a bias towards the technique of game technology as opposed to the technique of game design. Game technology is the systems created to support a given game design. One is not clearly separated from the other (especially in cases where the design and technology derive from a single individual), but we will not concern ourselves with the creation of game rules, or what is good or bad design, but only with how the material records of the development of a game’s underlying technology effect that design. Much of this has to do with our perspective; we are concerned with the confluence of historical materials that lead to an expression of a historical argument. Those materials are important to the construction of a game’s design, and the history of game

design will benefit from our analysis, but the structure of a game as a conceptual object tied to games as an expressive art form is out of scope. How a game expresses its design is based on its development history, but the analysis of a closed game object within the field of game design is not a constitutive material trace in the way that, say, a designer’s development journal might be. As such, our designer / developer is an entity that produces material traces that can lead to an understanding of a completed game object, and that contributes to the basis of the intermediary knowledge accumulation about the processes of game development and technology. For the purposes of this thesis, we will refer to the “practitioner” as a “game developer” or “developer”. This is to align it more with the software development aspect of games than the game design aspect. Game developer also connects more easily to the broader category of “software developer,” since the technology we are discussing is software based.

As mentioned in the general model, intermediary knowledge accumulation is a set of practices that collect and organize the material traces of a technology. In many cases, this results in a collection in an archive, or some institutionally-backed repository of source material. In other cases, the records are diffuse throughout a larger network, found in the journals and writings of practitioners, or in the collections and commentaries of enthusiastic amateurs. The landscape of the documentation of computer game software is suitably vast and diverse, but it is also almost exclusively not in institutional repositories or the object of any current activity in the organization of archives. As will be elaborated through a case study in Appraisal, the records of software development present new challenges to knowledge accumulation about software as technology. Additionally, much of the current accumulation of knowledge about the construction of computer game software is based in practitioner-disseminated knowledge. There is very little in

the way of concerted, comparative studies of technological change in software in general, let alone games software. As concerns games, the intermediary accumulation is the current state of historical investigation. The history of games as technology is then in the form of a proto-discipline, there is much raw material and intermediary accumulation of resources, but not much effort dedicated to the revelation of larger themes. In the domain of computer game software, especially, there is little communication about the practice of locating and using records of games or their development. It is our belief that a more granular discussion of sources, the archive, and its relation to the construction of the history of computer game technology will reveal common strategies and means to augment the intermediary, and to better communication between game history and the archive.

Given that the history of computer games as technology is nascent, the intermediary connections to our model's "historian" might seem a bit premature. It is hoped that in presenting the findings in this thesis we can begin to assemble a discipline and define a basis for a history of technology of computer games. We are not saying that no one is doing any history involved with computer games as technical objects, but that there is not awareness of a standard of practice, a set of goals, or a stable set of foundational inquiries that usually constitute a discipline. There is not a Society for the History of Computer Game Technology (nor of Software Technology); there are only professional conferences and outputs dedicated to active, contemporaneous practice and discussion. What is needed is to account for the accumulation of the knowledge about the history of computer game software in ways that address the challenges associated with its investigation.

The historian in our model is also connected to the expression of technical history as well. In fact, since the historian is the means of translation for the ac-

cumulation through exploration to expression, they are in a sense defined by and constituted from both intermediary sets of resources and practices. A definition of a history for — and a historian of — computer games as software will then flow from the discussion of the means of intermediary accumulation and the constitution of intermediary exploration and expression. By “constitution” we mean the ways in which the evidence from intermediary accumulation can be expressed uniquely and productively for computer software history. This again leads back to potential tools, methods, or other technological interventions that help support and reinforce each intermediary is a way that benefits scholarly practice.

The potential audience is the last piece of the model and the most speculative. To whom is the historical expression directed and to what end? This was an issue for the historians of technology above. They had a desire for a greater understanding and appreciation for the history of technology, and they linked that desire to an active attempt to widen the reach of the field. Explicit discussion of who the audience was was not clarified beyond an appeal to both laymen and practitioners alike. For the history of games, those practitioners would be the game developers, and so that is part of the audience. The “lay” target for historical expression is merely any destination to which the expression can contribute a better, more nuanced understanding of the organization of game software systems, and help to remove some of the confusion surrounding the “black box” of technology that encapsulates game development practice.

The model we described in Figure 1.4 reflects the basic structure of this thesis, which is split into three sections, stabilization, exploration, and expression. These align with the two points of intermediary intervention identified in the model above. The first section, Stabilization, is a set of case studies in archival practice for computer games that seek to organize the discussion of what is present

in the intermediary space, and what methods might be used to help make it more accessible and interpretable to historians. Exploration, the second section deals primarily with a detailed look at a new way to explore the history of computer game software through explicit reference and sharing of computer games in citation, and further tools for exploration and discovery. It is less of an overview than the first section, since it drills down into particular examples of historical exploration, reference and information retrieval. The third section, Expression, combines the insights from the first two sections and applies them to a theoretical model of game historical study and its application to a case study of the material traces of the computer game *Doom*. It is a look at the structure of the history of *Doom* as a software object, made with an awareness of its accumulation, and the potential for its exploration.

Both intermediary interventions also align with the archival processes outlined above, and each chapter, in addition to contributing to the stabilization and exploration of the historical record, also contributes to fundamental knowledge management practices in appraisal, description (with a nod toward storage), and access — discovery and citation.

1.7 Stabilization

Our discussion of the accumulation and stabilization of computer games grounds out into a set of preservation case studies into the appraisal and description of the material traces of games. These case studies present a set of processes and methodologies needed for the articulation of computer game software in archives, and the ontological considerations that make them discoverable at a later point in time. This section includes two chapters, Appraisal and Description, dealing with issues in the intermediary accumulation of resources that form the basis of

the historical record of game software.

Appraisal presents a detailed account of the ingestion of the computer game *Prom Week* and its development records into the University of California’s Merritt digital repository. The appraisal strategy — the methodological process of diving the valuable and historically salient items in a to-be archived collection — and its procession are borrowed from similar studies aimed at preserving the records of science and technology. We attempt to map out the full material accumulation produced by *Prom Week*’s team over three years of development activity. This chapter illustrates what the knowledge accumulation looks like for even a modest game development effort. This includes discussion of how practitioners manage and create documentation, challenges encountered in dealing with highly varied documentary sites — like the “cloud”, web services, and development tools — along with an oral history of the game’s production as illuminated by its accumulated records. The chapter concludes with a prototypical methodology for the storage of game development records, and a call for more research into the stabilization of the variegated outputs of software development.

Description focuses on efforts to standardize the cataloging and recording of games in an institutional collection, and the challenges faced in attempting to fit game software into records management systems designed for static, non-executable printed media. We elaborate on the creation of a controlled vocabulary for the categorization of game platforms and media formats. This schema is intended to promote the recovery of records for practicing software historians through the concept of “reasonable compatibility” — that future researchers will want to recover software records that accurately reflect the general class of compatible hardware and software needed for a game to run. This invites a discussion of the material constraints of retrieving game objects, and an ontological

discussion of the meaning of “platform” and “media format” for future historical researchers. The chapter concludes with a significant discussion of the contribution the vocabularies represent vis-a-vis their incorporation into both the Library of Congress’s Source Code Listing, and the MARC21 data format for cataloging records.

1.8 Exploration

In following with the general model describing the progression from documentary creation through to historical audience, the next two chapters, Citation and Discovery, contribute to the intermediary exploration and expression of computer game history. The intent of these chapters is to pivot from a focus on the constitution of the accumulation of records about computer games, to means for the expression of those records. In both Citation and Discovery, we present new means to explore the records of computer game history through their visualization, citation, and the creation of computational tools that leverage them. This ties to new expressions in allowing for the presentation of new information in computer game history, game studies, and software studies argumentation — as made possible by the Citation chapter’s tools outlined below — and for new means for examining and theorizing historical records.

Citation describes the motivations for and development of a system for the management and creation of references to games, their performances, and their run-time states. The system, the Game and Interactive Software Scholarship Toolkit (GISST), began as a means to help with the standardization and organization of game citations. It evolved into a larger toolset that allows embedding of computer game run-times into an active web browser, and therefore directly into scholarly arguments about games. This chapter first addresses the value of

allowing for new types of discursive presentation in scholarly works by borrowing concepts from discourse analysis and the history of science and technology. It then proceeds to describe the GISST's functionality, and how GISST's existence points toward a new class of software devoted to the automation and augmentation of game historical, game studies, and software studies arguments. We also present an expert evaluation of the tools as a means to validate its potential for enabling new forms of expression for game historians and scholars, as well as providing further impetus for the creation of executable software archives.

Discovery attempts a computational intervention into the information science concept of "discovery" within institutional collections. In an inverse to the issues of stabilization in the first two chapters, discovery is concerned with how the users of archives locate relevant material inside of collections of accumulated records. In this chapter we describe a suite of tools for the location of related games in collections based on community descriptions. We organized a latent semantic analysis (LSA) model of games as described on Wikipedia and GameFAQs, and then expressed that model through a collection of information retrieval tools and visualizations:

1. GameNet: A title based search engine that takes a specific game and compares it to others in the model. This results in listing of the most and least related games to the input.
2. GameSage: A free text search engine that takes an input description for a hypothetical game, or a game that is not described, or simply where the name cannot be remembered, and computes a new vector that it then compares to known games in the search space.
3. GameSpace: A three-dimensional visualization that reduces the many dimensional vectors to three and plots them as stars in a galaxy of games.

4. GameGlobs: A two-dimensional visualization that uses k-means clustering to organize games into a fixed number of groups.
5. GameTree: A two-dimensional radial visualization that uses hierarchical agglomerative clusterings to create a full graphical “tree” of all the games in the corpus.

The discovery tools and visualizations represent a new way for historians and researchers to explore the historical record of computer games by leveraging amateur, community sourced information. A scholar can take a game they are researching and use its description as a prompt for an exploration of related titles. In light of potential benefits for discovery, this chapter follows the presentation of the model and its implementation with evaluations of its usage among game scholars, student game developers, and general Internet users. The goal is to show how the discovery tools open up new avenues of exploration for historical game records, and how visualizations can be useful in expressing new considerations about game history.

1.9 Expression

The final chapter of the thesis is a theoretical exploration of the ways that the accumulation of knowledge about computer games can be expressed through new types of historical studies. In focusing on a case study of the computer game *Doom*, we present a model for computer game study based on the various layers of documentary records that can accompany such an object. The model pulls from the theories of Hayden White, in historiography and historical narrative, and John Law, in the history of technology, to highlight four layers of inquiry available for interrogation by game scholars and historians. *Doom* is analyzed through

four small studies into its historiography, compiled object, technical expressions (source code and assets), and the enacted and tacit knowledge of its production. This model is meant to help illustrate the various ways that the stabilization of game development documentation and game objects can help in the revelation of tacit knowledge and technical process highlighted earlier in this introduction in regards to the “black boxes” of technology and calls for more technical studies in the history of technology.

We conclude this thesis with a call for more consideration of the needs of technical history, both in the stabilization of records from the variety of sources described in the Appraisal and *Doom* chapters and through a call for more consideration of the need for historical preservation of software to be embedded more deeply in the practice of game and software studies.

Chapter 2

Appraisal

2.1 Compiling the Record

“Compiling the record” is drawn from Michael Mahoney’s introduction to the second day of the second History of Programming Languages Conference. The agenda called for a discussion of the primary material record of computing, of archives, objects, and the stuff of history. “The record is not quite as complete as it looks,” he maintained, with much of the information about the history of computing, and specific in this case the history of software, “buried in corporate records” and consisting of “needles in huge haystacks.”¹ However, the compilation of records was also a minor consideration next to the basic question of how to use them to construct a history for software. “We have lots of answers but very few questions, lots of stories but no history, lots of things to do but no sense of how to do them or in what order,” Mahoney states before launching into potential answers.² Some of these answers form the basic impetus for this chapter, in which

¹[133] Mahoney, Michael S. “Issues in the History of Computing.” In *History of Programming languages-II*, 772-781. ACM, 1996. <http://dl.acm.org.oca.ucsc.edu/citation.cfm?id=1057839>. pg. 772

²[133] pg. 772

we explore the material record of computer games in hopes of creating a basis for their history.

Mahoney's major concerns are for the maintenance and informed construction of the software record. "We have to pick and choose what to keep in store and what we set out," in order to make collection and preservation "consciously selective." This conscious accumulation can then "anticipate the history [it is] supposed to generate and, thus, create that history."³ That the primary historical record dictates the potential histories drawn from it is obvious, but Mahoney's point moves a step deeper into how that record speaks and how we need to listen. "In deciding what to keep, it may help to understand that historians look at sources not only for what is new and unusual but also for what is so common as to be taken for granted . . . Because all practitioners of a subject share [similar skill], they do not talk about it. They take it for granted."⁴ The records of software development and practice then unintentionally resist historical analysis because much of the evidence for the "how" and "why" is hidden inside the implicit knowledge of practitioners. To combat this resistance, we need not only to understand what records we have available, but also the objects that they describe.

Mahoney illuminates two critical strategies for reclaiming software history. First, historians must be able to adopt the mindset of engineers and become able to "read the artifacts as critically as they do the records."⁵ And second, more emphasis must be placed on investigating and retrieving the context of software artifacts creation. This context primarily includes the bases of knowledge and training used at the time of development, as well as the environment in which the objects and records were created. Mahoney's recommendations to focus on recovering artifacts, implicit processes and contexts align almost perfectly with

³[133] pg. 773

⁴[133] pg. 774

⁵[133] pg. 774

early concerns of science and technology archivists. Most specifically, archival concerns over appraisal, the act of selecting and locating records. It is at this alignment between the concerns of historians and archivists of technology that we can introduce a further mapping to computer game records.

This chapter presents an in-depth analysis of the material traces of the game development through a case study of the academically produced computer game *Prom Week*. Below, we outline the basic archival procedure known as “appraisal”, and argue that methodologies developed for the organization of documentary records of science and technology can be equally suited to the concerns of the history of computer game software. In addressing the stabilization of computer game development records through our case study, we provide a methodological roadmap for other such studies into other classes of game software and software in general. Additionally, as the history of *Prom Week* is intimately tied to its documentary outputs, we gain insights into its development history simply in the description and discussion of its accumulated resources. This foreshadows further work in this thesis on the link between the accumulation of knowledge about the computer game *Doom*, and how its historical records inform on its possible histories.⁶

2.2 Appraisal

Appraisal is the process of divining what parts of the archival — and therefore permanent historical — record should be maintained for future historical study.

⁶A significant amount of the work in this chapter is based on “A Unified Approach to Preserving Cultural Software Objects and Their Development Histories,” a guide to the appraisal of computer game software produced for the National Endowment for the Humanities [102]. Most of the methodological discussion of the case study of *Prom Week* is drawn from that document. However, the treatment of *Prom Week*’s history in light of its documentation is new material derived from unused analysis from the NEH Report.

An appraisal surveys the records of a particular collection of material, and — based on assumptions about that materials future pertinence — decides what stays and what goes. As such, appraisal is the “core of all archival endeavours” as the starting point for the acquisition of records into a collection or archive, and as a locus for methodologies and perspectives on what is important for humanity to preserve.⁷ Appraisal defines the boundaries of the historical record, and the scope of further archival theory about the place and position of archives in society.

Appraisal, also, as an act of selection and stabilization of historical records, has a history fraught with challenges of privilege and absence. Privilege for the records and voices of the powerful and popular in society, and absence for the records of marginalized communities and obscure processes and effects.⁸ Selection of the items in a historical record, echoing Mahoney, dictates the types of history available for study, and by extension the methods and expertise needed to recover and narrativize it. This chapter’s approach to appraisal continues a dialogue, through

⁷[66] Duranti, Luciana. “The Concept of Appraisal and Archival Theory.” *American Archivist* 57, no. 2 (1994): 328-344.

⁸A clarification here. Much of the discussion of appraisal theory implicitly deals with the dominant position of western, white culture in archives. That is, with implicit racism and problems of the subjectivity of archivists. The criticism mounted against each successive phase of appraisal theory mentioned in this section implicitly involved a concern for the records of marginalized peoples. We interpret these phases for the records and history of science and technology because they also happen to map to issues of digital technical documentation. However, we do not intend to equate people with documentation, nor to ignore the pressing needs to adopt archival appraisal to serve the needs of the underrepresented, thus this note. For more information on appraisal as it relates to records of minority experiences, see [38] Booms, Hans. “Society and the Formation of a Documentary Heritage: Issues in the Appraisal of Archival Sources.” *Archivaria* 24, no. 3 (1987): 69-107. [59] Cumming, Kate, and Anne Picot. “Reinventing Appraisal.” *Archives and Manuscripts* 42, no. 2 (May 4, 2014): 133-45. doi:10.1080/01576895.2014.926824. [76] Flinn, Andrew. “Community Histories, Community Archives: Some Opportunities and Challenges.” *Journal of the Society of Archivists* 28, no. 2 (October 1, 2007): 151-76. doi:10.1080/00379810701611936. [100] Johnson, Elizabeth. “Our Archives, Our Selves: Documentation Strategy and the Re-Appraisal of Professional Identity.” *The American Archivist* 71, no. 1 (2008): 190-202. Additionally, the game community is just barely coming to terms with issues of subjective experience and inclusion. In that way, game archiving is two steps behind, we don’t know what we have, and we can’t leverage that to figure out what’s missing. Both case studies here are thus very much a partial (and privileged) representation of the work of game developers and the conditions of their lives. More work is certainly needed.

games and software, about the abilities of institutions and archivists to both keep up with the deluge of digital record creation and make sense of how it can be useful for historical understanding. Terry Cook, in an article summarizing the history of modern appraisal theory outlines three successive theoretical positions for the archive:

first, the archivist as curator who did not do appraisal, but left that to the creator; secondly, the archivist-historian indirectly appraising based on values derived from trends in historiography; [and] thirdly, the archivist directly appraising based on researching, analyzing, and assessing societal functionality and all related citizen-state activities.⁹

The “first” archivist uncritically takes what they are given by a creator. This gave way to a cession of archival responsibilities to the historians themselves. Their considerations for what was important to maintain in light of historiographical perspectives helped to provide a seemingly more solid methodological grounding for selection criteria. The problem with that perspective was two-fold: first, historians are only one potential use-case for archival records, so their specific needs were not representative of the needs of other scholars and second, as revealed spectacularly by Hayden White and others offering postmodern historiographic critiques, the conduct of historical study is itself tied to the contemporary issues and passions of its day.¹⁰ This is a weakness as any historiographic position cannot cover the whole set of potential narratives, and risks explicitly excluding many perspectives.

⁹[57] Cook, Terry. “‘We Are What We Keep; We Keep What We Are’: Archival Appraisal Past, Present and Future.” *Journal of the Society of Archivists* 32, no. 2 (October 1, 2011): 173-89. doi:10.1080/00379816.2011.619688. pg. 182

¹⁰White’s work is already part of the impetus for the *Debugging Game History* volume [86] mentioned in the introductory chapter. For more of his historiographic work, see [219] White, Hayden V. *Metahistory: The Historical Imagination in Nineteenth-Century Europe*. Baltimore: Johns Hopkins University Press, 1973. [220] White, Hayden V. *Tropics of Discourse: Essays in Cultural Criticism*. Baltimore: Johns Hopkins University Press, 1978. Specifically chapters 1, 2 and 3. White’s critiques also form the basis for the Chapter 6, “A Model of Doom.”

A more recent approach is research-based appraisal of the processes and functions that produce the documentary record.¹¹ By looking into the creation and use of records by the practitioner, creator, community or institution responsible for them, an appraisal can try to account for what is most internally salient and valuable to the archival record.

Another modern appraisal technique, “proactive” as opposed to “reactive” appraisal is embedded in our research-based case study of *Prom Week*. In discussing proactive appraisal, Nicole Convery refers to the theory of a “records continuum” first proposed in the UK in the early 2000s.¹² Traditionally, even when conducting research appraisal, there was little theoretical interaction between records managers and archivists. Records managers in institutions (or as individuals) did not explicitly organize their documentation with consideration for the needs of future archival storage. The “continuum” removes this clean division between creation, appraisal and archives. With the growth and complexity of digital documentation management and web services, there is now a need to embed archival assumptions into digital management schemes when they are created and conceived instead of when they find their way to a collecting institution. Additionally, the process of “finding their way to a collecting institution” is also changing in significant ways. Digital storage is relatively cheap, and many online documents remain editable and shareable long after they have fallen into disuse. With physical documentation, records needed to be purged to physically make room for more records, but

¹¹Angelika Menne-Haritz ([151] Menne-Haritz, Angelika. “Appraisal or Documentation: Can We Appraise Archives by Selecting Content?” *The American Archivist* 57, no. 3 (1994): 528-542.) addresses the changing perspectives on archival appraisal, most directly in light of Hans Booms ([38] Booms, Hans. “Society and the Formation of a Documentary Heritage: Issues in the Appraisal of Archival Sources.” *Archivaria* 24, no. 3 (1987): 69-107.) call for historical research and societal context in appraisal decisions. Previously, most appraisal conversations only referred to the provenance — maintenance and recording of sources — for archival records and not how the documentation being appraised might influence collection decisions.

¹²[56] Convery, Nicole. “From Reactive to Proactive Appraisal.” *Archives and Manuscripts* 42, no. 2 (May 4, 2014): 158-60. doi:10.1080/01576895.2014.911676.

digital documentation does not have similar constraints.

As a goal of this thesis is to make the history of software more open to interpretation and more tangible, research-based proactive appraisal around software technology provides a methodological entry point. We can link modern appraisal's interest in technology praxis with Michael Mahoney's concern for revealing the implicit, tacit knowledge within it. In this way — in treating games as software technology — we can then disassemble game development documentation with the tools of historians and archivists of technology.

Before proceeding with a discussion of appraisal sources, Cook deserves one more note. The above quote traverses three stages of appraisal theory enroute to a fourth focused on community engagement and outreach. “Perhaps we are ready to share [the] appraisal function with citizens, broadly defined, where we engage our expertise with theirs in a blend of coaching, mentoring, and partnering.”¹³ This call for active participation with community, and a relinquishment of the “appraisal function” is brought up as both a pragmatic and necessary next step. Pragmatic because modern information technology produces much more information than can possibly be stored, so perhaps appraisal perspectives within a community could help stem the tide. And necessary for very much the same reason, that there will be too much information for even a reasonably prepared specialist to deal with, and that there are more communities and disciplines in need of archival work than ever before.¹⁴

¹³[57] pg. 182

¹⁴For games specifically, we will return to community archives in Chapter 6 “A Model of *Doom*” below, and it will be apparent that we may have more to learn from the community (as regards the “appraisal function”) than they do from us.

2.2.1 Related Work

Our case study into *Prom Week* is prefaced on three research-based appraisals into the records of the history of science and technology.

In the history of science and technology, efforts were made in the early 1980s to remedy what was perceived as a lack of documentary records about the most significant science and technology research of the 20th century. In 1983, the Joint Committee on the Archives of Science and Technology (JCAST) published a report, “Understanding Process as Progress: Documentation of the History of Post-War Science and Technology the United States.”¹⁵ It provided a basis for the appraisal of documentation resulting from scientific research process, and argues — as we do for games — that preserving and understanding the documentation about how a project was conducted is as historically important as the final results. Game development, like contemporary scientific research, is an often-collaborative process. It involves much exploratory and iterative work before one or more final products are produced. The records resulting from its processes are also likely to be unfamiliar to a non-specialist archivist. The JCAST report highlights three major problems in dealing with historical scientific data and records.¹⁶ These problems map, without much translation, to major issues in both computer software record and development record appraisal:

1. The amount of unpublished documentation in game development is not addressed (or categorized by) current archival practice, and it cannot be estimated based on experiences from other fields.
2. There is “an absence of professional consensus on guidelines for the appraisal and description of archival records of science and technology.” This lack of

¹⁵[70] Elliott, Clark A. “Understanding Progress as Process: Documentation of the History of Post-War Science and Technology in the United States.” Society of American Archivists, 1983.

¹⁶[70] pg. 6

consensus contributes to both ingest backlogs at repositories unversed in the material and, in some cases, might lead to the needless destruction of potentially valuable materials. Many institutions are unaware of what they have, what can be done with it, who would want it and what it is worth.

3. Too little is known about the potential users of game documentation or “about how adequately contemporary [archival] practices [meet] their needs.”

Essentially, game development is a unique technical phenomenon, over which there is a lack of consensus about appraisal and description, and a lack of knowledge about future historical value. The JCAST report elaborates on the need to save scientists’ and technologists’ research journals, research data, and other findings, in addition to pre-publication works and reports. It has little to say, however, on the process for saving and recording software, development data, or any other computational systems and artifacts that make up most of the record of game development. Digital assets and systems are not well covered in the JCAST report (most likely due to 1983 being still quite early in the digital era). As we will see, they now make up a majority of the records in game development. Our case study work is then a rearticulation of JCAST’s methodology for born-digital development assets and software records.

The JCAST’s reports concern for process (and by extension the implicit craft knowledge embedded in it) is emphatically based on the concerns of historians of science and technology. The report is an attempted communication between those producing historical documentation and the historical scholars interested in its study. There is a consistent admission, throughout the document, of the need to save these records to intervene in the loss of tacit knowledge about scientific progress. The lack of attention paid to the technical and scientific objects of study themselves — the experiments and their software and data — is also understand-

able. Although process is considered important, the idea that process can also be recovered from, and embedded into, the actual objects of research is beyond the scope and concern of the report. It would not be reasonable to attempt to save the potentially monstrous and complex apparatus of scientific work (at least not in documentary archives), and general understanding of the importance and future potential of software data had not yet developed. That apparatus and software are mentioned at all is kind of a testament to the forward thinking appraisal work in the report, but it could not attempt to solve everything.

The second major influence on our initial case methodology is the Charles Babbage Institute's (CBI) 1989 report *The High Technology Company*.¹⁷ Explicitly focused on the development processes inside the Control Data Corporation, the report outlines the production of the CDC 1604 computer. Using the 1604 and a collection of other company products, the report conducts a "documentary probe" of their development processes. These specific internal processes, like research and development, engineering, and public relations, are derived and then analyzed in light of their specific documentary outputs. All documentation relating to each product was located and reviewed to gauge the extent and diversity of the entire organization's records through this analysis of constituent process. This is a significant (and maybe the only) detailed research-appraisal conducted for technology records in the wild. Our case study in *Prom Week* specifically, owes a significant debt to this report, since it provided the basic notion of using a specific research output as a way into its internal documentary processes.

Finally, Haas et al.'s *Appraising the Records of Modern Science and Technology: A Guide* provided insight into how to organize a research-appraisal of a

¹⁷[43] Bruemmer, Bruce, and Sheldon Hochheiser. "The High-Technology Company: A Historical Research and Archival Guide." Charles Babbage Institute, Center for the History of Information Processing, University of Minnesota, 1989.

working laboratory.¹⁸ The guide also provides the basic structural framing for our case study of *Prom Week*, that of process, context, and document enumeration. It also connects most emphatically to Mahoney’s basic delineation between the processes of technological production and the contexts within which those processes are performed. The guide spends a significant amount of time describing the basic assumptions underpinning the conduct of natural science research at the time, and highlights the relationships and power dynamics that come into play as a result.

2.3 *Prom Week*

So what does a proactive research appraisal of computer game development look like? And how does it relate to the interpretation of a game by historians of technology? This case study into the game *Prom Week* seeks help answer these questions, and to propose some new ones about the conduct of material histories for software. We are heavily informed by the appraisal discussion and sources above. This research appraisal is both a “documentary probe” in that we are using a produced software “product” *Prom Week* as a target for a scrutiny of the processes behind it, and an exploration of their resulting documentation and context in line with the work of JCAST and Haas et al’s guide to appraising science and technology records. This section begins with an explanation for the choice of *Prom Week*, how it fits into the outlined appraisal methods, and a brief description of our specific methodology. We then proceed to discuss the documentary processes of *Prom Week* (the stages of the game’s development process), the context of their creation in a research laboratory, and the extensive documentary

¹⁸[87] Haas, Joan K., Helen Willa Samuels, and Barbara Trippel Simmons. “Appraising the Records of Modern Science and Technology: A Guide.” Massachusetts Institute of Technology Cambridge, MA, 1985. <http://www.getcited.org/pub/102582895>.

outputs those processes produced. We conclude with significant elaboration on the difficulties and challenges of the research appraisal, and what implications they have for future historical work with games and software.

Most of the documentary issues in this section relate to the born-digital production and storage processes of modern game development. Therefore, there will be less explicit focus on physical records. We realize that the amount of physical records is almost certainly a function of the time-period in which development took place, but that born-digital software development records are under-theorized and share similar, digitally specific, constraints regardless of the time-period of their development.

2.3.1 Choice of *Prom Week*

Our appraisal is based on the game *Prom Week*, developed by a team in the Expressive Intelligence Studio (EIS) in the School of Engineering at the University of California, Santa Cruz from 2010 through 2014. *Prom Week* is a social simulation of the relationships between a group of high school students in the week before their senior prom. It is an academic research game that incorporates a new artificial intelligence framework, Comme il Faut (CiF), allowing the students in the game to remember past events and build unique and nuanced relationships.¹⁹

Prom Week was selected because it functioned as a bridge object between the research appraisal strategies of the history of science and technology and our target, the history of computer game software development. *Prom Week*, as

¹⁹For more information on *Prom Week* as a research contribution see [143] McCoy, Josh, Mike Treanor, Ben Samuel, Aaron A. Reed, Michael Mateas, and Noah Wardrip-Fruin. “Prom Week: Designing Past the Game/Story Dilemma.” *Proceedings of the 8th International Conference on Foundations of Digital Games*, 2013. [144] McCoy, Joshua. “All the World’s a Stage: A Playable Model of Social Interaction Inspired by Dramaturgical Analysis.” University of California, Santa Cruz, 2012. and [211] Treanor, Mike. “Investigating Procedural Expression and Interpretation in Videogames.” University of California, Santa Cruz, 2013.

academic research, lends itself to the processes outlined above for appraisal of historical technology. It again, as a game, allows us to use the research appraisal of technology as a starting point for the investigation of the appraisal of computer software technology. *Prom Week* is also at the nexus of professional and academic software. It produced numerous well regarded papers (as well as material for four dissertations), and also received industry recognition for its accomplishments in AI models of social interaction. This allowed us to infer that some of the methodologies we developed for *Prom Week*'s appraisal would map to other professional software development activities by small studios.

This latter point, about industry recognition, and a further, about access, round out our initial selection criteria. Since we were looking for a way to bring established appraisal strategies to bear on software development, academic research software fit the bill. It was scientific research with additional considerations for software. *Prom Week* was selected because in addition to being successful research, it was highly regarded, and we had direct access to it (being part of the same laboratory). This access made it possible to fully exhaust the appraisal in a way not possible for most commercial software. The aim was to find as many forms of documentation as possible, and to explore as many of their concerns as we could to display a better picture of the task of software history.

2.3.2 Process

The documentary processes of *Prom Week* are important because they give us glimpse behind the enclosed outputs of scientific and software development. *Prom Week*, as released to the game-playing public and written up in scientific publications, is an object distilled, in retrospect, from the processes that led to its creation. Release or publication present a unified and reasoned account of the

messy world of mistakes, dead-ends, missed opportunities and failures inherent to the creative process. Science and technology in the making is a journey outward from initial hypothesis and prototypes to points unknown. It is usually possible to figure out where you are, how you got there and where you have been only after significant iterations, retrials and refinement of intermediary results. In this section we focus on the processes behind the stabilization of *Prom Week* as a playable, scientific object, as derived from an examination of its development's documentary outputs.

Our efforts began by contacting the developers and asking them to provide whatever documentation was already at hand. This resulted in a few primary repositories of development work. Namely, their shared folders on Google Drive and Dropbox, links to the completed game as playable on Facebook and Kongregate, the contents of the various project mailing lists, and source code access through the team's Subversion version control repository hosted at UCSC's School of Engineering. After a cursory review of the documentation (mainly to locate relevant names of contributors) a series of interviews was organized.²⁰ These provided context for the documentation available, allowed for questions about some trickier items to be sorted out, and provided context for how the developers organized and enacted their development process. They also pointed to further items missed or forgotten in the initial appraisal like personal webspaces and files stored

²⁰Interviews were conducted over the course of three months (August 2013 - October 2013) based on the availability of the researchers. In total 14 hours of interviews were conducted with 7 principal members of the team: advisors Noah Wardrip-Fruin and Michael Mateas, and graduate researchers Joshua McCoy, Michael Treanor, Benjamin Samuel, Aaron Reed, and Brandon Tearse. Oral interview process was derived from the works of Willa K. Baum including: [26] American Association for State and Local History, and Oral History Association. *Oral History: An Interdisciplinary Anthology*. Edited by David King Dunaway and Willa K. Baum. 2nd ed. American Association for State and Local History Book Series. Walnut Creek: AltaMira Press, 1996. [31] Baum, Willa K. *Oral History for the Local Historical Society*. Nashville, Tenn.: American Association for State and Local History by special arrangement with the Conference of California Historical Societies, 1987. [30] Baum, Willa K. *Transcribing and Editing Oral History*. Nashville: American Association for State and Local History, 1977.

offline on personal hard drives and computers.

Eventually a set of six processes took shape:²¹

1. Idea Formation
2. Physical Prototyping
3. Digital Prototyping
4. Iterative Development
5. Release and Dissemination
6. Revision and Continued Development

We will briefly address each of these processes in turn with a specific focus on how they manifested for *Prom Week*. This inevitably includes an example history of *Prom Week*'s development through the lens of its documentation. Below is an abbreviation of the appraisal results found in the NEH report mentioned in the Case Studies section above. For more detail on the application of appraisal to game development documentation, and more extensive lists of the types of resources available please consult that document.²²

Idea Formation

Most academic projects owe some debt to previously completed work. The goals of researchers are based in their previous experiences and interests. As a result, early project documentation is a collection of previous and potentially

²¹The first four processes roughly align, especially the prototyping steps, with game design methods from the textbook [78] Fullerton, Tracy, Chris Swain, and Steven Hoffman. *Game Design Workshop: Designing, Prototyping, & Playtesting Games*. CRC Press, 2004. There are many game design methodology books, and we found that Fullerton et al.'s development stages were well aligned with the processes identified through interviews with the *Prom Week* development team and their documentation.

²²More specifically, [102] Pg. 6-28.

quite disparate threads. In the case of *Prom Week*, its genesis sprung from the foundational goals of the Expressive Intelligence Studio, a computer lab at the University of California, Santa Cruz. The lab’s co-directors Michael Mateas and Noah Wardrip-Fruin both had backgrounds in the design and analysis of artificial intelligence systems that generated interactive story and narrative structures. Mateas is known for the creation of *Façade*, a computer game that had players mediating a marital dispute through the use of free-text input. In collaboration with Andrew Stern, Mateas designed ABL (A Behavior Language) and an underlying narrative management system to support the game.²³ The lab’s other director, Noah Wardrip-Fruin, specialized in the analysis of historical narrative systems to reveal deeper understandings of how their technical design differed from people’s perception of their outputs.²⁴

The environment fostered by the specializations of the lab’s founders drew graduate students interested in games, artificial intelligence and narrative systems. One such student, Joshua McCoy, began working with Michael Mateas on artificial intelligence for the competitive online game *StarCraft* before settling into an exploration of social systems.²⁵ McCoy’s background in sociology, in addition to computer science, lead to experiments in computational modeling of different sociological and dramatic frameworks. Some specific work, inspired by Erving Goffman’s dramaturgical analysis, produced the conference paper “The Computation of the Self in Everyday Life: A Dramaturgical Approach for Socially Competent Agents.” This publication is the initial seed (and first piece of

²³[137] Mateas, Michael, and Andrew Stern. “A Behavior Language for Story-Based Believable Agents.” *Intelligent Systems*, IEEE 17, no. 4 (2002): 39-47. and [138] —. “*Façade*: An Experiment in Building a Fully-Realized Interactive Drama.” In *Game Developers Conference, Game Design Track*, 2:82, 2003.

²⁴[216] Wardrip-Fruin, Noah. *Expressive Processing*. Cambridge (Mass.): MIT Press, 2009.

²⁵McCoy first encountered Mateas’ work in a graduate student orientation course. Mateas presented work on his ABL system for *Façade* and piqued McCoy’s interest, he started working with Mateas soon after.

documentation) for the work that would result in *Prom Week*.²⁶ After completing work on a social modeling project in summer 2009, McCoy began work on the *Comme il Faut* system that would form the technical basis for *Prom Week*. Based on the potential for McCoy's system (as perceived by the lab's founders) and the interests of other graduate students in the lab work began on a prototype, game-based expression for it. In fall 2009, a small team assembled eager to find a way to leverage McCoy's work into a playable prototype.

The start of the project initiated numerous email lists for team communication and planning, and are the first born-digital records produced by the project team (after McCoy's initial publications and experiment system development). All of the initial email lists are prefaced with "alt-prom," a result of an early brainstorming meeting in September 2009 that settled on a prom theme for the game. These lists are the earliest, non-interview-based evidence of the project team's work and thought processes.

Physical Prototyping

Physical prototyping involves the construction of physical analogs for components of a game play system and emphasizes design failure and testing. This form of prototyping is not part of all development strategies, especially for software without a graphical user interface. However, since most games and user interfaces require visual feedback it is usually possible to prototype small parts of a larger system. Physical prototypes are usually low quality, quickly designed demonstrations of game play systems or visual design.

²⁶Locating the beginning of a specific research trajectory is only possible after retrospective analysis of the citations and references found in later work. For *Prom Week* we are fortunate in that McCoy started on a new research direction after completing his Master's degree, and that it was a clear departure from previous work (competitive gameplay to social modeling). Many times an appraisal will not find such a clear dividing line in the documentation, which is where further contextual information and history can then help out.

Physical prototyping began roughly a month after the formation of *Prom Week*'s development team. The goal was to find a suitable way to merge the work McCoy had started on social modeling in the CiF framework with an entertaining playable expression that explored the system's potential. As mentioned, early brainstorming settled on a social simulation of a high school prom. McCoy developed a prototype digital interface for his underlying social modeling system — basically just a collection of numerical input fields displayed in a web browser — that allowed for tweaking various system values. This was paired with a set of paper playing cards and a game mat displaying a school prom. Two players, competing as either the “goth” or “emo” contingent, played cards representing various transformations in the mental state of the prom's DJ, Milton. The goal of the game was to cause Milton to play specific music based on his attitudes toward one or the other group. Gameplay proceeded by playing cards, with McCoy manually feeding their values into the web interface, and seeing how they affected Milton's mood. Initial prototype play tests occurred in late November and early December. After multiple sessions it became apparent that players were not receiving enough external feedback — not through the representation of Milton nor the values on the web display — about how their cards influenced the social model. The focus then shifted to finding a way to externalize the social model's reactions through gameplay.

The composite physical / digital prototype is still available for analysis. Adobe Illustrator files featuring all the game cards and play board are stored in the Merritt repository, along with the Adobe Flash SWF representing McCoy's social simulation interface. However, to play the game, one would still need McCoy to explain and demonstrate the input of values and point out any potential system bugs. Additionally, both types of Adobe files are highly dependent on the version

of the software that generated them. The output from this phase of the project is a present but unplayable prototype, and continued discussion on the mailing lists. Once the prototype feedback pushed for an externalized model represented by a cast of characters, an art mailing list was started to coordinate communication between undergraduates responsible for the digital prototype's graphical needs. McCoy also turned his initial digital prototype work into a publication for the Digital Arts and Culture conference in December 2009.

Digital Prototyping

Digital prototypes are smaller, less fulfilled versions of some larger, more complex technical goal, but they still often involve the need for organization of software development tools and workflows. It is at this point, with *Prom Week's* digital prototype, that the inherent complexity of technical documentation comes to the fore.²⁷ In documenting *Prom Week's* digital prototype, our appraisal intersects with concepts like development environments, platform specific resources, third-party software and libraries, and version control and development management tools.

Prom Week's initial digital prototype was developed between December of 2009 and March 2010. This interval is the period between initial physical prototype feedback and a demo of the digital prototype at the Game Developer's Conference on March 9, 2010.²⁸ The team setup up a variety of collaborative work spaces

²⁷The digital prototypes for *Prom Week* can be alternately referred to as "The Prom" or "Promacolypse." "The Prom" is the name associated with the actual demo's Adobe air application, `TheProm-GDCdemo-9March2010.air`, and was referenced numerous times in oral interviews as an early title for the game. "Promacolypse" is the name of the containing folder for the GDC demo in the SVN repository, `altprom/trunk/GDCDemo/Promacolypse`. It is also the name given to the game by the art team in their early notes on the shared Google Drive folder.

²⁸*Prom Week* appeared in [136] Michael Mateas' "AI and Interactive Storytelling: How We Can Help Each Other" presentation at the GDC AI Summit <http://www.gdcvault.com/play/1012421/AI-and-Interactive-Storytelling-How>.

online, beginning with a Subversion version control repository on the UCSC School of Engineering's internal network and shared online folders on the Dropbox and Google Drive web services. They also recruited a team of undergraduate artists and additional programmers to help create artwork and animation support tools for the game.

The prototype was an Adobe Air application, written in the ActionScript 3.0 language and employing additional Adobe Flash created animation assets.²⁹ It featured a cast of three characters working through a basic interaction with underlying CiF system. One of the developers, Brandon Tearse, had been re-implementing CiF in a Java-derived functional language called Scala. Initially, it was thought using a more robust and flexible language would help speed up the CiF architecture. However, Tearse's work was abandoned when it became clear that interfacing between a Scala backend and ActionScript 3.0 frontend was not well supported by either language. Instead of writing bridging code in addition to the prototype, the team decided to port Tearse's Scala implementation to ActionScript 3.0 to align with the rest of the front-end development.

The team had less than ten weeks to deliver an initial digital prototype that integrated the CiF engine into a playable game. Most of the developers recounted a significant crunch toward the end of the this period, with fixes and additions being added until the day of the demonstration. The digital prototype initiated most of the technical documentation storage infrastructure. For the remainder of the project, all source code, art assets, and secondary support tools and frameworks we

²⁹Adobe ActionScript 3.0 is a programming language based on the ECMAScript standard (as such it is akin to JavaScript). ActionScript 3.0 is a generalization of ActionScript 2.0, a scripting language designed for use in the Adobe Flash animation application. Adobe Flash is a graphical user interface for creating timeline based animations that Adobe expanded into a full fledge application development framework. Adobe Air is a supplementary software tool that allows Adobe ActionScript 3.0 applications to be compiled for native execution on Microsoft Windows and Apple Mac OS X systems.

added to the Subversion repository or online collaborative folders initially created for the prototype. Therefore, basically all development documentation resided on cloud services or shared development servers. Additional code was added to the Subversion repository, with additional demo art assets and tools split between that repo and the online shared folders.

Iterative Development

Primary development on *Prom Week* began with the Spring quarter in late March 2010 and continued until the game’s official release on February 14th (Valentine’s Day) 2012. The primary development team remained stable for the length of the development, with some members dropping out for a quarter or two to fulfill other graduate students obligations (like qualifying exams and particularly difficult coursework). Development work ebbed and flowed to two different cyclical structures. The first was a consistent schedule of conference and public presentations of the game at various academic and game industry functions. *Prom Week* or its CiF social engine appeared in some form at GDC, FDG, and AIIDE in, respectively, March, June and October of 2010, 2011 and 2012.³⁰ As most presentations of the game or its underlying technology generally included some technical demonstration the team would inevitably have to crunch to meet each specific presentation deadline. As the game became more feature complete and stable, the work required for each demo subsided mildly. Especially after the

³⁰GDC is the aforementioned Game Developer’s Conference. FDG is the Foundations of Digital Games Conference (<http://www.foundationsofdigitalgames.org>), and AIIDE is the Artificial Intelligence and Interactive Digital Entertainment conference (<http://www.aiide.org>). See [145] McCoy, Joshua, Mike Treanor, Ben Samuel, Brandon Robert Tearse, Michael Mateas, and Noah Wardrip-Fruin. “The Prom: An Example of Socially-Oriented Gameplay.” In AIIDE, 2010. <http://www.aaai.org/ocs/index.php/AIIDE/AIIDE10/paper/download/2141/2573>. [146] McCoy, Joshua, Mike Treanor, Ben Samuel, Noah Wardrip-Fruin, and Michael Mateas. “Comme Il Faut: A System for Authoring Playable Social Models.” In AIIDE, 2011. <http://www.aaai.org/ocs/index.php/AIIDE/AIIDE11/paper/viewFile/4080/4429>. and [136].

release of the game.

The second cycle was that of SCRUM, an iterative development technique in which each member of the development team decided on a short course of action (traditionally a week or two), attempted to complete it, and then reported back to the group for feedback and potential redirection. Goals were chosen from a SCRUM board of post-it notes describing tasks in need of completion. The development team did not follow this process religiously, but there was consistent effort to maintain a loose and flexible shared network of development responsibilities.

Following the March 2010 GDC demo, the team decided to reframe *Prom Week* as a series of interactive social puzzles. Each scenario provided a mix of high school characters with pre-existing relationships that would be altered by the player's actions. The basic structure usually focused on a series of interactions in the week before Prom, and each new scenario would reset the clock to focus on a different character or social group. Goals including getting characters to fall in love and go to prom, give bullies a comeuppance and create or destroy friendships. In order to deal with the complexity of narrative interaction, significant work was put into developing a support tool to author character dialogue. Additionally, three graduate student writers were brought on board in September of 2010, with one, Aaron Reed, remaining for the duration of the project. As the writing demands increased, the team recruited undergraduates to write additional dialogue. Developer Ben Samuel managed this team from April to September of 2011.

As members joined and left development, each added additional documentation to the various shared resources. Many created folders for their specific work or function, or forked and branched the source code repository to add new features or try out some experimental design. The main development team, lead

by Josh McCoy, continually enhanced the game and its engine over the course of development. Predicate logic and weighted relationship rules appeared during the initial digital prototype, followed by micro-theories of interaction in summer of 2010. That fall the Social Facts Database (SFDB), the shared knowledge base for the game’s characters, gained the ability to note the timing and order of various in-game events. McCoy’s last major addition, in April of 2011, added modal logic support to the predicates. Essentially instead of just noting that “X loved Y,” a modal addendum would qualify the extremity of that statement — X could now love Y a lot or a little.³¹

This phase of the project accounts for a majority of its documentation. More people and time contributed to development in this period, so the record reflects that effort. Most of the development documentation was stored in the shared Dropbox folder and in the SVN repository. The Google Drive account only held 46 documents by its last recorded use on December 14th, 2013, whereas the Dropbox and SVN each contained thousand of files.

Release and Dissemination

A computer game release represents the demarcation between the internal development process and the external experience of the game by a more general audience. Though no hard and fast rules apply, the release of a piece of closed-source software is usually the only historical object generally available (and thus considered by popular game histories). The objects of game history are released games. Meaning that, without the access we were provided *Prom Week*’s historical description and entry into history would start here.³²

³¹This is obviously a gross simplification of the actual CiF modal predicate logic, but our appraisal did not seek to comprehend the game’s technical systems, only to point to the documentation necessary for that comprehension.

³²*Prom Week* is also the subject of numerous academic publications over this time period, including a post-release publication in 2013 outlining the history of its various prototypes [143].

The team released *Prom Week* on February 14th, 2012 as a standalone Shockwave Flash file (.swf) uploaded to both Facebook and Kongregate.com.³³ Although the file was modified a few times in the next year (see below), this uploaded object is the primary result of the team’s two years of development effort. It is also a culmination of Josh McCoy’s initial social modeling work in late 2008 and early 2009.³⁴ The .swf is a compilation of ActionScript 3.0 source code (relationship rule set, social facts database) and the game’s creative assets (narrative text, art and music). In this development environment, everything needed to run the game is included in the executable file. This is actually a bit exceptional, and reflects the specific nature of Adobe Flash’s platform at the time. Most Flash applications were intended for web browser delivery, and thus did not have the luxury of a local file system. Additionally, for security reasons, an application loaded on an external portal, like Facebook or Kongregate, would not be allowed to access resources elsewhere on the Internet.³⁵ As such, the .swf format reflected the needs of self-sufficient web disseminated interactive object.

Prom Week’s release also entailed a significant push by UCSC in general, and the EIS lab in particular to publicize the release. This included entering the game into various independent games competitions, courting online game reviewers to take a look, releasing press releases, and publicly presenting the game at academic and industry events. Both the Independent Games Festival

For most non-academic games, no such level of introspection or publication is usually conducted.

³³As of this writing, the Facebook version is only playable by those with Facebook login credentials. The Kongregate version is still playable here (<http://www.kongregate.com/games/promweekplaya/prom-week>) (Accessed 8/31/2016), however certain game features, like full-screen viewing, are now behind a Kongregate login (which coincidentally can also be a Facebook login).

³⁴The other, presumably, being his dissertation based on this work [144] and its resulting doctorate.

³⁵This type of vulnerability is known as a cross-site scripting (XSS) attack. If not prevented, a SWF downloaded from one location could potentially load malicious code and resources from another without the knowledge of the player.

(IGF) and IndieCade Festival nominated *Prom Week* for its technical design and narrative components respectively. Since these festivals represent the major venues for independent game design work, additional development work was conducted to help improve nomination chances. A special IGF submission build was developed alongside the release candidate in September and October of 2011. The IndieCade nomination in June of 2012 (well after the February general release) prompted a full redesign of the game’s user interface. There had been numerous usability and visual improvements that the team had wanted to make. The IndieCade nomination allowed for more resources to be diverted to address them and for a UI consultant to be hired to help spruce things up. The version of *Prom Week* now available on Kongregate is the result of the IndieCade redesign. *Prom Week*’s originally released version of February 2012 is only possibly recoverable through previous revisions in the SVN repository.

Revision and Continued Development

The last phase of the *Prom Week*’s development is its slow imbrication into the future work of the EIS lab. After release, work continued on the game (as mentioned above) but it also formed the basis for new research directions and publications. Josh McCoy mined *Prom Week*’s online play traces (the game reported user statistics back to a development server) to analyze if the game’s CiF engine really did provide players with unique and challenging narrative experiences.³⁶ The research data streamed from the live *Prom Week* application is one of the major data sources not collected for our appraisal. This is due to a lack of infrastructure and resources to set up a continual backup of the streamed data (which is still being collected) past the end of our research appraisal work.

³⁶[143] McCoy, Josh, Mike Treanor, Ben Samuel, Aaron A. Reed, Michael Mateas, and Noah Wardrip-Fruin. “Prom Week: Designing Past the Game/Story Dilemma.” Proceedings of the 8th International Conference on Foundations of Digital Games, 2013.

In an inversion to idea formation, *Prom Week* is now an inspiration for other projects in the lab. Team members worked on ways to improve narrative authoring for CiF after it was found to be a major development time sink. They also ported CiF from ActionScript 3.0 to JavaScript for use in web-games and applications. Members of the EIS lab in general also took aspects of *Prom Week* in new directions. James Ryan, a natural language processing researcher, worked with undergraduates to annotate the play traces recorded from the live game. They used those annotations to develop a second narrative system that could generate new dialogue exchanges without the need of a human author. That work then inspired development of other narrative-focused research games, including *Bad News*, which won a CHI student game design award in May 2016 and an IndieCade Audience Choice award in October 2016.

It is at the end of a project, when its outputs fertilize the ground of new research, that documentation again confronts boundary and retention issues. The boundary is in deciding where the documentation for *Prom Week* ends. As a research object and a code base, modifications and advancements continue to this day (late 2016). Our appraisal's documentation and time-bounds are necessarily a bit arbitrary. We grabbed snapshots of the code base at the end of our primary research in August 2014 (even though some now abandoned work on an Apple iOS version of *Prom Week* was in process). If new documentation was being created, but we did not already have access to the sites of its generation, then those sites were now outside of our appraisal.

Retention, the process of establishing a means and method for the continual storage of documentation, immediately becomes an issue once a project is complete. *Prom Week's* researchers moved onto future research work (or left the university after completing their degrees) without any plan to save the documen-

tation of their work. The source code for the project, stored on the UCSC SVN repository server, actually moved quite soon after the release of the IndieCade version of the game. If Ben Samuel had not been actively working on the aborted Apple iOS port at the time, we would have had to locate and reconnect the SVN server ourselves. For other projects, without an active archival intervention in progress, the code would simply have been lost. All of *Prom Week's* cloud based documentation is still available, though it has not been updated since late 2013.³⁷ Its continued existence is based on the whims of the Google and Dropbox services and their longevity. All physical products of the research, including the early prototypes and some public relations material, were saved in the appraisal. All PR material not explicitly collected is no longer locatable, having been cleaned out of the lab at some point in the last three years.

This concludes our documentary timeline of *Prom Week*. The next two sections, Context and Enumeration, will elaborate more on the work constraints faced by the team, their interactions with one another and the research hierarchy, and the exact types of documentation produced inside the sites alluded to above.

2.3.3 Context

All software is created in a unique social and organizational context. As the history of *Prom Week's* process shows, the work of academic game development does not occur in a vacuum. The norms of the computer science community, the development team's goals and personalities, and the realities of institutional hierarchy and funding all contribute to constrain and influence research and development. Knowing how research context relates to the development process

³⁷Occasionally one of the developers goes back in and changes or alters some document. Therefore the most recent activity is from 2015, but it does not appear to be directly related to *Prom Week's* research.

is important for understanding the contents of development records. For *Prom Week*, much of the documentation and its production constraints are the direct result of institutional processes and social dynamics. In this section we look at the organizations involved in the creation of *Prom Week*, from the academic lab that birthed it to the individual groups responsible for its design and development.³⁸

The Expressive Intelligence Studio

The laboratory is a structure inside the modern research university devoted to a specialized scientific or technical tradition. It does not have to focus exclusively on one area of study and can be quite multidisciplinary, however the core values and intellectual valences of its inquiries are always somehow related. Individual research labs exist in a hierarchy and are affiliated above with academic departments and schools and below with specific researchers and students. Administrative decisions and the interests of lab members merge to dictate the course of research. The lab may also be part of a larger, non-departmental organization of multiple different departments and labs across a campus or multiple institutions.

The Expressive Intelligence Studio (EIS), the lab responsible for *Prom Week*'s development, is a part of the Center for Games and Playable Media at UC Santa Cruz. It is also affiliated with the Computational Media and Computer Science Departments in the Jack Baskin School of Engineering, and the Digital Arts and New Media Program (DANM).³⁹ The lab's initial focus was on expressive artificial intelligence systems and interactive narrative techniques for computer games.

³⁸As noted in the Process section, the contextualization work below is essentially a *Prom Week* focused summary of the NEH report's generalization. Please consult [102] pg. 29-35 for more detailed information.

³⁹The EIS lab originally resided in the Computer Science Department, but its directors founded a new Computational Media Department in 2015. The lab's co-director Noah Wardrip-Fruin is also a member of the Digital Arts and New Media faculty. The lab includes students from all three programs as a result of grandfathering (in the case of Computer Science) and drastic interdisciplinarity.

Over time it evolved to include digital humanities, general game studies and interdisciplinary art practice.

Projects in the lab evolve from the research of the co-directors in conjunction with the backgrounds and interests of their students. For *Prom Week*, as discussed in the Idea Formation section, Josh McCoy's interests in sociological analysis merged with Michael Mateas' expressive artificial intelligence system from *Facade*. *Prom Week* is a bit unique in that it was pursued mostly as a side project. Usually, lab research is a combination of motivation and funding. Research grants dictate the scope and direction of study, and as a result must also comport with the goals of larger funding agencies. Most of the graduate students on the *Prom Week* project pursued it on the side. This means that they also had teaching duties, other funded research work, and class schedules competing with the completion of the game. The only explicit funding came from Michael Mateas' NSF CAREER and student Ben Samuel's NSF GRFP grants. Both of these are provided for general research interest and were not specifically awarded for *Prom Week*.⁴⁰ In fact, many on the team admitted that getting funding for *Prom Week*, an AI social simulation game featuring high school politics, would have been a hard sell to most science and engineering grant programs.⁴¹

Work on *Prom Week* was dependent on the availability and time of graduate students and faculty already busy with other commitments. This is reflected in the documentary production for the project, as members of the core development drop out of the record for significant intervals based on the dominant whims of

⁴⁰NSF CAREER grants are awarded to beginning faculty to help bolster their initial research program and are not directed to any specific research (this allows for initial flexibility). NSF Graduate Research Fellowship Program (GRFP) grants are awarded to promising new graduate students for their general research.

⁴¹On a further ironic note, Tom Coburn (then a Republican senator from Oklahoma) included *Prom Week* on his annual government waste list. The next application of *Prom Week*'s underlying CiF architecture was a DARPA-funded troop training program aimed at helping soldiers integrate with and befriend local populations.

their degree program and funding schedules. For example, Josh McCoy left the project for the Fall quarter of 2011 to advance to PhD candidacy. He passed the project management mantle to Mike Treanor who had to finish steering the project to an October 2011 IGF submission and a February 2012 launch. In the positive direction, Michael Mateas' CAREER grant allowed for the summer hire of Mike Treanor in 2010 to help with some foundational programming. Likewise, Ben Samuel's GRFP allowed him to manage an undergraduate writing team in mid-2011 that helped implement significant narrative content for the game.

Social Groups

A laboratory's document production is embedded in the hierarchical relationships of its members. In the EIS lab, two co-directors manage an ever-changing cohort of around 20 graduate students. These students in turn may recruit undergraduates to help with minor tasks associated with development and research. In *Prom Week's* development, the lab directors initiated the development process by deciding to pursue a game based on Josh McCoy's work in addressing an open research question identified by work on Michael Mateas' Facade system. The directors contributed to initial brainstorming sessions, commented on prototypes, and helped with drafts of development resultant research articles. However, a majority of the technical design, implementation and testing fell to the graduate researchers — they and the undergraduates produced all of the artwork and code associated with the project. Grad researchers also managed all the sites of documentary production. This includes all cloud storage, version control, and mailing lists.

On *Prom Week* the undergraduate team contributed considerable amounts of the content including art assets, sound design and written dialogue. In line

with the discussion above, the undergraduate researchers did not implement *Prom Week*'s major research contributions. They did, however, design tools the development team used to insert social scenario content and dialogue into the game. This included the dialogue authoring Design Tool and the skeletal animation system for the game's characters (developed quite early for the original digital prototype). Undergraduate involvement was sporadic, with many moving onto other engagements and graduating before the completion of the project.

External Influences

In many cases, the influence of outside forces is instrumental in directing lab goals, policies and funding. As mentioned above, the outputs from *Prom Week*'s development, in addition to those associated with its development as a game, were attached to research schedules dictated by conference publications and presentations. Key presentations and demos indexed the activities of the research team, and framed their communication with research and industry peers, the gaming press, and the player community.

For academia, *Prom Week* needed to communicate and engage with trends in artificial intelligence and game studies. For industry, it needed to present a new approach to advanced artificial intelligence systems that could be clearly leveraged into new game products. Members of both peer groups met with the team and commented on its development. During the initial physical prototype, a couple professional game designers and artificial intelligence academics played and commented on the system.⁴² Symmetrically toward the completion of the project another round of external play tests took place. The team held eight playtests with game industry professionals and academics in the fall preceding release.⁴³

⁴²Henry Lowood, Curator of History of Science and Technology Collections, Stanford, and AI programmer Richard Evans both looked at paper prototypes of the game.

⁴³The playtesters were: Richard Lemarchand (Naughty Dog), Borut Pfeifer (Haunted Temple

They wanted to improve the IGF submission and final release builds of the game.

Prom Week's press attention ramped up significantly after its nomination for IGF and IndieCade awards. Officially released in February 2012, the industry nominations came in January (IGF) and June of that year. Numerous gaming sites covered and reviewed the game. The IndieCade nomination included a place at that conference's E3 expo both, leading to another round of game play reviews by industry press. Press response to the game generally praised its AI and social simulation features. Some critiques of the UI prompted development work over that summer to improve the game's showing at IndieCade in the fall.

Lastly, feedback from the player community, including commentary on Kongregate, led to various tweaks and improvements. Most of those were related to stability issues encountered by a small amount of the 65,000 or so people who played the game after release. Player exploration also fed the backend data logging used for further publication and research. Without a significant number of players, continued *Prom Week* research outputs would not have materialized. In this case, making a good game was also good research practice.

2.3.4 Documentary Enumeration

This section outlines the total extent of the documentary record for *Prom Week*, and is to our knowledge the first full attempt at an enumeration of the material records (digital and physical) for a software development project.⁴⁴ The

Studios), Ozlem Kalini (Sony), Nic Duchonaut (PARC), Alex Neuse and Mike Roush (Gaijin Games), Mirjam Eladhari (Gotland University), Clint Hocking (LucasArts) and Arash Keshmirian and Serban Porumbescu (LIMBIC Software).

⁴⁴However, work into the enumeration of game documentary outputs has been framed and described before. [227] Winget, Megan A., and Caitlin Murray. "Collecting and Preserving Videogames and Their Related Materials: A Review of Current Practice, Game-Related Archives and Research Projects." Proceedings of the American Society for Information Science and Technology 45, no. 1 (January 1, 2008): 1-9. doi:10.1002/meet.2008.1450450250. [228] Winget, Megan A., and Wiliam Walker Sampson. "Game Development Documentation and Institutional Collection Development Policy." In Proceedings of the 11th Annual International ACM/IEEE

larger report includes a significant overview of the material classes for the documentation, and is a bit too general for our discussion in this section.⁴⁵ Instead, we focus specifically on the documentation for *Prom Week*, where it was stored, and what problems it revealed in the archival appraisal (and retention) of its game development records.

***Prom Week* Organization**

The finalized output of the *Prom Week* project exists on Facebook and Kongregate.com. The complete project documentation, as a diffuse set of research and development code, art, and public relations material, is housed in five primary locations:

1. The University of California, Santa Cruz Subversion version control server at: <https://svn.soe.ucsc.edu/svn/altprom>
2. Shared folders on the Google Drive and Dropbox cloud storage applications
3. Correspondence (email listserves)
4. Team members files on their personal computers or websites
5. External third party websites describing, reviewing or hosting the game

Prom Week's version control repository holds the code and assets necessary to compile the final release version of the game as well as most of its earlier versions and demos. Cloud storage hosted additional files related to research demos, publications, and individual developer folders for most of the undergraduates and

Joint Conference on Digital Libraries, 29-38. JCDL '11. New York, NY, USA: ACM, 2011. doi:10.1145/1998076.1998083.

⁴⁵See [102] Pg. 38-59 for a more general discussion.

graduate students on the team. Both version control and cloud storage locations contain a lot of unused or preliminary documentation, including abandoned features, versions of the game, and half completed supplementary tools. While the version control repository has some basic organizational strategy — its folder organization is roughly in line with ActionScript 3.0 and Flex 4.0 application development — the cloud services offer no such guidance.⁴⁶ Every team member had access to cloud storage folders. There is no consistent scheme for directory names, each usually just references a specific task, like a conference publication (“AIIDE 2010”), or a specific contributor (“Ben’s Stuff”). Many of the files in both locations are a mystery to members of the development team. Some were created by people only temporarily assigned to the project, while others, due to the length of development, were simply forgotten. In one particularly illuminating example, a developer forgot the programming language they used to contribute to the project.

The *Prom Week* team used Dropbox and Google Drive for cloud storage services. The Dropbox folder contains a majority of the cloud-hosted documentation, with around 2GB of data and over 4000 files. By contrast, the Google Drive only holds 40 or so files and did not see as much use over the project’s development interval.

Prom Week’s official correspondence occurred on 7 separate internal mailing lists hosted on UCSC servers. Lists formed around specific team activities:

- Altprom-artgroup
 - All work on in-game art, sound and other creative content
- Altprom-authors

⁴⁶Adobe Flex is a UI framework for Adobe Flash applications. Flex project have a routinized structure for the organization of their source code and binary compilation outputs.

- All authors writing *Prom Week* in-game conversation text
- Altprom-closers
 - List devoted to a smaller group of developers focused on finishing the project after undergraduates left the project
- Altprom-commits
 - Emails sent by source control server describing recent code changes
- Altprom-general
 - List for everyone involved in the project, provided general announcements and information
- Altprom-group
 - Probably a smaller sub group of core developers (development team could not remember what this list was for)
- Altprom-tigers
 - Initial *Prom Week* email list, consisted of core development team only

All email lists are prefaced with “altprom” as this was the original name for the project based on early development meetings. The emails did account for everyone involved in development, and even helped the core team members remember when certain people joined and left the development team. Due to the length of the project some of the reasons for starting separate email lists are forgotten. Smaller ones were created for specific, short-term purposes and then changed focus or were abandoned. The personal development files consisted of preliminary work done by Josh McCoy on early CiF prototypes, the original digital component to the

physical prototype, and development blog posts made by Ben Samuel. External web sources included game review websites, and the promotional documentation for both the IndieCade and IGF submissions.

Documentation Problems

The work above outlines types of documentation, shows how they were organized, and provides historical context for their creation. This section deals with the problems encountered in trying to figure all that out. *Prom Week's* born-digital documentation resisted appraisal (and archiving) in a couple key ways that have implications for the historical study of game and general software. Broadly, there are significant issues with access, identification, and migration of digital files that will need more extensive methodology and consideration. These issues are also an impetus for our proactive approach to appraisal, for only in trying to sort out documentation before it comes to the archive can we anticipate and prepare for the difficulties associated with it. None of the issues described in this section are specific to *Prom Week's* documentation, and will certainly manifest in most other software development projects — probably with even more challenges depending on the scale of the project.

Access Access to documentation is key to its eventual survival. While this may seem an obvious point, what is not obvious is the new difficulties in securing access to born-digital project documentation. Gaining access means acquiring login credentials to each documentary source, and using software specific to each for file viewing and retrieval. *Prom Week's* main documentation was housed on a university controlled version control server, and two cloud storage services, Google Drive and Dropbox.

Both the version control server, Subversion, and Dropbox allow for easy down-

load to a local machine running either the “svn” command-line client or the Dropbox desktop application. The “svn” tool for Subversion allowed for the check out of the current source code revision or for a full dump of the entire project version history. Dropbox’s application synced the cloud project folder to a local one, copying over the entirety of the development documentation without much effort. For Google Drive, the situation was more difficult since there was no native sync available during our project (it was added afterward). Additionally, Google Drive’s native documents do not have a file type since they exist only exist as logical organizations of data expressed through a web interface. We will return to this point in Migration below.

Identification File identification and dependencies are a significant issue for the retrieval and comprehension of historical data sources. In “Ensuring the Longevity of Digital Information,” Jeff Rothenberg first addressed the mutability and fragility of digital data by stating that “old bit streams never die — they just become unreadable.”⁴⁷ He recounted a hypothetical story of his to-be distant ancestors happening upon their family heritage on a compact disc. Accompanying it was a physical letter describing the encoding of the disc’s data, a necessary physical support for its digital nature. If you have a collection of bits and no context or specification for their internal organization, it is little better than having a bunch of random ones. Even when you know what the data is and what it’s supposed to articulate, retrieving that articulation can be difficult without the initial software that created it. *Prom Week*’s files take on a significant range of types, and implicate a constellation of other programs intimately tied to the files’ expression and future historical comprehension. They also highlight issues of

⁴⁷[185] Rothenberg, Jeff. “Ensuring the Longevity of Digital Information.” *Int’l. J. Legal Info.* 26 (1998): 1. pg. 2

versioning and obscurity that result from the large, highly varied document types that accompany software development work on game projects.

In Table 2.1 below we have organized all 60 file types found in *Prom Week's* documentation. We have included the 30 or so dependent programs one would need to interpret them all.

Table 2.1: *Prom Week* File Formats and Dependent Programs

File Extension	File Description	Dependent Program	Plain Text?
.7z	7-Zip Archive	7-Zip	N
.ai	Adobe Illustrator project	Adobe Illustrator	N
.as	ActionScript 3.0 source	Any ActionScript compatible integrated development environment, any text editor	Y
.asproj	Flash Develop ActionScript 3.0 project	Flash Develop	Y
.au	Audacity block	Audacity audio software	N
.aup	Audacity project	–	N
.aup.bak	Audacity project backup	–	N
.avi	Audio Video Interleave	Any video player or video editing software	N
.bak	General backup	Dependent on backup type	?
.bat	Microsoft Windows batch process script	Microsoft Windows operating system	Y
.camproj	Camtasia Studio project	Camtasia Studio 7.0	Y

Continued on next page

Continued from previous page

File Extension	File Description	Dependent Program	Plain Text?
.camrec	Camtasia Studio recording	Camtasia Studio 7.0	N
.css	Cascading style sheet	Any web browser, text editor	Y
.csv	Comma separated values	Any spreadsheet, text editor	Y
.dll	Microsoft Windows Dynamic-link Library	Microsoft Windows Operating System	N
.doc	Microsoft Word document	Microsoft Word (pre 2007)	Y
.docx	Microsoft Word document, object oriented XML	Microsoft Word (2007 or later)	Y
.dropbox	Dropbox configuration	Dropbox	Y
.fla	Adobe Flash project	Adobe Flash	N
.fxp	Adobe Flash Builder Flex project file	Adobe Flash Builder	Y
.gif	Graphics Interchange Format	Any image software	N
.html	Hypertext Markup Language file (XHTML, HTML 4.01, 5.0)	Any web browser, text editor	Y
.java	Java source	Any text editor	Y
.jpeg	Joint Expert Group image file	Any image software	N

Continued on next page

Continued from previous page

File Extension	File Description	Dependent Program	Plain Text?
.jpg	Same as .jpeg	–	–
.js	JavaScript source file	Any web browser, text editor	Y
.lcl	Windows log in screen editor	Deviant Art community application (no longer available)	N
.m4v	Apple Video container	Any video player	N
.mov	Quicktime Video	Quicktime	N
.mp3	MPEG-1 or MPEG-2 Audio Layer III digital audio	Any music player	N
.mp4	MPEG-4 Part 14 multimedia format	Any video player	N
.mxml	Adobe Flex meta XML	Adobe Flash Builder, any text editor	Y
.odt	OpenDocument text	OpenOffice Writer	Y
.odp	OpenDocument presentation	OpenOffice Impress	N
.old	Backup	Dependent on backup type	?
.pages	Apple Pages document	Apple Pages	Y
.pbm	Portable bitmap	Netpbm	N
.pdf	Portable Document Format	Any PDF viewer	N
.php	PHP source	Any text editor	Y

Continued on next page

Continued from previous page

File Extension	File Description	Dependent Program	Plain Text?
.png	Portable Network Graphics	Any image software	N
.potx	Microsoft Powerpoint template	Microsoft Powerpoint (2007 or later)	N
.ppt	Microsoft Powerpoint presentation	Microsoft Powerpoint (pre-2007)	N
.pptx	Microsoft Powerpoint presentation, object oriented XML	Microsoft Powerpoint (2007 or later)	N
.psd	Adobe Photoshop project	Adobe Photoshop	N
.pyd	Microsoft Windows Python Dynamic-link library	Microsoft Windows operating system	N
.rtf	Rich Text Format	Any text editor	Y
.sql	Structure Query Language source	Any text editor	Y
.svg	Scalable Vector Graphics	Any image software	Y
.swc	Compiled Shockwave Flash file	Adobe Flash / Flash Builder	N
.swf	Shockwave Flash file	Adobe Flash / Flash web browser plugin	N
.txt	Standard ASCII text	Any text editor	Y
.vpk	VUE package file	Tufts Visual Understanding Environment	N

Continued on next page

Continued from previous page

File Extension	File Description	Dependent Program	Plain Text?
.vue	VUE concept map	Tufts Visual Understanding Environment	N
.wav	Waveform audio file	Any audio player	N
.xcf	GIMP project file	GNU Image Manipulation Program	N
.xlsx	Microsoft Excel spreadsheet, object oriented XML	Microsoft Excel (2007 or later)	Y
.xml	eXtensible Markup Language	Any text editor	Y
.xmpses	Adobe Premiere Elements DVD Marker	Adobe Premiere	N
.zip	Zip file archive	Any extraction program	N
No extension	File folder / directory	Dependent on operating system	?

In many cases, loading a file into a text or hex editor provided enough header information to determine the file type.⁴⁸ That is, if the file conformed to a particular standard and was not one specific to *Prom Week*'s development team. Luckily, this did not turn out to be the case. Regardless of the correct identification of a file it was sometimes difficult to determine which version of a program created it.

⁴⁸A "hex-editor" is a simple program that presents the organization of a computer file as its literal sequence of bytes. These programs allow for the analysis of signatures at the beginning of files in which a specific sequence of byte values can act as an identifying preface for the file's type and contents.

The Adobe Creative Suite programs are now certainly out of date. This would cause a more recent version of the software to convert the old format into a new one (and thus risk the lost loss of specific file structure information that may no longer be used). There are some archival software tools designed to deal with this problem, but most of them are focused on the narrow confines of archival image or static document collections. As such, they can tell you a lot about specific types of PDFs or image formats, but not as much about production files like the aforementioned Adobe Creative Suite items.⁴⁹

Other files types were just obscure. Two examples in Table 2.1 are `.vue` and `.lel` files. Without our research focus allowing time to do a thorough investigation it would probably not be feasible to identify each file type. The `.vue` file is a Visual Understanding Environment file for a Tufts University mind-mapping program, however this was far from immediately apparent. All common information online pointed to a `.vue` file being one full of 3D geometry, which made no sense in the context of *Prom Week*, a 2D game. After examining the header of a `.vpk` file (another mysterious entity) we determined that the `.vpk` was an archive file of `.vue` files. The header also mentioned Tufts University. This lead to a search for “Tufts” along with “.vue” and “.vpk” that resolved to the website of the originating program.

The `.lel` file was so obscure that we only have a guess that it is associated with a Windows 7 start up screen modification program. The online community DeviantArt provides user created programs to modify operating system themes and aesthetics. A program that modified Windows 7 log-in screens appears to have accepted `.lel` files. This is not verifiable since the program has long since

⁴⁹Identification programs include: DROID (<http://digital-preservation.github.io/droid/>), JHOVE (<http://jhove.sourceforge.net/>) and JHOVE2 (<https://bitbucket.org/jhove2/main/wiki/Home>). Bit Curator from University of Maryland might be more useful but was released after our investigation concluded (<http://www.bitcurator.net>).

been removed from the site. Our information comes from a collection of links posted to files that are no longer hosted.

Migration Migration of digital data from older to newer formats, and from older storage to newer storage is the principle concern of digital preservation efforts. Data is never fixed in a single location for very long. Eventually the media on which it is stored will fail and the data carried on to a new, still temporary location. Migration sometimes implies a change of form, a way to fit the old data into a newer structure without totally destroying the data’s meaning and interoperability. *Prom Week’s* data sources — outlined in the *Prom Week* Organization section above — presented two types of migration challenges. First is the need to agnostically copy files from one location to another without disturbing them. “Disturbing” in this sense would be to accidentally insert new files (usually in the form of operating system indexing files) into old directories or to change the modification and creation date of the files. Second is how to extract data from services and sites for which you cannot gain full file access. In light of these challenges, we briefly address their relevance to each data source for *Prom Week*.

Dropbox Dropbox synchronizes its cloud-based files to a user’s local machine and then forwards all user changes back to the cloud. It then propagates those changes to anyone else synced to those files and pushes the changes to their local machines. Initially, Dropbox did not preserve file creation dates when copying cloud-files to a user’s local directory. When we gained access to *Prom Week’s* Dropbox information, we initially thought we could directly copy our local Dropbox files and thus easily migrate them off the service. However, since the times and dates of creation and last modification were not correct, we had to resort to Dropbox’s browser interface to retrieve the items. This was significantly slower

and a bit inconvenient.⁵⁰ In looking into Dropbox migration, we also discovered that Dropbox’s application programming interface (API) includes access to metadata and file description information (like the last known user to edit a file) that were not present in the local copy on our machine. Collecting and organizing this additional metadata was out of scope since we didn’t have resources to finish the appraisal and develop scripts to scrape and link the file metadata. Also, if the data was retrieved, there was solution for where to leave it and how to describe it for the future.

Google Drive As mentioned, Google Drive’s documents, be they spreadsheets or word processing files, do not exist apart from their representation through their respective Google App. They have no fixed logical form. When downloading a document from the service, the file represented in the web browser is converted into a potential variety of formats based on a user’s request. Therefore, migration of the “original” file is not supported by the service because — technically — there is no stable, bounded file available as there would be from a local file system. Additionally, Google provides an API for accessing data on the service and, like Dropbox, that API provides metadata that is not available anywhere else.

As a quick example, most of the *Prom Week* documentation on Google Drive consists of text-based planning documents. They can be downloaded in one of six formats: Portable Document Format (.pdf), plain text (.txt), Rich Text Format (.rtf), Microsoft Office (.docx), OpenOffice (.odt) or web page (zipped .html archive). When exporting, all revision history and file modification information is stripped since — from the perspective of the local file system — each downloaded

⁵⁰Dropbox has since changed their service, so the file modification and creation information is now accurate for locally stored files.

file has just been created.

Google Drive documents also record extensive revision histories. One can link specific edits to specific users, and see how a document took shape over time. Although revision histories are accessible and downloadable from the browser client, recording all documentary revisions in a timely fashion would require interfacing with Google's API. Also, the revisioning system on Drive tracks more recent edits with a finer granularity than older ones. This exponential decay is sensible, as older edits are less likely to be frequently needed by users. It also reduces the storage burden on Google servers, and since they are not in the archival or history business they have no basic incentive to keep older information available.

Subversion The Subversion version control system is designed to manage frequent updates to files by a collaborative team. Each change (or set of changes) can be added as a discrete revision to the file system being tracked. A revision addition is referred to as “checking in” a set of changes, which allows other users to check it “out” before continuing any work that might overlap with the same files. These systems are a vast improvement for productivity and team communication, but they also create significant migration challenges. Since the current files in a Subversion repository reflect the state of a specific, indexed revision (usually the most current), all other revisions will not be copied along with the current directory. In order to get all the revisions, one needs to use secondary tools, like `svnadmin`, to dump an archive of all previous revisions, or use the main Subversion command line tool `svn` to check out each revision of interest and manually copy it.

Version control repositories are software programs that manage file revisions. Since they are software each repository is only compatible with specific Subversion software versions. This means that making use of the system's dumps or

exploring a legacy repository also needs a system capable of reading it. The Subversion software itself is then an appraisal and archival target in addition to the documentation it contains. This also means that migrating the full Subversion repository requires a full copy of the repository as it exists on a host server and not just downloaded to a user's machine. Since we had full development access to *Prom Week's* server, we could just copy the root repository directly. This would not be the case with most other projects stored on public Subversion services, since they are only designed for read-only access and revision "check out."

Email and Developer Files Email and personal developer files were largely unproblematic compared to the services above. The email lists were exported by the *Prom Week* developers into a collection of .mbox Mailbox email archives. These are essentially plain-text, but required secondary software to interpret and represent correctly. The personal files of developers were copied using flash drive connected with their laptops. All online web content was archived into WARC format by the Internet Archive's Archive-It service.

Storage

UCSC (as well as the larger University of California library system) provide digital repository storage services through the Merritt online digital repository. Files stored for long-term preservation are indexed and searchable based on title-level descriptors (like title and creator). Larger groups of files are compressed and uploaded to the service where content listings of individual files are automatically created. To upload *Prom Week*, its documentation was broken up into 24 zip archives containing all digital documentation for the project and recordings of the interviews conducted for the NEH report.

The storage process involved:

1. Organizing all documentation into coherent chunks for compression

Prom Week's documentation was grouped (by the development team) according to online services. That is, the original born-digital file organization consisted of:

- Shared folders on Dropbox and Google Drive
- Email archives for development mailing lists
- Source code and finalized creative content in Subversion version control

Individual interviews with the development team were stored, with topical indexes, in separate compressed archives. All data stored on online services was left with its original file organization and hierarchy, even if that made organization less clear.

1. Creating a file manifest for the compressed documentation

The manifest is a spreadsheet describing the title (general description), original creation date and creator for each archive, in addition to a hashed checksum for file verification. Descriptions are necessary for search and indexing in the digital repository. Titles included an overview of the contents of each archive and (when applicable) the version of the storage software used.

1. Uploading finalized compressed content to the repository

After validating the files according to the generated checksums, each file was paired with its descriptive information and made available for download from the repository.

Prom Week's combined documentation thus included:

- Contents of the development team’s shared Dropbox folder
- Contents of the development team’s shared Google Drive folder in multiple formats
- A combined archive of email archives in mbox format for all project mailing lists
- 13 development team interviews
- An archive of the online demonstrations and publications (development blog) of the *Prom Week* development team as collected by Archive-It.Org’s web crawler
- The most current version of the source code and creative content stored in version control
- A plain-text dump of the contents of the version control repository
- A copy of the entire version control repository, including all code and content revisions

2.4 Conclusion

In our research-appraisal of *Prom Week*, we have touched on the history of its development processes, the context of their creation and the documentation that resulted from it. The point was to get a sense of the historical requirements for historical software documentation, both in its preservation and study, and to show the actual “haystack” of documentation alluded to by Mahoney at the beginning of this chapter. Our proactive approach revealed the ways in which digital documentation for software practice, which is now the dominant form of

documentary accumulation, resists historical investigation and reveals previously unknown contours of practitioner’s practice.

Here, we tried to display the “what” of the documentary record, and how that material could help substantiate a “how” and “why” for the production processes of *Prom Week*. It also highlighted the difficulty in dealing with digital records, and the importance of access to the development process as a paramount form of access to the development object. Everything discussed above is not contained inside the single executable .swf file distributed as the final, historical game object. Hopefully, we have shown how that file contains multitudes, and how it encapsulates and expresses the practices of *Prom Week*’s team. Imbricated as it were, with layers of their social context and technical backgrounds.

Echoing the opening of the chapter, the process of appraisal, through selection and retention, dictates the material record from which we can create historical narratives and to which we can target historical inquiries. Without the efforts of this chapter, the only remaining remnants of *Prom Week*’s development process would have been its final online release, and the publications describing some of its development and system design.⁵¹ The stored records of *Prom Week*’s development now include most of the team’s correspondence, its entire technical development, and hours of interviews about its design process. This allows for a much richer and multi-faceted look into its history, should a future scholar want to investigate the academic development process in the early 2010s, artificial intelligence and interactive narrative system design, or even the evolution and development of *Prom Week*’s code. The appraisal work above not only reveals the commonly hidden aspects of technical development, but also provides an organizational foothold for future historians that wish to roll up their sleeves and dive in. At least they have

⁵¹*Prom Week* being a research project in many ways created outputs that would not normally exist for other classes of games, in which cases only the final released game would remain.

something to dive into — and basic bearings to begin more specific investigations.

This chapter also serves as a preface to the coming elaboration on the development records of *Doom*. In conducting a full research-appraisal of *Prom Week*'s documentation, we can proceed — in the final chapter, “A Model of *Doom*” — to an analysis of *Doom*'s with some of the foundational methodological concerns taken care of. Migration will be difficult in some cases, identification and access in others. The documentary record will reflect the processes and context of the early-1990s instead of the early 2010s, but the “stuff” is still of software development. There is still development practice, release and dissemination and most of the other processes already described. Therefore, while this chapter laid out a bit of the accumulation of *Prom Week*'s history, of how its documentation was produced (and supported its study), it did not have a competing historical narrative or other documentary base to confront. With *Doom*, we do not have the luxury of starting from scratch in defining the historical basis for a construction of its narrative. This is good, in that there is an incredible amount of accumulation already present in the game's community. It is bad, however, in having to serve as a corrective or potential diminishing of some claims made about the game, its place in game history, and the practices of its development team.

A final goal motivating this chapter is to begin a conversation about the complexity inherent in software design and documentation, and ways to reduce both the technical and organization burden for historical scholarship.

Chapter 3

Description

3.1 Introduction

The construction of any historical scholarship is fundamentally based on the availability of its sources. For the study of computer game software, this means access to and description of game software in a way that allows it to be interpreted (or at the very least retrieved in some way) in the future. Modern cultural memory institutions, like libraries and archives, are the sites responsible for the maintenance of historical knowledge and artifacts. Therefore, they are also a necessary component of any stable, future practice in the historical study of computer games. This chapter, at its core, is about the description of computer games inside institutional collections, how that descriptive practice can benefit from collaboration with historically minded game scholars, and the process involved in couching descriptive work in both institutional standards and solid methodology. Heartily implicated in this endeavor is the vast and somewhat confusing network of standards, guidelines, and practices for collection cataloging and metadata. Marrying these processes to more coherent, practical structures that can guide institutions and scholars to some common understanding is a fraught undertaking. There is

a lot of conceptual grounding needed to communicate the constraints of collecting institutions to scholars, and conversely, to explain the needs of scholarship to institutions. A further problem is that the relative nascence of computer game history means that the needs of the field are still in development. Historians in the future might not have a coherent understanding of what information will be useful, which complicates the discussion of their current needs. Throwing the inherent and growing complexity of software objects — and their networks of dependencies — into the morass only further clouds the waters. We end up needing to adopt historically contingent inter- and intra-institutional collection processes to cover the unknown future needs of scholars dealing with inherently unstable technical objects.

While it may feel like — to ape a bit from Michael Mahoney in his “Histories of Computing” — we are pacing before a dense jungle thicket, pensively holding a machete and trying to find a way to hack on in, we can also look skyward and use that same instrument to cut down some low hanging fruit.¹ In the case of this chapter, our thicket is the tangled web of the scholarly, collection institutional, and ontological concerns of computer game software. Our low hanging fruit is a description of the computational platforms and media formats that support the expression and dissemination of software.

We use an elaboration of a controlled vocabulary for such platforms and formats as something to chew on while considering the dense network before us. We will briefly introduce the concept of a controlled vocabulary and then describe,

¹“Finally, there is a small body of professionally historical work, dealing for the most part with the origins of the computer, its invention and early development . . . It is meant as no denigration of that work to note that it stops at the point where computing becomes a significant presence in science, technology, and society. There historians stand before the daunting complexity of a subject that has grown exponentially in size and variety, looking not so much like an uncharted ocean as like a trackless jungle. We pace on the edge, pondering where to cut in.” [132] Mahoney, Michael S. “The History of Computing in the History of Technology.” *Annals of the History of Computing* 10, no. 2 (1988): 113-125. pg. 115

in the remainder of this introduction, how it will help us make better sense of the needs of software objects in historical collections. This includes how game scholarship can lend a hand in better articulating its historical goals and needs, and a more detailed description of the explicit steps necessary to construct the vocabulary.

3.1.1 A Brief on Controlled Vocabularies

A controlled vocabulary, to use a definition from Arlene Taylor’s *Organization of Information*, is “a list or database of subject terms in which all terms or phrases representing a concept are brought together. Often one of the terms or phrases is designated as the preferred term or authorized phrase to be used in metadata records in a retrieval tool.”² There is a bit to unpack in that definition, coming as it does from an information science textbook, but the basic statement is that a controlled vocabulary is a maintained, stabilized, and authoritative set of descriptions for some specific field in a metadata schema. The mention of “subject terms” is specific to the context of records in institutional collections, in which explicit terminology for the “main idea” is required to conceptually link topical areas. The most prevalent controlled subject vocabulary is the Library of Congress Subject Headings (LCSH). Any record in a finding aid at a research library will have a “subject” field populated with LCSH terms.³ The terms in any vocabulary need to be substantiated and vetted, and be specific to the task at hand. LCSH terms have actually come under significant scrutiny, as their use in records sometimes says more about the categorical opinions of the cataloger than

²[204] Taylor, Arlene G. *The Organization of Information*. Westport, Conn.: Libraries Unlimited, 2004. pg. 334

³For example, the entry for the Modern Library edition of “Moby Dick, or, The whale” in the UCSC online finding aid lists six LCSH subjects: “Ahab, Captain (Fictitious character) — Fiction”, “Whaling ships — Fiction”, “Ship captains — Fiction”, “Mentally ill — Fiction”, “Whaling — Fiction”, and “Whales — Fiction”.

the object in question.⁴

Although some controlled vocabularies are strict lists of terms mapped to preferred broader terms, other types of structures are also commonly considered to be controlled vocabularies. Taylor includes both thesauri (strict hierarchy) and ontologies (flexible hierarchy) in her extended definition of vocabularies.⁵ Our vocabularies, as elaborated in the “Vocabulary” section below, fall somewhere between a preferred term listing and an ontology. An ontology, in this categorization, is simply the most potentially expressive form of a controlled vocabulary.⁶ Ontologies “[bring] together all the variant ways of expressing a concept and [show] the relationships of a concept to broader, narrower, and related concepts.” The relationship between computer game platforms and media formats is an attempt to reconcile both the preferred designations for each term, and how those terms related to things like localization and hardware / software dependencies. Another key aspect of ontologies is that they “aim to capture consensual knowledge in a generic way” in a “model agreed upon by a community.” In our case, the controlled vocabularies we will present are a combination of game community, game academic, and library knowledge bases. And as we mentioned above, this type of functional interaction between memory institutional and scholarly conceptions is exactly the type of approach needed to deal with the dense network we find ourselves outside, poking with an analytical machete.

⁴Library of Congress Subject Headings are particularly problematic for games, a topic we will return to later in this chapter, and in discussion of game categorization in the Discovery chapter. For more information on the problems with subjectivity and LCSH terms see, [106] Knowlton, Steven A. “Three Decades since Prejudices and Antipathies: A Study of Changes in the Library of Congress Subject Headings.” *Cataloging & Classification Quarterly* 40, no. 2 (2005): 123-145. and [199] Stone, Alva T. “The LCSH Century: A Brief History of the Library of Congress Subject Headings, and Introduction to the Centennial Essays.” *Cataloging & Classification Quarterly* 29, no. 1-2 (2000): 1-15.

⁵Taylor considers thesauri to be “strictly hierarchical” because all entries connect a single term to its “narrower” synonyms.

⁶Footnote elaborating on ontologies in CS knowledge organization as a contrast to the more limited role they take in library land.

3.1.2 A Course Through the Thicket

The controlled vocabularies organize communal knowledge about computer game platforms and media formats into a structured ontology that is then mapped to relevant positions in the world of library and archival cataloging. The approach we are taking will fully describe the vocabulary, its construction and its intent, and then show how it flows into the metadata and information retrieval structures of collecting institutions.⁷ This way, we will only need to involve contextual description of the facets of library and knowledge organization practices germane to future game historical practice.⁸ The intent is to show game historians the material and descriptive constraints imposed in the institutional formalization of bibliographic records, and what those constraints mean (and how they should be amended) for historical software objects.

In describing computer games through formalized vocabularies and records, this chapter is a shift in focus from the material concerns of the last. Appraisal dealt with the material constraints, description and storage of documentation about the creation and formation of computer games as enclosed objects.⁹ It peeled back the development curtain to reveal the mess and historical processes of production. Here we are no longer concerned with evidence of a computer game's development, but with its existence as an object. Institutions and scholars

⁷This work resulted from a collaborative effort in game metadata and ontology, the Game Metadata and Citation Project. As such, while much of the theorization and organization of the ontologies is my own, it is based on a significant collaborative effort. Thus the “we” plural is used, similarly to some of this chapter's source text, which appeared in [104] Kaltman, Eric, Noah Wardrip-Fruin, Mitch Mastroni, Henry Lowood, Greta de Groat, Glynn Edwards, Marcia Barrett, and Christy Caldwell. “Implementing Controlled Vocabularies for Computer Game Platforms and Media Formats in SKOS.” *Journal of Library Metadata* 16, no. 1 (January 2, 2016): 1-22. doi:10.1080/19386389.2016.1167494.

⁸Our approach also tries to avoid the deafening cacophony of acronyms unleashed in most discussions of information management standards.

⁹“Enclosed” in the STS sense; having reached “closure” in social and technical discourse to the point of being a “black box”. No longer subject to general introspection or change. See the introductory chapter to this thesis, specifically footnote 8.

must manage that existence so the object is still discoverable and accessible in the future. This work is then a contribution to the shared knowledge of both institutions interested in gaming collections and the scholars eager to work with them.

For institutions, the description of an object in a collection invites a raft of different potential use cases and divergent assumptions about descriptive necessity. As Jerome McDonough stated in the findings of the Preserving Virtual Worlds (PVW) game preservation project, collections are curated — and archives are described — for use by the “designated community” of an institution. In his case, at the University of Illinois at Urbana-Champaign (UIUC), this designated community is “the faculty, staff, and students of UIUC, as well as . . . visitors to the campus and members of the general public.” This presented a problem because the archival standard he needed to use, the OAIS model for preserving digital archival information, derived from the aerospace community and their relatively homogeneous data use needs and consolidated knowledge bases. Adapting OAIS for computer game software, in accordance with the potential use case of everyone who could visit UIUC’s research library and archives, would not be straightforward. The contextual information attached to a game record is significantly different in a case where a future scholar might want to actually run the software as opposed to observe its packaging and read the manual. McDonough asserts that the game community, and the community of game scholars should work to provide input and a basic knowledge base that libraries and archives can use to improve their records practices. A task we mildly take up with the vocabularies in this chapter.

Another concern highlighted by McDonough, and echoed by other preservation minded scholars, is the need to clearly define the boundaries separating digital

games from one another.¹⁰ With different versions, platforms and formats abounding over the past 40 years of game history, making sure to include the correct information needed to interpret a game in the future is particularly daunting for institutions. Couple this with the expectation that future software collections will require emulation, and you present institutions with the additional responsibility of preserving software's contextual documentation and dependencies. Clearly there is a need to begin organizing descriptive practices for software versioning, dependencies and derivatives, along with the (in information parlance) representation information required for reconstituting executable programs. Again, McDonough recommends splitting the work and allowing institutions to collaborate on different standardizations for the description of the content in their collections that might be useful to others. Our vocabularies fit into this recommendation, as will be shown the dissemination work below.

The controlled vocabularies, therefore, begin to address a fundamental need for game scholars to help in both the articulation of their practical needs for games in collections, and in the contextualization and organization of knowledge about them. For game scholars, the idea that games are a highly variegated art form representing a host of different ontological distinctions and properties is not new. There are many ontologies and formal characterization schemes for games. What is new here, however, is focus on the material description of games as historical objects in need of retrieval and representation in the future. Rarely are the historical material dependencies and differences between computer games brought to the fore in ontologies or classification schemes; concerned as they are with games as systems of play and interaction, rather than as systems of

¹⁰See [161] Newman, James. "Ports and Patches: Digital Games as Unstable Objects." *Convergence: The International Journal of Research into New Media Technologies* 18, no. 2 (2012): 135-142. and [127] Lowood, Henry. "The Hard Work of Software History." *RBM: A Journal of Rare Books, Manuscripts and Cultural Heritage* 2, no. 2 (2001): 141-160.

technologies. This chapter should serve as a prompt for a closer look at the mundane technical dependencies of games as software objects by scholars. The controlled vocabularies are organized around assumptions about the descriptive needs for future scholars.

On a higher level, this chapter is dealing with the organization and description of knowledge about computer games and software, and attempting to ground that knowledge in sound ontological theory. By “ontological theory” we mean to draw on work in the “conceptualization” of objects — how they function as concepts in organization schemes — and the practical recommendations of the library and information science community regarding the labeling and hierarchical description of items in collections. There is a growing body of work in the description of computer game and software records,¹¹ but many are only theoretical and not effectively attempting to mediate the practice of game historical scholarship into their descriptive frames or methodologies.¹² The challenge is to find a way to couch our descriptive methodology in approaches that speak to the future of game studies, software studies, and game historical needs, and make sure they are stabilized in ways commensurate with many potential future use cases.

The goal is to outline the controlled vocabulary work and position it such

¹¹Espen Aarseth has maintained a steady publication stream of ontological discussions of games, for examples see: [21] Aarseth, Espen, and Gordon Calleja. “The Word Game: The Ontology of an Undefined Object.” In *The Philosophy of Computer Games Conference*, 2009. [23] Aarseth, Espen, Lev Manovich, Frans Mäyrä, Katie Salen, and Mark JP Wolf. “Define Real, Moron!” *Some Remarks on Game Ontologies*. In Stephan Götz, Michael Liebe & Dieter Mersch (Eds.), *DIGAREC Series 6* (2011): 50-69. [22] Aarseth, Espen, Solveig Marie Smedstad, and Lise Sunnanå. “3. A MULTI-DIMENSIONAL TYPOLOGY OF GAMES,” 2003. Also, [234] Zagal, José P., Michael Mateas, Clara Fernández-Vara, Brian Hochhalter, and Nolan Lichti. “2. Towards an Ontological Language for Game Analysis.” *Worlds in Play: International Perspectives on Digital Games Research 21* (2007): 21.

¹²One notable exception is Jin Ha Lee’s work in use-cases for games’ metadata, which provides some basic archetypes for scholarly use. See [122] Lee, Jin Ha, Joseph T. Tennis, Rachel Ivy Clarke, and Michael Carpenter. “Developing a Video Game Metadata Schema for the Seattle Interactive Media Museum.” *International Journal on Digital Libraries* 13, no. 2 (March 1, 2013): 105-17. doi:10.1007/s00799-013-0103-x.

that we clear a little space for thought about the future interface between game scholarship and game stewardship. We begin the chapter with a description of the purpose of the controlled vocabularies, and how they are designed to help with the description and location of games in archival and library collections. After that, we elaborate on the ontological structure of the vocabularies, and the issues associated with delineating their individual terms. This makes use of a practice of “ontological enactment”, whereby the description of objects in an ontology is tied to their networks of use. Here we also confront problems with versioning, internationalization, and peripheral dependencies. This leads to the conception of a criterion of “reasonable compatibility” for the vocabulary terms aimed at ensuring the future scholars can make “reasonable” guesses as to the platform requirements for particular items. The chapter continues with a description of where the terms fit into the library retrieval systems in use by collection institutions. This invites a brief description of the historical treatment of games in the library retrieval apparatus. It also provides a more detailed elaboration on the issues of community use and object description mentioned above. The chapter closes with a description of our efforts to disseminate information about the vocabularies to the library community, and how the vocabularies themselves function in a future library landscape of semantically linked data. A key concern throughout is the revelation of the practical conditions of computer game software in memory institutions, and how to make those conditions more open and comprehensible to game historical scholarship.

3.2 Controlled Vocabularies

Creating vocabularies is a significant undertaking, mostly due to the sticky nature of developing any formalized ontology. During the initial formalization, a

vocabulary is arranged based on the ontological commitments of an “aboutness statement”; what are the terms in the vocabulary “about” and what function are they intended to serve? As Arlene Taylor writes in her general principles for controlled vocabularies, “in some cases, it may take staggering effort to remain faithful to the aboutness of the resource, while following specific application rules in addition to following general principles of applying controlled vocabulary.” The terms in a vocabulary must be consistent in their relationships with one another, and be readily applicable to their targeted descriptive domain. For computer game platforms and media forms, their aboutness and application proved a bit more problematic than we initially assumed. (In fact, we chose both because we thought they would be more objectively salient than other, more subjectively tangled descriptors like genres or subjects.) Therefore, the main goal of the vocabulary work was (and is) to provide a stable and authoritative set of labels for platforms and media formats for game records in retrieval systems. By promoting a unified, coherent and consistent set of terms, the vocabularies are designed to enable better description for future scholars.

This section outlines the controlled vocabularies’ development according to the methods of the library science and knowledge representation communities. It lays out the basic descriptive problems encountered while trying to stabilize the concept of “computer game platform” within the historically disjunctive progress of technological development. Implicit in any description of a platform is the media formats that it accepts, since those formats are what is actually being described and placed into collections.¹³ Delineating platforms then also involves linking to its corresponding media formats. The vocabularies encode these links into a linked data format known as the Simple Knowledge Organization System

¹³Some of the collections we mention below do contain platform hardware as well, but the overwhelming majority of game records are records of game data stored in a media format.

(SKOS), a machine-interpretable ontological language designed for semantic web applications. After addressing the aboutness criteria for the vocabularies, we elaborate on the vocabularies' encoding schema and conclude with the steps taken to disseminate it to the greater library community.

3.2.1 A Brief Record Example

Before addressing the ontological issues with a definition of “platform,” a brief example of the basic problems with game records is in order. As outlined in the PVW final report, a major concern for any modern collection (archival or librarial) of software objects is the ability to accurately represent them in the future.¹⁴ This implies the need to either keep working hardware (and dependent software) available for patron use or migrate the software data from physical formats into emulated environments supported by a digital repository. There is a general assumption that physical data records and hardware will eventually degrade, leaving migration and emulation as the ultimate source for future historical investigation. Regardless, if a scholar currently wants to investigate a software record in a library catalog — either to find it physically and play it or to run its data through emulation — the record itself should be well described. This means that the information in the record should allow for a scholar to know if the resource has the specific technical properties to be worth further investigation. Because most catalogers and archives do not deal with software records, there is sometimes not enough expertise available to describe a software resource in a consistent and coherent

¹⁴The use of “represent” here is derived from the language of the report and the general guidelines of the library community. Representation of a piece of software means its use in the future by a patron of the library. Later in this thesis, particularly in the Citation chapter, we discuss what “representation” means in relation to the execution of software data (and how that term might not quite fit the bill). Library and archive professionals are concerned with the practical issues of “representation” of digital data, but not usually with the grander metaphysical implications of such language.

fashion. This leaves more work for the scholar, who might retrieve the item and find out it is incompatible with her needs.

The PVW report provides two examples of deficient description. The first is a record of the game *Doom* in Stanford’s online finding aid. The record is detailed but does not remark on the version of *Doom*, simply stating that it is “Doom. Episode 1.” dated “November 1994.” This leaves one to assume (if one is a *Doom* scholar) that the version in question is a physical shareware distribution of *Doom* version 1.666.¹⁵ Additionally, the system requirements are directly copied from the packaging as:

System requirements: 386 or better IBM compatible PC; 4MB
RAM; DOS; VGA; CD-ROM player

This description also leaves a significant amount of contextualization work to the historian. “DOS” is probably a version of Microsoft’s “MS-DOS” and based on the date most likely version 6.22. “VGA” is a reference to the specifications of the video display (and capabilities of the graphics drivers) needed to run the game. “386 or better IBM compatible PC” refers to the compatibility class of the CPU. In this case a version of the x86 instruction set. And so on.

The second example in Table 3.1, from the Stephen Cabrinety Collection in the History of Microcomputing at Stanford University, provides a record for the game *Star Raiders* from that collection’s archival finding aid.

Even by the standards of the previous example, these records leave a bit to be desired. However, they are representative of most archival records, where, according to the report, “item-level description is minimal at best.”¹⁶ As with the

¹⁵The Preserving Virtual Worlds Final Report [149] actually notes that “by November 1994, DOOM was already in version 1.7a; in fact versions 1.2 through 1.7a were all released during the first eleven months of 1994”. In fact, version 1.7a is specific to *Doom II* releases.

¹⁶In fact, many archival collections are usually only described to the box-level, leaving the exact contents a mystery until some researcher stumbles along and sorts it out.

Table 3.1: Star Raiders Entry from Cabrinety Collection Finding Aid

Box	Company	Title	Year	Physical Description
136	Atari, Inc.	ST Star Raiders	1986	1 computer disk; 5 1/4 in. Atari
72	Atari, Inc.	Star Raiders	1982	computer cartridge Atari
134	Atari, Inc.	Star Raiders	1982	computer cartridge Atari
142	Atari, Inc.	Star Raiders	1980	computer cartridge Atari
152	Atari, Inc.	Star Raiders	1982	computer cartridge Atari

Doom example, historical assumptions take the place of information that could just appear in the record. To quote from the PVW analysis:

Presumably we can depend on the fact that Box 136 contains the version of Star Raiders for the Atari ST, but as for the rest, whether the cartridge in Box 142 is intended for an Atari 2600 system or one of the Atari 400/800 systems is unclear, and for the remaining three boxes, the cartridges could be in theory be for the Atari 400/800, 2600 or 5200 systems. And only a scholar with knowledge of the exact release dates for the various editions and platforms would be able to deduce that information from this finding aid.¹⁷

Both examples serve to show the motivation for the creation of a controlled vocabulary for platforms and media formats. The *Doom* record lacks specificity for its basic system requirements because it copies a contemporaneous description of its dependencies. At the time, “IBM PC compatible” and “DOS” may have been good enough descriptions for someone purchasing it in a store or ordering it from id Software. Today, however, more historical context needs to be embedded in those records to prevent future contextual confusion. The platform vocabulary provides an explicit, standardized reference instead of the highly fragile description on the resource. In our case, “DOS” becomes “Microsoft DOS Version 6.22”, and that label is linked to further contextual description and even other versions of the platform.¹⁸ We will address the use of “platform” as a proxy for “oper-

¹⁷[149] McDonough, Jerome P. *Preserving Virtual Worlds: Final Report*. Graduate School of Library and Information Science, University of Illinois at Urbana-Champaign, 2010. pg. 35

¹⁸Many times box cover requirements only consist of these contextual, contemporaneous ref-

ating system” below. Because we are still only twenty or so years removed from *Doom*, all the information in the record is still relatively interpretable. As time progresses, however, more and more of the “obvious” historical context for *Doom* — and other games and software — will require more specialization and research to retrieve and interpret.¹⁹ By linking some of a game’s technical requirements to a standardized and supported vocabulary, we stabilize its description for future historical research.

The Star Raiders archival example also highlights the need for a better description of physical media formats. Most records of historical software are also records derived from their material existence. The data for Star Raiders is conveyed to its platform through the physical carrier of an Atari cartridge. As noted above, the basic description in the Cabrinety record does not provide enough detail to ascertain what version of the game is in each box. Therefore, any potential platform and system requirements are also a mystery. With a controlled vocabulary of media formats, the level of description is targeted to the specific platform of a resource. In the above 1980 version of Star Raiders the ambiguity of “computer cartridge Atari” would be replaced with “Atari 2600 cartridge”.

A more practical reality underlies the lack of specificity in both examples. Working with cataloging standards and descriptions is onerous and difficult, especially if there are no explicit rules or standards tailored to the objects in the collection under review. In the *Doom* example, a professional Stanford University librarian created the record without much awareness of future scholars’ descriptive

erences. Part of the goal for the vocabulary work is to help non-specialist better describe materials, in which case the “DOS” reference, but a date range might allow for a narrower initial description. More on this below in our discussion of the labeling hierarchy in our semantic web constructs.

¹⁹*Doom* is still one of the most well-known games of all time, so while — in practical reality — it might remain more accessible, other titles from the same period will most certainly not receive the same treatment, and therefore greater occasion for loss.

needs. In the second archival example, finding anyone with descriptive expertise in game archives is going to be a tall order, given that there are only a handful of experienced game archivists on the planet (including the one writing these words). The lack of clarity in the Cabrinety Star Raiders record is therefore indicative of common descriptive challenges for archival collections. Finding someone who could correctly identify and describe an Atari 2600 cartridge, including where to look for the relevant information, would already be difficult. Add in the knowledge of how to — in a future conscious and consistent manner — map that information to common archival metadata fields, and it begins to make sense that most archives are happy to just have a note about the general contents of every box or folder, let alone the items inside them.²⁰ Time is also a significant consideration. For example some boxes in the Cabrinety collection may contain 50-60 different software titles, for a variety of platforms in a variety of formats. Simply attending to all the item descriptions, even with specialist knowledge, is a daunting proposition. Conducting it without leads to descriptive errors or a lack of coverage. To address these practical concerns, the controlled vocabularies try to contextualize all the information they provide, link that information to further relevant resources, and in the case of media formats, include images and illustrations to allow for visual comparison in identification.

To sum up, the controlled vocabularies formalize the information present in game and software records to help with future contextualization and comprehen-

²⁰In many cases, even having a trained expert in a field might not be enough to overcome the incompatibility between a cataloging data format, the ontological condition of its object, and the pedagogical needs of patrons. From [232] Yasser, Chuttur M. “An Analysis of Problems in Metadata Records.” *Journal of Library Metadata* 11, no. 2 (April 2011): 51-62. doi:10.1080/19386389.2011.570654, “it was found that trained librarians used many more elements than could support the purposes of the project, but they could not capture contextual aspects of the resources and neither resource authors nor trained librarians were able to handle pedagogical aspects of the resources.” In other words, more training lead to ineffective (and over described) records.

sion issues. Additionally, as most current physical formats will have their contents migrated to digital collections; assuring that the initial record provides some indication of the data’s format is imperative to providing for stable future access.

3.2.2 Vocabulary and Ontology Best Practices

Creating a vocabulary from scratch requires relying on the practical and categorical recommendations of the information science and knowledge representation communities. This section briefly describes the best practices followed in the construction of our platform and media format vocabularies, and outlines the guidelines we followed from both the information science and ontological engineering communities.

Definition

Technically, the controlled vocabularies are semi-formal domain ontologies. They are semi-formal because they are “expressed in an artificial and formally defined language” but not to the extent of “formal semantics, theorems, and proofs.”²¹ In this case, they are expressed in the Simple Knowledge Organization System (SKOS) a subset of the Resource Description Framework (RDF). RDF is a subset of the Web Ontology Language (OWL), a formal description logic.²²

²¹[80] Gómez-Pérez, Asunción, Mariano Fernández-López, and Oscar Corcho. *Ontological Engineering*. Vol. 139. Springer Heidelberg, 2004, contains a significant literature review of the classification and organization of ontologies. They define four classes of ontological rigor: highly informal, semi-informal, semi-formal and rigorously formal. Most historic controlled vocabularies would be highly informal or semi-informal (depending on their application) because they are expressed in less constrained, natural languages instead of more formally defined ones.

²²OWL actually has a variety of restricted implementations that allow for differing levels of formal rigor. The OWL-DL is the most formal specification of the language, allowing for completeness proofs and other formal verifications well exceeding the needs of our controlled vocabulary.

Aside from formal classifications, another means of categorizing ontologies is by the breadth of their subject matter. Top-level (or upper-level) ontologies like Cyc or the IEEE’s Suggested Upper Merged Ontology (SUMO) provide for the ontological organization of foundational reality. They include categorizations that begin at general concepts like “Thing” (Cyc) or “Entity” (SUMO) and work down from there.²³ Top-level ontologies usually form the basis for a descending trail of lower ontologies.²⁴ Domain ontologies are positioned right below the top-level as they are specific to a certain professional domain, but not designed for a specific set of tasks or direct application in a specific knowledge management system.

The controlled vocabularies are domain ontologies. They define a limited set of concepts and are application agnostic; being a set of unified terms not targeted toward a specific knowledge organization system or application. The vocabularies are also not currently linked to higher-level terms in any top-level ontology.²⁵ They do, however, represent a hierarchical knowledge set with a deeper tree than many controlled vocabularies. This is partly a result of their encoding into SKOS, which provides for and enforces conceptual hierarchies for use in the semantic web. Even though controlled vocabularies are, in some cases, considered to be the simplest form of ontology, the controlled vocabularies below, in their formalization through

²³“Entity” entry in [13] SUMO (<http://54.183.42.206:8080/sigma/Browse.jsp?kb=SUMO&lang=EnglishLanguage&flang=SUO-KIF&term=Entity>). “Thing” entry in [9] OpenCyc, <http://sw.opencyc.org/concept/Mx4rvViA9JwpEbGdrcN5Y29ycA>. “Thing” is not a root of the Cyc ontology because the ontology has a formalized understanding that it is an ontology. That is, “cyc vocabulary term” and the notion of the Cyc ontology as a collection of collections of terms is embedded within Cyc itself.

²⁴Again, [80] *Ontological Engineering* [80] describes six basic ontological levels: top-level, domain, task, domain-task, method and application ontologies. The more specific the ontology, the less reusable and shareable its content.

²⁵Equivalence relationships might be possible but are outside the scope of the current work. Both the aforementioned top-level ontologies Cyc and SUMO provide concepts for computer hardware, software, etc. So the mapping should be pretty straightforward. However, based on the practices of the information and library science communities (where no controlled vocabulary we could find was tied to KR ontologies) these connections are still waiting to be made (and constitute potential future work).

SKOS, expand the expressive power of the “controlled vocabulary” concept.²⁶

Ontological Criteria

The process of organizing the conceptual basis for an ontology is known as “conceptualization”, an “abstract, simplified view of the world that we wish to represent for some purpose.”²⁷ This is similar to the “aboutness” of controlled vocabulary, and is used as a way to explicitly contain the scope and purpose of an ontological set of relations. When developing any set of controlled terms or concepts, we also inure ourselves to a set of specific ontological criteria and qualifications.

Ontological criteria, drawing on design philosophy from Gruber, refer to a set of implicit agreements between an ontological conceptualization and its use in practice. Gruber outlines a number of foundational commitments, most of which map to similar concerns in controlled vocabulary guidelines.²⁸ They are: clarity, coherence, extendibility, minimal encoding bias, and minimal ontological commitment. Clarity and coherence are pretty much a given. If an ontology does not clearly delineate terms and / or allows for unintended inferences based on

²⁶[117] Lassila, Ora, and Deborah McGuinness. “The Role of Frame-Based Representation on the Semantic Web.” *Linköping Electronic Articles in Computer and Information Science* 6, no. 5 (2001): 2001, organizes an ontological spectrum, with “catalog / IDs” falling at one end, and “general logical constraints” at the other. “Catalog / IDs” are described as controlled vocabularies in that they are “a finite list of terms”. The intent of our vocabularies actually falls under their definition of a “thesauri [with] ‘narrower term’ relation”; that is, “a list of terms and meanings . . . [with] some additional semantics.” The implementation of the vocabularies also places them closer to the logical constraint end of the spectrum, since SKOS, by their definition, is “frame-based.” This means that each vocabulary term is also a class of “Concept” with “slots” (properties) and formal (is-a) hierarchical (narrower / broader than) relations.

²⁷[84] Gruber, Thomas R. “Toward Principles for the Design of Ontologies Used for Knowledge Sharing?” *International Journal of Human-Computer Studies* 43, no. 5 (November 1, 1995): 907-28. doi:10.1006/ijhc.1995.1081. pg. 908

²⁸See chapter 10 in [204] Taylor, Arlene G. *The Organization of Information*. Westport, Conn.: Libraries Unlimited, 2004. and [16] “ANSI/NISO Z39.19-2005 (R2010) Guidelines for the Construction, Format, and Management of Monolingual Controlled Vocabularies - National Information Standards Organization.” National Information Standards Organization, 2005.

relationship predicates it probably needs a bit of reworking.²⁹ Extendibility refers to an awareness of the domain of use and how it may change over time. This is similar to the concept of maintenance in controlled vocabulary guidelines. You want any knowledge organization to adapt to new requirements in the domain, and to provide a convenient means for adding new concepts or altering some hierarchical relationships. Since each term in the vocabularies is a SKOS Concept node (with each leaf node representing a specific term entry), manipulating the arrangement of terms is relatively easy.

Enforcing a minimal encoding bias means that your ontology “should be specified at the knowledge level without depending on a particular symbol-level encoding.” This removes a potential bias that can develop if an ontology is constructed with a particular encoding or data format in mind. The structural limitations of that encoding or data format can constrain the expressive power of the ontology, and tie it to a specific syntactical model that might not be as shareable with other systems. Controlled vocabulary guidelines call for a greater “interoperability” between vocabularies and different knowledge management systems. In many areas of knowledge organization there is generally a proliferation of data formats, encoding schemes and systems. Any contribution should attempt to be widely applicable, especially if the knowledge is intended for a top-level or domain level organization.

The controlled vocabularies we developed suffer a bit from the encoding biases of the SKOS system. A detailed discussion is found in the SKOS section below. Briefly, the Open Metadata Registry (OMR), a non-profit SKOS semantic web server, hosts a practical implementation of SKOS that is available for institutional

²⁹Gruber’s notion of clarity also comes with an assumption (with which we disagree) that an ontology is an objective structure. One can try to form definitions without the influence of social factors or the implicit constraints of a domain, but they will never be completely removed. See next note for more.

use. They use a subset of SKOS, which prevented the use of some predicates. Instead of modeling the vocabulary using the full set of SKOS predicates, we intentionally limited the vocabularies' expression to those that were available for active use. We felt that getting the vocabulary embedded in systems in current use by institutions, and including more terms instead of more specifics was the better approach. Regardless, the point about encoding bias is valid, and should mindfully be considered by anyone looking into ontology or controlled vocabulary construction.

The last ontological criterion is minimal ontological commitment. Basically, if you are defining a domain of knowledge, make sure that you keep scope in check. Each term should be specific to the limited intent of your domain, and not try to encapsulate any more than is necessary. This criterion is tied to the notion of “specific and co-extensive entry” in controlled vocabularies, where you want each term to adequately cover the extent of the domain and also be in specific alignment with the object that it represents. For example, if you are developing an ontology of cats you need to decide if you are describing specific breeds, and if so, to what granularity. If you encounter a mixed-breed do you now apply two different breeds to a single cat, or create a new “co-extensive” entry to cover this particularly contingent kitty? Also, is the particular breed specific enough for your use, or would the same breed in different regions need to be considered? If so, you want a more “specific” entry for that cat, “North American Tabby” over “Tabby”, and “Tabby” over simply “cat”. Specificity and co-extensivity are significant concerns in the controlled vocabularies since platforms, specifically, are a rather ill defined domain (as shown below). The level of effective granularity (localization region, television standard, etc.) paired with the ability for most computational devices to function as game platforms actually makes the construction of the platform

vocabulary predominantly an exercise in specificity and co-extension.

Ontological Enactment

Any ontological conceptualization involves the definition of concepts, properties, and relationships that implicitly shape the expressive potential of an ontology. This conceptualization is not neutral. The foundational act of categorization brings with it the socio-cultural assumptions of the person making the categories and also embeds the ontological norms of a particular domain.³⁰ This section elaborates on recent works in science and technology studies that talk of an “ontological turn” in which the organization and existential condition of objects is brought to the fore. This connects to both our larger thesis objective of tying methods from science and technology studies to issues of the stabilization of historical software records, and more locally, provides some background on the approach we took to the conceptualization of the terms in the controlled vocabularies.

Steve Woolgar and Javier Lezaun, in an introductory discussion of this turn for a special issue of *Social Studies of Science* describes the contribution of this ontological mindset:

The fundamental contribution to STS of the ontological turn is its power to draw renewed critical attention to objects the might otherwise appear “finished” or “ready-made,” to scrutinize those entities that a conventional STS analysis would often consider “black-boxed” and no longer controversial . . . Investigating the composition of ontological realities would thus be a way of challenging any presumption of order or completion in the world — especially those forms of order and completion that have been dear to STS scholarship.³¹

³⁰Cantwell-Smith, in *On the Origin of Objects* [194], refers to the foundational assumptions of any ontological scheme as “a priori commitments.” This is an echo of Latour’s (and other ANT practitioners) notion of an “irreduction criteria,” which basically means that any foundational categorization of the world is inherently flawed due to the limiting effects of its *a priori* axioms. Anyway you slice it, you’ve sliced it, so it (the slice) cannot actually represent the whole.

³¹[230] Woolgar, Steve, and Javier Lezaun. “The Wrong Bin Bag: A Turn to Ontology in Science and Technology Studies?” *Social Studies of Science* 43, no. 3 (June 1, 2013): 321-40. doi:10.1177/0306312713488820. pg. 323

Woolgar and Lezaun further elaborate on the “forms of order and completion . . . dear to STS scholarship,” stating that an ontological focus “short-circuits the tendency to rephrase questions about the reality of multiple worlds as questions about the multiple ways in which a singular world is represented.”³² Although this requires a bit more unpacking, the basic target of this methodological shift is the pluralistic underpinnings of the social construction of science and technology (or anything, really).

The ‘social construction’ in STS is based initially on Thomas Kuhn’s theories of paradigms and incommensurable worldviews.³³ He argued that the development of new scientific knowledge is less a progression of ideas through proof and agreement, than a competition between foundational epistemes. Different groups function with different foundational definitions about the possibility and significance of different structures. Kuhn essentially argued that scientific discourse, or the agreed upon state of scientific, rational “reality,” lay in the assumptions about that reality shared by a dominant group of practitioners. That reality shifted as certain anomalies within the contemporary episteme built up to a point that new perspectives were needed to address them. Generally, the younger members of the community, less invested in the current state of a field, pushed on these anomalies and created new and different perspectives. At a certain point, the general agreement about reality shifted because a majority of practitioners now ascribed to the new, modified world-view. That this view was a function of negotiation and time — since young outlasted the old — meant that social organization and community had a significant influence over rational discourse. It was an attack on the “positivist” and “realist” schools of thought that presupposed a clean and purely rationalist reality detached from human conception and perspectives. So-

³²[230] pg. 322

³³[112] Kuhn, Thomas S. *The Structure of Scientific Revolutions*. 3rd edition. Chicago, IL: University of Chicago Press, 1996.

cial construction invited with it a “pluralistic” discourse, in which one needed to situate themselves within an assumed community and epistemic context before continuing with investigation (or the inverse, in which one presupposes that an investigation must lead to the explanation of a single perspective that sits in relativistic relation to many others).³⁴

Another facet of “social construction” implicit in Woolgar and Lezaun’s comments is the social construction of technical objects. A great example of this is found in Bijker’s work on the history of the bicycle.³⁵ Briefly, the current design of the bicycle, with two same-sized inflatable wheels, did not arise through a long, logical process of refinement and retuning, but from a crush of different constituencies’ views on what a bicycle should be and whom it should be for. The fundamental conceit, however, is that after long periods of negotiation and social fracas, certain forms of a technical object stabilize while other competing ideas fall by the wayside. This “closure” of innovation over the technical object, in this case the form of the bicycle with two inflatable wheels (as opposed to the asymmetric and quicker design of the penny-farthing), effectively erases the fraught historical period during which there was no unified, dominant form of “bicycle”. The current bicycle, once stabilized, sheds the history of its formation, and becomes a historical “black-box”, no longer the site of intense innovative social forces. The “black-boxes” mentioned by Woolgar and Lezaun are a whole assortment of similarly constructed objects that are now used uncritically as the basis for theorization. The move toward ontological concerns and again towards

³⁴The notion of “social construction” arose gradually out of the work of Kuhn and many others over the course of the 1960s, 1970s, and 1980s. This brief note is just pointing out that Kuhn did not use the term — or even conceptualize his conclusions as supporting it — since he preceded (and in some ways founded) considerations of “social construction”.

³⁵See chapter 2 “King of the Road: The Social Construction of the Safety Bicycle,” in [34] Bijker, Wiebe E. *Of Bicycles, Bakelites, and Bulbs: Toward a Theory of Sociotechnical Change*. Cambridge, Mass.: MIT Press, 1995.

these “black-boxes” is also one that does away with the concept of something that is closed and foundational.

To ground this again in the bicycle example, the closure of its current form is, by the ontological move, never actually complete; even if its form is stabilized, its use and meaning to various discourses is still in flux. To get supremely local, a mountain bike on the campus of UCSC — a significant destination for mountain bikers — is either a source of enjoyment and adrenaline or an agent of destruction for local mountain trails. In this way, analysis of the bicycle vis-a-vis concerned locals and foreign thrill-seekers opens up the mountain bike to renewed ontological scrutiny. In the social constructionist mode, both groups exist in different world-views with perspectives on the correct use of the mountain bike. In the ontological shift, the use of the mountain bike by a rider, and its interaction with the path, create the occasion for both opposition viewpoints. That is, the organization of reality here is created by the actions and interactions of the bike, people, and path. Woolgar and Lezaun summarize,

Objects are brought into being, they are realized in the course of a certain practical activity, and when that happens, they crystallize, provisionally, a particular reality, they invoke the temporary action of a set of circumstances. . . the new approach eschews the implication that the world pre-exists representational practices and favors instead the assumption that practices (which can no longer be considered merely ‘representational’) perform the world. (Emphasis in original)³⁶

This shift from social perspectives to ontological performance is neologized as “ontological enactment”. The objects in an ontology are “enacted in practice”, deriving meaning and worth from their interactions and movements. For our purposes, we take the idea of enactment and use it to frame the discussion of computational platforms. The vocabularies are intended for a specific use in the

³⁶[230] pg. 323-324

clarification of records, but they are actually communicating a specific use by the scholar or patron. By framing the platforms as engaged in a form of enactment, we imagine them as defined by the choreography of their use.³⁷ The scholar-player, the data conveyance of the media format, and the “platform” apparatus all work fulfill the play experience. The enactment of a platform, therefore, forms the basis for its conceptualization in our controlled vocabulary.

3.2.3 “Aboutness” of Platform

A computer platform is a notably indistinct entity. Definitions from common reference sources are contradictory and ontologically confusing. Wikipedia states that a “computing platform” is, typically, “a hardware architecture, an operating system (OS) and runtime libraries” used to run a piece of software.³⁸ However, they also list alternate types of platforms that contradict this definition and in some cases each other. There is a constant conflation of “operating system”, “system”, and “console” with “platform” throughout the entry; in many cases the site uses the terms interchangeably. Merriam Webster also falls into an indistinct ontology, designating “operating system” and “platform” to be synonymous terms.³⁹ Clearly the constitution of a computing platform is a combination of different parts all enlisted and mobilized for the representation of software data.

³⁷[230] Woolgar in summarizing Cussins, “Other STS accounts have made use of the idiom ‘choreography’ to describe the practical work of alignment that creates a commensurate world.” pg. 325. Alignment is needed to coordinate and explain the ways in which a set of objective interactions can handle or bear the weight of multiple agents. In Cussins’s specific example, he recounts experiences with ethnographic research into infertility medicine, and how the patients engaged with the infertility apparatus existed in many different, though coordinated, subject positions based on their ontological positioning vis-a-vis doctors, other patients, and medical instruments. The choreography of these various ontological items (here people are also items) constitutes their meaning and agency in the world, and these ontological constellations must align and overlap to support the relative positions of the doctors, patients and instruments.

³⁸“Computing Platform,” Wikipedia. https://en.wikipedia.org/w/index.php?title=Computing_platform&oldid=772442103 (Accessed 3 Apr 2017)

³⁹While this may ring true in certain instances, they are hardly an equivalence class. As we will see, sometimes an operating system is part of a platform instead of the definition for it.

This section argues for an “aboutness” of platforms that is useful for the purposes of the controlled vocabulary. It looks at how platforms exist at multiple levels of abstraction, and shows that those abstractions cause the ontological confusion and diffusion seen in discussions about platform. Additionally, it proposes a soft metric of “reasonable compatibility” as the framing conceptualization for the vocabulary drawn from a consideration of the enactment of platforms in research and community practice.

The above definitions of computational platform are drawn from common reference sources. Neither Wikipedia nor the dictionary are specifically concerned with a definition of platform secure enough for a standardized list of terms. Even in communities dedicated to the study of computer games and their history, the concept of a “platform” is intentionally complicated and amorphous.

Probably the most well known critical consideration of game platforms is found in the Platform Studies collection edited by Nick Montfort and Ian Bogost for MIT Press. Books in the series show how the specific technical constraints of different computer game platforms limit and focus the expressive potential of the computer games developed for them. Notably, the definition of “platform” provided by the editors is all encompassing, and does not try to nail down a fixed set of criteria for what counts as a platform under review. From the introduction to the series from *Racing the Beam*, the inaugural platform studies text,

A platform in its purest form is an abstraction, a particular standard or specification before any particular implementation of it. To be used by people and to take part in our culture directly, a platform must take material form. . . This can be done by means of the chips, boards, peripherals, controllers, and other components that make up the hardware of a physical computer system. The platforms that are most clearly encapsulated are those that are sold as a complete hardware system in a packaged form, ready to accept media such as cartridges . . . In other cases, a platform includes an operating system. It is often useful to think of a programming language or environment on top of

an operating system as a platform, too. Whatever the programmer takes for granted when developing, and whatever, from another side, the user is required to have working in order to use particular software, is the platform. In general, platforms are layered—from hardware through operating system and into other software layers—and they relate to modular components, such as optional controllers and cards.⁴⁰

A platform in this definition is not a particularly fixed entity. It can be a collection of physical hardware parts, “chips, boards, peripherals, controllers, and other components”, or a “programming language or environment” or what is “taken for granted” in development or what is “required” for use. It can be an organization of physical or logical constructs (in the first two cases), or the potential of the constructs as enacted by developers or players. A “platform” in some cases has a material identity and a set of ontologically distinct parts, and in others is defined through those parts and their enactment in practice. In the post-modern veins of deconstruction and subjectivity, the lack of a center for a definition of platform allows for a freedom of academic expression and inquiry. A platform can be any collection of computational structures that constrain and evoke particular expressions and that manifest communities of practice. This definition in its use of “abstraction” and the even more ambiguous “whatever” allows each author to argue for their particular “platform” as one stable enough for critical discussion. This latitude for the conceptualization of a platform is found in other academic work. Some argue for the meaning of platform to extend into the game’s internal systems — for example, Dylan Lederle-Ensign’s argument for the id Tech 3 engine as a platform — that pushes the definition into the technical expressive apparatus itself.⁴¹ Others move in the opposite direction to more general (and

⁴⁰[158] Montfort, Nick, and Ian Bogost. *Racing the Beam the Atari Video Computer System*. Cambridge, Mass.: MIT Press, 2009. pg. 2-3

⁴¹[121] Lederle-Ensign, Dylan, and Noah Wardrip-Fruin. “What Is Strafe Jumping? idtech3 and the Game Engine as Software Platform.” *Transactions of the Digital Games Research Association* 2, no. 2 (2016). <http://todigra.org/index.php/todigra/article/view/35>.

problematizing) positions, as in Laine Nooney’s argument for drafting utensils and kitchen table as platform, or Caetlin Benson-Alcott’s conceit that maybe people are platforms too.⁴²

The ambiguity of “platform” as an abstract support for computational expression and as a physical construction for the real execution of data even bleeds into definitional structures in the history of computer games. *Debugging Game History*, a recent MIT Press publication aimed at sussing out the historical foundations for game history, focuses each chapter on a specific foundational concept in the history of games. Alongside entries on “Menu”, “Game Camera” and “Character”, there are also separate entries for “Platform” and “Console”. One is Benson-Alcott’s

⁴²In [32] Benson-Alcott, Caetlin. “40 Platform.” *Debugging Game History: A Critical Lexicon*, 2016, 343-49. Benson-Alcott discusses the multiple potential interpretation of “platform” as interpreted by researchers in a variety of disciplines. She notes Laine Nooney’s discussion of gender’s influence on “platform” and design:

As a wife and mother in central California in the late 1970s, Williams designed her first game, */Mystery House/* (Online Systems, 1980), on the platforms she had at hand — legal pads, unrolled wrapping paper, and the dining-room table . . . Nooney argues that Williams’s gender had a material and thus a socially meaningful impact on */Mystery House/*. Some platform scholars might stop their investigation at the Apple II, with an analysis of how the five-and-a-quarter-inch floppy disc and Apple DOS influenced */Mystery/ /House/*’s structure and its player’s experience. Nooney challenges them to look deeper, to consider the platform underneath the platform, the relations among gender, wrapping paper, tract houses, and game maps that make this intra-action of hardware and software, this game experience, possible.

While this note appears to conflate social-cultural contexts with the materiality of “platform” as a useful concept for the ontological description of software objects, the point is still valid. There is benefit in trying to fully ground out the material constraints of any design, even if they ground out of the hardware and software stacks we’ve been discussing in this chapter. Additionally, Nathan Altice, in [25] Altice, Nathan. *I Am Error: The Nintendo Family Computer/Entertainment System Platform*. Platform Studies. Cambridge, Massachusetts: The MIT Press, 2015, probes into not only the Nintendo Entertainment System’s constraints, but also the affordances of the cartridge technology underlying the Nintendo Game Pak. He notes that considering a platform as a fixed temporal structure, even on a technical level, is problematic given the advances to game design and additional features embedded in Nintendo Game Paks over the systems decade of dominance in the industry. See chapter 6 “Expansions” for more information on this point.

aforementioned look into the fluidity of platform as an abstract concept, and the other is a material history of consoles and systems by Raiford Guins.⁴³ That both exist within this volume, which itself is functioning to define the domain of game history (and implicitly the valid modes of its study), shows that the ambiguous duality of “platform” is baked into the foundational discourse of game historical studies. There will probably not be any attempt to clear it up. While this is not an issue for historiography, a field well acquainted with a critical focus on its own epistemological and ontological assumptions, it is a problem if we want a legitimate source for the construction of a basic ontology of platforms. We therefore need to re-focus our conceptualization (in light of the above) to how the enactment of platforms can function in the service of practical, material considerations for game history. This requires that we describe, briefly, why platforms are difficult to differentiate (in some cases) and also how they are (and will be) used by historians and the game’s community.

3.2.4 Levels of Abstraction

The order of this and the next section is a bit arbitrary. Here we lay out some basic levels of abstraction and types that help explain the different sites commonly labeled as “platforms”. In the next section we show how such labels are “common”, and discuss our community-based methodology for coming up with platform terms in the vocabulary. The abstraction discussion in this section, and its organization of types of platforms is based on the reality of how platform names are used “in-the-wild”. That is, our methodology produced a set of types and categories that is useful in explaining its results. We therefore need to either explain the methodology, defining terminology in an ad-hoc manner as we go, or

⁴³[85] Guins, Raiford. “8 Console,” *Debugging Game History: A Critical Lexicon*, 2016, pg. 63-79

we can front-load that terminology to enable a discussion of why it is needed. This section ordering is a choice of the latter.

So what is a platform for our purposes? And how can we use that definition to explain the assortment of material objects and systems that fall within it? One way, as we will see in the next section, is to draw a definition from the people playing games on platforms and making games for them. Another way is to try and coordinate all the various definitions into a structure that allows for the expression of all the other forms. A platform, in our definition, is a layered and networked structure, any slice or extension of which can be encapsulated under a unified label that allows another person to come along and use it to execute particular data. The definition of a platform intimately involves its media formats since the platform functions as a specific construction that enacts the format's data.⁴⁴

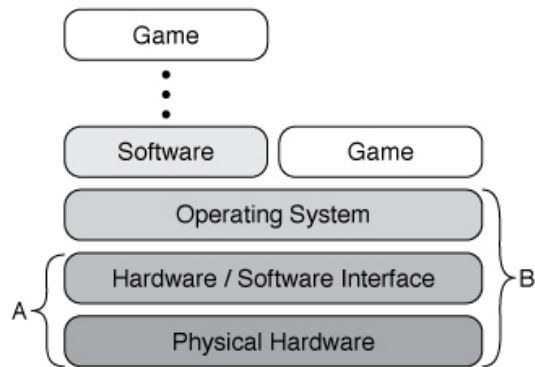


Figure 3.1: Platform Abstraction Diagram

If a platform is a collection of layers, then what are they and how to do they relate? Figure 3.1 presents the basic abstract strata that collapse in various ways into platforms. The layers are a simplification of the common abstraction

⁴⁴Our terminology here is a bit loose, since data structures, data, and executable programs (games) are not cleanly synonymous terms or concepts. However, the basic notion that a platform is organized for the enactment of an organization of data still stands.

layers in a computational system. Each layer maintains a specific set of interfaces with those above and below. Their specific interactions, while important, are not germane to their labeling as platforms.⁴⁵ The important thing is that a platform can exist as multiple combinations of these layers, and those different combinations cause them to manifest at different levels in the hierarchy of our vocabulary terms.

The bottom layer, physical hardware, includes the physical components of a computational system. In addition to a processor, memory, storage, and display components, this layer also includes physical media interfaces devices and any peripherals required for user input (keyboard, mouse, game controller, etc.). Above physical hardware is the hardware / software interface. This layer allows for communication between executable program code and physical hardware. It can take on a variety of forms, but in our definition represents the most basic interface a programmer can use to interact with (and affect) a platform.⁴⁶ The operating system is essentially a software program that manages the hardware / software interface for other software programs. Many of the basic tasks in programming

⁴⁵The basic abstraction layers of computational systems are thoroughly (and probably better) described in numerous foundational textbooks in computer science and software engineering. For a much more detailed look, see [88] Hamacher, Carl, Zvonko Vranesic, Safwat Zaky, and Naraig Manjikian. *Computer Organization and Embedded Systems*. New York, NY: McGraw-Hill, 2012. [91] Hennessy, John L., and David A. Patterson. *Computer Architecture: A Quantitative Approach*. 4th Edition. San Francisco, CA: Morgan Kaufmann Publishers, 2007. [92] —. *Computer Organization and Design: The Hardware - Software Interface*. 4th Edition. San Francisco, CA: Morgan Kaufmann Publishers, 2009. We hope this caricature is not too abhorrent or simplistic for our technically inclined readers.

⁴⁶Any simplification of a complex system limits the comprehension of its constraints and potentials, and mollifies or compresses the amount of knowledge needed to interpret it. In many, many accounts we feel that there is never a significant enough nod to the full complexity of technical things and how that complexity always undermines any ontological scheme or conceptual framing. For the hardware / software interface then, we admit that we are glossing over a lot of potentially problematic terrain, not least of which that sometimes there is not clear distinction between hardware and software, or that programming can occur completely without software or, or, or, or, or. To gain a significant understanding of the nuances of hardware and software interaction we recommend working through [162] Nisan, Noam, and Shimon Schocken, *The Elements of Computing Systems: Building a Modern Computer from First Principles*. Mit Press Cambridge, MA, 2005, it will provide a significant, embodied understanding of computational systems (and might even start you on a new career path!).

rely on highly repetitive and complex management of interfaces to hardware components. An operating system abstracts away those processes and allows for the creation of even more complex software.⁴⁷ The general software layer sits atop the operating system, and can either be game software, or provide extra software support for another layer of software. In actuality, the layers of software can extend for many layers (theoretically infinite) above the operating system. However, for the purpose of definitions, each software layer is object dependent and specific to the particular game in question.⁴⁸ We will therefore stop here.

A platform label encapsulates the specific combination of layers generally required to support a piece of game software. Our vocabulary is aimed at the recoverable use of historical software in collections, and so the labeling is targeted towards the acts of play and representation, instead of software development. We delineate two types of platforms: hardware and software. Each type encapsulates a different combination of the layers above. A hardware platform is either a combination of physical hardware and a hardware / software interface (designated as A in figure 1), or a combination of physical hardware, a hardware / software interface, and an operating system (B in figure 1). The second designation may be a little confusing since an operating system is software; however there are many platforms that are almost always explicitly referred to and labeled relative to their physical hardware and form factor without any mention of their operating systems. These platforms are generally dedicated specifically to gaming,

⁴⁷We have engaged in a bit of ontological gymnastics here that warrant some clarification. An operating system both manages and is constituted by the interfaces provided to the hardware layer. Operating systems are actually quite layered themselves, with foundational BIOS (basic input output systems) bootstrapping more complex interfaces that result in systems such as Microsoft Windows or Apple Mac OS. Operating systems are also manifest in a way that hardware is not, since the operating can only exist after a collection of hardware interfaces are available for management.

⁴⁸“Object dependent” in that each game object has a specific set of dependent software layers that support it, therefore enumerating all the possible combinations of software layers is not possible here since it is highly contingent.

and usually accept a single media format. Both hardware A and B are treated as a unified type, hardware platform, for the remainder of the chapter (unless otherwise specified).

A software platform consists of either the operating system layer, or one or more software layers. The synonymous relation between a software platform and an operating system is based on the common requirements found on physical packaging. In many cases the operating system is the major component necessary to run a game because it is assumed that if the operating system is present it will have access, through abstraction, to a correct set of hardware / software interfaces and the physical hardware they support.⁴⁹ A software layer “platform” is an abstraction layer above the operating system that provides a holistic interface capable of running a game. Web browsers, for example, could be considered a software-platform for web-based games because the specific operating system supporting the browser is irrelevant from the perspective of the game contained within it.

A computer game platform is then any computational system capable of supporting an instance of game software. Since most general computing devices (those not dedicated specifically to games) also support game software, general systems are included in the vocabulary terms.

3.2.5 Derivation of Terms

The vocabulary terms are derived from extensive research into community and collection institution practices. There is no authoritative source for information on the proper designations for platforms and media formats. Standard vocabulary

⁴⁹One of the strengths of the much maligned Microsoft Windows operating systems is their ability to manage (and function atop) a incredibly diverse set of physical hardware and hardware / software interfaces. This breadth of support is also a cause of some of the operating systems’ issues and reputation.

operating procedure states that new knowledge organizations should be based on the community labels used by practitioners in a given domain. For games this meant looking into a variety of online community sources and investigating the current practices in library collections. This section briefly describes the aggregation work that formed the base data set for the selection of vocabulary terms.

To begin the investigation, we collated all references to platforms and media formats we could find, both online and in the collections at Stanford University Library and UCSC Library. Initially, based on the sheer number of available references, we limited our scope to hardware platforms. We specifically focused on dedicated gaming platforms that would most likely be found in library and archival collections. The initial research pass aggregated information from online finding aids and a collection of websites that we thought covered most of the colloquial and technical names for platforms. The online sources were split into three categories, Wikipedia, game community sites and commercial sites. Below is a table of the number of individual platform references found both online and in our local collections.⁵⁰

The numbers in Table 3.2 represent the number of unique platform names after disambiguation and cross-referencing for similar labels.⁵³ For each aggregated

⁵⁰The Wikipedia listing is definitely a significant under-estimate, but we only grabbed references that were explicitly stated as being computer game platforms.

⁵¹Wikipedia's number of platforms is specifically dedicated to computer game hardware entries. Virtually every computing device ever made, which is many thousands, has an article, but they are not counted here. Also, Wikipedia does record media formats, but there is no singular listing of storage media available on the site.

⁵²Due to an archival setup that does not distinguish between types of hardware, the platform listing for the Cabrinety Collection is of the total number of hardware items in the collection. Potential platforms will likely be much lower (around 200).

⁵³The numbers here illustrate the breadth of platforms and their number. No two resources used the same level of technical granularity; therefore, the operating system numbers likely include many minor versions that will eventually be removed. Also, many sites did not distinguish between platforms and operating systems.

Table 3.2: Aggregated Number of Potential Platforms and Formats Per Collection

Source/Collection	Platforms	Operating Systems (if listed separately)	Media Formats (if listed separately)
Wikipedia ⁵¹	182	Over 1,000	–
Amazon	44	–	–
Ebay	43	–	–
GameStop	29	–	–
Giant Bomb	142	–	–
MobyGames	168	–	–
PlayAsia	46	–	–
Universal Videogames List	179	–	–
Video Game Console Library	110	–	–
UCSC Library	27	13	–
Stanford’s Stephen Cabrinety Collection	446 ⁵²	790	53
GAMECIP Aggregated Listing	410	–	150

platform name further research was conducted into its media format, peripherals, alternate names and versions, and any other information that helped with better description and identification. An example of this extended information is shown in Table 3.3 the Atari Video Computer System, more commonly known as the Atari 2600.

Table 3.3: Example Research Entry for a Platform (Before Disambiguation)

Platform	Alternate Name	Alternate Version	Format	Operating System	Peripherals
Atari 2600	Atari Video Computer System (VCS); Sears Video Arcade	Atari 2600 Jr.; Atari 2800 (Japan); Coleco Gemini (clone)	ROM cartridge; Cassette tape	None	Joystick; Paddle

The sheer number of potential platforms (and the extensive amount of information online) caused us to limit the initial terms to those immediately applicable to items within our local collections, or those that would be most likely available

in non-specialized game collections. Additionally, the decision to avoid software platforms proved untenable. A significant amount of games in both local collections required some form of Microsoft Windows (and provided no other salient platform information aside from that operating system on their packaging).

3.2.6 Format, Conceptualization and Reasonable Compatibility

A computer game media format is any collection of game data encapsulated in physical medium that requires a platform for presentation to and interpretation by a human being. Media formats generally consist of some means for non-volatile data storage, and physical interface to connect to a platform. For a quick example, a floppy disk is a magnetic storage disc (non-volatile storage), enclosed in a square protective sheath (physical interface) that is inserted into a floppy disk drive (connection to platform) and interpreted.

The relationship between a media format and a platform is essential to the delineation of platform terms in the vocabularies. A game resource in a given collection consists of a media format storing the game's data and complementary packaging that, hopefully, provides enough information to determine the resource's target platform. Because the structure of a game resource's media format requires a specific supporting platform, that platform can therefore be defined as a computational configuration that supports a class of media formats of which the resource is a member. That is, the enactment of a media format with respect to its platform is also a way to define a platform as a technical configuration that provides support for a media format and its data.

When dealing with actual game resources in historical collections, being able to determine their supporting platform — the one they are compatible with —

ends up as the paramount descriptive concern. In this way the correct assignment of platform information to a media format (and the data contained within it) is the crux of the whole vocabulary operation. The conceptualization we use in delineating different platforms, or in considering their correct, canonical names, is based on the assumed needs of the people who will make use of or describe a game's data in the future. We assume these needs are to get the game running again, even in some limited capacity. Compatibility is then a key consideration because anyone coming to a resource at a future point will need to know about the basic requirements for its use. This use (or enacted) assumption means that the differences between platforms manifest in their compatibility with different classes of media formats (and by generalization, different data formats). Discrete platform terms in the vocabularies occur when they provide for a different set of compatibility classes for a resource. In this way, we link the conceptualization of the terms in our vocabularies to a presumption about the terms' future use is helping to identify the correct platform for a resource. The conceptualization for a platform term is then to provide a label for the configuration of abstraction layers compatible with game data stored in a media format. And further, the conceptualization is aligned with the enactment of a media format's data as the basis for a platform's definition.

There are two types of incompatibility accounted for in the vocabularies: region and version. Region incompatibility is when a platform is modified into mutually incompatible versions based on the geographic region of its commercial release. This incompatibility is usually the result of business decisions to prevent cross-market sales, or the result of conflicting technical standards for physical hardware components. Version incompatibility refers to situations where a modification to a specific platform results in it being unable to support software

created before the modification. This is most present in the versioning of software platforms, like operating systems, where the main version number stands as an indicator of compatibility. The granularity of a versioning change can have a variety of effects on its broader compatibility with games. Therefore, many incremental versions (especially in operating systems) will require a new platform term.

Versioning and compatibility is a slippery ontological slope, considering that although minor changes in operating systems, software libraries, and application programming interfaces (APIs) are never supposed to break compatibility, in practice they do so frequently. This presents a problem for the platform vocabulary since our conceptualization is tied to compatibility, and any conceptualization should account for specific entry for each term. Recall that specific entry is the need to only make a term as specific as a particular conceptualization requires. In choosing compatibility as our metric, have we created a situation that will invariably lead to thousands of potential terms for each version of every software platform? We certainly want the compatibility information to be as specific as possible, but we also want the vocabulary to be approachable for non-specialist cataloging. To deal with the potential proliferation of software platform terms, we adopt a criterion of “reasonable compatibility” with respect to versioning. Reasonable compatibility means labeling a particular media format with a platform label to the extent that a future, historically aware and motivated patron, could make a reasonable assumption about further platform details and needs. It also, inversely, applies to the application of platform terms in records, where a cataloger can make a reasonable assumption about the platform label based on information from packaging or contextual documentation.

A good example of reasonable compatibility is the *Doom* cataloging example from this chapter’s introduction. The current record simply says that *Doom* is for

DOS, and is probably a reflection of the information available on the packaging. Without any further information, the term recommendation from our vocabulary would be “Microsoft MS-DOS” or if the packaging declared further “Microsoft MS-DOS 3.X” (if we used the actual minimum requirement for *Doom*). A researcher wanting to use that resource today would at least know that the game is for a version of MS-DOS, and luckily, the DOSBox emulator supports most versions of that operating system. Therefore, a reasonable level of description for this version of *Doom* does not need to get very specific before we have enough information to recover it. Now, the assumption of reasonable compatibility does not preclude the continual extension of the vocabulary into a giant list of sub-versions, but it allows for less specialist investigation and knowledge in the use of the vocabulary terms when a specific granularity is not present. Additionally, as we will see in the cataloging discussion below, historical practice has required the full description of system requirements as appears on packaging, and those descriptions include versioning specifics in many cases. A purpose of the vocabulary terms is to provide a mapping from the non-standard system requirements for game software to standard platform labels. As a result, a search for a specific platform term in a finding aid would return, within an assumption of reasonable compatibility, a listing of game resources that will probably have further system requirements germane to their specific compatibility constraints.⁵⁴

After sorting through the conceptualization and describing the motivation and methodology behind the vocabularies, we are ready to provide their template

⁵⁴Reasonable compatibility also applies to media formats in a more limited capacity. As Nathan Altice’s work in *I am Error* has shown [25], the specific versions and internal technical configurations of Nintendo Entertainment System Game Pak’s lead to a variety of emulation reproduction issues. While specific labeling of versions for media formats might be possible, it does not fall under the reasonable compatibility criterion, especially in regard to non-specialist catalogers who might not be aware of the nuances of format engineering nor able to divine the version of a media format without a literal internal inspection.

naming conventions. For platform entries we use:

{Company / Corporation Name}
{Platform full commercial name}
{Region (if applicable)}
{Version (if applicable / reasonable)}

An entry for a platform, in this case Sony PlayStation 2 released in North America would read:

Sony (Corporation) PlayStation 2 (Platform) NTSC-U/C (Region)

Media format entries have a similar organization:

{Company / Corporation Name}
{Format full commercial name}
{Region (if applicable)}

An entry for a media format, in this case a Nintendo Entertainment System cartridge would read:

Nintendo Entertainment System Game Pak (Company / commercial name)
NTSC (Region) ⁵⁵

The company or corporation responsible for its manufacture prefaces each vocabulary item. Many packaging styles we examined feature truncated or alternate names and in some cases the manufacturer is not explicitly mentioned. We felt that future search needs of users would most definitely include the company names associated with the platforms and wanted to make sure they appeared in each record. The truncated or contemporary colloquial names available on packaging were also sometimes not distinct, so we enforced use of the full commercial name for a platform when possible.

⁵⁵Nintendo generally precedes all of their proprietary format names with their company name. In cases where the company name was embedded within an official commercial name the Company / Corporation Name field is ignored.

3.2.7 Semantic Web Integration

This section provides an overview of the controlled vocabularies' integration into the Semantic Web through linked data frameworks. It briefly describes the motivations behind the Semantic Web and linked data, and then describes the controlled vocabularies encoding in the Simple Knowledge Organization System (SKOS). This is the final section outlining the development of the controlled vocabularies, after which we proceed to cover their effective use in modern information retrieval and knowledge representation systems.

Linked data, as outlined in the best practices in Bizer et al.'s *Linked Data — The Story So Far*, “refers to data published on the Web in such a way that it is machine-readable, its meaning is explicitly defined, it is linked to other external data sets, and can in turn be linked to from external data sets.”⁵⁶ Linked data is contextual semantic information embedded in online data in such a way that knowledge organization systems can use it to make inferences about that data and its relationship to other heterogeneous sources. The motivation for the use of linked data to describe and represent the vocabularies is driven by the current (though still evolving) library and information science practices. Essentially, the future of online catalogs and other knowledge organization systems for collection institutions is going to be based on linked data. The prime mover in this area is the Library of Congress and its proposed shift from the legacy MARC format (discussed below) to the new Bibliographic Framework (BIBFRAME).⁵⁷ BIBFRAME is a linked data format, in that its metadata fields are composed of the same linked data URIs that constitute the Semantic Web. As such, many current controlled vocabulary efforts are structured in linked data in anticipation of future integration into the Semantic Web. We cover the vocabularies' use in cataloging and

⁵⁶[36] pg. 2

⁵⁷Bibframe reference

archival standards in the next section, since there is a bit of groundwork to cover in explaining how they work. For the remainder of this section, we elaborate on how the vocabulary is embedded into the SKOS linked data framework, including the conceptualization of reasonable compatibility.

SKOS vocabularies are implemented by way of linked data triples; concept-predicate-concept relations composed of universal resource identifiers (URIs). SKOS is also specifically designed to encode controlled vocabularies and other thesauri-like organizations of terms. Each platform and media format entry is a SKOS “Concept,” meaning that the label for the “Nintendo Entertainment System (NTSC)” is attached to a URI pointing to “Concept”’s definition. If we want to link the “Nintendo Entertainment System (NTSC)” to its media format “Nintendo Entertainment System Game Pak (NTSC)” we used a SKOS “related” property, meaning that the platform “Concept” is linked to the media format “Concept” by way of a “related” property.⁵⁸ Figure 3.2 below illustrates this basic triple, consisting of two Concepts, two string properties, and a relationship property.

As shown in Figure 3.2, each entity is linked to a URI that provides a stable, definitive reference to its meaning. URIs link both the platform and media format Concepts to their vocabulary definitions. Those definitions are linked, again via URI, to the definition of a SKOS Concept. The related property is also linked via URI to its definition in the SKOS namespace (as shown in Figure 3.2).

SKOS provides a number of descriptive and relational properties that are useful for controlled vocabularies. The platform and media format vocabularies make use of: narrower and broader relations to organize the specificity of terms; preferred

⁵⁸A SKOS property is synonymous with a logical predicate. A predicate is a simply a logical statement reflecting a binary (true/false) relationship between two things. In this case the SKOS “related” property means that the platform Concept has a related media format Concept. The property is technically a predicate with the label “has related”, meaning that is it “true” that the platform has a related media format.

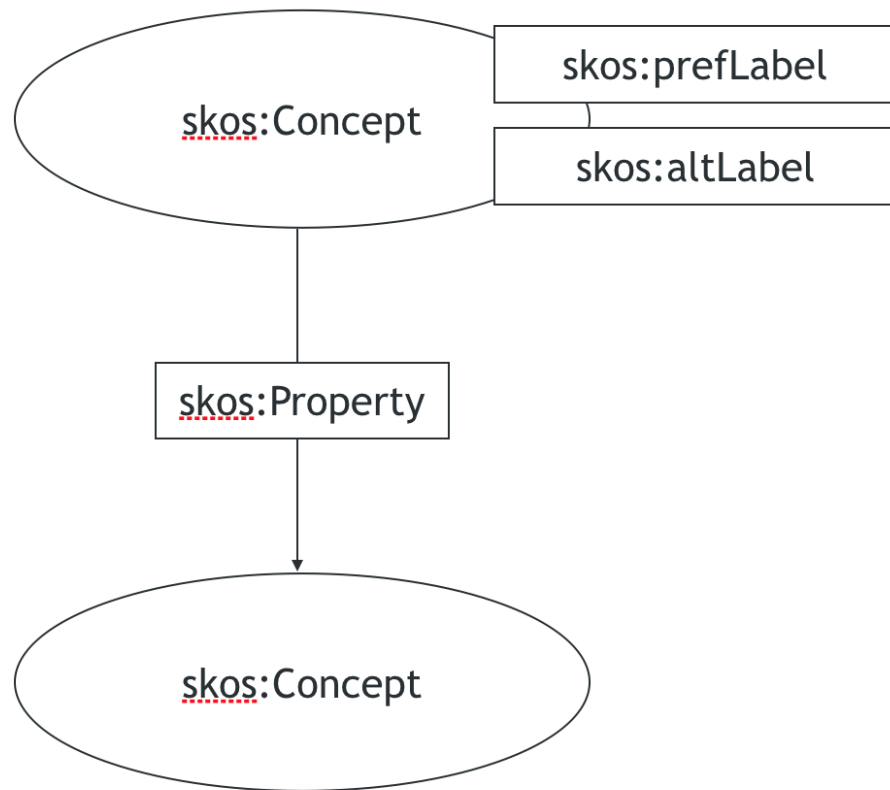


Figure 3.2: Basic Simple Knowledge Organization System (SKOS) Diagram

and alternative labels to position colloquial names underneath our canonical ones; notes to provide additional description and context; and inter-vocabulary related relations that map media formats to compatible platforms. Figure 3.3 presents the basic hierarchy of terms. All SKOS vocabularies are organized as tree structures, with a unifying top-level concept that is linked to more specific (narrower) concepts below. Each concept node provides a higher-level parent concept (and definition) for the concepts in its respective sub-tree. In the platform hierarchy, higher-level concepts include (in order of descending levels) “computer game platform,” “hardware platform,” “software platform,” “general hardware platform,” “dedicated hardware platform” and “operating system”. The media format hierarchy is less complex, only differentiating between “general” and “dedicated” formats. This is mainly used to signify that the “general” formats will probably work with a larger variety of platforms.⁵⁹

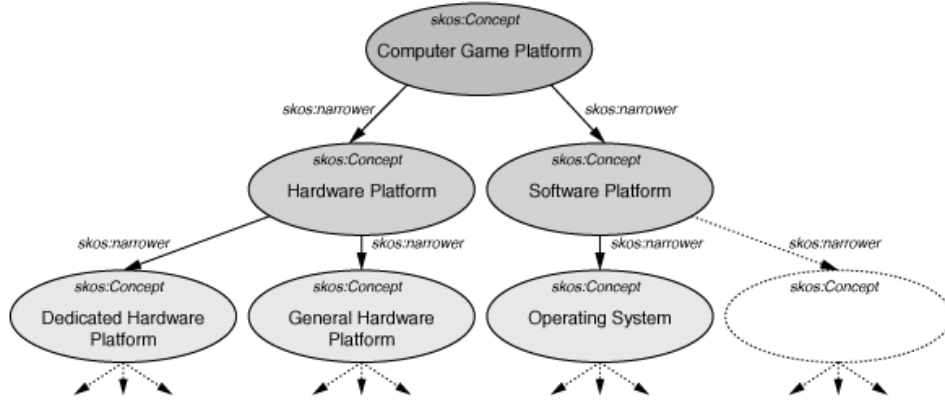


Figure 3.3: SKOS Terminology Hierarchy

Figure 3.4 shows version and region compatibility as expressed in the SKOS hierarchy. The hierarchies express the principle of reasonable compatibility through

⁵⁹The media format vocabulary is also less specified since it was an off-shoot of the work on platforms. “Future Work” below outlines some means of improvement in the classification scheme.

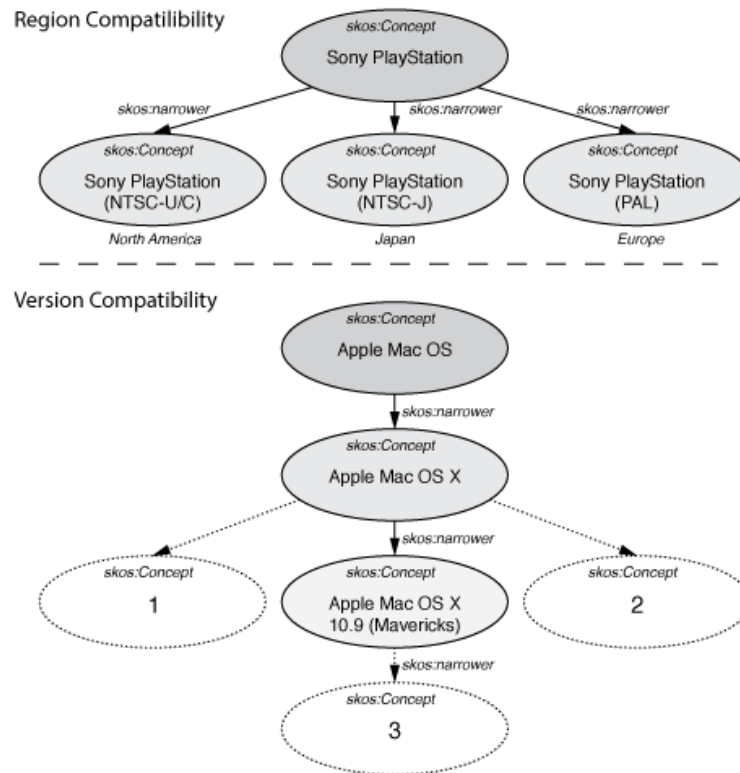


Figure 3.4: Region and Version Compatibility. 1 and 2 signify potential previous and future versions, whereas 3 allows for further specific versioning.

recommendations to use the term that most specifically aligns with the information available on a resource. In some cases the specific media format or platform might not be comparable with a term at a leaf node in the tree. The vocabulary usage guidelines recommendation traversing up to the most identifiable parent node. In some cases just knowing that a resource uses any version of Microsoft Windows is an improvement to most cataloging and archival records.⁶⁰

3.2.8 Vocabularies in Institutional Practice

This section outlines the controlled vocabularies' use in institution knowledge organization systems. The focus is primarily on library catalog finding aids, and item-level description in archival records. Both use cases are drawn from actual, practical work being done at both Stanford University Libraries and UCSC Library. The goal is twofold: to show how the vocabularies are a contribution to current cataloging practices in both archives and libraries, and to serve as a mild probe into the state of institutional knowledge management of software resources. In line with the goals of this thesis, which is a look into the state of games as objects of historical study, the actual practice of recording and managing the objects of game history is certainly a significant part. Rarely do historically minded scholars collaborate directly with library and archival personnel to improve the description and access to records. As game studies scholars, our work with these professionals is so far the only collaboration that has had a direct influence on the practical cataloging work of games or software. Future work discussion at the end of this section will elaborate on the continued potential for further collaboration.

⁶⁰The description of the SKOS hierarchy in this section as well as the above sections on levels of abstraction and methodology are based on the aforementioned article, "Implementing Controlled Vocabularies for Computer Game Platforms and Media Formats in SKOS". We recommend looking to the publication for more information on the SKOS descriptive work and the methodological issues that arose in trying to organize platform terminology based on online resources.

Hopefully this work is just the beginning.

Institutional Context

Before describing where, exactly, the controlled vocabulary terms are positioned in cataloging and archival practices, a little housekeeping is in order. This section outlines the basic differences between general collections and special archival collections in regards to cataloging.

Cataloging practices vary greatly depending on the goals and infrastructural constraints of a particular memory institution. For library general collections, like university stacks or public library collections, most items are described in an online catalog or finding aid. This item-level description, for software at least, usually includes a title, publication information, and the relevant call number (for physical items) or a link to the proper information in an online repository.⁶¹ Library staff curate collections based on the needs of the communities they serve. This is a main distinction, since general collections support the circulation of items, those items need to be managed and described as individuals.

For archives (or special collections), collections are not usually curated at the item-level. Archives generally deal with collections of documentation organized by outside parties, like corporations or individuals. These collections are accessioned, and then maintained in manner that explicitly avoids tampering with their natural organization. This preservation of the potential disorder of an archival collection is known as the *respect des fonds*. That is, there is contextual value inherent to the organization of documents, and any significant disruption of that organization might negatively impact their historical study. This is tied to the

⁶¹Currently there are very few institutions that provide online access to software data. This is another motivation behind the controlled vocabulary work; to make the designation and standardization of platforms and formats a more ingrained and established process so that when migration to digital repositories occurs in the future software data will not need re-description.

notion of provenance, in which every reasonable effort is made to maintain the authoritative source and position of an archival item. When provenance is ignored or fonds disrespected, there is potential for items to become lost or decontextualized to the point of irrelevance. Therefore, in many instances archival collections do not and cannot use the same systems as general catalogs, being as they are more diverse, and dependent on more specialized knowledge.

Given that most research institutions have significant cataloging needs, and that those needs are probably similar in respect to collection management, most places ascribed to a common set of standards. In North America and Europe, most institutional catalogs describe items according to the Resource Description and Access (RDA) guidelines. These guidelines provide a set of core descriptive elements that should exist in any cataloging record. Cataloging records themselves are generally implemented in data formats designed for use with knowledge organization systems, like online catalogs. Although RDA does not require a specific data format for record description, the largest unified catalog in the world, maintained by the Online Computer Library Center, implements all its records in the MARC21 format. As such, a majority of the research institutions in the United States and Europe now use the MARC standard in their collections. MARC stands for “machine readable card,” an anachronistic acronym derived from its initial implementation for punch cards in the late 1960s. MARC21 is simply the current, internationally unified version of the MARC standard.

Data requirements for archival institutions vary greatly in comparison to general collections. Because each archival collection is a unique, and each institution’s archival specialty distinct, archival management standards are not universally implemented. As a result, while standards do exist for archival collection description, their breadth of use cannot be assumed in the same way as the cataloging

standards. As such, most of the discussion of the vocabularies' contribution to archival records practices is based on the practical working conditions of Stanford and UCSC — and therefore cannot be as comfortably extrapolated to other institutional contexts.

The transition to digital object production over the last 40 years significantly upended the standards and approaches of traditional catalogs and archival repositories. The MARC format and RDA's predecessor, the Anglo-American Cataloging Rules Version 2 or AACR2, arose out of a world dominated by print media production. Both RDA and MARC carry with them a structural bias for print media description. They have retroactively adapted to cover the needs of digital object description, but the fit is awkward to say the least.⁶² Similarly, archival institutions are also dealing with the shift from physical to born-digital collections. Luckily, there are quite a few emerging standards for object description in digital archives can adequately handle software data.⁶³

Controlled Vocabularies in Context

Any contribution to the practice of library and archival professionals has to conform to their standards, and fit in with their practical constraints. This section lays out three of the current possible use cases for the vocabularies based on their status as Library of Congress term source codes and hosted semantic web

⁶²The next section will cover this claim in a bit more depth.

⁶³The Preservation Metadata: Implementation Strategies (PREMIS) Data Dictionary [12] covers description of common transformations for archival data, including migration. It also provides for somewhat extensive technical dependency descriptions, however software applications are primarily included in PREMIS as a means for the representation of other data files. This does not preclude them from being described in the standard, it just makes them slightly tougher to fit in. McDonough et al. [149] conducted significant research on the Open Archival Information System (OAIS) and its applicability to Don Wood's and Will Crowther's Adventure text adventure game. They concluded that OAIS is not a perfect fit, but can be articulated to cover software instances with certain caveats mostly related to the level of specification for software dependencies.

vocabularies. The purpose is to illustrate how the vocabularies can potentially affect and improve practice in each case area.

The three use cases for the vocabularies are:

1. As a recommended source for platform terms in RDA best practices for computer games
2. As a recommended source for platform and format terms in GAMECIP metadata mapping guidelines for computer games
3. As a potential source vocabulary for platform and format terms in archival item-level descriptions

The use cases progress from the world of general collection cataloging (1) to archival collections (3) by way of the mapping guidelines (2). Before diving into the use cases, there are two phrases passed by in the preceding paragraph that we need to grab hold of first and explain: “Library of Congress term source code” and “hosted semantic web vocabulary.”

The Library of Congress maintains a significant collection of controlled vocabularies that function as the canonical sources for specialized information in record data formats. Having a “source code” in this sense is to have a label for the source of the information found in a record. It is an official way to tie the value of a certain metadata field to an authorized, canonical source for that value. In the case of the controlled vocabularies, being a product of the Game Metadata and Citation Project (GAMECIP), the official source codes for the vocabularies are “gcipplatform” and “gcipmedia.” This means that when a term from either vocabulary is used in an archival or cataloging record it can be linked back to an official source in the Library of Congress that can provide more contextual explanation of the term. A source code also allows gives the cataloger an assurance that the

term has been validated and approved for use by requisite experts. The first use case will explain how this source code fits into current general cataloging practice.

The Library of Congress source code listing for both vocabularies is directly linked to their entry point in the Semantic Web. As mentioned, the Open Metadata Registry hosts and maintains Semantic Web controlled vocabularies and metadata schemes. The Semantic Web is essentially a giant semantic knowledge graph shared by a growing number of sites online. The OMR provides for the hosting of SKOS vocabularies and links them to the larger semantic web graph through a SPARQL endpoint. All this means is that when a request is made to search for semantic web information, the OMR provides that search with its hosted vocabularies. So, for instance, if another group started a computer hardware vocabulary and linked to the platform vocabulary through a SKOS “related” property that “related” connection could potentially return information from the platform vocabulary when the hardware vocabulary is searched. The OMR also provides for resolution of human-readable description for controlled vocabulary terms. Each term in the platform vocabulary is a SKOS Concept node with a specific URI. That URI, always prefaced with “<http://gamemetadata.org/platform/{platform id}>” or “<http://gamemetadata.org/media/{media id}>”, is a link to a website describing a specific platform or media format term. In summary, the controlled vocabularies being semantically hosted means they are accessible to higher-level semantic web search structures, and that interested parties can always find out the meaning of a vocabulary term in an online catalog or archival record if the term is annotated with its official LC source code. The use of semantic controlled vocabularies is a rather new practice, but falls in line with similar moves by other institutions, most notably the RDA cataloging standard (which hosts its terms on the OMR), and the controlled vocabularies for public broadcasting (PBCore).⁶⁴

⁶⁴Full, consistent standards for semantic web data, and linked data for institution records

The ability to find values in a record metadata field and link them back to contextualizing information (as well as making their semantics machine-interpretable) is one of the reasons why “linked data” is so named, and why it is exciting to the world of knowledge organization.

With the explanation of the controlled vocabularies’ location as hosted semantic web vocabularies with an associated LC source code, we can now proceed to show what that means in practice.

RDA and Best Practices for Computer Games

As mentioned, Resource Description and Access (RDA) are a set of cataloging rules that apply to most collections in North America and Europe. RDA is technically the third version of the Anglo-American Cataloging Rules (version 2, thus AACR2, drafted in 1978). As an update to previous cataloging rules based on print media, RDA inherited a bit of ambiguity in regards to computer games and software records. Fortunately, as a result of collaboration with the GAMECIP project, the Online Audiovisual Catalogers (OLAC) recently published a set of best practices for computer games.⁶⁵ Additionally, one project member, Greta de Groat, also published a detailed history of cataloging rules for computer games. Both documents allow for a significant improvement in the treatment of computer

are still in their infancy. For further, thoroughly worked examples, see [36] Bizer, Christian, Tom Heath, and Tim Berners-Lee. “Linked Data-the Story so Far.” *International Journal on Semantic Web and Information Systems* 5, no. 3 (2009): 1-22. [195] Southwick, Silvia B. “A Guide for Transforming Digital Collections Metadata into Linked Data Using Open Source Technologies.” *Journal of Library Metadata* 15, no. 1 (January 2, 2015): 1-35. doi:10.1080/19386389.2015.1007009. PBCore [10] is the Public Broadcasting Core Metadata Elements and Controlled Vocabularies.

⁶⁵The Game Metadata and Citation Project (GAMECIP) [7] is a larger research project from which both the best practices and the controlled vocabularies arose. It was fortunate that the project tackled many issues in cataloging concurrently, so that each contribution could connect up with and build on each other. In this section specifically de Groat’s History of Computer Game cataloging was an earlier project output that then informed the approaches taken with the implementation of controlled vocabularies that themselves found a way into best practices.

games in catalogs, and provide much needed context for historical records of computer games. This section briefly describes the historical condition of computer games in cataloging records as a way to illustrate the historical contingency of different descriptive strategies for software in collections, and how fleeting and brittle they can be. It then shows how the current best practices — with assistance from the controlled vocabularies — help to strengthen new record creation, adding more stability and providing a deeper methodology than has previous been directed to software records in catalogs.

The history of game cataloging begins in 1978, with the inclusion of “machine-readable data files” in chapter 9 the AACR2 cataloging rules.⁶⁶ That chapter also optionally provided for a general material designation (GMD). An additional label for non-print materials appended after the title. For instance, *Star Wars*, for the Atari 2600, might be given the title “*Star Wars* [machine-readable file],” if inserted into a catalog in this period. The use of “machine readable file” and chapter 9 in general mark the initial inclusion of any descriptive rules for computational data in cataloging records.

Computer games, as an explicit category of computational media, first appear five years later in Nancy Olsen’s 1983 *A Manual of AACR2 Examples for Microcomputer Software and Video Games*. They are described as “electronic toys or games that are typically issued in cartridge carriers and are manipulated by hand controls.” This equivalence between video games and electronic toys led to a recommendation in Olsen’s document that games be considered for coverage under AACR2 chapter 10 rules. de Groat notes that chapter 10 pertains to physical realia. The updated AACR2 recommendations, therefore, placed computer games

⁶⁶Most of the quotes in this discussion of the history of computer game cataloging are drawn from Greta de Groat’s aforementioned article, [62] de Groat, Greta. “A History of Video Game Cataloging in U.S. Libraries.” *Cataloging & Classification Quarterly* 53, no. 2 (February 17, 2015): 135-56. doi:10.1080/01639374.2014.954297.

in the same ontological basket as physical toys and games, and led to a general designation of “[game]” instead of “[machine-readable file]” in cotemporaneous records. Olsen’s manual reflects the guidelines put forth by a “computer software in cataloging” task force report from around the same time. That task force recommended that the program statement — a description of the organization of the computer file resource — include some mention of the physical media storing the program. Previous program statements had included the number of individual files, and in some cases the number of lines of source code, but had left out — or had no vocabulary for — the physical form of the data’s carrier.

A year later, in 1984, the American Library Association (ALA) began recommending the inclusion of a System Requirements note to consolidate “all technical information, such as make and model of machine, memory, operating system, and so on . . .” for a computational resource. There was no thought to standardization of the system requirements information given the rapidly changing state of the computer industry, and the lack of industry standards for requirement description. System requirements became a required field in the 1988 update to the AACR2. This update also began the recommended practice of attempting to retrieve title information from a game’s title screen (in line with principles from film cataloging). If the cataloger could not play the game, physical packaging would suffice.

Other significant additions in the 1988 AACR2 update were another change in the general designation, this time from [machine-readable file] to [computer file], and the inclusion of a small controlled vocabulary for physical data carriers (media formats). The physical carrier types at that time: computer chip cartridge, computer tape cartridge, computer tape reel, computer laser optical disk, and computer laser optical card, are very general and apparently are not drawn from

anything other than the types encountered by catalogers at the time.⁶⁷ The change in designation from “machine” to “computer” is appropriate given that a computer (as a sub-class of machine) probably appeared to be a rather anachronistic notion by the late 1980s. It is important to understand the historical contingency of these descriptions, however, in thinking about how our current terminology and conceptions might become dated or confusing in the near future.⁶⁸

de Groat mentions that an additional historical anomaly, the early 1990s discussions of “interactive multimedia,” produced a complementary (and sometimes highly contradictory) set of recommendations. These recommendations grappled with how to describe computer-based resources “residing in one or more physical carriers . . . or on a computer network” that consisted of “(1) user-controlled non-linear navigation using computer technology and (2) the combination of two or more media (audio, text, graphics, images, animation, and video)”. de Groat comments (and we agree with post-convergence hindsight in 2017), “these seem like relatively ordinary characteristics for computer resources.” As a result the general designation of [interactive multimedia] is still available and sometimes applied to game resources.

The last major change in AACR2 rules came in 2001, with another shift in general designation, from [computer file] to [electronic resource], and a recommendation to describe the physical carrier in “conventional terminology” of the time. Both changes appear to be a retreat from clarity. The shift from “computer” to “electronic” permeated all descriptions. “Computer data” became “electronic data” and “computer program” became “electronic program.” The [electronic re-

⁶⁷In discussion with de Groat she mentioned that she could find no evidence of the reasons for these five specific carrier types or their level of granularity.

⁶⁸It is this author’s opinion that most computation qualifiers currently in use, like “digital”, “computational”, and “data”, will decline rather precipitously over the next generation. Digital humanities → humanities, computational media → media, and data science → science. That things are “computational” will be long taken for granted.

source] designation appears to have been implemented in response to print-based resources making the jump to digital representations. Computer programs apparently received secondary consideration after “electronic books” and “electronic journals.” The use of “conventional terminology” — and a lack of controlled vocabulary — led to a “wide diversity of physical description formulations for computer games,” specifically in the description of optical disc formats (like those targeted for specific game platforms, Sony PlayStation or Microsoft Xbox, or different rewriteable formats, CD-R, DVD-R, DVD-RW, etc.)

One final shift occurred with the new RDA rules in 2010. This is the current standard cataloging rule set, and in updating from AACR2 RDA adapted a more deconstructed notion of content. RDA did away with general material designations, instead presenting a set of content, media and physical carrier types to be used in item descriptions. Available content types are now only “computer program” or “computer dataset,” and more problematically, computer games are specifically listed as “two-dimensional moving images” or “three-dimensional moving images” with no explicit requirement to label them as computational. Furthermore, there is now no distinction between a CD-ROM containing static video data and one containing computer game program data. That is, no explicit way to differentiate interactive versus static content directly in resource descriptions.⁶⁹

As a quick recap, the general designation for computer game changed from [machine-readable file] to [game] to [computer file] to, potentially, [interactive multimedia] to [electronic program] to “two-” or “three-dimensional moving image” over the last 40 years. Notes on the physical carrier type for a resource now rely on a handful of generic terms for computational media and a recommendation on historically contingent “conventional” terminology. Additionally, system

⁶⁹This is a hypothetical situation based on the reading of the actual requirements. In practice other aspects of the record will probably make it rather clear whether the item is a video dataset or a computer game program.

requirements, first recommended in 1985 and required in 1988 are now no longer required “beyond what is normal and obvious for the type of carrier or type of file.” Uses of “normal,” “obvious,” and “conventional” are problematic since they rely on the implicit, unexamined assumptions of the individual doing the cataloging at a specific time.

The controlled vocabularies now enter the picture as a way to (pun intended) set the records straight. The aforementioned best practices for computer games focus on the application of the RDA rules to computer game resources described in the MARC data format. Recall that the MARC format is the US and European standard catalog record format. It is essentially a set of coded three digit fields whose contents are recommended by RDA guidelines.⁷⁰ Each field allows for further sub-field qualifiers, “\$” symbols followed by alphanumeric characters. For instance, the “title” of a resource is placed in field 245 following a “\$a” sub-field qualifier:

```
245 00 $a Need for speed:  undercover71
```

The MARC format is designed to ease the sharing and organization of bibliographic records, especially records that describe the same item located in two different institutional collections. Common practice among catalogers is to first check with WorldCat, the OCLC international database of MARC records, to see

⁷⁰This is a gross simplification but serves our comprehension purposes. A better description is provided by de Groat, “The MARC bibliographic formats have different fields and byte sequences that are conditional on a byte in the Leader. Leader/06 determines the format of the record, and determines the workform that a cataloger will see in OCLC or other cataloging system when they input a record. Leader/06 ‘m’ is the value for ‘computer file,’ which controls the fields defined for the MRDF format.” MRDF stands for “machine readable data file” and is a computer interpretable encoding of the MARC standard.

⁷¹The “\$a” qualifier stands for “title proper”, as in “this is the proper title of the resource”. Due to issues derived from book series titles, franchise names are usually separated from sub titles in conventional records. This creates a problem with search, since most games in franchises would be difficult to locate based solely on the franchise title (there would be a lot of “Need for Speed”s). The \$a designates that the colon(:), which would usually demarcate a sub-title (and thus be placed after a \$b subtitle qualifier) is (in fact!) part of the full title of the resource. The lowercase title is a standard for catalog descriptions.

if the item they need to describe already has a full record. If so, that record is adapted to the local collection's context, leaving most of the information unchanged. This practice is known as “copy-cataloging” and while it is convenient for some materials, it is particularly problematic for computer games and other software. As we saw in the brief outline of cataloging history, the designation and description of computer software records changed frequently and significantly over a relatively short period. This means that any item entering a collection now could have (depending on its age) a rather arbitrary set of descriptions for its content, system requirements, physical carrier and general designation. The best practices are an attempt to prevent some of this mess going forward. The vocabularies, in conjunction with the best practices, are an effective first step towards more historically stable records.

Finally, the controlled vocabularies find a place in this morass of cataloging history and minutiae. The best practices recommend the use of the platform vocabulary in both the system requirements and platform designation fields of the MARC format. The system requirements field 538, is that catchall description field for any requirements found on packaging (or online in the case of born-digital games). However, since the descriptions are not standardized, the recommendations call for the explicit use of a vocabulary term when discernable. Additionally, they recommend that the same term be placed in the platform designation field 753. The 753 field, as a result of GAMECIP's efforts, now supports a linked-data compatible sub-field qualifier for LC source codes. This allows for the use of an explicit URI, or the LC source code shorthand to be included in all future computer game records. For example:

```
538 __ $a System requirements: Sony PlayStation 4.  
753 __ $a Sony PlayStation 4 $2 gcipplatform
```

or:

```
753 __ $a Sony PlayStation 4 $2 http://gamemetadata.org/uri/platform
```

The ability to use a controlled term for platform designation is a significant contribution to the practical cataloging of computer game software; one that is already beginning to see use at both Stanford and UCSC Libraries.

Sadly, the guidelines for physical carrier description are still stuck with the current batch of seven controlled terms.⁷² The ones most applicable to games are “computer disc,” “computer chip cartridge,” and “online resource.” Future work mentions steps that could be taken to embed the controlled format vocabularies into RDA and MARC in a similar fashion to the platform vocabularies.

That even the platform vocabularies now have an official place in the most widely used cataloging guidelines and data format in the United States and Europe is a significant improvement over previous historical efforts. As described throughout this section, the historical description of computer game and software records ached for a more considered approach steeped in practice and in interface with practicing scholars. Now there is a toe-hold in the larger descriptive cataloging apparatus for the coherent and consistent description of game platforms (and through the best practices recommendations, an optional consideration for media formats). Any further extensions to the vocabularies therefore no longer need to navigate the systems and standards of institutional catalogs, but are simply able to add more terms and more granular descriptions whenever new items appear and there are resources to catalog them.

⁷²Unlike the 753 field, which is a designated for make and model of machine, there is no comparatively applicable field for media formats. Certain fields exist for physical carrier description, but all require the use of an official RDA carrier type. Thus the seven options.

GAMECIP Metadata Guidelines

The vocabularies second use is their application in the GAMECIP Metadata recommendations for computer games. These recommendations are a set of 20 metadata elements for game resources. An element is an abstract recommendation for describing some part of a resource. For example, the RDA rules above consist of a set of core elements that are a required presence in resource records. The RDA guidelines assign the core element title to field 245 of MARC. The element is a requirement for description of a specific type that is then placed into an explicit metadata data format field. The best practices document discussed above is essentially a description of which RDA elements are relevant to games, and how those elements should be described in MARC fields. It is a two-step process, with the information to be described separated from its implementation in a specific data format. The GAMECIP Metadata recommendations are a set of elements designed to fully describe a game resource. These elements are then mapped to analogous elements in different metadata schema. The process of mapping elements (and indirectly field implementations) of different metadata schemes is generally known as a “crosswalk.” A crosswalk allows descriptions designed in one domain to be implemented in others. Some of the crosswalks connect metadata schemes with different levels of granularity, so they are rarely completely equivalent. It is also common for many elements from one scheme to map to a single element in another, or conversely, for a single element in one to map to many in another.

The recommendations map GAMECIP elements to complementary ones in Dublin Core, MODS, Schema.org, and RDA. Each metadata scheme contains a different set of elements based on the domain they intend to describe. All of those mapped to by the recommendations are general metadata schemes, meaning they

are intended to cover a wide variety of domains with limited specialist depth. This is an intentional move by the recommendations to show how game description could be improved in current, actively used formats. One simply describes a resource according to the GAMECIP elements, and then follows the crosswalk to the actual scheme used by their institutional knowledge management system. RDA, as a set of guidelines, has already been extensively covered and applies mostly to general collection catalogs. The other schemes, schema.org, MODS, and Dublin Core, find use in a variety of settings including online repositories and archival collections. Stanford University uses MODS for most of their archival record description needs (a point that will return in the next section on archival records).

The recommendations use the controlled vocabularies for three distinct GAMECIP elements: Platform, Media Format, and Extent. The first two elements, Platform and Media Format, are explicit containers for the vocabulary terms, and function as a means of recommended usage for the vocabularies in other metadata schemes. The last element, Extent, is short for “physical extent” and is a description of a “game distribution media.” This element is similar to Media Format, but is a more general physical description. A game may have a Media Format of “5 1/4 in. floppy disk” but a Physical Extent of “4 5 1/4 in floppy disks.”

Archival Records

The last use case for the vocabularies is in the metadata schemes mapped out in the GAMECIP recommendations. This is the most speculative territory, since we can only rely on local examples due to the locality specific nature of archival practices. Specifically, the mapping of platform and format terms through the GAMECIP Platform and Metadata elements, allows for their use in the Metadata

Object Description Schema (MODS). Stanford University Libraries use MODS for the description of items in archival collections. The GAMECIP mapping connects Media Format and Extent to MODS’s “physical extent” field, and Platform to a standardized use in a MODS “note” field. The latter case is less than ideal but illustrates how more general schemes commonly lack the explicit technical and contextual information relevant to computer game software records.

The primary collection of computer game software at Stanford is the Stephen M. Cabrinety Collection in the History of Microcomputing Collection. Containing over 10,000+ pieces of game software, the collection is easily one of the largest in the world. Work is currently under way to migrate all of the data off of Cabrinety’s physical media and position it in Stanford’s Digital Repository. This task, a collaboration with the National Institute for Standards and Technologies’s (NIST) National Software Reference Library (NSRL), also calls for the accurate description of the data imaged from collection items. The vocabularies for media formats and platforms are therefore helpful in attaching a canonical platform and format reference, through the GAMECIP MODS mapping, to each imaged Cabrinety item. While not a perfect solution, this is the first step towards more clarity in game records that have historically had none.

3.3 Future Work

Work is currently underway to organize support for the controlled vocabularies in three ways:

1. The organization of a consortium for the maintenance and growth of the vocabulary
2. The creation of instructional information sheets to allow non-specialized

library staff to better identify historical formats and platforms

3. The creation of online, human-readable descriptions of vocabulary terms

While the first point on consortium is well outside the scope of this work, we will briefly describe the current efforts in information sheets and online resources.

The informational sheets are a proposed set of printable PDFs documents that describe each leaf vocabulary term. As illustrated in Figure 3.5, the sheets are divided into sections that align with their SKOS properties. For the media format sheets, a collection of to-scale SVG images has been created to allow for comparison when identifying media formats. The impetus for the sheets is both their capability for physical comparison in identification, and to allow them to be stored in Stanford University's digital repository. The repository storage allows for the creation of a permanent URL for each document, and takes advantage of already existing library infrastructure. This is important because creating new, technological solutions for library management of linked data structures requires significant institutional investment.

As mentioned above, the Open Metadata Registry provides for basic functionality for semantic SKOS vocabularies. Sadly, this does not include any provision for extensive description of each term, or usage rules and guidelines for a vocabulary. Most prevalent vocabularies on OMR use it for RDF generation and as a SPARQL end-point but provide additional, external guides to vocabulary terms. For linked data, attaching human-readable descriptions to what are essentially arbitrary URIs is a necessary part of the architecture. As such, making semantic links interpretable by people usually requires creating web pages devoted to term description or allowing browsers to resolve URI links directly to HTML. Semantic web specifications provide for different levels of semantic data clarity, and while they are not well-defined, general consensus is that web informational pages are a

UC SANTA CRUZ UNIVERSITY LIBRARY STANFORD UNIVERSITY LIBRARIES ET EXPRESSIVE INTELLIGENCE STUDIO

Nintendo Game Boy Color Game Pak

Format Variants: Color cartridge, Clear Color cartridge

DESCRIPTION:

Two cartridge formats used by Nintendo Game Boy Color platforms. Standard Color game paks are backwards compatible with the Nintendo Game Boy. Clear Color game paks (translucent) only function on Game Boy Color and later models.

Dimensions: 2.000" x 2.250" x 0.210"
 Identification: <<media_identification>>



PACKAGING:

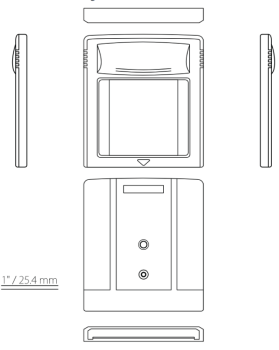
<<packaging_features>>

Dimension: <<dimensions>>
 Weight: <<weight>>



<<box_art_features>>

Variant 2: Clear Color Cartridge



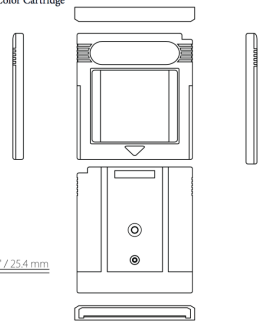
1" / 25.4 mm

REFERENCES:

© Public Domain 2016 by UCSC:SOE

IDENTIFICATION:

Variant 1: Color Cartridge



1" / 25.4 mm

Figure 3.5: Information Sheet Example

basic necessity.⁷³ The gamemetadata.org domain is currently being organized as the official source for information about controlled terms. Each page is devoted to a SKOS concept, and provides links to any node connected to it. The site also provides a description of use, and a basic alphabetical controlled vocabulary listing. The listing is automatically formatted based on the alternative title SKOS property. This means that alternative names for platforms and formats note a redirection to the preferred term, which is useful in the case of items that only bear alternative labeling. The figure below shows a basic platform and media format entries. In the case of media formats, all the images and SVG tracings created for the information sheets are also shown, and links to the information sheets permanent URLs are also provided.

3.4 Conclusion

Now that we have journeyed through the thicket, carving one narrow path among many with our investigation into platforms and media formats, we'll take a moment here to recount the journey and its contribution to the goals described at the outset. Recall that before starting, we noted that the ability to describe objects in ways that might resonate through the future is a fraught exercise. Times change, and with them the stability of basic knowledge categorizations and fundamental metaphysical conceptions of reality.⁷⁴ As the work above illustrates,

⁷³Tim Berners-Lee — one of linked data's biggest proponents — ascribes “star” ratings to linked data implementations. See [195] for how this relates to human-readable information pages, and look at PBCore [10] for a live example.

⁷⁴For more information on categorization as a socially constrained act see [40] Bowker, Geoffrey, and Susan Leigh Star. *Sorting Things Out: Classification and Its Consequences*. Cambridge (Mass.): Massachusetts Institute of Technology, 1999. [67] Durkheim, Émile, Marcel Mauss, and Rodney Needham. *Primitive Classification*. Routledge Paperbacks. London: Cohen & West, 1970. [114] Lakoff, George. *Women, Fire, and Dangerous Things: What Categories Reveal about the Mind*. Chicago: University of Chicago Press, 1987. [115] Lakoff, George, and Mark Johnson. *Metaphors We Live by*. Chicago: University of Chicago Press, 1980.

even for the seemingly unproblematic categorization of the physical and technical dependencies for computer games much work needs to be done in simply defining the purpose and purchase of those terms within the frame of game historical study, and in finding ways to stabilize those terms in ways amenable to future work. That is to say, you need a reason for why the terms you are using as described in the way that they are, ways to preserve the intentions of those descriptions going forward, and a strategy for their scholarly use, maintenance and upkeep. You need to plan a path, construct it, and then make sure others can use it and keep it well-trodden.

The work of this chapter needed to first ascertain why a controlled list of platform and media format terms would be useful, and whom it would be useful for. It then needed to back those terms up with a conceptualization of how they might be used in the future. Here we chose the concept of ontological enactment as a means to probe the “meaning” of computing platforms and formats, and ground them out into concise terminology. In thinking through that enactment, we hit upon a criterion of “reasonable compatibility” as an intermediary space between technical descriptions too intense for common practice, but not too subtle as to lead researchers astray. Again, we wanted the path to each described object to be reasonably well defined and free of weeds or false routes. Furthermore, the stabilization of the terms into an encoded, machine-networked and interpretable format automatically connected them to existent knowledge organization systems, and made them semantically interoperable and locatable. The SKOS encoding also allows for further extension and elaboration without our direct input or control, which means that if conceptualizations or use-cases change in the future our work can be mapped onto newer structures just as we mapped ours onto older ones.⁷⁵

⁷⁵This mapping occurred in multiple ways: the terms are connected to MARC21’s 753 field,

Finally, in embedding our vocabularies into existing descriptive structures, we illuminated some of the past issues with software record description and actively contributed to changing them in practice instead of just theoretically. That the terms are now recommended for use — and being used — in contemporary game catalogs is a testament to all our preparatory work in basic research, ontological conceptualization, and descriptive structures. This chapter then also functions as methodological guide for further future imbrication of newer records of game software and technology into legacy organizational systems.

they are linked semantically to RDA physical carrier terms and through the GAMECIP metadata elements, also tied to related fields in the MODS, Schema.org, Dublin Core, and RDA element sets.

Chapter 4

Citation

4.1 The Pivot

This chapter marks the transition from knowledge stabilization to knowledge exploration. The previous chapters dealt with the enumeration and description of both game development work and the games that resulted from it. Here we speculate on the potential for computer games stored in stable collections to provide a basis for new kinds of game and software historical scholarship. As many institutions contemplate migrating their software collections to digital repositories and expanding their born-digital holdings, it behooves us to find new opportunities and use-cases that leverage these records born-digital nature. For history, these new collections of software provide for a deeper examination of their reference and retrieval as historical objects. This examination also requires the staking out of new territory in the requirements and potential for software archives. As recent surveys of the digital humanist landscape make abundantly clear, the future of historical scholarship will be tied to reconciling older, print-based, qualitative

practices with newer, networked and quantitative ones.¹ The meaning of the historical archive is changing, and the ways to enable and maximize its exploration must also adapt. Below we focus on the particular case of game historical scholarship, and the use of historical citation, reference and source retrieval in providing new means of discussion about games.

Below we outline the current practice and requirements of citation in game historical study, and then supplement that practice through the description of a tool for the reference and resurrection of game software data. This resurrection is used as a shorthand for the re-execution of legacy software data inside a new computational context.² Our tool also aligns with digital humanist arguments, most prominently those of Stephen Ramsay, who calls for the acceptance of computational tools *as* arguments for and about new humanistic expression. This sentiment is echoed in Burdick et al.’s recent polemic on the state of digital humanities, *Digital_Humanities*.³ They assert that, “the next generation of digital experimenters could contribute to humanities theory by forging tools that quite literally embody humanities-centered views regarding the world.”⁴ In this sense, the tool presented below is the embodiment of a potential future for game historical practice.

Burdick and fellows also call out the coming dissolution of humanistic and

¹[39] Borgman, Christine L. *Scholarship in the Digital Age: Information, Infrastructure, and the Internet*. Cambridge, Mass: MIT Press, 2007. [193] Siemens, Raymond George, and David Moorman, eds. *Mind Technologies: Humanities Computing and the Canadian Academic Community*. Calgary: University of Calgary Press, 2006. [202] Svensson, Patrik, and David Theo Goldberg, eds. *Between Humanities and the Digital*. Cambridge, Massachusetts: The MIT Press, 2015. [217] Warwick, Claire, Melissa M. Terras, and Julianne Nyhan, eds. *Digital Humanities in Practice*. London: Facet Publishing in association with UCL Centre for Digital Humanities, 2012.

²Resurrection could also apply to the physical reconstruction (or acquisition) of legacy hardware. However, this chapter is only able to comment on the data resurrection through emulation — for reasons that will become clear below.

³[45] Burdick, Anne, ed. *Digital_Humanities*. Cambridge, MA: MIT Press, 2012.

⁴[45] pg. 104

preservationist foundations.

As concepts of authorship, document, argument, provenance, and reference become increasingly unstable, concepts that are fluid, iterative, and distributive, but less “authoritative,” are taking their place.⁵

While the previous work in this thesis might serve as a push toward the capacity and need for “authoritative” records and concepts, game citation directly confronts — and calls for answers to — many of the issues outlined in the above quote.⁶ We will attempt to draw attention to problems of “authorship, document, argument, provenance, and reference” in this and the following chapters.⁷

The work in this chapter foregrounds the issues of reference as regards the notoriously unstable nature of computer games and software.⁸ These objects, in their requirements for future retrieval and use in historical arguments call for the ideation of what Bethany Nowviskie refers to as a “speculative collections”.⁹ These collections are those that do not yet exist, but will be required for future digital scholarship. The requirements of both game citation and the tool illustrated below invite the creation of a speculative future collection for managed and retrievable historical software data. Speculative collections call for the telegraphing of future scholarly use, and the creation of criteria for the descriptive, curative, and managerial requirements that would likely result.

⁵[45] pg. 109

⁶That “less authoritative records are taking their place” is actually more a symptom of not creating systems and “authoritative” sources that can better deal with new types of records, as we attempt to show both through the citation system below, and in the previous chapters on the appraisal of new forms of game development documentation and new descriptive apparatus for game software objects.

⁷Hopefully we have already scratched some of these itches in the preceding work.

⁸See [149] McDonough, Jerome P. “Preserving Virtual Worlds: Final Report,” Graduate School of Library and Information Science, University of Illinois at Urbana-Champaign, 2010. [161] Newman, James. “Ports and Patches: Digital Games as Unstable Objects.” *Convergence: The International Journal of Research into New Media Technologies* 18, no. 2 (2012): 135-142.

⁹[163] “Speculative Collections.” Bethany Nowviskie, October 27, 2016. <http://nowviskie.org/2016/speculative-collections/>. (Accessed 5 Apr 2017).

The future of game historical scholarship is then not only based on the stabilization of records for future retrieval, but also in the ways that that retrieval is enacted in practice, and available for further exploration and exploitation by critical computational methods and tools.

4.2 Citation

Citation is a foundational act in modern scholarship. Regardless of the discipline, any scholarly argument is based on or a reaction to previous work. Different fields use citation practice in different ways, but the major functions remain consistent.¹⁰ Citation operates on two fundamental levels. Within a text, it legitimates the knowledge claims made by an author, and provides support for their arguments and findings. It also, through the connections a citation makes to related works, ties an author to the social organization of the discipline(s) to which they are contributing. In fact, any discipline is essentially constituted by these networks of citations, the collections of links that form being dense enough to support further claims and disclaimers, rebuttals and denials. Studies of citation occupy the thoughts of numerous fields, from the quantitative analysis of bibliometrics, with its h-indexes and impact factors, to the applied socio-linguistic study of discourse analysis. The dual roles of citation practice, both in the form and content of knowledge links, and as the base for social practices within disciplines, are certainly ripe enough for a pluck into the basket of game historical studies. Before diving into the morass of game citation, and even more specifically, game data citation, the rest of this section will set up some background definitions and support structures from citation-adjacent fields. These will then be leveraged into

¹⁰[200] Sula, C. A., and M. Miller. "Citations, Contexts, and Humanistic Discourse: Toward Automatic Extraction and Classification." *Literary and Linguistic Computing* 29, no. 3 (September 1, 2014): 452-64. doi:10.1093/llc/fqu019. pg. 454-455

a fuller discussion of game citation and the citation tool as a speculation and intervention into future practice.

4.2.1 Citation in Use

As discussed, citation functions both within a text as a marker to other sources, and without as a tie between an author and a discipline. The citations found inside texts follow the prescriptions of the common practices within a field of study. Common guides for the humanities include the Modern Language Association (MLA), American Library Association (ALA), and the University of Chicago Manual of Style. Most engineering disciplines, including Computer Science, follow similarly organized research guides. In CS's case, a majority of the works are organized around the major conference guidelines provided by the Association for Computing Machinery (ACM) or the Institute of Electrical and Electronics Engineers (IEEE). These guides are the products of the study of bibliography, with its most prominent scholarly effect being the enumerative bibliographies — the “Works Cited” lists — found after the conclusions of monographs or conference publications. The constitution of bibliography entries is the result of practices in descriptive bibliography, a subset of analytical bibliography.

While bibliographic practice in the age of the Internet is in some corners lamented as both a lost art and potentially unnecessary,¹¹ we believe that coherent, consistent and standardized bibliographic description is still essential. Additionally, the practices of analytical bibliography, probably most impressively displayed in Frank Manchel's magnum opus, *Film Study: An Analytical Bibliography* can help to reconcile the production and history of academic disciplines.¹² Manchel's

¹¹[167] Parks, Tim. “References, Please.” The New York Review of Books. <http://www.nybooks.com/daily/2014/09/13/references-please/>. (Accessed Apr 5 2017.)

¹²[134] Manchel, Frank. *Film Study: An Analytical Bibliography*. Rutherford: London: Fairleigh Dickinson University Press; Associated University Presses, 1990.

4-volume, 2500 page work enumerates and describes the entire breadth of English language film and film studies produced between 1965 and 1990. Interesting, in line with the current project of scholarly support through tools, Manchel's work is emphatically indebted to SCRIPT/VS word processing system and the IBM 6670 Laser Printer for help in maintaining and organizing the necessary subject and author indexes required for his work.¹³ The practice of organized analysis of the works in a field can help to reveal new research directions and provide a solid base for future claims. Game historical work needs more time and effort devoted to these foundational, field-constitutive activities.

Returning to in-text bibliographic citation practices, we find that an explicit function of citation is the legitimation of claims. In the most recent MLA Handbook, the editors write that citation practice involves, "demonstrating the thoroughness of the writer's research, giving credit to the original sources, and ensuring that readers can find the [sources] . . . to draw their own conclusions about the writer's argument."¹⁴ Additionally, authors must provide a "comprehensible, verifiable means of referring to one another's work . . . to give credit to the precursors whose ideas they borrow, build on or contradict and allow future researchers interested in the history of the conversation to trace it back to the beginning."¹⁵ Authors need to legitimate their claims, both by crediting original sources and supporting dialogue with other scholars with whom they may agree or disagree. Citation also works as a means to pay an intellectual debt, as any addition to knowledge, being based on previous efforts, should acknowledge others' contributions.¹⁶

¹³[134] pg. 28

¹⁴[153] Modern Language Association of America, ed. *MLA Handbook*. Eighth edition. New York: The Modern Language Association of America, 2016. pg. 4

¹⁵[153] pg. 5

¹⁶[69] Eco, Umberto. *How to Write a Thesis*. Cambridge, MA; London: MIT Press, 2015.

In history specifically, the formation of modern historical discourse is prefaced on the “scientific” practice of accurate historical sourcing. Commonly attributed to Leopold Von Ranke and his continental predecessors, the development of footnote and endnote showed that the author had “done an acceptable amount of work, enough to lie within the tolerances of the field.”¹⁷ Citation persuades others to pay attention to a scholar’s work and thoughts. It cannot “explain the precise course” taken by a historian, but can “give the reader . . . enough hints to make it possible to work this out — in part. No other apparatus can give more information — or more assurance — than this.”¹⁸ Anthony Grafton, in further discussion of the preceding quotes, notes that the function of footnotes — the historian’s preferred mode of citation — is to give legitimacy to a claim and promote the authority of the claimant. Footnotes also provide means for enforcing the social space of historians through the inclusion or exclusion of particular works. In many cases, a notable omission provides for a deeper criticism than an antithetical reference because at least the latter claim is being recognized and confronted instead of ignored.

4.2.2 Citation as Discourse

Given that this chapter is devoted to the use and abuse of citation practices in game historical texts, we need to develop a suitable framing and terminology to discuss it. As alluded to by the discussion above, the act of citation involves the coherent and consistent description of a source within a text. This is usually in the form of a bibliographic footnote or endnote, or list of entries (enumeration) at the back of a text. Bibliography dictates that there is sufficient description

¹⁷[83] Grafton, Anthony. *The Footnote: A Curious History*. Revised edition. Cambridge, Mass: Harvard University Press, 1997. pg. 22

¹⁸[83] pg. 23

to allow a future researcher to recover the described source. This creation of a knowledge link to another's work, in light of bibliographic practice, is then a matter of practicality. One needs adequate description for future retrieval. What bibliographic notions do not cover is the use of a citation, a knowledge linkage, within the text itself. The correct form of a description for a source is not the same as how an author activates that source inside their text as a functional part of their argument. We then have to split the act of citation in two. The first part is 'simply' the description and positioning of a citation inside a text, and the other is an analysis of how that citation is used as a way to further the objectives of an author, and by extension that author's discipline. Luckily, a lengthy discussion of citation's effect on both textual composition and the social formation of disciplines is already quite progressed in the fields of bibliometrics and discourse studies. Building on all their hard work, we will swoop in and grab hold of a few key concepts, dangling them around a bit to knock off some unnecessary implications before dropping them into the nest of game practices.

The intersection of one text within another is an instance of *intertextuality*. Although intertextuality is used in a number of fields, our definition of it here is drawn from discourse studies, a sub-field of applied linguistics. One of the founders of the field, Norman Fairclough, describes intertextuality as "the property of texts have of being full of snatches of other texts, which may be explicitly demarcated or merged in, and which may assimilate, contradict, ironically echo, and so forth."¹⁹ Intertextuality, in Fairclough's consideration, implicitly calls to account the production, distribution, and consumption of texts. For production, the key consideration is on the "historicity" of texts, how they add to previous knowledge and expand a specific discursive chain of thought.²⁰ Distribution re-

¹⁹[72] pg. 84

²⁰"Discursive chain of thought" is basically the organization of the knowledge in a specific

flects on the network of texts and how they transform and flow into different types and fields.²¹ Fairclough uses the example of political speeches transformed into news reports. With consumption, “the intertextual perspective is helpful in stressing that it is not just ‘the text,’ not indeed just the texts that intertextually constitute it, that shape interpretation, but also those other texts which interpreters variably bring to the interpretation process.”²² All three implications of Fairclough’s “intertextual perspective” have a significant bearing on the interpretation of citation and reference in academic works. That texts are engaged with a disciplinary train of thought, flow and re-form based on context, and intimately involve presuppositions about the other texts they do (and could) use, are all points of reflection for both the citation practice of games, and the implications for a better use of the tool proposed below; it being a means of a new type of intertextual link for game history.

Fairclough’s intertextuality is drawn from a more elaborate framework based in French discourse analysis that traces all the way back to Michel Foucault’s *Archaeology of Knowledge*. Again, conveniently, Fairclough spares us a significant jaunt into continental linguistic theory by clearly delineating two notions of intertextuality, manifest and constitutive, that are both relevant to the discussion of citation. Manifest intertextuality is “where specific other texts are overtly drawn upon within a text,” forming a “heterogeneous constitution of texts.” To clear that up a bit, the quote ending the previous sentence is an example of manifestation, as is “Fairclough’s intertextuality” at the beginning of the paragraph. Both are specific, overt calls out to other texts, with the direct quote being a

discipline. All practitioners are contributing to the historical accumulation that furthers the course of their discipline and the history of its claims.

²¹There are parallels here with Latour and the translation of concepts in networks, as well as the use of inscription in the creation of research works. We will bring in some of the history of science and technology perspective on text formation, with its focus on practice, briefly below.

²²[72] pg. 85

more emphatic kind of manifestation. Constitutive intertextuality (for Fairclough “interdiscursivity”) is effectively the means of intertextuality for a specific text. How the configuration of its references, allusions, and implications for other texts, both explicitly mentioned and implicitly demanded, align to form a specific type of discourse for a specific audience.²³ Bibliographic actions, like enumerative bibliography, footnotes, and in-line citations are then all types of manifestation while the act of citation itself, as a social and knowledge linking activity, is a constitutive act.

Before extending the application of constitutive and manifest intertextuality to game historical discourse, two more key insights from Fairclough are useful. The first is the idea of a “presupposition” of a text. Sometimes presuppositions are just “propositions that are given for, and taken for granted by, text producers.” When engaged in the act of writing and assembling an argument, authors bring into their writing numerous pointers to other works — through manifestation — or ideas from other works that are assumed to be part of the general knowledge of an assumed reader. Or, crucially, part of the knowledge that one is assumed to take from an “other text” that is insinuated in the current one. “In many cases of presupposition the ‘other text’ is not an individual specified or identifiable other text, but a more nebulous ‘text’ corresponding to general opinion (what people tend to say, accumulated textual experience).”²⁴ A presupposition, through manifest citation, of a particular “other” author or “other” work, brings with it a host of assumptions based on an assumed experience on the part of the reader. This gap between presuppositions, as intended by an author and interpreted by a

²³We are careful to note that the assumption of an audience here is not only referring to the actions taken by an author, with an audience in mind, to clarify and align their text with others’ expectations, but also to the implicit knowledge that a potential reader will bring to a text given that it is in a specific discursive form. Fairclough goes into more depth on discursive types in his *Discourse and Social Change* [72], specifically in chapters 2 and 3.

²⁴[72] pg. 121

reader, leads us to a second point about ambiguity and the constitutive “surface” of a text.

In drawing together the heterogeneous network of texts that constitute a new one, there are times when certain parts may not fit as well as others.

Texts vary a great deal in their degrees of heterogeneity . . . [they] also differ in the extent to which their heterogeneous elements are integrated, and so in the extent to which their heterogeneity is evident on the surface of the text . . . Again, texts may or may not be “reaccentuated”; they may or may not be drawn into the prevailing key or tone (e.g. ironic or sentimental) of the surrounding text. Or again, the texts of others may or may not be merged into unattributed background assumptions of the text being presupposed. So a heterogeneous text may have an uneven and “bumpy” textual surface, or a relatively smooth one.²⁵

That we should pay attention to qualities of a textual surface — and the ways in which its imbricated texts do and do not comport with each other — motivates our framing for the discussion of the citation work below. If we are combining references in text to games, and other systems based on non-discursive experiences, their constitutive intertextuality needs to be examined, along with its effects on the resulting historical discourse.²⁶ We may want the alignment to have a specific texture, but we also need to be aware that that texture, that surface, is something deserving of reflective consideration and thought. How do the citation of games, and the juxtaposition of program and text affect the reader’s experience and comprehension of the argument? What does this do to the issues of presupposition and scholarly assumption? Below we discuss the current practice of game citation in light of both manifest intertextuality and presupposition.

²⁵[72] pg. 104

²⁶The organization of citations in a text and, in our case, the organization of text and running executable programs does also call out to various traditions relating to visual design and the juxtaposition of text and image, specifically in art criticism see [33] Berger, John, Sven Blomberg, Chris Fox, Michael Dibb, and Richard Hollis. *Ways of Seeing*, 1973.

One last implication of the heterogeneous constitution of texts is that it results in what Fairclough refers to as an “ambivalence” of argumentative meaning. “If the surface of a text may be multiply determined by the various other texts which go into its composition, then elements of that textual surface may not be clearly placed in relation to the text’s intertextual network, and their meaning may be ambivalent; different meanings may coexist, and it may not be possible to determine ‘the’ meaning.” The ambiguity inherent to any textual argument results from the fact that I — as the author — am removing other texts from their original context and constituting them into my own. As such, the onus is on me, the researcher, to both explain and refine the “other” texts in a way that supports my argument and that is interpretable to the reader. However, whenever I pull in other’s sources, especially in the case of a manifest quotation or below as a manifest executable, there is a resulting set of risks to my argument.

1. Disjunction — The referenced text or object could afford an interpretation that is different from the one I intended
2. Manipulation — I could be intentionally obscuring the reference or manipulating it to argue against itself, or to support a claim it does not make
3. Presupposition — The presupposition I am making in using the reference does not align with the actual experience of the reader nor, potentially, with my own previous experience because I am misremembering

All of these implications are ever present, but the third, presupposition, is the most relevant to game citation. “Presuppositions are effective ways to manipulate people, because they are often difficult to challenge . . . Manipulative presuppositions also postulate interpreting subjects with particular prior textual experiences and assumptions, and in so doing they contribute to the ideological constitution

of subjects.”²⁷ As a clarification, the use of “manipulative presupposition” is different from (2) intentional manipulation above. We are distilling the notion of the difficulty in confronting — or identifying — the built in assumptions made by authors and readers. We also extend the “textual experience” to “played experience” in the case of game studies, because, as the discussion below shows, the problem of presupposition does make claims in games “difficult to challenge” and also leads to the “ideological constitution of subjects” in regards to game genre, classification, and historical positioning and importance. Presuppositions are difficult to correct if they go awry because both the reader and the author are under a similar delusion about the content and shape of a reference. Poking below the surface of the text and retrieving the shared context for a presupposition requires significant effort, and in the case of games, might not currently be possible due to a lack of access (as highlighted in the previous two chapters).

4.3 Bibliography and Citation in Game Studies

In the previous section we outlined the function of citation as an act of both descriptive bibliography and as an intertextual mechanism in the creation of textual discourse. This section brings both of those concepts to bear on the current practices of game citation in game studies and game historical texts. While both fields are large, with a significant amount of publication activity, we will see that the extent of game citation is currently rather limited. Additionally, in game studies works there is a good deal of presupposition about the accumulated played experiences of both the reader and author. These assumptions are a major reason for the current lack of specific bibliographic guidelines. As Nathan Altice writes, in one of the only other analyzes of game citation practice,

²⁷[72] pg. 121

Our familiarity with and access to videogames is taken for granted, since many of us are old enough to recall first-hand experience with the entire history of videogames — a claim that cannot be made by scholars of other media. There is an implicit assumption that we all know what a *Super Mario Bros.* cartridge looks like, so why bother with thorough descriptions?²⁸

This “implicit assumption” of *Super Mario Bros.* is the result of the presupposition at work in game studies texts. It is not uncommon for game references to be scant or potentially non-existent. This is a problem because the assumption of contemporary, tacit experience with historical games cannot hold up past the current generation of scholars. Game citation practices, like those of appraisal and description in previous chapters, need to be addressed with a mind toward future historical scholarship and needs. The rest of this section will describe how citation functions in game studies works, and briefly point to further recommendations for their improvement. All of this is a set up to the introduction of the citation tool in the next section as a tool-assisted intervention into both the issues of citation standardization and presupposition of game play experiences.

As already noted, computer game bibliography and citation practice is wont for a set of consistent and thorough standards. Again, from Altice,

To claim that videogame bibliography demands a closer allegiance to the practices [of enumerative, and analytical bibliography] assumes that a unified practice called “videogame bibliography” even exists. At their best, videogame citations adhere to the barest enumerative models. Even in those texts that most seriously grapple with electronic artifacts as objects that exhibit physical properties worthy of description, such as Kirschenbaum’s *Mechanisms* or Montfort and Bogost’s *Racing the Beam*, videogames are still afforded scant bibliographic information.²⁹

²⁸[25] Altice, Nathan. *I Am Error: The Nintendo Family Computer Entertainment System Platform*. Platform Studies. Cambridge, Massachusetts: The MIT Press, 2015. pg. 334

²⁹[25] pg. 333-334

Altice’s claim of the lack of a “videogame bibliography” practice is not difficult to substantiate. As he states, many works that are intimately tied to the exploration of the material constraints and expressive as technical artifacts do not share consistent bibliographic practices. Both works mentioned above, Matt Kirschenbaum’s *Mechanisms* — a treatise on the oft-overlooked ambiguities in the expression of digital data — and Ian Bogost and Nick Montfort’s *Racing the Beam* — a platform study into the inner workings of the Atari 2600 — come from the same publisher, are intimately involved with the technical distinctions of computer software, and do not share a consistent practice in their bibliographies.³⁰ *Racing the Beam* is a part of a larger series (as mentioned in the Description chapter) of works on specific platforms. Each book investigates the constraints that a particular platform imposes on the expressive potential of its software. Each book also takes a different position on the placement, organization and depth of its enumerative bibliography of games. Some volumes, like Jimmy Maher’s on the Commodore Amiga, eschew any explicit listing of the games referenced in the text.³¹ Contrarily, works like Altice’s own *I AM ERROR* on the Nintendo Entertainment System, adopt meticulous, platform and media format specific reference schemes.

Now, given that there is a not a standard set of bibliographic and citation practices for software, and that most major scholarly organizations — like MLA, University of Chicago, and surprisingly even the ACM — lack any guidelines for software bibliography, it is not surprising that academic book publishers and authors do not maintain consistent practices. The work of the Game Metadata

³⁰[105] Kirschenbaum, Matthew G. *Mechanisms: New Media and the Forensic Imagination*. Cambridge, Mass.; London: The MIT Press, 2012. and [158] Montfort, Nick, and Ian Bogost. *Racing the Beam the Atari Video Computer System*. Cambridge, Mass.: MIT Press, 2009.

³¹[131] Maher, Jimmy. *The Future Was Here: The Commodore Amiga*. Platform Studies. Cambridge, Mass: MIT Press, 2012.

and Citation Project (GAMECIP), which looked at hundreds of game studies citations across a variety of online and print sources, also validates Altice’s (and our) assumption about the lack of consistent practice. In fact, even within the same journal, *Game Studies*, which does have an explicit bibliographic policy, there was still lax enforcement of descriptive citation practice.

Altice’s concerns also link bibliography and citation to the descriptive concerns of the previous chapter. He notes,

As a Famicom scholar, I may possess the terminology to describe that platform’s media but meanwhile lack the platform-specific knowledge to properly cite a PlayStation 2 game . . . Granting each [reference] its due description poses a sizable research challenge. One solution is to build up a body of platform-specific descriptions that others may use as a model . . . but such shared knowledge will take time and work.³²

Computer game bibliography is distinct from other media forms mainly in the complex of technical requirements needed to retrieve the object. Our platform and format vocabularies, outlined in Description chapter, speak to Altice’s call in a limited way by attempting to codify and standardize some basic descriptive information for computer games. The larger issue, however, is that “rich bibliographic records require a baseline technical understanding of the objects they describe.”³³ For game scholars concerned with the technical underpinnings and object materiality of software, each new platform, format, and data configuration incurs a significant descriptive cost. For items in Altice’s enumerative bibliography, each specific game is listed according, in part, to the configuration of components inside a Nintendo Entertainment System Game Pak, and in the case of emulated sources, the header information of a particular game data file. Clearly, for his arguments to validate, this level of depth is necessary, and it would benefit future scholarship

³²[25] pg. 337

³³[25] pg. 336

if others could take advantage of his classification schemes or even extend them into their own specific sub-domain of software.

Altice and others in the platform studies purview are more concerned with the material and technical conditions of games than other historical scholars. What works for platform studies might be overkill for other subdisciplines. However, at the very least, the technical information provided in a bibliography should be correct, and involve a level of detail specific enough to allow an unacquainted reader a fair chance to recover the source.³⁴ The lack of consistency in the description of computer game sources can damage the legitimacy of game historical arguments. As mentioned in the last section, one key way that a citation can fail is through a mistaken presupposition about the source it is referencing. This misalignment between the author’s expectation — or recollection — of a game play experience and that of a future reader’s is only exacerbated by incomplete and inconsistent citation practice. We now take up discussion of a particularly salient example of presuppositions in game historical work.

4.3.1 Presupposition of DOOM!

One significant difficulty in game citation is that games are not as recoverable as other media forms. Many institutions do not have software collections, and those that do struggle against the technical and material constraints of hardware maintenance and access. When these limited access scenarios collide with a lack of rigor in citation practice, the result is that many outputs of game scholarship rely on only the barest descriptions for games. They are used more as pointers to the concept of a particular game, as presupposition, than to an emphatic, playable

³⁴This is in line with the recommendation for “reasonable compatibility” in the Description chapter. If you’ll recall, it stated that a game resource in a collection catalog should provide granular enough information to give a researcher a reasonable guess at the technical apparatus required for the resource’s recovery.

instance of one.

To take a particularly salient example, Dan Pinchbeck's book *DOOM: SCARY-DARKFAST* relies, almost exclusively, on presupposition of game citations. The work contains manifest citations, mostly through in-line references, to 130 other computer games. Most are used in passing to articulate how a particular structural, thematic, or kinesthetic element from each game relates to those of *Doom*. The in-line references are of the form (Title, Year), leaving the reader to fill in the blanks based on their assumed knowledge of each title. Furthermore, given the breadth of games mentioned, it is likely that Pinchbeck has not played, or at least recently played, many of them. The references hang on a presupposition of his past experiences with the titles, and hopefully they still resonate in ways commensurate with his arguments. The references are, as we mentioned above, presupposed shorthand for the shared played experience of both author and reader.

To illustrate how this form of manifest, presupposed citation functions, take this set of paragraphs describing the progress of the first-person perspective from Pinchbeck's book:

We need to consider the context into which DOOM arrived. The very first FPS game was Maze War, created by Steve Colley, Howard Palmer, and Greg Thompson (and other contributors) at the NASA Ames Research Center. Colley estimates that the first version was built during 1973, as an extension of the earlier game Maze, which offered a first-person exploration of a basic wireframe environment. At some point during '73 or '74, networked capability was added, enabling multiplayer FPS play. The genre was born out of networked death-matching. After Thompson moved to MIT, he continued to develop Maze War, adding a server offering personalized games, increasing the number of players to eight, and adding simple bots to the mix. Twenty years before DOOM, all of the prototypical features of the FPS were in place: a 3D real-time environment, simple ludic activity (look, move, shoot, take damage), and a basic set of goals and win/lose conditions — all this and multiplayer networked combat.

Around the same time, Jim Bowery developed Spasim (1974), which he has claimed to be the very first 3D networked multiplayer game. Spasim pitted up to thirty-two players (eight players in four planetary systems) against one another over a network, with each taking control of a space ship, viewed to other players as a wireframe. A second version expanded the gameplay from simple combat to include resource management and more strategic elements. Whether or not Bowery's argument that Spasim precurses Maze War and represents the first FPS holds water, its importance as a game is undiminished — even if for no other reason than because Spasim is a clear spiritual ancestor of Elite (Braben and Bell 1984) and its many derivatives. It perhaps even prototypes a game concept that would later spin out into combat-oriented real-time strategy (RTS) or even massively multiplayer online (MMO) gaming.

What certainly differentiates Spasim from Maze War is the perspective. Like other early first-person games, such as BattleZone (Atari 1980) and id's Hovortank 3D (1991), the game is essentially vehicular, with no representation of the avatar onscreen other than a crosshair. It is interesting that, aside from occasional titles such as Descent (Parallax 1995) and Forsaken (Probe Entertainment 1998), the genre very swiftly settled down into the avatar-based perspective, abandoning vehicular combat more or less completely. It's also interesting that contemporary shooters often opt for a shift to third-person when including vehicles, such as with Halo: Combat Evolved (Bungie 2002) or Rage (id Software 2011). Half-Life 2's (Valve Software 2004) first-person car sequences are actually quite unusual.³⁵

These three paragraphs make reference to twelve games spanning a period from 1974 to 2011. *Doom* does not receive a full in-line citation since it is the topic of the book, and is addressed with in-line references in a previous section. Ignoring the general argument and focusing only on the citations and their relationship to the assertions being made on their behalf, we already encounter some significant issues.

Firstly, the citations are not particularly specific. *Descent*, for example, was released in six different versions for three different platforms in three different

³⁵[168] Pinchbeck, Daniel. *Doom: Scarydarkfast*. Landmark Video Games. Ann Arbor: University of Michigan Press, 2013. pg. 6-7

localities in 1995 alone.³⁶ The description “(Descent, 1995)” then does not provide enough information for a reasonable assumption about the particular version Pinchbeck played (or presupposed). Second, the citations presuppose a significant amount of knowledge on the part of the reader. When Pinchbeck remarks, “Spasim is a clear spiritual ancestor of Elite . . . and its many derivatives,” we (the reader) are required to understand — through presupposition — that *Spasim*, a first-person, cockpit oriented space exploration game is echoed in *Elite*, a similarly-perspectived first-person space simulation game. Clearly, that assertion incurs a familiarity with both games, and by extension knowledge of Elite’s derivatives. Finally, even though a game might be recoverable through the sparse citation provided, much of the discussion is still presupposed on the memory of played experiences of both Pinchbeck and his readers. In referencing the vehicular segment of *Half-Life 2*, Pinchbeck is theoretically requiring a future researcher, should they want to experience that sequence, to spend many hours of game time reaching and evaluating it. There is nothing inherently wrong with this, but we must highlight the extent of the assumptions being made of the reader. Either you already have contemporary experience of *Half-Life 2*, and incidentally remember this game play sequence, or you are relying on Pinchbeck’s memory of his contemporary play. Both positions presuppose a temporally situated accumulation of played experiences that aligns with the year of this work’s publication. Future researchers, at a remove from a contemporary, played understanding of the game, must assume that Pinchbeck is not committing any of the intertextual no-nos — like misinterpretation or incorrect presupposition — listed in the previous section. Otherwise, they will need to recover *Half-Life 2* for themselves, and assume that their version contains the vehicular sequence in question and that it is reachable

³⁶According to MobyGames, <https://web.archive.org/web/20160421081803/http://www.mobygames.com/game/descent/release-info> , *Descent* has 15 different releases, 6 of which occurred in 1995 in the United States, Japan and Germany for DOS, Mac and PC-98.

through play.

While it may seem that we are being a bit drastic in this example, we cannot take for granted that our own presuppositions about *Half-Life 2* or any other game discussed in the quote above (or, for that matter, in this text) will align with the presuppositions of future scholars.

Pinchbeck's references are intended to evoke a general idea of a specific title, relying primarily on the presupposition of reader knowledge. The referenced games in these cases stand in metonym for their specific constitutive function in the text. *Halo*, *Rage*, and *Half-Life 2* for their comparative vehicular segments; *Spasim*, *Elite*, *Descent*, and *Forsaken* for their 360-degree cockpit viewpoint; and *HoverTank 3D*, *BattleZone*, *Maze* and *Maze War* for their advances to first person representation. Concrete, retrievable instances of these games are secondary to the structural or thematic conceptualizations of them as presupposed into Pinchbeck's argument.

In contrast, recalling Altice's more extensive, object-based citations, we see that many of his claims are rooted in the minutiae of a single platform and its technical constraints. For Altice, his argument is dependent on the specifics, on the material differences between games rather than the higher level concepts they can evoke. He commonly uses emulated versions of games to illustrate points about Nintendo Entertainment System rendering techniques. Because rendering functions differ between the many versions of, say, *Super Mario Bros.*, Altice's citation of a specific version of the game's data is important, his analysis would not be possible — or legitimate — without it.

The lesson is not that anything Pinchbeck is saying is particularly incorrect, but that the onus for clarification is heavily weighted toward the reader, and in particular, a presupposition about the reader's accumulated knowledge of games.

Pinchbeck’s work is intended for a game savvy audience, and is certainly not attempting to be a rigorous, formal history of *Doom*. But the type of underspecified, game-as-shorthand reference structure is endemic to a significant swath of game studies. It will also make these types of work less relevant the farther they are displaced from the contemporary titles to which they refer. Again, in clarification with Altice, “Most contemporary game scholars are old enough to remember most of the entire historical arc of computer games, so further clarification for them, and audiences like them are not currently required.” Those in the future, unversed in the early history of computer games, will need to do a significant amount of work to recover all of the implicit game history embedded within Pinchbeck’s references.

Another note is that Pinchbeck’s citations are more the norm in current game citation practice. The GAMECIP study of citation practice analyzed citations in over 300 publications relating to computer games. Of those, 102 different styles of citation were found, and *of those* only 31 included any information about game platform. A majority simply focused on title, developer and year of publication. The main problem with this lax citation practice is that without at least a foundational set of descriptive elements tied to some expression of technical constraints and requirements, locating and replaying games referenced in scholarly works might be very difficult in the far future³⁷. Altice’s end of the spectrum, with its acknowledgement of computer game materiality grounding out into the literal byte order of a file header, is more historically secure in theory but requires a level of technical understanding that might turn off scholars with less techno-materialist concerns.

In the end, a probable solution is to provide a minimally viable set of descrip-

³⁷For more information on recommendations for citation guidelines as a result of the GAMECIP work, please refer to our unpublished citation recommendations.

tive bibliographic fields based, again, on assumptions of reasonable compatibility and retrieval. These minimum specifications and recommendations for citation practice can be found in forthcoming (as yet unpublished) work from the GAME-CIP project. The main thrust, however, is that for the legitimation of any argument made about or through a game, there is a requisite depth of citation that aligns with the claim. From the above, the depth of Pinchbeck's arguments dealt with apparent surface characteristics of games. Characteristics that would hopefully be apparent to anyone playing one of the games cited. In the case of platform studies arguments, the claims are more chthonic and dependent on citation at a different depth, one close to the actual material existence of the program.

Hopefully, this section illuminated some of the problems with current manifest intertextuality in games, most specifically that due to the current limitations of textual description, the field of game studies is dependent on a presupposition of played knowledge that is not tied to any specific material instances of games. The next section looks at this problem from the perspective of constitutive intertextual relations and provides a basis for our partial solution in the form of the citation tool for executable reference. This constitutive work is the result of confronting the current limitations of citations as they have been described so far. Primarily, when even the citation of specific, material data is not enough or of a kind with the expression of new historical claims.

4.4 Reduction and Intertextual Expression

Intertextuality is a fickle phenomenon. As noted by Fairclough above, when making one text manifest within another, work needs to be done to mold the "other text" in a form commensurate with the discourse surrounding it. Otherwise the textual surface is disturbed, and the flow of thought for the reader becomes more

difficult to reconcile and interpret. (Of course in some instances, this might be desired as a way to remark on the disjunction between different textual forms and different ways of reading.) Fairclough looks at newspapers, medical interviews, and other forms of discourse dissimilar from the academic text within which he is operating. He focuses on the ways in which each discourse's intertextuality contributes to its existence as a distinct genre, a distinct type of expression. This notion of constitutive intertextuality, the ways in which different discourses make use of and interact with other texts, is a fundamental aspect of discourse analysis. The constitutive act of bringing together "other" texts through manifest actions like citation, as noted by Ken Hyland in his study of academic citation practice,

links text users to a network of prior texts depending on their group membership, and provides a system of coding options for making meanings. Because they help to instantiate or construe the meaning potential of a disciplinary culture, the conventions developed in this way foreclose certain options and make some predictions about meanings possible.³⁸

The organization of disciplines and regions of thought and inquiry, both in the humanities and the sciences, are then dictated through the intertextual relations of publications. These publications organize into networks that then enforce and negotiate the boundaries of disciplines, and the specific intertextual discourse required for group allegiance. Additionally, it stands to argue — as we will for the rest of this section — that the intertextual surfaces of these genres of discipline make use of certain conventions that can preclude certain types of meaning and the expression of certain types of thoughts.

By relying on standard conventions of manifest intertextuality, and therefore prescribing limitations on the expression of academic claims, we are preventing certain discussions from taking place. In the interest of this thesis, we are most

³⁸[96] Hyland, Ken. *Disciplinary Discourses: Social Interactions in Academic Writing*. Applied Linguistics and Language Study. Harlow; New York: Longman, 2000. pg. 156-157

concerned with the explanation and historical positions of computer games as systems and technical objects. We remarked in the last section on the presupposition at work in discussions of game history. How references stand in metonym for more complex thematic components and system interactions. In a sense, the discussion was really about the limitations of current textual discourse about games that relies on the narrativization of game play or the accumulated knowledge of the reader as player. That game academics use text as the major form of expression is understandable. Michael Lynch, when discussing scientists use of text over visuals, notes:

The fact that writing is the dominant medium of academic discourse is not incidental; while pictorial subject matter is alien to written discourse, and requires a *reduction* to make it amenable to analysis, written subject matter can be iterated without any “gap” with the textual surface that analyzes it.³⁹

Games are even more removed from the textual surface than the visualizations Lynch is investigating and their insertion into textual discourse filters through many different levels of “reduction”. Lynch’s work focuses on the reduction of the worked scientific reality of the life sciences to the written page. “Scientific research teams are described as agencies of mediation between an uncertain and chaotic research domain and the schematic and simplified products of research that appear in publications.” Researchers select and distill the appropriate data and reduce it to visualizations and textual description to align with the constraints of printed publications. This reduction of the chaos of a research project to the streamlined and validated prose of research publications is of a kind with the work of some game historical scholars in their attempts to better mind the “gap”

³⁹[129] Lynch, Michael. “The Externalized Retina: Selection and Mathematization in the Visual Documentation of Objects in the Life Sciences.” *Human Studies* 11, no. 2 (1988): 201-234. pg. 201

between the played expression of games and their appearance and function in text-based academic arguments. Most of the time, game studies uses presupposition as a form of reduction, a way to fit the complex system interactions inherent to the experience of game play into readable discourse. This approach, however, largely limits the field of game discussion to their existence as “objects played by a researcher in the past”, preempting other means of using games in argument.

The notion of reduction is important to the overall discussion of intertextual surfaces and their effects on comprehension and expression. Reduction functions on a spectrum aligning with the goals of a particular discourse. The reduction from computer game to in-line textual citation is the most extreme form. Many others make use of, in progression: still images, sequences (or juxtaposition) of images, video, interactive visualization, and in limited cases, emulation. In monographs, and examples like Pinchbeck and Altice above, only still and sequenced images are available. Pinchbeck narrates key areas of *Doom* with comparative juxtapositions of different game versions, and single images of key aesthetic and level design features. Altice makes extensive use of emulator tools for the visual display of internal memory states, or again, like Pinchbeck comparative juxtapositions of key sequences or different passes of a rendering function. Outlining the full extent of image usage in game studies monographs is well outside our scope, but the important consideration is the jump in textual mediation that occurs in the transition from collections of images to video, interactive visualization or emulation. The textual surface described for the majority of this chapter is one of physical print and the constraints of its intertextuality. The newer forms of reduction are not mentioned by Lynch because they still remain unleveraged in the sciences — there are very few online publications in any field that leverage digital documents new textuality. Digital humanists cry out for more active, digital intertextual

presentation (citations here), but their codified expression is only standardized in a handful of online publications.⁴⁰ Linking back, the expression of academic discussions of games is then dependent on the forms of manifest intertextuality that are available and commensurate with the dominant constitutive discourses that currently exist. When people want to engage with games in ways that are not commensurate with textual description they make use of less encumbering reductions. In our case, when trying to either explain embodied system interactions or complex dynamic processes, it is helpful to have more than textual discourse as the only tool in the tool chest.

4.5 Types and Examples of Reduction

To clarify a bit, there are two key considerations at work in our discussion. The first is the intertextual surface of discourse and how it makes use of manifest actions, like citation, quotation, and images, in the constitution of a text. The flow of an argument is aided by the integration of “other” texts such that the discursive flow is smooth.⁴¹ Whenever one is reading through a discussion and needs to refer to a figure, table, or other interstitial manifestation, there is then, borrowing from Lynch, a “gap” in the textual flow and thus in the discursive progression. An aim for any apparatus that allows for new types of manifestation must also consider how that manifestation affects the constitution of a text, and the ways that manifestations can augment or potentially further distort a discursive surface. This surface is also, with advances in on-line technology and distribution methods, not only just a physical sheet of flattened, dead wood, but the transmediatic —

⁴⁰Prominent examples are Kairos, Scalar, and Vectors. <http://kairos.technorhetoric.net/>, <http://scalar.usc.edu>, and <http://vectors.usc.edu> respectively.

⁴¹Unless, as indicated above, the interruption of the flow serves a discursive function. The disjunction of meanings being relevant to some point or elaboration. Sometimes contrasts in discursive presentation inform on limitations of either one.

interactive — surface of the computer screen and networked document. The medium of expression, in the case of computer games and systems, is now of the same stuff of the medium being described and discussed. There is potential for a better and more forceful alignment of textual surface with digital system expression.

The second key consideration is how reductions assist intertextual integration to enable new forms of argument. We are not the first to venture down this intermedial path, and by illuminating some further examples we can highlight the new types of expression that we hope to enable with the to-be-described tool. This section is mainly devoted to an elaboration of the second point about methods of reduction in light of the first’s concern for intertextual alignment and comprehension. The section is a collection of related and motivating work.

4.5.1 Video

Recall that the methods of reduction not discussed by Lynch are embedded video, interactive visualization and emulation. “Embedded” is key since this allows us to present them as manifest intertextual objects (and later use some of the discourse analytical apparatus to discuss their effects). Reduction is a reduction of the embodied act of play to a form amenable with the constraints of the particular intertextual surface being created. Video reduction is fast becoming one of the major means for the dissemination of knowledge about how games are played, and the surface characteristics of their presentation to the player. As a phenomenon, this is beneficial to the progress of game historical study since at the very least there will likely be some video remnants of game play available to future preservation efforts.⁴² The availability of video has also prompted some aca-

⁴²[160] Newman, James. *Best before: Videogames, Supersession and Obsolescence*. Routledge, 2012, actually calls for game preservation policies to *prioritize* videos of gameplay on the

demics to begin experimenting with embeddable video as a means for discussion. For example, Doug Wilson, in an extended discussion of the game *Spelunky* for the Polygon website, makes extensive use of embedded YouTube videos to support a discussion and walkthrough of one of the game's most difficult achievements, a no-death solo eggplant run⁴³. By interweaving textual description with multiple embedded videos keyed to specifically salient moments, Wilson telegraphs a new type of intertextual surface where narrativized gameplay description is aligned with video supports. The technical mastery at work is made more apparent and visceral through the accompanying videos.

4.5.2 Visualization

Interactive visualizations as embedded arguments, the next step up the reduction ladder, are not a significant practice in academia. Certainly, visualizations — in the form of static images embedded in text — are very common in physical and social sciences, and in humanistic analysis of textual corpora. Woolgar and Lynch preside over two volumes dedicated to representation in scientific practice that are mostly focused on the constitutive power of manifest visualizations in scientific texts. The fact that the volumes are separated by 30 years belies the continuing influence of visualization on scientific work. Analysis of the effect of visualizations on humanistic practice is probably most expressed in the recent attention to algorithmic criticism in the works of Stephen Ramsay — for corpora analysis

assumption that executable access is a less likely future scenario.

⁴³As described in [226] Wilson, Douglas. “A Breakdown of 2013’s Most Fascinating Video Game Moment.” Polygon, December 23, 2013. <http://www.polygon.com/2013/12/23/5227726/anatomy-of-a-spelunky-miracle-or-how-the-internet-finally-beat>. (Accessed 5 Apr 2017). The “solo eggplant run” is a secret completion achievement for the computer game *Spelunky*; a rogue-like dungeon exploration game modeled on Indiana Jones and “explore the tomb” type motifs. The eggplant run involves carrying a useless item from the beginning of the game through completion with losing it or losing one’s life. It was so difficult that it took years before its first completion by Bananasaurus Rex in 2013.

— and David Staley — in historical visualization — among others.⁴⁴ However, the use of non-static, interactive visualization of dynamic systems is absent from the intertextual presentation of findings in most scholar fields. The groundwork is actually being laid more by those interested in the pedagogical application of visualizations.

Bret Victor and his collaborators are at the forefront of so-called “explorable explanations”, juxtapositions of online text and embedded interactive visualizations designed to reveal the functionality of systems.⁴⁵ Some examples include a long explanation of the basic dynamics of simple animated pathing, and Vi Hart and Nicky Case’s interactive model of Thomas Schelling’s group segregation theories.⁴⁶ Each example works to mollify the “gap” between the textual presentation inherent to the browser window and the machinations - and interactive features - of the supporting visualizations. Victor’s work espouses a pedagogical philosophy of system comprehension through manipulation and tacit experience. Readers are invited to read the expository prose, and then play with the interactive models of the phenonema on the page. The hope is that through tacit manipulation and play a deeper understanding of the underlying system will develop.

Victor refers to his online visualizations as “reactive documents” that allow

⁴⁴[172] Ramsay, Stephen. *Reading Machines: Toward an Algorithmic Criticism*. Topics in the Digital Humanities. Urbana: University of Illinois Press, 2011. [196] Staley, David J. *Computers, Visualization, and History: How New Technology Will Transform Our Understanding of the Past*. Second Edition. History, the Humanities, and the New Technology. Abingdon: Routledge, 2015.

⁴⁵However, being at the forefront of something is not the same as “inventing” it. And Victor’s work is heavily inspired by Alan Kay’s in “active essays” and Ted Nelson’s educational musings in “No More Teacher’s Dirty Looks” from [159]. For more information on the genealogy of the term see [231] Yamamiya, Takashi, Alessandro Warth, and Ted Kaehler. “Active Essays on the Web.” In *Creating, Connecting and Collaborating through Computing*, 2009. C5ãĀŽ09. Seventh International Conference on, 3-10. IEEE, 2009. <http://ieeexplore.ieee.org/abstract/document/5350160/>. .

⁴⁶[90] Hart, Vi, and Nicky Case. “Parable of the Polygons.” <http://ncase.me/polygons>, (Accessed Apr 5 2017). and [212] Victor, Bret. “The Ladder of Abstraction.” <http://worrydream.com/#!2/LadderOfAbstraction>, (Accessed Apr 5 2017).

the reader to test out the various models presented and gain insight through those interactions. The goal is to develop an “active reader”, someone who uses “the author’s argument as a springboard for critical thought and deep understanding.”⁴⁷ Active reading owes much to the foundational pedagogical insights of Seymour Papert. Papert devoted his career to the development of computational tools to aid in programming and algorithmic thinking.⁴⁸ He also believed that tacit experience with reactive systems would support better modeling capabilities in confronting new problems and challenges. He traced this potential to a youthful fascination with gears that implicitly enforced a tacit understanding of complex differential systems. His gears functioned as a model that enabled a better understanding of math “than anything I was taught in elementary school”.⁴⁹ The ability to present information in different and interactive ways led to a new model for the exploration of further knowledge. Papert linked this notion with the educational theories of Jean Piaget that espoused the “assimilation” of concepts into a learner’s world view. Papert’s gears functioned as an “affective model,” a mapping (assimilation) between their relational dynamics and other mathematical concepts. Victor’s work is then an attempt to embed these “affective models” as interactive visualizations into online documents.

The design and application of affective models that encourage comprehension of technical concepts is important for our larger argument about the potential for new forms of intertextuality to support new argumentation. Victor presents a prototypical means of doing more with online texts, and trying to engage the reader with the systemic processes under discussion. Victor’s reactive documents

⁴⁷[214] Victor, Bret. “Explorable Explanations,” 2011. <http://worrydream.com/#!/ExplorableExplanations>. (Accessed Apr 5, 2017.)

⁴⁸A note here that Papert later career focus on computational tools is best described in [165] Papert, Seymour. *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books, Inc., 1980.

⁴⁹[165] pg. vi

are a new discursive surface. One populated with interactive features aimed at creating a new type of active reader. They are also the result of a reduction from larger, complex system dynamics to concentrated, pedagogical visualizations designed to support textual arguments. The reduction, however, is much richer than an image or video, since it can support the creation of a tacit, embodied argument. Instead of referencing an image or video of a system processes, a smaller part of the system can be introduced into the discourse describing it. Or better yet, use the interactive surface as an argument in of itself for a particular point of view or affective process.

4.5.3 Emulation

Before discussing emulation as a form of reduction — in line with the progression outlined above — we need to clarify some basic technical distinctions and provide some related examples. This is necessary because the use of emulated systems as a form of argumentation about software history — or really any topic for that matter — has not before, to our knowledge, been explored or theorized.

Emulation, as a computational process, is the use of one system in reproducing the functionality and output of another.⁵⁰ Emulators, the programs responsible for emulation, are used in many corners of the software industry to allow for testing of applications on a variety of devices. For instance, most mobile phone applications are not developed on mobile phones. They are programmed in emulation on laptops and desktops more conducive to long bout of typing and heartache. As

⁵⁰More technically, it is “a technique for implementing a virtual machine on a host computer whose instruction set is *different* from the host computer’s.” [183] Rosenthal, David SH. “Emulation & Virtualization as Preservation Strategies,” 2015, pg. 2. Although the line between virtualization and emulation does get a bit murky by this definition — the computer I’m typing this on, a MacBook Pro running Apple Mac OS X 10.11 shares the same basic instruction set with a Sony PlayStation 4 — the encapsulation of one system within another is the basic function of emulation.

noted by Nathan Altice, emulation has a long history tracing back to the historical origins of software development. In the 1960s IBM developed the first sets of commercial emulators to allow software written for one mainframe model to be compatible with another.

More recently, in the 1990s, enthusiast game communities began to create emulators of popular game systems, like the Nintendo Entertainment System. Targeting then-current operating systems like Windows 95, these emulators allowed for the replay of older titles that might no longer be available for purchase, were released in foreign territories or might otherwise be difficult to acquire. The “games” in this usage were data dumps extracted from the physical cartridges, and other forms of magnetic and optical media. For cartridge systems, these files are known as “ROM”s since they are copies of a cartridge’s read only memory. A more general term for data extract from a media format is a “data image,” more commonly shortened to “image”.⁵¹ Emulators operate on data formatted for a specific system, and over the last 20 years, emulation development and the extraction of legacy data from physical formats has flourished. Emulators now exist for thousands of different computational platforms, and are a ripe source for the exploration of software history. That is, assuming one ignores some significant legal issues.⁵²

⁵¹The use of “image” in referring to an extracted data set is drawn from operations in mathematics. When you use a function to operate on a set of numbers, you are mapping the input values with potential outputs. In the case of $y = x + 1$, y is a function of x ($f(x)$) with x standing for a range of input values, and y for the output values of the function. The input values in this case “map” through the function to a specific set of output values. This resultant map, which is just the set of inputs each incremented by 1, is an “image” of those inputs in a new domain. Similarly, the data extracted from a physical medium is not the same data (nothing changes from the physical medium’s point of view) but a mapping of the data stored on that medium to an equivalent set of data now migrated from the media to another machine. Thus, “imaging” is very directly the process of mapping the data stored on a specific physical media to an identical configuration on a separate machine with its own storage. This distinction between types of data, and the means by which they are migrated, mapped and translated between machines will become relevant below for issues of reduction.

⁵²Emulation, specific in its copying and use of potentially copyrighted data, is in murky legal

Important for our discussion of manifest intertextual presentation are recent developments in web browser technology (and speed improvements in general) that now make it feasible to run emulators inside online documents. The potential for embedded emulation has not yet been exploited or thoroughly explored. However, because most web browser now run highly optimized JavaScript compilers, in-browser emulation is a growing phenomenon. The most prominent example is the JavaScript *Multiple Arcade Machine Emulator* (MAME) project. MAME began as a system for emulating arcade machines. Complementarily, the *Multiple Emulation Super System*, using a fork of MAME’s code base, provided support for most personal computers. MAME recently went open source and MAME and MESS are now merged. The combined infrastructure supports thousands of different arcade machines and personal computers released over the last 40 years⁵³. Initially a large C++ project, the Internet Archive, along with a collection of motivated developers, ported key components of MESS — before its integration with MAME — to JavaScript to create the Internet Arcade, a playable archive of the Internet Archive’s imaged software collection. After the open-source combination, JavaScript became one of a number of compilation targets for the entire MAME-MESS code base.

Similarly, many other emulators began organizing compilation to JavaScript. This included the emulators *DOSBox*, for legacy x86 MS-DOS machines; *FCEUX*, for the Nintendo Entertainment System; and *Snes9x*, for the Super Nintendo Entertainment System⁵⁴. A technology known as Emscripten, initially developed

territory. We will not significantly address the issues here, since they are out of scope with the discussion, and also out of spirit with the desire to reclaim and explore the history of games. For more extensive legal information on this topic, check out the aforementioned [183] and [149].

⁵³The MAME project is infamous for its attention to detail and support for esoteric systems. One recent addition included a Sonic the Hedgehog popcorn vending machine with embedded display that was marketed only in 1993 in certain Japanese cities.

⁵⁴Many other emulators now have JavaScript versions, the three listed above are the three

by the Mozilla Foundation, hastened the emulation porting process by providing a means to compile emulators written in the C programming language directly to JavaScript. We will discuss Emscripten more technically in the citation tool description section below.

Now that emulation is available in browser, it is possible to place both a running program, and the text describing or commenting on it, onto the same intertextual surface. In the progression of reductions from images to interactive visualization, there was always a clear notion of how each step still represents a deficient copy of some object or system outside the text. For images and video, as mentioned by Lynch above, researchers put in a significant amount of effort to both make their samples more photogenic and thus more interpretable when presented on a textual surface. In dynamic visualizations, there is an implicit understanding that we are being presented with part of a system that has been distilled for comprehension and reader engagement. The very act of visualization is to provide a specific perspective (of many) on the data or system under discussion. With emulation, there does not appear to be a similar process at work. While the emulation is a program designed to conform to the constraints of a digital document, it is not a distillation of a system but a full version of the system itself. This challenges the basic premise of intertextuality presented above, that the texts made manifest in and reduced to a specific discursive surface are under the basic control of the author. While some manipulation or presupposition — in cases where the author is misremembering or functioning with a divergent set of assumptions in regards to the reader — is always a potential issue, that the manifest references might have a mind or operation of their own was never

we use in the citation tool and so are explicitly mentioned. For a full listing of emulators (including JavaScript) check https://en.wikipedia.org/w/index.php?title=List_of_video_game_emulators&oldid=784194013 (Accessed June 8 2017) which keeps a running list of projects and compatibility.

imagined. In bringing emulation into the text, we therefore encounter a new type of intertextual interaction, and with it a different model of reduction. One that requires significantly more effort in the legitimation of claims and the interpretive exercise. Presupposition of played experience is no longer possible because the system — the *same* system — is available to both author and reader.

The imbrication of emulation into argumentative texts has only been lightly attempted in the past. Nick Montfort wrote an article for the Electronic Literature Review that embedded a Java plugin-based emulation of the Infocom game *Deadline*.⁵⁵ However, his argument does not particularly integrate the emulation so much as position it in the article to simply show such a move is possible. Others have used online emulation for deeper systemic analysis. One notable example is Ben Fry's early online visualization of the internal memory state of the Nintendo Entertainment System. The "deconstructulator" is a Java-based in browser emulator based on a modified version of the NES Cafe emulator⁵⁶. The visualization presents three different windows, a rendering of the full sprite memory of *Super Mario Bros.*, a playable emulation of that game, and the active memory contents of the NES's PPU (Picture Processing Unit), a component that manages sprite rendering on positioning on screen. As the player plays *Super Mario Bros.* the contents of the sprite map highlight the currently active sprites in use, and the PPU map show the current state of each 8x8 tile in the PPU. By moving Mario around, the player can see how the different animations and changes to the

⁵⁵[156] Montfort, Nick. "Cybertext Killed the Hypertext Star | Electronic Book Review," December 30, 2000. <http://www.electronicbookreview.com/thread/electropoetics/cyberdebates>. (Accessed 5 Apr 2017). The Java-based plugins are now flagged as security threats, and even forcing the browser to ignore those warnings did not result in the emulations executing correctly.

⁵⁶On a preservationist note, while the "deconstructulator" is still available online, it requires a Java-plugin to function. Due to security concerns over the last decade, Java support has been dropped or disabled in many browsers. There is no longer any mention of the NES Cafe emulator on its creator's website, and its most recent update (July 22, 2008 according to [8]) is over 8 years old as of this writing in 2017.

game’s background, enemies and platforms modify the NES’s system memory. Fry designed the piece for his “Visually Deconstructing Code” series, a set of small projects aimed at unearthing some of the hidden processes at work in NES ROM code.

Other examples of emulation as a revelatory mechanism exist within the communities devoted to forms of what James Newman refers to as “superplay”. Activities like speed-running, both human and tool-assisted, glitch-hunting, sequence-breaking, and other forms of “performative mastery” of games benefit from research conducted with emulation. Many community emulators support tools for memory analysis and even scripting languages for the live manipulation of a running game. This allows player-performers interested in, for example, shaving that last second off of a run or getting past a boss without attacking, to dig beneath the representational surface of the game and mine its system for potential solutions. Some online streamers, like Clyde Mandelin, write custom emulator modifications that allow for live streaming of both their gameplay and aggregated statistics or interactive visualizations of the underlying system.⁵⁷ As we will discuss below, it is becoming clear that a community of practice is developing around the expressive potential of emulators. It’s also a sign that the products of community historical efforts are becoming more aligned with digital humanist insights about technical collaboration between academia and amateurs.⁵⁸

In placing a running program into an online text, its reduction to that surface implicates a raft of potentially disingenuous assertions about the historical play experience. As outlined extensively in the work of preservation minded historical

⁵⁷<https://www.youtube.com/user/ClydeMandelin>

⁵⁸See chapter 10 “Unreliable Archivists” in [177] Rinehart, Richard, and Jon Ippolito, eds. *Re-Collection: Art, New Media, and Social Memory*. Leonardo. Cambridge, Massachusetts: The MIT Press, 2014. for a more detailed discussion of the benefits and perils of incidental community archival practices.

researchers, emulation, in its erasure of the original executable context, denies the experience of the original hardware.⁵⁹ The modern web browser, as a displayed surface, is very different from an Atari-era CRT, and most modern machines do not have way to interface with original peripherals. Additionally, many emulators try to make the played experience smoother by modifying speed for the sake of accuracy.⁶⁰ They also remove old constraints on the swapping of disks to load parts of the program in piecemeal and internal memory limitations. However, not all aspects of emulation are a historical loss, since the position of the emulation as running inside a host process allows for the introspection and revelation mentioned in the above examples. The ability that many emulators provide to save and load memory states is also, as we will see, a boon to players and researchers hoping to encounter difficult, confusing or novel locations inside games.

4.5.4 Closing

A key note about the reductions above is their ability to bring something from “out there in the world” into the text. Usually those studying academic discourse, or the social construction of academic arguments, focus on how that external evidence is transformed into a manifest object in the text. For scientific work, we have mentioned both discourse analysis and science and technology studies as fields that theorize on the reduction and distortion of tacit laboratory knowledge into written discourse.⁶¹ In the humanities, citation and reduction are less epistemologically fraught, since the aim of humanistic discourse is not gener-

⁵⁹[103, 158, 160, 128, 74]

⁶⁰Some emulators running in constrained environments, like JavaScript emulators in web browsers, need to cut corner to get processing up an acceptable speed. Other emulators running in native execution contexts, like Microsoft Windows applications, sometimes intentionally slow down processing in order to match the timings of older machines.

⁶¹See [229] Woolgar, Steve. “On the Alleged Distinction between Discourse and Praxis.” *Social Studies of Science*, 1986, 309-317. for a discussion of the different philosophical roots behind discourse analysis (Continental philosophy) and STS (Anglo-American Analytic philosophy).

ally to re-organize some empirical object or finding into a textual form suitable for publication.

Within history, the use of manifest intertextuality is critically important to the sustenance of the field, but the act of manifestation does not usually imply a reduction of a finding “out there”; the “out there” of historical sources being mainly other texts. Rarely is the historical object, if there is one, reduced to a form commensurate with the textual surface. In fact, much historical work into objects specifically addresses this issue, a good example being John Law’s work in aircraft design that explicitly constructs different historical strands of documentation to reveal the fractal nature of the object in question. In his case, he looks at the construction of the British TSR2 strike and reconnaissance aircraft, and how it exists as an object of engineering, marketing, and an embodiment of the projection of hegemonic force. The aircraft is viewed along different evidentiary axes to support a conclusion about how objects exist in myriad ways depending on how they are documented and narrativized.⁶² This again ties back to ideas from discourse analysis, mainly in how the constitutive intertextuality at work in the history of science and technology defines the objects of analysis; a summary from Steve Woolgar:

Surely, it is often said, it is absurd to say that we cannot distinguish between a thing and what is said about that thing. But the constitutive view does not prohibit such distinctions. It offers us a way of seeing these distinctions as actively created achievements rather than as pre-given features of our world. In particular, the distinction between talk and objects-of-talk is seen from the constitutive perspective as the upshot, rather than the condition, of discursive work.⁶³

The fundamental take away is that previously, describing any technical system as a historical object necessitated various forms of reduction and other intertextual

⁶²We will be discussing John Law’s effect on our own historical modeling in Chapter 6, “A Model of *Doom*”.

⁶³[229] pg. 314

strategies to remediate and insert it into a text. With embedded emulation, and to a lesser extent dynamic visualization, embedding the system itself must now be reconciled. When John Law describes his aircraft, he could not bring the aircraft into the text and let the reader hop into the cockpit, and with computational systems increasingly the site of construction and reception for scholarly work, but with technical historical objects that are also computational systems, we can literally transcribe them into discourse and invite the reader to take the flight stick.

4.6 Back to Citation and Archives

In bringing a non-text-based object into textual discourse, like the reductions of image, video, visualization and emulation above, there is a key link to archives and citation that has not been made explicit. In the case of online documents that incorporate various reductions, those texts are not singular objects but networked organizations contingent on access to the various sources of reduction. If one prints an image alongside text, the image is now part of the textual form, and is, from an archival standpoint, part of the same object. With online work, every page of information is an aggregated object. The basic markup for the page comes from one source, the styling of that page from another, and all the various images and other embedded entities from still others within that same domain or from some other (hopefully trusted) source. When something is embedded, as the images, videos, visualizations, and emulations are, they necessitate and depend on the existence and maintenance of stable links to recover their data.

The maintenance of these links is a significant issue for the stability of knowledge online. Whenever a link leaves its local namespace (assuming that internal network links are maintained, which is not always a given) it relies on the ex-

istence, capabilities and restrictions of a remote hosting repository. For videos, most content links resolve thanks to the embedded link architectures of mass scale video sites like YouTube or Vimeo. This allows the embedder to not have to maintain their own video server nor provide the bandwidth necessary for playback. It also removes responsibility for intellectual property management and ties access to embedded content to the whims of the content provider. In the Spelunky example above, Wilson toyed with the constraints of YouTube's embedded video player to reveal specific salient content inside the game. That action only made possible by the affordances of YouTube, the repository hosting the content. In the case of historically stable online academic discourse, it should be apparent that any new ability to share or link to digital data incurs a commensurate necessity for a functional and stable repository. The current solutions for video leave a lot to be desired given that they are bound to the corporate imperatives of actors not emphatically concerned with preservation or link stability.

The tool below is an attempt to organize a prototypical archive for embedded emulated content, and to try and reconcile some of the manifest intertextuality present in game historical work to a more stable set of practices regarding citation and linking. In the case of embedded emulation, the data has similar issues to that of video. Namely, that the IP rights for the distribution of streaming copyrighted gameplay data need to be correctly managed, and that the embedded content be presented in such a way to make it useful for inclusion into texts. The consideration for future scholarly use of emulated content is a way to dictate what a speculative collection of such works would look like at a larger, institutional scale. Additionally, the consistent citation of this content, as an initial condition of the system's functionality, should be a concern for any future work in the creation of links to new forms of digital expression and reduction.

4.7 A Tool for Descriptive and Manifest Citation of Games

This section outlines the design and functionality of the citation system in the Game and Interactive Software Scholarship Toolkit (GISST). GISST is a suite of tools aimed at helping with common game studies and game historical tasks, and includes a system for the management of manifest citation of both game emulation and game bibliographic references.⁶⁴ The citation component of GISST — described below as the “citation tool” — is designed to partially address numerous issues presented above:

1. The need for a more consistent bibliographic citation information for computer games
2. An example use case for the placement and manipulation of various reductions of computer games into online text. In this case, images, videos and live emulation.
3. The need for a managed archive of the reductions used in (2)

The citation tool functions on three classes of objects, games, performances, and game system states and applies points 1-3 to each. Game objects are collections of data about a game. This includes both basic descriptive metadata, required for correct bibliographic entries, and links to executable data needed to run the game in browser emulation. Performance objects are records of games as played or viewed by a player or group. These records are also split in a way similar to game objects with viewable performance data being paired with descriptive

⁶⁴GISST as a full system has yet to be realized, and the citation functionality described below led to a potential for a larger future toolset.

metadata. By “viewable performance data” we mean either a collection of frames — gifs or video — representing some situated act of play, or replay data — input stream recordings for emulators or replay files for a specific game engine. Game system states are snapshots of a game’s run time memory, either saved by the emulator as a separate file or extracted directly from a system’s memory. The tool manages game, performance, and game state records in a server side database, and allows for the embedding of any (assuming executable or viewable performance data is available). The rest of this section briefly accounts for the inclusion of performance in our citation apparatus and then lays out the functionality and potential future work for the tool.

4.7.1 Game v Performance

The discussion above mostly dealt with the citation of game objects as a means of presupposing their content and the contours of their gameplay. However, game performances *as events* are also commonly referenced in scholarly works. Performances result from two activities, games-as-performance, in the case of games tied to explicit geo-temporal contexts — ARGs, installations, etc. — or gameplay performance. Gameplay performance is the play of a game that is not explicitly tied to a geo-temporal context, but that gains relevance from being situated in one. An example is a particular match at a fighting game tournament, where the event itself circumscribes gameplay performance. The game in this case is not the operative site of performative relevance, its game play at the tournament is. If the same match occurred in practice in a dorm room, no one would care.

Game performances as significant historical sources are well discussed in the literature. Clara Fernandez-Vara, in her *Introduction to Game Analysis* notes that, “we may want to analyze a game that is an event, a be-there-or-be-square

type of thing, a performance.”⁶⁵ She describes the need for secondary sources — “paratexts” like video or firsthand description — in helping to reconstruct and corroborate information about a performance. This sentiment is echoed in Henry Lowood’s work on the reconstruction of events that take place in massively multiplayer online games (MMOGs).⁶⁶ In this case, the study of virtual world game play is more akin to anthropological work. The game itself, while it could be recovered and run through emulation in the future, is devoid of the community that created meaning through the performative space and affordances the world provided. Lowood remarks on the fallacy of an ideal “perfect capture” of every event and input supplied to the virtual world.

Even if we save every bit of a virtual world, its software and the data associated with it and stored on its servers, along with a replay of every moment as seen by players, it may still be the case that we have completely lost its history. The essential problem with this approach is that it leaves out the identification and preservation of historical documentation, and these sources are rarely to be found in the data inside game and virtual worlds or on the servers that support them.⁶⁷

Even with access to game replay files, or reproductions through emulation, evidence of a game performance must also be paired with secondary information to substantiate and analyze it. Our inclusion of performance citations in the tool is to enable a link between the game object’s data and description, and further contextualizing performances. Additionally, the ability to embed emulation in line with historical performance video and description, adds further potential for somatic contextualization of game play. By bringing the emulated system to the reader, they can gain a sense of what Steve Swink refers to as “game feel”, an

⁶⁵[74] Fernández-Vara, Clara. *Introduction to Game Analysis*. New York: Routledge, 2014. pg. 44

⁶⁶[128] Lowood, Henry. “Perfect Capture: Three Takes on Replay, Machinima and the History of Virtual Worlds.” *Journal of Visual Culture* 10, no. 1 (April 1, 2011): 113-24. doi:10.1177/1470412910391578.

⁶⁷[128] pg. 118

embodied understanding of the game system as felt through the act of play.⁶⁸ Pairing this embodied understanding of a game play system with performances adds another level of intuitive understanding to historical game play acts.

4.7.2 Citation Tool

The citation tool has two primary components:

1. A command line interface (CLI) responsible for the ingestion of game and performance data, the generation of citations, and the management of the citation database
2. A web application (the “app”) that enables the live emulation of ingested game data, the live recording of game play performances, and the live recording of computational game states

For the rest of this section we will use CLI to refer to the first component, and “app” to refer to the second. Their functionality is significantly inter-related, for example the CLI command “serve” launches the app, and the app’s backend server calls the CLI for certain processing tasks. We will attempt to the best of our ability to be clear about the particular component under discussion. The next two sections provide a brief technical overview of both components. Additionally, the image below illustrates the relationships between the various components and we will refer to various features by number throughout this section.

⁶⁸Swink’s “game feel” is focused exclusively on continuous input games, like platformers or action titles. Doug Wilson has argued that “game feel” should extend to other types of interactions with computational feedback systems, from menu systems to mouse interaction in strategy games. We take the latter, more liberal view of game feel in the context of providing an emulated system in argument for the significance of a performative act or as a means of elaborating on a deeper understanding of embodied play experiences. See [203] Swink, Steve. *Game Feel: A Game Designer’s Guide to Virtual Sensation*. Burlington, MA: Morgan Kaufmann, 2009. and Wilson’s talk, A Tale of Two Jousts: Multimedia, Game Feel, and Imagination. <https://www.youtube.com/watch?v=JkbCNMAS0qI&feature=youtu.be>, (Accessed Apr 5 2017).

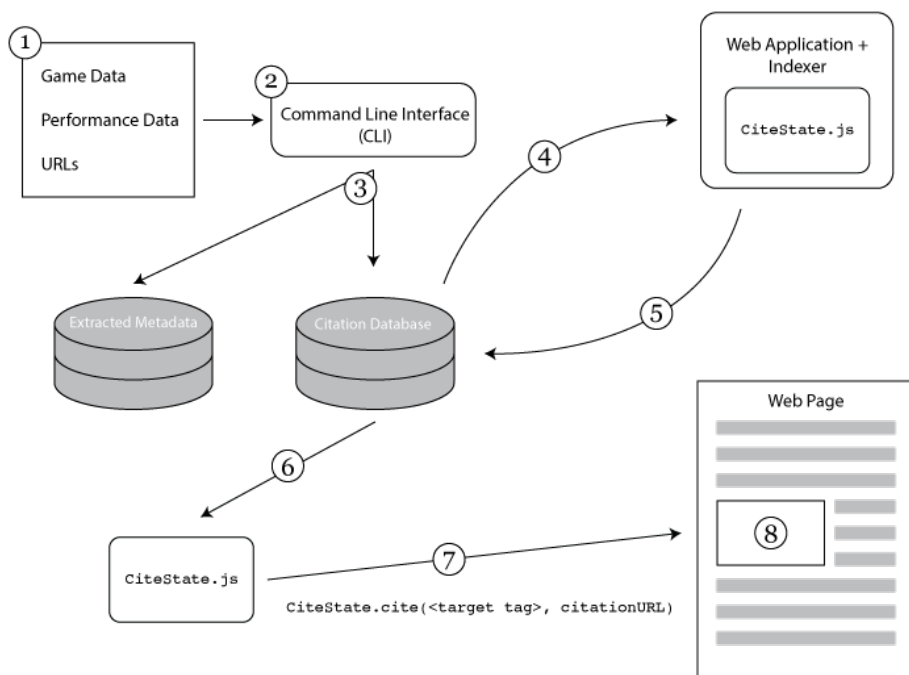


Figure 4.1: GISST components and pipeline.

Input resources (1) are fed to the CLI (2) that extracts their information (3) into an extraction table — for URLs — or the citation database — for performance and game data. The Web Application reads from the citation database (4) and the Indexer uses `CiteState.js` to create further citable resources (5). `CiteState.js` can then use those resources' permanent URLs (6) and its cite function (7) to embed an executable program into a target HTML tag (8).

4.7.3 Command Line

The CLI is a collection Python command line scripts that manage the extraction and citation of game and game performance data — 1-3 in Figure 4.1.

Usage: `gisst [OPTIONS] COMMAND [ARGS]...`

Options:

`--verbose` To everything that's going on.
`--no_prompts` Turn off all user prompts (use with care).
`--version` Show the version and exit.
`--help` Show this message and exit.

Commands:

`cite_game` Create a game citation.
`cite_performance` Create a performance citation.
`clear` Clear local data
`delete` Delete a citation by uuid
`extract_file` Extract metadata from a compatible file.
`extract_uri` Extract metadata from a compatible url.
`gif_performance` Create a gif from a performance citation.

<code>search</code>	Search for citations with a game partial.
<code>serve</code>	Run local access server for citations.

The CLI's help command outputs the above list of commands and options. Of interest to the current chapter are the citation commands, `cite_game` and `cite_performance`, and the extraction commands, `extract_file` and `extract_uri`. The CLI manages a simple database of information about games and performances. "Games" are simply a collection of descriptive metadata elements populating a table in the database, with the potential for associated game data. Similarly, "performances" are any play recording associated with a particular game reference. The performance metadata is stored in the same database as the game references. Each performance can be linked with viewable performance data, usually either a video recording or a tabulated input format for one of the tool's supported emulators.

The CLI provides for two basic actions, extraction and citation. Extraction commands accept either local filenames or Universal Resource Identifiers (URIs). For games, the extraction files are either game data files, or directories containing game data. Providing a URI to the game extraction command assumes that the link provided hosts information about a particular game. Currently, extraction supports game reference URIs for either MobyGames or Wikipedia. Performance extraction only accepts URIs from YouTube.

Extraction performs a first, unedited pass on the data provided by a specific source. If a file source is not recognized, it will be stored as a generic object in the database. Unrecognized URIs result in errors. Extraction is needed to construct a stable citation because the information provided by a potential resource may exceed the constraints of the descriptive metadata scheme or require further disambiguation. As an example, the Wikipedia page for *Super Mario Bros.*, originally released for the Nintendo Entertainment System, combines all information

about the title, in all of its different versions and releases, onto a single page. The extractor presents all of this information to the user, and allows them to choose the particular version of *Super Mario Bros.* they wish to later cite.

Citation enables the creation of a fresh game or performance record. If source information is provided and is specific enough to ascertain a unique game or performance, then the system will automatically create a citation and check for duplication. Any citable game or performance data is then available for closer inspection in the app. The basic extraction and citation pipeline is illustrated in Figure 4.2 below.

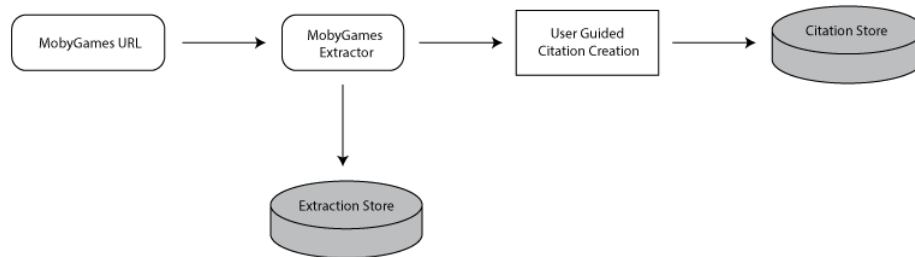


Figure 4.2: Basic CLI Pipeline

The CLI currently allows for the extraction of a variety of sources and file types, as shown in Table 4.1 below:⁶⁹

Any source data that is extracted, currently game files and videos, is linked to a dependent citation entry. This allows for the recovery of source data in the web application interface through either emulation or video playback.

⁶⁹The table does not include game states because the CLI does not ingest arbitrary state data. This is mainly due to the fact that emulated state data is specific to both a game and the emulator supporting it and effectively useless without those dependencies. Additionally, in the case of some of GISST's supported emulators, like DOSBox, there are no independent save state formats, just data derived from the live emulation during run-time.

Table 4.1: GISST Supported Resources

Citation Type	Supported File Types	Support URI Source
Game	.NES ROM Format .SMC ROM Format Any directory containing a DOS compiled executable .z64 ROM Format (partial)	MobyGames Wikipedia
Performance	FM2 replay format Generic Video Files	YouTube

4.7.4 Web Application

The app is standard browser-based web application, with a JavaScript/HTML/CSS front-end designed for use with the Chrome web-browser, and a backend interface that is linked to the same database and ingestion commands as the CLI. This is marked as steps 4 and 5 in Figure 4.1 above. The app allows for the play, recording and citation of game states through the use of JavaScript based emulators. The basic architecture is outlined in Figure 4.3 below.

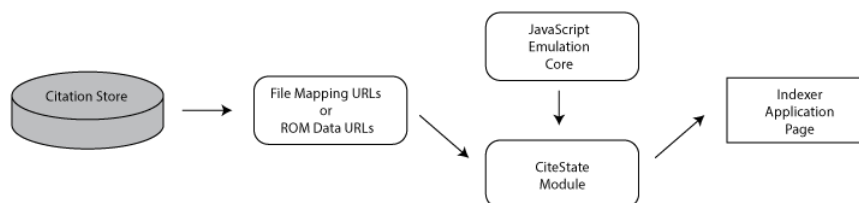


Figure 4.3: Data Flow to GISST Indexer

The major components of the app front-end are a collection of JavaScript modules that support emulation and video recording of games in browser. The underlying technology supporting these modules is known as Emscripten. As mentioned above, Emscripten is a C-to-JavaScript cross-compiler that enables programs written in C to be executed as independent JavaScript modules in-

browser. The supported emulators are modified ports of C programs, with an additional interface layer that allows for citation and recording by the web application. Each emulator supports a basic set of API calls to load and save game state, control input and audio source, and restart if necessary. The wrapped emulators are stored as a set of compiled JavaScript files, that along with a wrapper class, `CiteState`, allows for the loading of a set of game files and a save state into a targeted HTML5 tag.

The app's interface supports four basic views of citation data:

1. A basic listing page for the game and performance citations in the database
2. A full text search page that includes all citation records and game save state descriptions
3. A citation listing page that provides active links to previous save states and, for performances, the ability to create quick gifs animations based on a performance video
4. An indexer page that allows for examination of an emulated game, and the creation of game save states and video recordings

Entries 1-3 are common boilerplate tables of information and do not warrant further discussion (with the exception of the “active links” in item 3). The indexer, however, is the generative heart of the endeavor and we will briefly describe its basic functionality and architecture.

As mentioned above, `CiteState.js` is a small JavaScript module that manages the emulation interface as well as video and audio recording of the target HTML canvas in which the emulation runs. `CiteState.js` can theoretically manage an unlimited number of emulations per page, but there is a hard limit of 6 concurrent

emulations due to the constraints of current web browsers⁷⁰. If a seventh emulation is added to a page, the oldest (first loaded) one will be automatically deactivated. The CiteState object manages only the interface and data flow into and out of the emulation. All major data transfer functions, save and load state, screen capture, and recording provide JavaScript callback functions that are triggered on the completion of each process. At that point, the data management is handed over to the indexer code that manages communication and transfer of state and recording data to the backend Python server. The next four figures describe the communication processes for the interaction between the Indexer, its CiteState instance, and the storage server.

Of note in the above diagrams is the concurrent management of save (Figure 4.6), load (Figure 4.4 and 4.5) and video recording (Figure 4.7) functions. In the case of save and load state, for the DOSBox emulator specifically, each save state also included the full file structure of the system at the time of the save. As a result of the architecture of Emscripten, which relies on a fixed, pre-allocated block of memory for each running application, uncompressed states from DOSBox are around 100MB. This well exceeds the bandwidth capabilities of common web usage. As a result, each file and state is compressed before transfer to the server. The concurrent operation of each major function also allows for video recording and state save and load to occur simultaneously or for multiple emulator instances

⁷⁰The specific issue is that each emulation runs as a separate application within the web page. Emscripten is designed to generally run a single application per page. As such, it provides no support for management of multiple applications, and in some cases this assumption of singular page execution bumps up against browser limitations. In our case, each new emulation creates its own AudioContext to produce sound through the browser. An AudioContext is a JavaScript object responsible for audio playback and routing. Currently, Google's Chrome browser (the most advanced in this specific regard) only supports 6 concurrent AudioContexts per page. This makes sense given that most web pages will only need one source of audio. In our case, it would be possible to circumvent this limitation by writing an AudioContext manager. The main impediment is that each Emscripten-based emulator conversion is unique and thus we would need to rewrite significant portions of each emulator's audio management code to allow for an alignment to a global AudioContext.

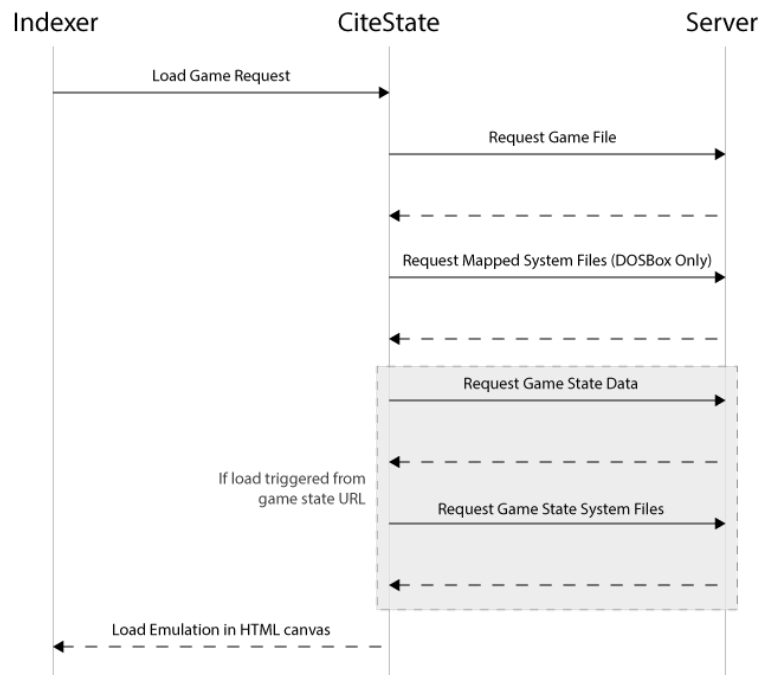


Figure 4.4: Network Diagram for Loading a Game into the Indexer. Note the optional save state or save data load if provided to in the URL.

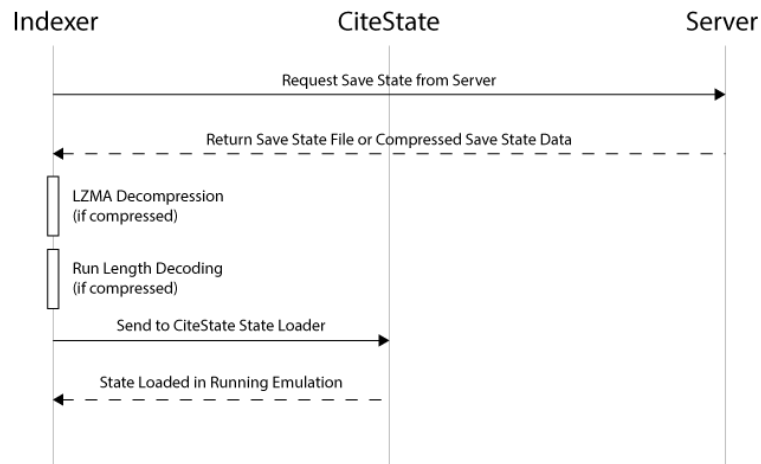


Figure 4.5: Network Diagram for Loading a Save State. Decompression is required if the save data is not an emulator produced save file. See next figure for more. All compression functions run in a separate worker process to avoid locking the browser during play.

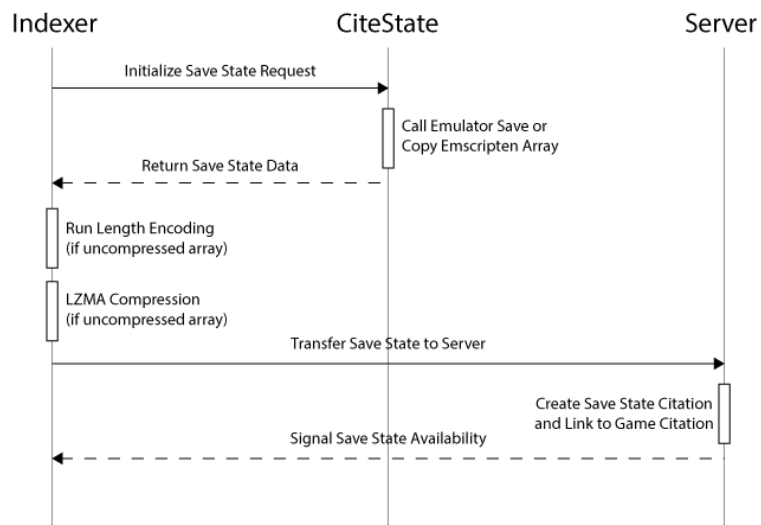


Figure 4.6: Network Diagram for Saving a Save State.

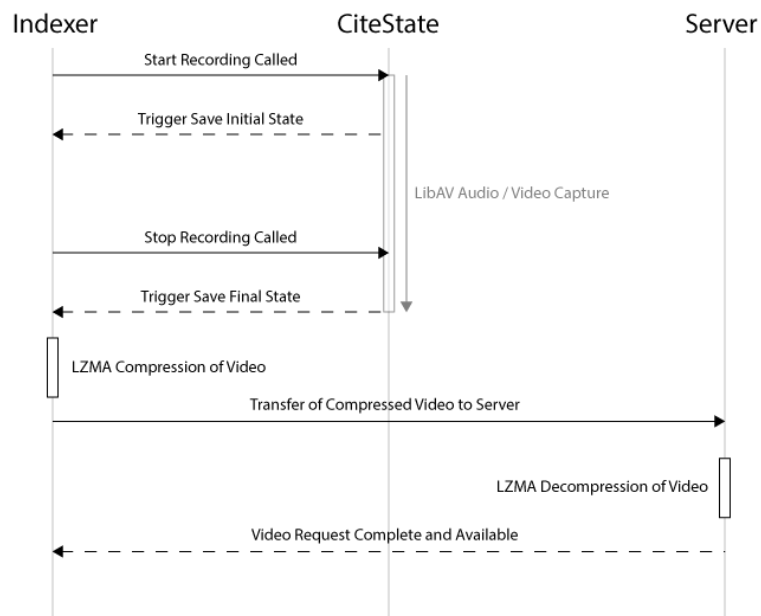


Figure 4.7: Network Diagram for Video Recording. The recording process runs on a separate worker process in the diagram. Video compression is also handled in a separate process.

on a single page. The video recording process is actually also an Emscripten-based cross-compilation of “libav” a simple C library designed for video recording. This means that when loaded in the indexer each emulation is supported by multiple concurrent Emscripten-based C applications. We believe that the potential for the operation of multiple, cross-communicating applications inside the browser will provide for a significant advance in the capabilities of client-side JavaScript applications.

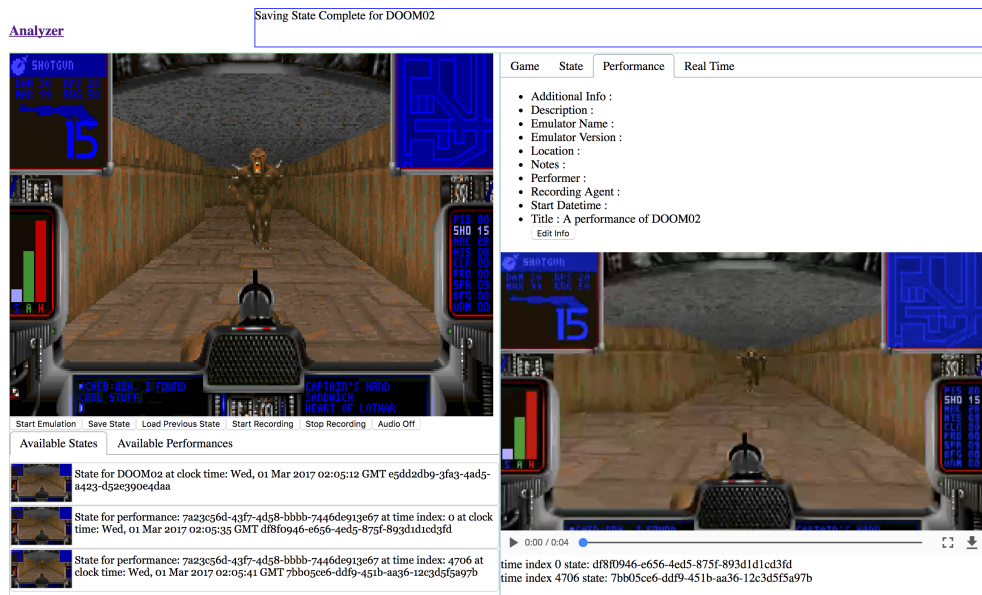


Figure 4.8: Indexer User Interface. The emulation window is on the left, with a recording of that window displayed on the right.

Proceeding to the app’s user interface, as displayed in Figure 4.8, we will briefly describe each button and component. The buttons under the main emulation window provide for most of the data recording features. The user can load the emulation, save a state, load the most recently saved state, control video recording, and mute audio. When a state is saved, it is logged in the “Available States” tab along with a screen shot and generated descriptive information. By clicking on any available state, the emulator will immediately load that state into the main

window. After a state is selected, the “State” tab in the left side bar lets the user change its descriptive metadata. All changes logged in the side bar are propagated to the server and will show up in search queries. Video recording functions in a similar way. Any time a start-stop sequence is completed, the beginning and ending state of the emulation will be saved along with the video. Each recorded performance appears in the “Available Performances” tab. Clicking on a performance updates the sidebar’s “Performance” tab, allowing for the review of a recorded video and editing of its generated metadata.

Any performance or state saved in the analysis tool will appear in the main citation listing page, and on individual pages for each respective game and performance. The state links on each individual page as “active links” in that they will load up the indexer page with the correct state preloaded into the emulator. This makes each active link a link into a running emulation as a specific point. We take up the pedagogical and analytical implications of this in our expert evaluation review in the next major section of this chapter.

The indexer, in creating and storing the saved states and performance recordings, provides the source material for future links created by the `CiteState.js` module. The `CiteState.js` interface allows for a simple description of a target page element and an id from the citation database. `CiteState.js` then automatically handles the loading of a game, performance or state, and places them into an HTML element that can be aligned by the user through CSS styling or other means of element positioning (this is steps 6-8 in Figure 4.1 above). This completes the chain from source ingestion through shared linking of game emulation in browser.

4.7.5 Future Work

The citation tool opens up numerous opportunities for the dissemination and standardization of game historical sources. For bibliographic record purposes, all the information in a specific citation store could be exported into forms compatible with common citation database formats, like BibTeX, or linked with citation systems, like Zotero.⁷¹ These citations could also automatically include information about a compatible emulator and the specific file specific data required by more technical scholars, like the platform and software studies folks mentioned above.

In the realm of reductions, since the emulation is a full computing system running in a web page, its memory and operations are totally available to introspection via other concurrent JavaScript processes. We are already working on including memory manipulation functionality in the `CiteState.js` interface, which would provide dynamic visualization of a complete program to occur coincidentally in the browser surface.

Lastly, since each of the citation types, games, performances, and game states, also require a linkage between the citation and some form of born-digital data, new forms of storage and retrieval will be necessary. There is some work on storing and loading emulated systems or sharing the results of emulation produced on cloud-based servers, but still no general solutions for the storage and retrieval of executable software, nor support for citation as envisaged in the functionality of the tool above.

The next section, an expert evaluation of GISST's citation component by practicing game studies scholars and library professionals, also presents some significant ideas for future work.

⁷¹[14] <http://www.zotero.org>

4.8 Evaluation

As described, the citation component of GISST is an argument for more rigorous citation of computer games, and for the augmentation of their expression in game studies discourse. We believe the tool can ease the citation burden for game scholars and allow them to create new types of arguments and expressions about games. To corroborate this belief we conducted a speculative expert evaluation of the tool, inviting comment from a group of professionals engaged with game study and preservation. The goal of the study was to ascertain if the intentions of the tool were clear, if our thoughts above aligned with those of practicing scholars, and to invite constructive commentary. This section outlines the evaluation and its responses, and how those responses aligned with our goals and ignited ideas for future work and collaboration.

We sent the evaluation to select group of practitioners consisting of game designers, game studies scholars, and librarians. These groups align with those we hope will benefit most from the citation tool and, in general, the work of this thesis. The evaluation consisted of a set of 11 questions to be answered based on a 5-minute introduction to the CLI and web app components of the tool. All those chosen were already aware of GISST, and the video served as a reminder of functionality that had at some point been demonstrated to them in person. Responses were collected from seven people through an online form. Respondents included: Henry Lowood, curator the History of Science and Technology and Film and Media Collections at Stanford University; Chaim Gingold, a game designer and historical researcher; Nathan Altice, a professor and game historian at the University of California, Santa Cruz; James Newman, a professor and game historian at the University of BathSpa; Glynn Edwards, head of technical services in Special Collections Stanford; Shane Denson, an professor of Art History at Stan-

ford; Douglas Wilson, a game designer and professor at RMIT University; and a professor who wished to remain anonymous. The remainder of this section will describe the responses to the tool, followed by recommended improvements.

4.8.1 Discussion

Overall, the responses were overwhelming positive, with one game studies scholar stating that the availability of the tools “could be huge” for the field. The respondents hailed from an overlapping set of backgrounds, but the responses aligned along two basic paths. The first was how the tool could affect game studies and game historical practices in citation, and what the tool could contribute, through state citation and retrieval, to students and game studies scholars. The second turned toward more of the potential for preservation that the tool presents in its management of game citation and game states.

The potential influences noted for game studies practice included (1) the formalization of game citation practices, (2) the removal of obstacles to game access, (3) the automation of game history tasks currently taken on in an ad-hoc manner, and (4) the presentation of deeper, and more comprehensive historical analysis. Multiple respondents noted the tools implicit call for a more “formal and robust” citation practice for games, with Henry Lowood stating that the tool provided a first take on a “citation framework where there was none.” The tool functioned as a way to call attention to the potential of better citation practices. James Newman explained, “a contribution of the tool will surely be to heighten discussion of citation and [its] limits and variations in current practices.” This therefore aligns with our arguments for more consistent citation practice in the games studies section above.

Altice, Newman and Shane Denson highlighted the tools’ ability to provide an

easier route to specific game locations and gameplay sequences. Altice and Denson specifically work on comparative analysis of game versions and emulators, so the potential for the tool to make parts of games more reachable was appreciated.⁷² This concern for access to game history also extended to the other scholars, who all remarked on the ability of the tool to make classroom lectures more engaging, and according to Altice provide for “in-class play that isn’t contingent upon equipment or playing skill.” He felt this allowed for a “wider breadth of examples” since lengthy equipment set up or hours spent trying to get to a particular spot in a game could be removed from the equation. Altice also felt that this approach could make exploring games like “flipping to a relevant page in a book, which could make citation more prolific and illustrative.” He also believed that playing a games citation would have a more powerful rhetorical effect than other forms of reduction, nicely pairing with our claims about rhetoric and game feel above. Chaim Gingold also felt that the tools provided a significant new way to share content with students. He imagined providing state citations to students in the future as one might today assign videos on YouTube or Twitch.

The tools as an automated solution for citation also struck a chord with the researchers who already use an assortment of ad-hoc solutions for emulation, and gameplay recordings and analysis. James Newman already organizes a rather complex chain of tools for gameplay capture and analysis, and the tools provided a way to alleviate some of the burden in getting multiple programs and systems to work together. Gingold also noted that the tools could allow students to engage in the types of historical analysis that are only available to interdisciplinary scholars with programming and humanities backgrounds. Students could incorporate “their interaction into their scholarship (like us!),” and provide them a starting

⁷²For Altice, see [25], for Denson’s work, see [65] Denson, Shane. “Digital Seriality.” Accessed March 5, 2017. http://shanedenson.com/stuff/visualizing_digital_seriality/digital-seriality.html. (Accessed 5 Apr 2017.)

point for more detailed analysis of design and game play interactions.

The last major response was of the ability for the tools to provide a new level of analysis for game studies and game history. Newman noted that having access to a GISST-like system would remove the need to engage in extended descriptions of game play or game scenes. If a citation is also a playable instance, he could rely on the reader to pay what he was talking about, and then focus his time on deeper analysis instead of front loading arduous amounts of descriptive text to set up his points.

In principle, being able to refer to a persistent recording or savestate would give me increased confidence in writing more detailed analyses of sequences of gameplay and would, hopefully, alleviate some of the need for description in favour of close commentary and annotation.

Denson concurred, stating that “arguments can [now] be illustrated directly (through video, for example) and even mounted through hands-on engagement (gameplay) rather than merely discursive description.”

The second major thread in the responses highlighted the tools’ implicit effects on game preservation and the organization of game history. Some viewed the tools as an argument for more robust digital repositories able to handle and retrieve executable content. One noted that the tools displayed the power of centralizing documentation about games and how the coordination of tools could foster new expressions through the linkages of different technologies. In this case, the concordance of emulation, documented citation, and gameplay videos invited a discussion of the need for coherent underlying infrastructure to support preservation of those outputs. Lowood agreed, saying that the tools put “issues around documentation, archiving and gameplay preservation front and center.” Glynn Edwards also focused on the needs to create consistent metadata schemes for the emulated save states and companion documentation. There was a general agreement among the preservation professionals that the tools’ existence, in and of

themselves, functioned as an argument for better preservation practice, and they were excited to begin working towards solutions.

Another small preservation note that aligns with some coming discussion in the next chapter, was that of the tools' ability to allow for quick validation of game files, and game data integrity. By ingesting an executable into the system, one can easily check if it is compatible with a specific emulator, and if it is actually the file it claims to be. Lowood also noted that the tool could push repositories to negotiate better IP rights access to executable software, or, at the very least, further reveal the need for that work to be figured out.

4.8.2 Improvements and Future Work

Given the overwhelmingly positive response to the work, most of the critical discussion of the tools pointed solely to means of immediate improvement and new features. In the main, many respondents wanted the tools to continue development of better UI and user accessibility features. Right now, most of the ingestion apparatus occurs on the command line, and Altice correctly felt that “freeing the tool from dependence on the CLI” would be necessary since “this would be a non-starter for many (most?) scholars without a technical background.” Many also pointed out that while the technology had obvious potential, as noted in the last section, it ached for a set of coherent examples and illustrative case studies. The tools represented more of a “starting point” for new discussions in game studies and game preservation but were not really yet a solution (though with work they could be). This appears to show that the tools are a ripe ground for future work, as just making them more accessible and easier to use excited many of the respondents. One even requested a basic tutorial for the current alpha prototypes since they felt they required hands-on access to fully appreciate the prototypes

potential use cases.

Another general request was for the inclusion of more emulators than the four currently available to allow for both comparison of the emulators themselves, and to help with further preservation questions in the description of the configured environments needed to support game data. Many noted that the tools, in both their analytic and preservation potential, pointed to uses outside of games, and would be a boon for software studies and general software preservation. This was edifying, since another thread of our overall thesis is that most advances for game software history are also significantly applicable to broader classes of software.

Future work dovetailed with the requests for more system support and usability. Many wanted to see what a larger, shared repository of game state citation could afford, either in a classroom setting or for executable collections in libraries and archives. There was also a call for more explanation of the citation description formats, and perhaps even integration with current scholarly citation toolsets like Zotero. Multiple respondents also mentioned the potential for annotation tools to add voiceovers to videos, and record and remark on game play input traces. Gingold specifically was excited about the ability to introspect on the running emulations, and visualize their system dynamics and memory states, similar to our own thoughts on future work above.

In closing, the responses to the tool essentially agree with the arguments presented in the preceding chapter. Respondents believed that embedded emulations represent a new form of expression for game history, and that tools themselves function as an argument for further work on technical system visualization, documentation management, game citation, and preservation. We believe that this suitably validates the tools, the methodologies they support, and theories behind them in ways that not only legitimate them as contributions to numerous fields,

but as a starting point for more significant future work, and perhaps even future disciplines.

4.9 Conclusion

At the beginning this chapter we set out to address citation practices as a significant lacunae in scholarly practices around games. This proceeded through a more in-depth discussion of the purpose and functionality of citation in both scholarly discourses in general, and towards the ways in which games are a new and special case. As a result, the system we described to manage and create playable manifest citations required not just novel engineering effort but a consideration of the types of objects within and around games that could be leveraged in arguments. The efforts in creating the citation components of the Game and Interactive Software Scholarship Toolkit (GISST) actually preceded GISST’s conceptualization.⁷³ In attempting to find ways to cite games, we incidentally had to create the technical means for those citations, and figure out how those citations could be made available and useful for argumentation. This then contributed new means of historical expression, in that we realized that the results of the citation system creation represented a new general class of activity in the design of systems to support game scholarship. This is why the citation system is now a component of GISST, because we believe that the citation work is only a preface to a whole range of possible tools for game history, game studies, and software studies works.

The citation work elaborated on a means for the manifest citation of new objects into scholarly discourse, but it also supported those objects creation and storage. As a result, this opened up those objects (performance videos, executable

⁷³Note that this is conceptualization of GISST as a system and not “conceptualization” in the ontological sense as discussed in Chapter 3, “Description.”

game play, and indexed game states) to further analysis. In addition to functioning as a form of reduction in supporting textual discourse, the objects are also now organized and manipulable by any potential future extensions of GISST's toolset. As mentioned in Future Work above, this could mean input analysis and replay of game play — a means to further look at instances of “superplay” like speed running or glitch-hunting — and introspection on the game's system state and run-time memory. In tracking the needs for a citable base of games, performance, and states, we have opened up a whole new set of resources for exploration and expression through scholarship. Yet another example of how the stabilization of historical resources can lend itself to new uses and articulations.

Chapter 5

Discovery

5.1 Intro

The previous chapters remarked on some of the key phases of historical knowledge management: appraisal, description, and retrieval through citation. They outlined how new methodologies and tools informed by game studies scholarship and computer science can help improve game and software history. This chapter argues that in addition to the stable description and retrieval of games for historical examination, we also need to think about the ways that scholars find games to write about, analyze, and discuss. In the realm of library science, the issue of locating items in collections is known as “discovery”. The growing problem with discovery is that the proliferation of things that can be discovered makes it more difficult to locate works that are potentially relevant to a scholar’s research. Additionally, for areas that have historically been under researched and therefore inadequately described, like games, the discovery problem is compounded¹. Not only are there many new games being created with each passing moment, but

¹Please refer to the previous three chapters for evidence of the contemporary lack of description and classification methodologies for games history.

also historical games that never received attention in the first place are increasing lost in the deluge. In addition, the structures that have historically organized games, mostly commercially ordained game genres, are themselves fraught with inconsistencies and threaten to retroactively re-organize the history of games by privileging certain types of classification over others. Below, we outline game discovery and its issues in more detail, and then present a solution for computer games in the form of tools and visualizations that leverage machine learning natural language processing (NLP) models to display and search corpuses describing historical games. The tools, GameNet and GameSage aim to operationalize the related characteristics of games to allow for new forms of discovery aimed at game designers and game historians. The visualizations, GameGlobs, GameSpace, and GameTree, in addition to being helpful for discovery also contribute to our larger argument for the application of computer science methods to the revelation of software history. The chapter concludes with evaluative validation of the tools potential by both game studies scholars and game design students and then looks to applications that further support game historical scholarship.

Some of the work in this section is based on a series of six publications, and we will direct the reader to them for further clarification of topics not directly related to game discovery. Another note, in many cases we found ourselves writing “tools and visualizations” when making points about this chapter’s technical contributions. We shorten this to “tools” only, and will only refer to the visualizations if there is a topic for which they are particularly apt, or that does not include GameNet or GameSage.

5.2 Game Discovery

The problem of game discovery is two-fold: (1) there are too many new games being created to have a reasonable chance of keeping up with, let alone finding, titles that might be research relevant; and (2) that most historical titles are under researched and under described, and therefore hidden from historical inquiries.

The first problem is substantiated almost commonsensically by the massive production of games over the last decade. Since its launch in 2009, the iOS app store has accumulated over 769,706 games, which is 25% of all available applications.² To put that in perspective, there are more games currently available on iOS than have been created for all other non-mobile platforms for the entire history of game production.³ Furthermore, even within the realm of non-mobile games, the proliferation of tools for game production, easier means of digital distribution, and the rise of games as a socio-cultural force have brought about a computational mediatic renaissance over the last decade. Services like Steam boast more games for the current version of Microsoft Windows than games created for all other Windows versions for the last 25 years. Wherever you look, the contemporary levels of creative production dwarf previous, non-networked epochs of human production.⁴ This makes current game discovery a significant challenge, and as Simon Carless notes this “excessive” choice and availability of games risks marginalizing potentially significant works that cannot mount the marketing or outreach needed to compete in online marketplaces.⁵ The extensive amount of new

²Based on April 2017 stats from <http://www.pocketgamer.biz/metrics/app-store/> (Accessed 5 Apr 2017.)

³For more on this issue see: [101] Kaltman, Eric. “Current Game Preservation in Not Enough.” *How They Got Game*, June 2016. <http://web.stanford.edu/group/htgg/cgi-bin/drupal/>. (Accessed 5 April 2017.)

⁴[218] Weinberger, David. *Too Big to Know: Rethinking Knowledge Now That the Facts Aren't the Facts, Experts Are Everywhere, and the Smartest Person in the Room Is the Room*. New York: Basic Books, 2011.

⁵[49] Carless, Simon. “Why Game Discovery Is Vital - Introducing Games We Care

titles poses a problem for the future history of the medium. With so many titles available, and with so little attention being paid to their coherent and consistent description (most commercial distributors are not particularly concerned with future-proof metadata), we need to find new ways to locate and reveal games. Ones that do not wholly depend on dedicated, consistent description efforts. The sheer scale of production will always incur a significant lag in cultural institutions efforts to keep up. Our model and tools below work to address this issue by leveraging crowd-sourced content for discovery tasks.

The previous work of this dissertation backs up the second problem of discovering historical games. Since work on game preservation at academic institutions only began in earnest within the last decade, enthusiast communities have historically picked up most of the slack. While the work of these communities is admirable and essential, they do not follow standards that are particularly consistent with each other, or with the practices of the library and academic communities. As shown in the description chapter, there is a significant bureaucratic burden imposed in attempting to change and progress national and international library standards. As a result, while our work in this thesis and on similar classification and metadata projects is helping to reorient the ship, the potential for discovery of games in historical collections is still far from any kind of parity with other media forms. Also, due to the lack of significant games collections, and in the case of physical collections, their highly varied form, even browsing games history is a difficult task. As we will take up in the next section, the serendipity of discovery is a significant part of the historical scholarly experience both in archives and in the wandering of the library stacks. With the advances in computer mediated — and therefore highly regimented and logical structured — knowledge retrieval,

About.,” June 6, 2014. http://www.gamasutra.com/blogs/SimonCarless/20140606/218988/Why_game_discovery_is_vital__introducing_Games_We_Care_About.php. (Accessed 5 Apr 2017.)

online tools and finding aids are not organized for the discovery of new information through serendipity.⁶ Since we aim to support game historical study, it is of importance that our tools for helping with discovery do not further exacerbate the issues inherent to other systems.

Of course, other media also have discoverability problems, but in games the problem is compounded by the high dimensionality of their ontologies as media artifacts. This is a reason that discovery systems organized around other forms, and described with metadata specific to those form's ontologies, present significant challenges for games. Computer games incorporate text, sound, moving images, and non-trivial interactive feedback, each of which may function as a viable search path for one type of discovery system and a potential obstruction for another. The only way to alleviate this discoverability problem is to develop dedicated tools for game discovery that index along a composition of all of these dimensions. This allows historical research to operate over them and increases the potential for locating relevant research items along with serendipitous discovery.

5.2.1 Forms of Discovery and Their Limitations

Typically, the means of discovery in collections boils down to two modes, direct and indirect access (cite). Direct access refers to when a researcher knows the exact topic or item they are looking for and simply consults a direct method of retrieval. This could be naming an explicit property of an object, like its title, or through reference to an alphabetic listing of subjects⁷. Indirect access is when discovery is based on the syndetic connection between one subject

⁶[170] Race, Tammera M., M. P. Popp, and D. Dallis. "Resource Discovery Tools: Supporting Serendipity." *Planning and Implementing Resource Discovery Tools in Academic Libraries*, 2012, 139-152.

⁷Direct subject lists need not be alphabetically organized, but must refer explicitly to the subject at hand without an indirect, syndetic (linked) connection to another topic.

and another, as in classed hierarchies like Library of Congress Subject Headings (LCSH). By classed, we mean just a specific collection of subjects that share a similar, top-level class, like “Transportation—Railroads” and “Transportation—Auto Mobiles”. This method of discovery is indirect because one must look to top level categories and then work down the tree to specific sub-classes that contain items of interest. Indirect methods became necessary, historically, because the rapid development of new subjects and domains of knowledge — and their dependent publications — quickly out paced the ability to simply locate a specific topic on a list. Since each item named might have multiple potential names or possibilities for description, indirect methods allow for the linkage between topics that manifest either through the creation of relations and equivalences (like “see also” connections) or the aforementioned hierarchical grouping of concepts.

A significant problem with the creation of subject listings is the potential to embed both (1) societal and personal bias and (2) problematic epistemology into any categorization scheme (cite, cite). In the case of the LCSH, the development of subject listings for its first 70 years or so emphatically assumed a white, wealthy, hetero, male researcher and organized headings accordingly.⁸ For some topics this was not particularly problematic, but for those attached to racist, genderist and sexist biases indicative of their relative cultural moment. A particularly salient example being a “see also” connection between homosexuality and mental disorders.⁹

The epistemological issue, most directly confronted by Eleanor Rosch through prototype theory, and George Lakoff’s complementary categorization work, is that knowledge organization in the Western world was based, until the latter half of the

⁸[106] Knowlton, Steven A. “Three Decades since Prejudices and Antipathies: A Study of Changes in the Library of Congress Subject Headings.” *Cataloging & Classification Quarterly* 40, no. 2 (2005): 123-145.

⁹[106] pg. 133

20th century, on antiquated, non-empirical, and abstract philosophies ascribed to tradition as opposed to lived reality¹⁰.

Rosch's prototype theory essentially argues that any categorization scheme is a reflection of an idealistic, biased "prototype" that is not the result of external reality but the experience and society of the person constructing it. A common example is to have someone think of the "best" example of a bird. Most people will think of a robin or pigeon (or more flowery, feathery forms like eagles) but not a penguin or turkey. The prototypical bird is effectively arbitrary within the class of birds, and that class of birds is a further construct of the contemporary zoology and biology, and not a natural or emergent class that exists apart from humanity's conception of it. Rosch's two main moves are then, as summarized in Lakoff:

First, if categories are defined only by properties that all members share, then no members should be better examples of the category than any other members.

Second, if categories are defined only by properties inherent in the members, then categories should be independent of the peculiarities of any beings doing the categorizing; that is, they should not involve such matters as human neurophysiology, human body movement, and specific human capacities to perceive, to form mental images, to learn and remember, to organize the things learned, and to communicate efficiently.¹¹

So the categorization scheme used is mainly the result of the experiential and socially inherited conditions of the categorizer and not a result of a "natural"

¹⁰We use "reality" here lightly, and not in a realist philosophy sense since that is the exact epistemology that Rosch, Lakoff, and others (Feyerabend, Kuhn, etc.) were reacting against. This more to contrast a traditional view of what is "real" with more rigorously investigated knowledge and language claims. For more see, [75] Feyerabend, Paul, and Ian Hacking. *Against Method*. Fourth Edition edition. London: New York: Verso, 2010. [112] Kuhn, Thomas S. *The Structure of Scientific Revolutions*. 3rd edition. Chicago, IL: University of Chicago Press, 1996. [114] Lakoff, George. *Women, Fire, and Dangerous Things: What Categories Reveal about the Mind*. Chicago: University of Chicago Press, 1987. [181] Rosch, Eleanor, Carolyn B Mervis, Wayne D Gray, David M Johnson, and Penny Boyes-Braem. "Basic Objects in Natural Categories." *Cognitive Psychology* 8, no. 3 (July 1976): 382-439.

¹¹[114] pg. 7

reality. Lakoff extends this idea to cover all of our conceptions in his analysis of metaphor, and connects nicely back to our concern for game discovery in the preceding section. Namely, that “our concepts of objects, like our concepts of events and activities, are characterizable as multidimensional gestalts whose dimensions emerge naturally from our experience in the world.”¹² Note that “emerge naturally” is tied to embodied, situated, and social constructed conceptions and not external categorizations detached from human consciousness. For games in particular, their multimedial mode of expression is perhaps best captured in a more holistic and embodied way, and not through the rough genre forms that currently dominate most non-critical ontological discussion¹³. The essential take away is that there are no actual objective descriptions for concepts and phenomena, but a socially contingent and embodied subjectivity that is adhered to at various levels of social organization.

In library science, and specifically in the area of subject analysis devoted to the classification of subject headings and controlled vocabularies, these concerns have also arisen in line with their acceptance by cognitive science. In guides to subject analysis, a distinction is drawn between the older cognitive rationalist model of description, based on philosophies of logic that assume the existence of external and eternal axiomatic truths, and a social constructionist model based on the influence of the different “discourses” that shape an individual’s interaction with society.¹⁴ The terminology here is getting a bit confused, however, since the cognitive model described in subject analysis is not the cognitive science model

¹²[115] Lakoff, George, and Mark Johnson. *Metaphors We Live by*. Chicago: University of Chicago Press, 1980. pg. 121-122.

¹³Here we are referring to the basic game genres, first person shooter, real-time strategy, and the like, which are combinations of marketing terms, and multiple confused ontological levels (granularities).

¹⁴[164] Olson, Hope A., John J. Boll, and Rao Aluri. *Subject Analysis in Online Catalogs*. 2nd ed. Englewood, Colo: Libraries Unlimited, 2001. pg. 265-269

discussed by Rosch and Lakoff. Regardless, the basic notion that positioning and locating items via subject headings or other classification schemes invites bias is essentially the same. Libraries are well on board the post-structuralist boat.

Another brief note is that subject analysis extends this division between a rationalist organization of knowledge, and a cognitive, experientialist one to the interpretation of potential users of a system. That is, as far as discovery of materials go, most systems were — and by means of institutional inertia still are — oriented toward a specific societal discourses (biases). This limits the potential for discovery systems because they effectively hamstring users into asking (through system input) and answering (through returned information structured in that discourse) only the most conventional questions. These “conventions” are enforced by the discourses that constructed them, and in many cases cannot be easily altered to return knowledge in a different discursive form or from a different knowledge paradigm. Our tool work is very aware of this issue, as will be addressed in the discussion of the Wikipedia and GameFAQs corpuses below, as well as in the definition of game “relatedness” derived from our NLP model.

Returning to the organization of information about computer games, it is not hard to see that certain genres, discourses, and other means of enforcing specific, socially constructed models of games might limit the access to their history, or in many ways contort that history to fit the expectations and nostalgic recollections of dominant gaming social groups.

Games and other forms of expressive entertainment are dependent on market forces for their creation and sustenance. In this case, marketing certain classes of games that are familiar to specific audiences that have historically purchased them ends up reinforcing a bandwagon effect.¹⁵ The best-marketed — and most

¹⁵[24] Adler, Moshe. “Stardom and Talent.” *The American Economic Review* 75, no. 1 (1985): 208-212.

identifiably genred — titles reach the largest audiences, which in turn persuade the industry that only similar titles can have similar effects, and the ouroboros of 13 Call of Duty titles in 13 years begins feeding.¹⁶ The tools for discovery must work against these conventions if the history of games is to be recoverable and discoverable.

5.3 A Goal for Discovery

While all computer game stakeholders are susceptible to the discoverability problems outlined above, both in the difficulty of their recovery of games, and their embedded biases, we are most concerned with how the discoverability affects scholars of game history and the designers that operate on that history to produce new works. For game historians, access to past games is imperative to the progress of the field. Discovering titles that fell through the cracks, that might have inspired a more prevalent design pattern, or that progeniated some genre, allows us to organize alternative and deeper histories of the field, and improve on the contemporary, journalistic state of game history. For designers, improved discovery allows them to seek out prior work that could inform their process, or incur the development of new designs. Lacking tools that meet this need, especially those that could facilitate the discovery of novel or notable titles, many designers function without access to, and thus without a rich awareness of, the medium in which they operate. This is arguably one aspect in which computer games show their immaturity relative to more established media.¹⁷ In this chapter, we limit

¹⁶https://en.wikipedia.org/w/index.php?title=Call_of_Duty&oldid=772473470 Lists 13 games in the “main” Call of Duty franchise, and 12 others in ancillary roles.

¹⁷This blindness — through lack of awareness driven by a lack of tools and paradigms adapted to historical investigation — is also a significant problem for the history of software in general, and most of the discussion of this section could be ascribed to other classes of software, or computational art forms.

the scope of our discussion about the tools to the scholarly and design-centric use cases. The remainder of this section will clarify our goals for the discovery tools, in light of some of the current literature on general discovery systems.

First, game discovery is not game recommendation.¹⁸ Recommender systems (sometimes just called recommenders) are often part of larger commercial applications such as online retailers,¹⁹ and a prototypical task of these systems is product recommendation.²⁰ In contrast, discovery tools (also called exploratory search systems)²¹ promote user learning above user purchasing. While in the recommendation task there is often an explicit notion of the correctness or accuracy of a recommendation,²² the parallel concern in discovery is the usefulness of an item. (Of course, this distinction has made evaluating discovery tools a much trickier issue.)²³ Finally, while recommenders are susceptible to a popularity bias by which a small proportion of the item space is recommended exponentially more often²⁴ — a phenomenon that may actually aggravate the bandwagon effect we described in the last section — discovery tools, as mentioned above, also aim to provide

¹⁸The underlying model for our discovery tools has also been evaluated as a potential recommender system, see [187] Ryan, James Owen, Eric Kaltman, Timothy Hong, Michael Mateas, and Noah Wardrip-Fruin. “People Tend to Like Related Games.” Proceedings of the 10th International Conference on the Foundations of Digital Games, 2015. We leave out this discussion because we are mainly focused on the operationalization and benefit of discovery for scholars and practitioners and not player / consumers.

¹⁹[116] Lam, Shyong K., and John Riedl. “Shilling Recommender Systems for Fun and Profit.” In Proceedings of the 13th International Conference on World Wide Web, 393-402. ACM, 2004.

²⁰[166] Park, Deuk Hee, Hyea Kyeong Kim, Il Young Choi, and Jae Kyeong Kim. “A Literature Review and Classification of Recommender Systems Research.” *Expert Systems with Applications* 39, no. 11 (2012): 10059-10072.

²¹[135] Marchionini, Gary. “Exploratory Search: From Finding to Understanding.” *Communications of the ACM* 49, no. 4 (2006): 41-46.

²²[176] Ricci, Francesco, Lior Rokach, and Bracha Shapira. *Introduction to Recommender Systems Handbook*. Springer, 2011.

²³[223] White, Ryen W., Gary Marchionini, and Gheorghe Muresan. Evaluating Exploratory Search Systems: Introduction to Special Topic Issue of Information Processing and Management. Pergamon, 2008.

²⁴[98] Jannach, Dietmar, Lukas Lerche, Fatih Gedikli, and Geoffray Bonnin. “What Recommenders Recommend—an Analysis of Accuracy, Popularity, and Sales Diversity Effects.” In International Conference on User Modeling, Adaptation, and Personalization, 25-37. Springer, 2013.

diverse, serendipitous offerings. Neither should game discovery be considered as merely game information retrieval, primarily for the reason that a game-discovery tool will index games, not information about them. Moreover, in game discovery, getting back results is not a resolution — the offerings provided by a discovery tool are meant to be analyzed and explored.²⁵ Game discovery should always be user-centric, while in information retrieval ancillary concerns (namely algorithmic nuances) tend to take center stage, a truth that bears out in the offline, batch-style evaluation methods that have prevailed in the field for decades.²⁶

So, let us distill what distinguishes discovery tools from related applications into a succinct specification for what good game-discovery tools should do: it should index games along all dimensions of their high-dimensional ontologies, afford queries that may operate over the same dimensions, and present diverse, serendipitous, explorable offerings.

5.4 A Model for Discovery

The latent semantic analysis model explained in this section sufficiently addresses the above conditions for a successful discovery system foundation. It also contributes to both the newly emerging fields of game discovery and the application of natural language processing to game studies. Our model is actually a set of NLP methods applied to various corpuses of descriptive text about computer games. All of our discovery outputs described in the next section are based on the models described in this one. Below we discuss some related work in the application of NLP to the study of games and their histories, and then proceed to a

²⁵[222] White, Ryan W., and Resa A. Roth. “Exploratory Search: Beyond the Query-Response Paradigm.” *Synthesis Lectures on Information Concepts, Retrieval, and Services* 1, no. 1 (2009): 1-98.

²⁶[223]

detailed discussion of the methods and outputs of our model. This section is intended to highlight the discovery work’s contribution to NLP for game studies, the next sections outlining the tools and visualizations, will account for contributions to game discovery.

5.4.1 Related Work in NLP for Games

There is a growing body of work in which NLP techniques are employed in game-studies research, centered in large part around the efforts of José Zagal, Noriko Tomuro, and their (former) colleagues at DePaul University. More precisely, this work is characterized by its application of techniques from statistical natural language processing, a subfield of NLP in which bottom-up statistical methods are applied to large collections of natural-language text. In this section, we provide a review of this literature, before explaining latent semantic analysis, the statistical NLP technique powering the models. Throughout, we attempt to explain these concepts in such a way that readers who are not NLP practitioners may understand them²⁷.

In the first project to use statistical NLP for game studies, Zagal and Tomuro studied the specific language used to evaluate games across a collection of nearly 400,000 game reviews submitted by users to the website GameSpot.²⁸ First, they gathered 723 unique adjectives that modify the word “gameplay” in some review, and then, treating these adjectives as the core vocabulary with which game appraisal is expressed, proceed to examine them more deeply according to the

²⁷A more complete literature review can be found in the paper on which this section is based: [189] Ryan, James Owen, Eric Kaltman, Michael Mateas, and Noah Wardrip-Fruin. “What We Talk About When We Talk About Games: Bottom-Up Game Studies Using Natural Language Processing.” Proceedings of the 10th International Conference on the Foundations of Digital Games, 2015.

²⁸[233] Zagal, José P., and Noriko Tomuro. “The Aesthetics of Gameplay: A Lexical Approach.” In Proceedings of the 14th International Academic MindTrek Conference: Envisioning Future Media Environments, 9-16. ACM, 2010.

contexts they occur in.²⁹ Specifically, they compiled the 5000 words that most frequently appear either directly before or directly after the adjectives. From here, they represented each adjective by its distribution with respect to these various contexts — in machine learning parlance, this is called feature representation — and proceeded to cluster the adjectives. Clustering is a procedure whereby objects are grouped together such that ones in the same cluster are more similar to one another (with regard to their feature representations) than to objects in other clusters. For this task, the authors used k-means, one of the standard clustering algorithms.³⁰ Here, k is a hyperparameter — a parameter whose value is set by the user prior to runtime (as opposed to a parameter whose value is “learned” by the algorithm itself during runtime) — that specifies how many clusters the algorithm will partition the input set of objects into. After some initial exploration, the authors set k to 30, and then used a subset of these 30 adjective clusters to propose a typology of what they call “primary elements of gameplay aesthetics.”³¹ With this typology, they attest to the existence of a rich language for appraising aesthetic aspects of gameplay, but note that the specific vocabulary used by players appears to be different from that employed by scholars and designers.

In a further journal article, Zagal, Tomuro, and Shepisten argued for the use of NLP in game-studies research through three example studies.³² Here, we only outline the first. In this brief study, the authors applied readability metrics to 1500 professionally written game reviews extracted from GameSpot. A readability metric is a formula used to determine, ostensibly, the level of education needed to

²⁹Note that the use of the term “appraisal” in the descriptions below refers to the assessment of games and not archival “appraisal” of game documentation or objects.

³⁰[130] MacQueen, James, and others. “Some Methods for Classification and Analysis of Multivariate Observations.” In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, 1:281-297. Oakland, CA, USA., 1967.

³¹[233] pg. 12

³²[235] Zagal, José P., Noriko Tomuro, and Andriy Shepitsen. “Natural Language Processing in Game Studies Research: An Overview.” *Simulation & Gaming* 43, no. 3 (2012): 356-373.

understand a text. Typically, these formulae operate on the number and length of the syllables, words, and sentences of a text. Using three common metrics — SMOG,³³ the Coleman-Liau index,³⁴ and the Gunning fog index³⁵ — the authors find that the reviews are written at a secondary-education reading level. From these results, they argued against criticism that game reviews are written poorly and for a young demographic.

In another article, Raison and others extracted fine-grained player appraisals of games (found in amateur reviews) and used these to cluster the games themselves.³⁶ These fine-grained appraisals are in the form of co-clusters derived from Zagal et al.’s listing of 723 adjectives (that modified “gameplay” in a review) and the contexts they occurred in, which we described above. Whereas in standard clustering one set of objects (all of the same type) is partitioned into clusters of similar objects, in co-clustering two sets (having different types of objects) are simultaneously partitioned such that the elements of a cluster in the first set are bonded by being similarly associated with the elements of a particular cluster in the other set. This produces a set of co-clusters, rather than a set of regular clusters. In the study, each of the authors’ 3000 derived co-clusters comprises a cluster of adjectives and a cluster of contexts such that those particular adjectives all tend to occur in those particular contexts and, likewise, those contexts all tend to feature those adjectives. As an example, one of the co-clusters they list has {“great,” “amazing,” “excellent,” . . . } as its adjectival cluster and

³³Simple Measure of Gobbledygook (SMOG) rating. See [141] Mc Laughlin, G. Harry. “SMOG Grading — a New Readability Formula.” *Journal of Reading* 12, no. 8 (1969): 639-646.

³⁴[53] Coleman, Meri, and Ta Lin Liau. “A Computer Readability Formula Designed for Machine Scoring.” *Journal of Applied Psychology* 60, no. 2 (1975): 283.

³⁵[28] Armstrong, J. Scott. “Unintelligible Management Research and Academic Prestige.” *Interfaces* 10, no. 2 (1980): 80-86.

³⁶[171] Raison, Kevin, Noriko Tomuro, Steve Lytinen, and Jose P. Zagal. “Extraction of User Opinions by Adjective-Context Co-Clustering for Game Review Texts.” In *Advances in Natural Language Processing*, 289-299. Springer, 2012.

{“graphics,” “look,” “sound,” . . . } for its contextual cluster. Extrapolating from these co-clusters, as well as statistical associations between clusters of the same type, the authors argue about player perceptions of games more generally. For instance, they suggest that games that are perceived as being addictive, fun, or exciting are also perceived as being unique, deep, and innovative. Finally, the authors used their co-clusters as a feature representation with which to represent games themselves, which they then cluster using k-means. That is, they represent a game by a feature vector that specifies how many times particular adjectives were used to evaluate particular gameplay aspects in reviews for that game. From their clustering analysis, they observed (among other things) that clusters could not always be understood at the level of gameplay — for example, they cite a cluster of games that came from different gameplay genres but that were each based on animated television series.

As before, we find that the very nature of these results is rooted in the authors’ bottom-up method of inquiry. The fact that some of their clusters included games from multiple conventional genres highlights a key argument for this approach — when games are clustered according to how people actually talk about them, the resulting bottom-up typology contradicts the dominant top-down one. It intrinsically embeds the cognitive model, or dominant discourse, of those writing the texts into the results, and therefore reifies, to borrow from Lakoff, the writers’ specific multidimensional gestalt for each game.

As part of a larger exploration of cultural differences in game appraisal, Zagal and Tomuro studied lexical differences between Western and Japanese game reviews.³⁷ Specifically, for 221 games released in both the US and Japan, they compared the nouns most frequently occurring in user reviews submitted to GameSpot

³⁷[236] Zagal, José Pablo, and Noriko Tomuro. “Cultural Differences in Game Appreciation: A Study of Player Game Reviews.” In *FDG*, 86-93, 2013.

to those submitted to GameWorld, a Japanese website. Among other differences, they observed that Japanese reviews are more critical of technical issues, while replayability appears to be more central to Western concerns.

In “A Linguistic Analysis of Mobile Games: Verbs and Nouns for Content Estimation,” Lindsay Grace conducted two lexical analyses of developer descriptions of mobile games.³⁸ After compiling and analyzing the 38 distinct verbs used in developer descriptions of 70 best-selling games across the five most popular genres in Apple’s App Store, he offered three higher-level game-verb categories: verbs of elimination (“shoot”, “kill”, “destroy”, ...), categorization (“match”, “separate”, “choose”, ...), and transformation (“move”, “jump”, “rotate”, ...). In the second study, Grace compared the language used in Amazon descriptions of the 20 best-selling adult-fiction books of 2011 and 2012 to Apple App Store descriptions for that platform’s 20 best-selling games for those years. From these admittedly small samples, his findings suggest that books may include more violent, morbid content than games do.

Finally, in a series of recent papers published in the human-computer interaction (HCI) community, Zhu and Fang (and others) processed game reviews using a lexical approach similar to that of Zagal’s (though they appear unaware of this earlier work).³⁹ These authors conceive of their method as a refinement of ear-

³⁸[82] Grace, Lindsay D. “A Linguistic Analysis of Mobile Games: Verbs and Nouns for Content Estimation.” Proc. FDG, 2014.

³⁹[237] Zhu, Miaoqi, and Xiaowen Fang. “Developing Playability Heuristics for Computer Games from Online Reviews.” In International Conference of Design, User Experience, and Usability, 496-505. Springer, 2014. [238] —. “Introducing a Revised Lexical Approach to Study User Experience in Game Play by Analyzing Online Reviews.” In Proceedings of the 2014 Conference on Interactive Entertainment, 1-8. ACM, 2014. [239] —. “What Nouns and Adjectives in Online Game Reviews Can Tell Us about Player Experience?” In Proceedings of the Extended Abstracts of the 32nd Annual ACM Conference on Human Factors in Computing Systems, 1471-1476. ACM, 2014. [241] Zhu, Miaoqi, Xiaowen Fang, Susy S. Chan, and Jacek Brzezinski. “Building a Dictionary of Game-Descriptive Words to Study Playability.” In CHI’13 Extended Abstracts on Human Factors in Computing Systems, 1077-1082. ACM, 2013. [240] Zhu, Miaoqi, and Xiaowen Feng. “Using Lexicons Obtained from Online Reviews to Classify Computer Games,” 2013.

lier lexical approaches that in psychology led to the formulation of the famous five-factor model of personality.⁴⁰ From a collection of 696,801 game reviews submitted by users to GameSpot, IGN,⁴¹ and GameStop.com,⁴² they compiled the 4,843 most frequently occurring adjectives. Using the popular lexical database WordNet,⁴³ they merged together all synonymous adjectives to yield 788 adjective groups. Next, they proceeded to represent each adjective group by a feature vector specifying which documents adjectives from that group occurred in. From here, they submitted these adjective-group vectors to a statistical technique called factor analysis.⁴⁴ In factor analysis, statistical patterns among a set of observed variables (in this case, the adjectives) are exploited to construct a much smaller set of unobserved variables — called factors — that can still explain the full data set quite well. The idea is that the factors will represent core, higher-level concepts that underpin the data domain; as such, some form of factor analysis is often used in exploratory research that works bottom-up from a large amount of data.

Like these projects, the model underpinning ours could only be built using NLP and machine-learning techniques — it would not be feasible to hand-code (using a top-down approach) representations for several thousand games. That being said, we present a novel innovation of the methodology represented by the above projects. While the majority of research has processed game reviews — a text domain that is inherently evaluative in its tone and purpose — we use encyclopedic text and game walkthroughs, which are more objectively descriptive in tone and more ontological in purpose. As a major advantage of the particular

⁴⁰[147] McCrae, Robert R., and Paul T. Costa. “Validation of the Five-Factor Model of Personality across Instruments and Observers.” *Journal of Personality and Social Psychology* 52, no. 1 (1987): 81.

⁴¹<http://www.ign.com> (Accessed 5 Apr 2017.)

⁴²<http://www.gamestop.com> (Accessed 5 Apr 2017.)

⁴³[152] Miller, George A. “WordNet: A Lexical Database for English.” *Communications of the ACM* 38, no. 11 (1995): 39-41.

⁴⁴[89] Harman, Harry H. “Modern Factor Analysis,” 1960.

text sources we use, our model includes several thousand more games spanning a larger historical period. Furthermore, our use of latent semantic analysis (LSA), as mentioned above, is novel, and its first application to the realm of digital games.

Lastly, we avoid a fundamental shortcoming of the work that has been done in this area in the development of the tools and visualizations that sit atop our model. None of the previous models can be engaged beyond the publications describing them, which is troublesome given the complexity of machine learning models and the resulting difficulty of adequately describing them. We hope that future research in this area will follow our example of building and releasing tools by which machine learning models can be explored. This helps to enable better critique of the model's implicit assumptions, and reveals the potential for historical analysis of them.

5.4.2 Latent Semantic Analysis

The tools are supported by a latent semantic analysis (LSA) model trained on Wikipedia articles describing videogames, and GameFAQs game walkthroughs. LSA is a natural language processing (NLP) technique by which words are attributed vectorial semantic representations according to their contextual distributions across a large collection of text.⁴⁵ From a corpus of text, a co-occurrence matrix of its words and documents is built; this matrix specifies which words occurred in which documents (and thereby which documents words occurred in). The columns and rows in this matrix can be thought of as vectors that represent the meanings, in an approximate sense, of the words and documents that they correspond to — this is called a vector space model of semantics. LSA is an example of such a model, but its hallmark is that it reduces the dimensionality of these

⁴⁵<http://www.gamespot.com> (Accessed 5 Apr 2017.)

vectors by a matrix factorization algorithm. Remarkably, doing this allows the model to infer semantic associations that are not encoded in the full co-occurrence matrix. This ability to learn global associations from local co-occurrences is the achievement of LSA and what led to it becoming one of the major NLP techniques of the last twenty years. Having an LSA model, one can easily calculate how semantically related any of its documents are by taking the cosine between their LSA vectors. In corpora in which each document pertains to a specific individual concept, these relatedness scores can reasonably be utilized as a measure of the relatedness of the concepts themselves. Relying on this notion, we trained an LSA model on a corpus comprising Wikipedia articles for 11,829 videogames, and on GameFAQs walkthroughs for 5,739 games.⁴⁶ With these models, we can quantify how related any two of these games are by taking the cosine between their LSA vectors. Semantic relatedness between documents is typically calculated by taking the cosine between the documents' k-dimensional LSA vectors. If this is not intuitive, try conceiving of an LSA model as a k-dimensional space in which each document is placed at its k-dimensional coordinates. In this space, the semantic relatedness of two documents is reified as the distance between the documents' positions in the space — this distance is what the cosine represents. In corpora in which each document pertains to a specific individual concept, such as a corpus comprising encyclopedia entries, these relatedness scores can reasonably be utilized as a measure of the relatedness of the concepts themselves. As we explain below, this is how the tools, GameNet and GameSage, reason about game relatedness, and are part of how the visualizations, GameGlobs, GameSpace, and

⁴⁶Note that I am still working with James Ryan on updating the models and visualizations, so there will be a note on that work in here somewhere in the future. We are currently at around 16,000 games, but I left these numbers in for now since I don't have the final tally for the new set yet.

GameTree, represent them.⁴⁷

5.5 Tools for Discovery

Below we describe the two search tools based on our LSA model, and briefly recount the only other known work in game discovery systems.

5.5.1 Related Discovery Work

Videogame discovery is an emerging application area for which very little work has yet been done. In what would appear to be the first published effort in this domain, Lee et al. (2015) present Vizmo, a discovery tool that indexes games by visual style and mood. Influenced by earlier work in constructing browsers for other media, the tool was built using a faceted-metadata approach. Specifically, Vizmo is underpinned by a database of games that have been manually annotated for their visual style and mood using a subset of a larger videogame metadata schema that was created by the same group.⁴⁸ Users can browse the games in this database by setting filters for different combinations of visual style and mood, and the results are displayed in a chronologically oriented chart. From an expert evaluation conducted with nineteen game professionals, Lee et al. found Vizmo to be an aesthetically pleasing tool with potential use for game discovery along aesthetic or historical concerns. Though promising, Vizmo is an early prototype that currently houses only 604 titles, many of which are platform variants of the

⁴⁷For more detailed information on the derivation process for the model, please see Section 3.2 “Model Derivation” in [189] Ryan, James Owen, Eric Kaltman, Michael Mateas, and Noah Wardrip-Fruin. “What We Talk About When We Talk About Games: Bottom-Up Game Studies Using Natural Language Processing.” Proceedings of the 10th International Conference on the Foundations of Digital Games, 2015.

⁴⁸[123] Lee, Jin Ha, Sungsoo Ray Hong, Hyerim Cho, and Yea-Seul Kim. “Vizmo Game Browser: Accessing Video Games by Visual Style and Mood.” In Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, 149-152. ACM, 2015.

same game. That so few games are yet included is not surprising given that each must be manually annotated in order to be indexable by the tool. Until more games are added, it appears that Vizmo will not offer extensive practical use for game discovery (a point we return to later in discussing the results of our experiment).

If Vizmo may be thought of as taking a traditional top-down, metadata driven approach to game discovery, one in which humans handcraft indexable representations of games (which a discovery tool may then operate over). Our model above takes a decidedly bottom-up approach. There are numerous trade-offs between these two approaches. Employing a semiautomated method, we could quickly build full-fledged discovery tools comprising several thousand games. But relying on a statistical procedure to derive indexable game representations, we were left with tools that operate over rather opaque notions of what games are made of — as discussed above, our tools reason about games in terms of arcane statistical features of their textual descriptions. Vizmo, on the other hand, reasons over games purely in terms of human-crafted specifications. As such, it will always be clear how Vizmo indexes games, because its indexable representations are simply human annotations. (One advantage of unsupervised reasoning already noted is the ability to reveal systemic bias in description, or to draw connections that could not be reasonably conceived by a human indexer.) But while Vizmo’s game indexing is perhaps more reliable and certainly more transparent, GameNet offers reliable-enough indexing for ten to twenty times as many games depending on the model selected. See the evaluation section below for a substantiation of this point.

5.5.2 GameNet

GameNet is a tool for game discovery in the form of a network in which related games are linked. The tool is a front-end for our two different LSA models. “Ontology” is the model derived from Wikipedia articles, and is labeled as ontological due to the highly varied and unstructured nature of its game descriptions. “Gameplay” is derived from GameFAQs walkthroughs that almost exclusively describe game play structure and actions. Each game’s entry includes links to GameNet entries for other games that are related to that game, as well as to gameplay videos and other informative sources found elsewhere on the web. [fig:gamenet-wall-street] below shows excerpts from the GameNet homepage and its entry for the Nintendo Entertainment System game *Wall Street Kid* (1990).



Figure 5.1: GameNet Search and Results for Wall Street Kid

At the GameNet home page, the user first indicates which model to use, and the inputs the title of the game she wishes to find. An autocomplete handles correction of typos on the input, and then loads up the game GameNet entry. In the page header, the game’s title and year of release are prominent, as well as links to the game’s Wikipedia article and Google Images and YouTube search results found using autogenerated queries; included as well is a summary of the game that was extracted from Wikipedia. Below these elements is the core of the entry, which is a colorcoded listing of the fifty most related games to the game at hand. GameNet judges how related any two games are by taking the cosine between their documents’ LSA vectors. To promote exploration, the related games are

stylized as hyperlinks to their own GameNet entries. Finally, below the listing of related games is a listing of the most unrelated games to the game at hand. These links can potentially serve as portals to corners of the medium that were previously unknown to the user.

5.5.3 GameSage

GameSage is a tool that takes free-text input describing an idea for a videogame and lists existing games that are most related to that idea. This tool utilizes the notion in LSA of folding in, whereby a new document that was not used during model training is fitted with a representation in the semantic space derived by the chosen model. By treating the user’s input text (which specifies her game idea) as a corpus document (on par with the Wikipedia or GameFAQs texts we used to train our LSA models) and folding it in, we are able to derive an LSA vector for the idea. From here, we determine which existing games — from among GameNet’s ontological or gameplay models — are most related to the game idea by using cosine similarity, just as we did in constructing GameNet.

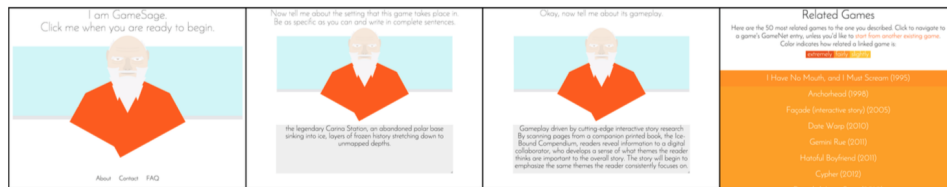


Figure 5.2: GameSage Query for Non-Corpus Game

Figure 5.2 shows excerpts from a session with the tool in which a user describes the in-development indie game *IceBound* and gets back a list of related games (including other indie games, story-focused games, and games used in narrative-technology research). At the GameSage home page, the system is personified as

an animated sage character who proceeds to ask the user seven questions about her game idea, each pertaining to a particular facet of the (prospective) game. After the final question is answered, the system concatenates the responses and preprocesses this text using the same procedure we enacted to preprocess the selected corpus. From here, the preprocessed text is attributed an LSA vector by folding it into the user's chosen model and the model's games are then ranked according to how related they are to the user's idea. Finally, GameSage makes a request to GameNet to generate an entry for the game idea, which is then presented to the user.

There is obvious discovery potential in finding games that are related to an idea for a game, which could prove helpful as a way of gathering insight during the early stages of game development. It also functions as a way for historians to locate games related to ones they cannot find in the model, but might be significantly related in some way. The evaluation section below provides two different studies that evaluated both game designers and game studies researcher for both prospective use cases.

5.6 Visualizations for Discovery

In this section we describe three visualizations created as expressions of our LSA model. These visualizations are intended to supply other potential vectors for game discovery, and to illuminate the current historical space of games as recounted by the discourses present on Wikipedia. Each visualization makes use of common visualization techniques that have not previous been applied to games history. Approaching visualization algorithms as more than just a pretty picture, we view them as a new means for argumentation through computational techniques. That algorithms can function as humanistic arguments, or significantly

support them is not a new idea (Ramsay, Staley), but there is still not much significant work in the area due to the need to combine a humanistic question with a visualized answer. In the case of the visualizations below, we feel they can form the basis for argumentation about the state of discourse and the typological reality of games at both particular moments in time, and based on particular sets of texts (discourse). The visualizations only make use of the games from the Wikipedia corpus, and we discuss the GameFAQs corpus work more directly in the Future Work section.

5.6.1 GameGlobs

GameGlobs is a two-dimensional visualization of various clusterings of the games in our LSA model. A user selects how many clusters (groups of related games) she would like to see the 11,829 games of the Wikipedia model partitioned into and is presented with such a clustering, as shown in Figure 1. Each cluster is drawn as a circle that can be clicked to display the games it contains, which are stylized as hyperlinks to their entries in GameNet. The clusterings themselves were derived by applying the classic k-means algorithm to the games' LSA vectors.⁴⁹ GameGlobs includes clusterings using several values for k (number of clusters) spanning between 2 and 2500 and utilizes two key visual cues: clusters with more games appear larger, and clusters are positioned semantically, such that clusters whose games are more similar are nearer one another. To achieve the latter effect, we used a technique called multidimensional scaling (MDS), which is a way of building low-dimensional visualizations of high-dimensional data.⁵⁰ This technique is represented by a suite of algorithms; we submitted the LSA vectors of our cluster centroids to a variant called locally linear embedding (LLE) to derive

⁴⁹See note 29.

⁵⁰[58] Cox, Trevor F., and Michael AA Cox. *Multidimensional Scaling*. CRC press, 2000.

their 2D coordinates.

5.6.2 Gamespace

GameSpace is an explorable three-dimensional ontological space in which each game in the model is represented as a data-rich star whose positioning is semantically meaningful. Specifically, games are placed in the space such that their most related games are nearby.⁵¹ Three-dimensional coordinates for the games were derived by submitting their LSA vectors to multi-dimensional scaling, as in GameGlobs, however the specific algorithm is not LLE but TSNE.⁵² The user can fly freely through the space using conventional 3D game controls, and upon encountering a game can click on it for more information, including: its title and year of release; an embedded YouTube player with a Let's Play video preloaded; an embedded pane displaying its Wikipedia page; and link to share the game and its position on Twitter.

5.6.3 GameTree

GameTree, shown in Figure 5.6 is a massive two-dimensional visualization of a hierarchical taxonomy of the games in our model. The underlying representation is a tree that was built bottom-up by submitting the games' LSA vectors to an algorithm called hierarchical agglomerative clustering,⁵³ which works as follows: each of a set of objects is initialized to be its own cluster; on each iteration, the two clusters whose centroids are most similar are merged into a higher-level cluster, whose centroid gets set as the mean of those two centroids; this repeats iteratively

⁵¹Try GameSpace out at: <http://gamespace.io>

⁵²This is due to the visualized space being more human interpretable in three dimensions with t-Distributed Stochastic Neighbor Embedding (t-SNE).

⁵³For more information on the visualizations see, [186].

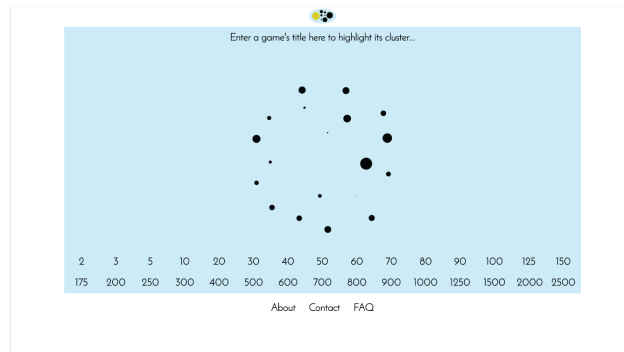
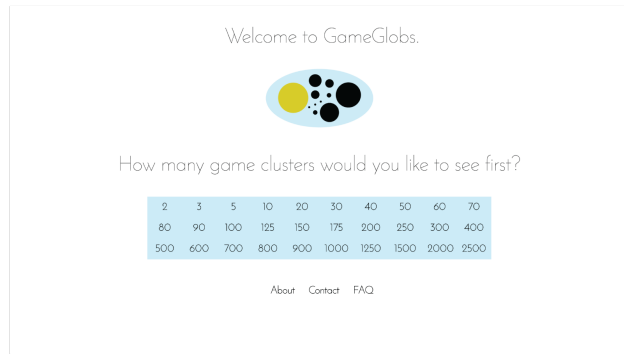


Figure 5.3: GameGlobs Showing 20 Clusters

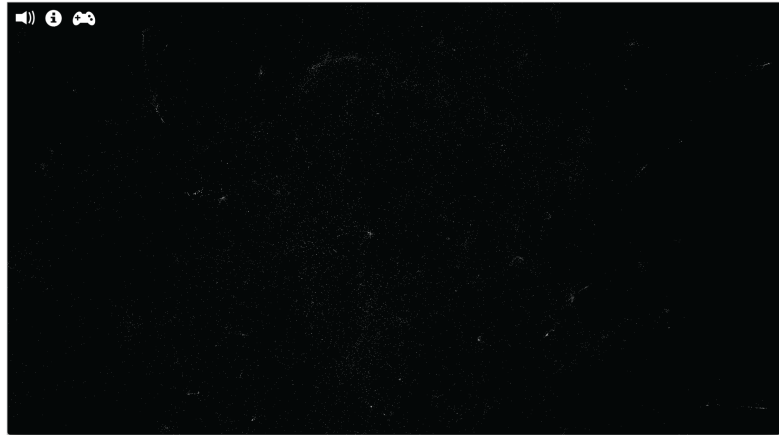


Figure 5.4: GameSpace Intro Screen and Main Space

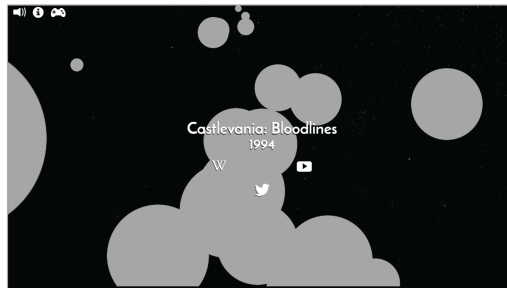


Figure 5.5: GameSpace Game Selection with Wikipedia and Youtube

until a root node is formed by merging the last two remaining clusters. Figure 5.6 shows the visualization, which is a radial tree.

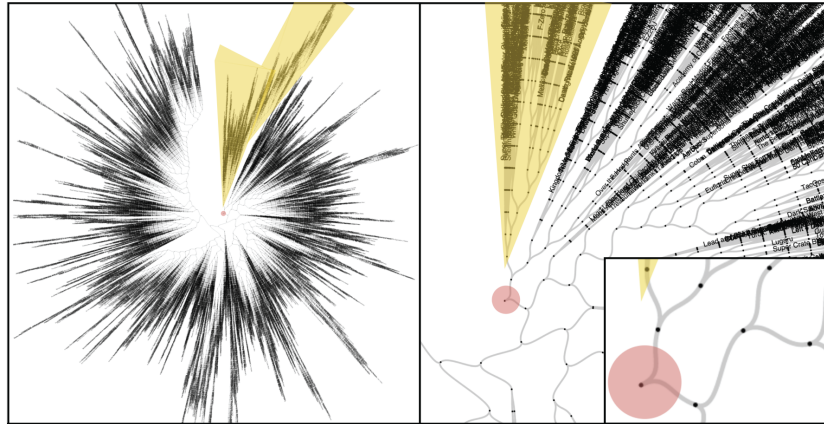


Figure 5.6: GameTree with Racing Game Branch Highlighted

5.7 Evaluation

Both use cases for the tools, as (a) a means for designers to locate historical titles related to their design ideas, and (b) for practicing game scholars to find titles related to their research interests, have been subject to published evaluations. For game studies researchers, we conducted an expert evaluation similar to that carried out for the CiteTool in the previous chapter. For game designers, we evaluated the tools utility in an introductory game design course where students needed to locate examples similar to their recent designs. Both evaluations were on the whole, very successful and validated the effectiveness and promise of both game discovery and computational front-ends to NLP models. One note is that both studies were conducted before the addition of the GameFAQs model, therefore

while there is future potential for comparative study that was not of concern at the time.

5.7.1 Expert Evaluation

We asked six published game scholars (who had recently conducted studies for which our tool could have conceivably proved helpful) to use GameNet for fifteen minutes and answer a series of questions about the experience. As a preliminary question, we asked the individuals what scholarly approaches they had employed in their recent projects to research games related to the specific titles or topics they were writing about. Interestingly, though not surprisingly, the scholars listed several methods in total. These included, in no particular order, using Google Scholar and other sources to find related scholarly work; searching Wikipedia for articles describing individual games; playing games using both native hardware and emulation; reading game criticism found online, as well as newspaper articles, magazine reviews, game guides, and game tips that were written at the time of the game’s publication (for older games, these included scans and transcriptions and were found across various web sources); watching Let’s Play videos and other YouTube footage demonstrating speed runs, glitches, walkthroughs, and general gameplay; and, finally, referencing other resources produced by fans, such as walkthroughs and FAQs, as well as a domain-specific informational database (IFDB, the Interactive Fiction Database). We note that the wide variety of approaches these six scholars employed highlights the absence of any single tool for game-studies research that incorporates all the various types of media that they utilized. Interestingly, though, GameNet does include pointers to both Wikipedia articles and Let’s Play videos, which were each among the enlisted approaches.

Upon answering this initial question, we instructed each of the scholars to start

at the GameNet entry for a specific game that was related to his or her recent project. Unfortunately, three of the scholars had hoped to start at games that do not have Wikipedia articles, and which are thus not included in GameNet (each instead settled on another recent game of study). Our six scholars and the games they started from were as follows: D. Fox Harrell, *Ultima IV: Quest of the Avatar*; Katherine Isbister, *The Sims*; Dylan Lederle-Ensign, *Quake III Arena*; Soraya Murray, *Assassin's Creed III: Liberation*; James Newman, *Super Mario Bros.*; and Aaron A. Reed, Thomas M. Disch's *Amnesia*. (For Harrell and Lederle-Ensign's projects see note; the rest are currently in submission or still in progress.)⁵⁴ Upon reaching the entry for their respective games of interest, the scholars each used the tool for at least fifteen minutes before completing our questionnaire.

We asked whether GameNet would have provided a faster way to locate games related to their recent topics of study, relative to the scholarly approaches they had previously employed. Here, the responses broadly indicated that, as domain experts for their respective topics, they had used the scholarly approaches mentioned above to probe more deeply into specific titles, rather than to seek out additional games related to the topic. Generally, the scholars indicated that, while this would not have helped in their particular recent projects, the tool could prove especially useful as a first method for exploring an area of games that is unfamiliar to the user. "It felt as if it would be more useful to get broad connections in a space I wasn't as familiar in," Reed explained. Isbister, however, appreciated GameNet affirming more tenuous connections between games that she already had in mind.

⁵⁴[124] Lim, Chong-U., and D. Fox Harrell. "Revealing Social Identity Phenomena in Videogames with Archetypal Analysis." In Proceedings of the 6th International AISB Symposium on AI and Games, 2015. and [121] Lederle-Ensign, Dylan, and Noah Wardrip-Fruin. "What Is Strafe Jumping? idtech3 and the Game Engine as Software Platform." Transactions of the Digital Games Research Association 2, no. 2 (2016).

This feeling of being in agreement with the tool on games she already knew led her to be more interested in the games it listed that she did not know about.

When asked whether their fifteen minutes on GameNet led to the discovery of a game that was previously unknown to them or that they had not realized was relevant to their topic, the scholars answered in the affirmative. Lederle-Ensign found multiple titles he had not considered discussing in his study, while Harrell had this to say: “[I came upon] one game I had not thought about much since childhood and seeing it described now made me realize that it had some interesting features relevant to my research.” Similarly, Isbister remarked, “I definitely found games that looked promising that I did not know about.” Reed, an expert on interactive fiction [44], was surprised to discover an Infocom title he had not known existed. Starting from *Super Mario Bros.*, Newman found three obscure games in Famicom exclusive *Armadillo*, Commodore 64 fan sequel *Mario Bros. II*, and Wisdom Tree’s *Bible Adventures*. “[These] weren’t titles I would have got to so quickly, if at all,” he remarked. Additionally, Newman was intrigued to find that these games seemed to not be directly related to *Super Mario Bros.*, but more precisely seemed two degrees removed from it by way of Nintendo *Game & Watch* title *Mario Bros*, *Super Mario Bros. 2* [U.S. version], and *Super Mario Bros. 3*, respectively. He added, “Getting to games that were similar to ones similar to my original search was quicker with this tool.”

As domain experts in the particular areas they explored, Harrell, Isbister, Lederle-Ensign, and Newman all endorsed the connections between games that GameNet listed. Murray and Reed, however, explained that the connections they saw were rather broad relative to their more specific research angles. Interested in finding other titles that took up Amnesia’s simulationist approach to interactive fiction — or that, like that title, were authored by a famous fiction writer (in

Amnesia’s case, this is science fiction writer Thomas M. Disch) — he instead found GameNet’s connections to be at the level of genre grouping. That is, the related games he found were merely other examples of text adventures, rather than titles that shared the particular gameplay and production attributes he was interested in. Similarly, Murray was seeking out other games that, like *Assassin’s Creed III: Liberation*, have strong female protagonists, but instead found all the other titles from that series (which all have male protagonists) and other games that she felt were related according to broader notions of genre.

Lastly, we requested any additional feedback that the scholars felt like giving. Lederle-Ensign and Reed took this opportunity to praise the interface, and several expressed that GameNet is simply fun to use. “Using it free-associatively (rather than staying based around one core game) is a lot of fun,” commented Reed, adding that it is “interesting to see the connection trails it finds.” In a similar vein, Newman noted, “there’s pleasure in figuring out the connections, particularly as you get further from the original selection.” Both, however, wished that GameNet would specifically characterize the nature of the connections it lists, a notion that was central to Murray’s feedback as well.

5.7.2 Novice Game Designer Evaluation

Influenced by the notion of task-based evaluation of exploratory search systems,⁵⁵ in which a system is evaluated for its adequacy in the natural context of its user task, we conducted a user experiment in which game-design students sought out games related to their own using GameNet and GameSage and also a baseline method in which they were permitted to use any resources available online. We

⁵⁵[111] Kraaij, Wessel, and Wilfried Post. “Task Based Evaluation of Exploratory Search Systems.” In Proc. of SIGIR 2006 Workshop, Evaluation Exploratory Search Systems, Seattle, USA, 24-27, 2006.

chose this baseline method because we believe it represents the (lack of a) state of the art in game discovery today. The primary variables of interest to our analysis are the number of games discovered using both methods, the diversity of games discovered (i.e., percentage that were unique), and the proportion of discovered games that were unfamiliar to users prior to the experiment. We will briefly describe the procedure and results of the evaluation, but refer the interested reader to our full published account, which goes into significantly more detail on the full experimental procedure and its caveats.⁵⁶

Participants

182 participants (20% women) took part in the experiment, with ages ranging between 18 and 27 ($M = 19.45$). The participants were all undergraduate students enrolled in an introductory game-design course. This course fulfills a general-education requirement and has no prerequisites; as such, the students hailed from diverse academic backgrounds encompassing 42 different degree programs. In a preliminary questionnaire, we asked the participants about their level of game-development expertise: 34% reported no game-development experience prior to enrolling in the class, 57% claimed novice-level expertise, and the remaining 9% called themselves experienced.

Experimental Task

Prior to us conducting the experiment, each of the students had completed an assignment in which he or she created a game emphasizing exploration in an unusual or metaphorical space. For both experimental conditions, the task was

⁵⁶[188] Ryan, James Owen, Eric Kaltman, Michael Mateas, and Noah Wardrip-Fruin. “Tools for Videogame Discovery Built Using Latent Semantic Analysis.” Proceedings of the 10th International Conference on the Foundations of Digital Games, 2015.

the same: participants were provided with an online form and asked to spend fifteen minutes finding games related to their respective games, entering their titles into the online form as they did. On this form, the participant was asked to place related games, as they discovered them, into one of two categories—familiar games, games they were already familiar with prior to the experiment, and unfamiliar games, games they were not familiar with prior. We let the participants operationalize their own criteria for familiarity, but we provided a specific notion of relatedness: students had an upcoming extra credit assignment in which they would be tasked with writing about games related to their own games that were not discussed in lecture, so we suggested that they deem a prospective game to be related if they would consider discussing it in this assignment.

Results and Discussion

The evaluation of the tool was based on three initial hypotheses about their effectiveness towards game discovery tasks. Each one is premised on an assumption that the models presented through GameNet and GameSage would allow for a greater exposure to the breadth of historical games, and function as a means for game designers to get quicker exposure to titles closely related to their own designs.

- Hypothesis 1: Participants will discover more related games using our tools
Our results strongly support this hypothesis. Indeed, participants discovered more than twice as many related games using the tools (nearly one per minute) than they did using assorted web resources. This difference is especially remarkable when considering that none of the games participants discovered in the baseline condition could be counted toward their discoveries in the tools condition. For our baseline condition, we chose to allow

participants to use any available web resources because we believe this best represents the (lack of a) state of the art in game discovery.

- Hypothesis 2: Participants will discover a greater diversity of games using our tools The results did not support this hypothesis, as the difference in condition proportions was not statistically significant. While we had anticipated that the proportion of unique titles would be significantly higher in the tools condition, the fact that it even approximates that of the baseline condition is still remarkable. Because the effective search space yielded by all resources on the web is orders of magnitude greater than that represented by the 12,000 games in GameNet, one might expect that, all other things being equal, a much larger proportion of games discovered in the baseline condition would be unique. This was indeed not the case, however, and in fact 47% of the unique titles discovered during the baseline condition were games that are included in GameNet. This suggests that, had the tools condition been ordered first in our study design, participants might have discovered more games with the tools, as well as a greater diversity of games (since games included in GameNet that were already discovered by a participant in the baseline condition could not be discovered by that participant in the tools condition).
- Hypothesis 3: Participants will discover a greater proportion of unfamiliar games using our tools Our results strongly support this hypothesis. We made this prediction from the intuition that, in lieu of a dedicated tool for game discovery, participants would tend to seek out (as a sort of scaffolding) games they could already name as related to their own. Indeed, roughly half of the games discovered by participants in the baseline condition were already known to them. We believe that the demonstrated facility of GameNet

and GameSage to provide users with extensive listings of games that were previously unfamiliar to them is perhaps the system's greatest strength.

5.8 Future Work

The tools and visualizations described above provide significant opportunities for discovery in archival and library collections, and therefore for the reclamation of computer game history. Each provides a new way of seeing a collection, and encourages a form of directed serendipity by allowing for the indirect alignment of titles by relatedness. Additionally, since the models are representations of specific discourses about games and their history, they reveal lacunae in those records. Further work on the tools is focused on leveraging these properties of our discovery techniques and applying them to actual physical and digital collections.

We have mapped the models to titles available in the Stephen Cabrinety Collection at Stanford University (Edwards et al.). This affords two immediate oppositional uses for the tools: (1) the serendipitous discovery of historically obscure titles, and (2) pointers to omissions in the specific discursive model. The first use benefits from the descriptive text that the community has organized for games. Many titles in the Cabrinety Collection are relevant to game history but are not apparent in that history due to their unavailability. By linking those titles to online descriptions, the work of one intrepid Wikipedia or GameFAQs user can save titles from historical obscurity and bring them into alignment and parity with other more famous ones. Contrarily, titles that could not map to the model are implicitly underdescribed in a particular discourse. This means that they cannot be linked to the other historical items, and may suffer more in the future as community knowledge of them fades. In these cases, the mapping can point that out and ask others to step in and provide descriptive text.

The visualizations are especially useful as a means of filling out a discovery listing. In exploring each representation, researchers may realize a particularly significant omission, and can attempt to remedy it by providing text that could then be used for future visualization. Our future work on GameSpace is oriented around this premise, and we are developing means to automatically update the model on a fixed cycle, or use the folding-in technique from GameSage to populate the three-dimensional GameSpace space with new titles as they are released. Any update methodology for the underlying models also inherently benefits all of the tools, since they are all similarly derived.

Another vector for future work is further validation of the tools. We are currently in the process of analyzing data from an evaluation study conducted with students studying computer game history. The experiment was an attempt to determine how well the tools covered particular historical spaces. Students were asked to use GameNet and GameSage to try and predict the games that would be discussed at certain future points in the class. The aim was to see if our discovery tools could be useful indicators of the historical salience of a particular title, and to find out if certain historical or topical areas were obviously missing from the underlying model. We foresee future similar evaluations for GameSpace once we update our model with games released in the last two years.

5.9 Conclusion

This chapter set out to introduce a new application of computational modeling to issues in the exploration and expression of computer game history. In using LSA as a base for the organization of the collected social knowledge of games — and as a reification of a solution to issues of game discovery — we provided a new window into games history, and leveraged it in the creation of new tools

for visualization and retrieval. That the model was so fruitful — GameNet and GameSpace in particular have been used by thousands of people — shows the power of computational methods to contribute to digital humanist inquiries and supplement older systems of record organization and representation.

Chapter 6

A Model of *Doom*

6.1 Introduction

In this chapter we present a case study of the computer game *Doom* that leverages the insights of the previous chapters. We have now dealt with issues of the appraisal and organization of development documentation, the sticky issues of game objective description, means for the retrieval and citation of games and their play, and simply finding relevant, related historical examples for comparative work. All of these actions are tied to two primary motivations. The first is to find ways to leverage (and make visible) the heterogeneous accumulation of documentation that surrounds software objects. This includes how those accumulations dictate or enact specific histories and limit the presentation of others. The second motivation is to point out that the different types of histories that result from different historic sources require different considerations for historical expression. As mentioned in the previous two chapters, which explored embedded emulation and machine learning in the exploration of computer game history, certain classes of documentation — in this case executable objects, and large corpora of descriptive text — demand certain types of analysis to support new historical expressions

and make them comprehensible to a reader. Our previous investigations offered a glimpse of the immensity of the work required to categorize and stabilize the basic records of computer games. This chapter takes a look at the accumulation of resources available to software historians, and argues that greater consideration needs to be paid to not only the surface outputs of software developments, the commercial game packaged for consumption, but also the histories of all the intermediate stages of development, both in the design and construction of software, and even the multiple layers of abstraction built into the object itself.

Recounting the history of any object, any “thing”, be it a person, concept, piece of software, or social movement, is dependent on the documentation available about its existence. Any history of it is contingent on the way that a historian takes that documentation and pushes it through their own ideologies, theories, and presuppositions about the world. In this way, all histories are incomplete, contingent, and limited because it is not feasible to reveal (or really comprehend) the totality of any thing in the world, only various aspects of it.¹ As mentioned below, these ideas of incommensurate historic viewpoints, and the inherent biases of any historiographical position, are not new. The field has arguably been dealing with them since the birth of the historical profession.² Our innovation is to acknowledge the crush of potential readings of *Doom*, and then remind everyone of how each of those views, each of those angles is dependent on the resources, methodologies, and tools available to the historian. We take each reading as an example of a methodological intervention into a particular aspect of *Doom*’s material history. Too often historians pay little attention to how their sources are maintained

¹[221] White, Hayden V. *The Practical Past*. FlashPoints. Evanston, Illinois: Northwestern University Press, 2014. pg. 11

²Hayden White, in *Metahistory* [219], begins with the influence of Hegel’s theorizations of an enacting “spirit” for human history and progress. White’s later works [221] eventually land in a post-modern admission of the impossibility of historical “objectivity”.

and preserved, and how those who are subject to their histories — here gamers and software developers — produce and preserve their own resources. For game history, this may seem like an exaggeration, but our work in this thesis presents the first attempt at informed descriptions of computer games for libraries, the second appraisal of game development work, and the first example of software-based citation of games. Clearly there is still an incredible amount of work to be done.

The remainder of this chapter is split into two major sections. We first lay out a new scheme for the interpretation of computer game history based on ideas from John Law, in the history of science, and Hayden White, in historiography. They provide nice, concise legitimations for interpreting history through the use of multiple angles of inquiry and leveraging new types of historical expression. We also highlight and make use of previous schemas for game software historical analysis to help with the divination of new angles from which to view *Doom*.³ The model of history presented in the first section is designed to tease out the lacunae in the software documentary record. As will become apparent, most works of game history focus on chronicles of events applied to historical presuppositions about games instead of literal game objects. Recall from the Citation chapter that a “presupposition” of a game is an author’s (or player’s) recollection of that title divorced from any concrete instance of a game object. Because most histories of games only refer to these top-level signifiers — for instance, the game *Doom* actually exists in many versions with many different game play experiences and configurations — there are areas of the history of game software that do not receive attention either historically or methodologically. This initial model and section

³Discussion focus primarily on [74] Fernández-Vara, Clara. *Introduction to Game Analysis*. 1st edition. New York: Routledge, 2014, [110] Konzack, Lars. “Computer Game Criticism: A Method for Computer Game Analysis.” In CGDC Conf., 2002. <http://www.academia.edu/download/31458323/konzack-tampere2002.pdf>. [157] Montfort, Nick. “Combat in Context.” *Game Studies* 6, no. 1 (December 2006). <http://gamestudies.org/0601/articles/montfort>.

are then arguments for filling in these holes in the history of computer games and further, the history of software.

The second part of this chapter focuses on the layers of the historical model described in the first. Each layer is not actually constitutive, in that the model does not argue that each layer supports another, but rather that there are different potential targets for historical investigation occurring at different documentary vectors for a piece of software. It is not that layers do not interact, but that the model does not make direct claims that each layer is required or necessary for all historical arguments, but that each might have specific evidentiary details that could affect general technical historical claims. Basically, one needs to be aware of how each layer might affect arguments, and potentially look into these areas as a part of their broader historical methodology. This section then applies this layered consideration to specific sets of documentation about the computer game *Doom*. Specifically, there is focus on how *Doom* exists as a presupposed historical object, how *Doom* versions contribute to an understanding of its development, what *Doom*'s source code reveals — and fails to reveal — about its design history, and finally, how looking into the professional knowledge of *Doom*'s creators — their tacit knowledge of game design — and how they acquired it points to a more nuanced appraisal of *Doom*'s technical contributions to game software. These smaller pieces function to bulwark our larger, thesis-level assertion about the need to reveal software history through a focus the material conditions of its documentary accumulation (stabilization), and on better ways to illustrate that accumulation to readers (exploration and expression).

6.2 Fractal History

Pulling back significantly, we first address the basis for a layered or multi-vectored approach to historical study. As noted by historian Hayden White in his many discussions of the challenges inherent to historical narrative,⁴ the post-modern quagmire caused by revealing “realistic”, “scientific” or “true-to-life” histories to be, at best, partial myths betrayed by both the limits of a specific era’s discourse and the implicit ideological biases of the historian, are well rehearsed. White points out that since the end of the nineteenth century, the delimitation between science and art left history in a bit of a bind.⁵ Being neither formalized enough for the sciences, and too limited, rote, and constricted to be an art.

In sum, when historians claim that history is a combination of science and art, they generally mean that it is a combination of *late-nineteenth century* social science and *mid-nineteenth century* art. That is to say, they seem to be aspiring to little more than a synthesis of modes of analysis and expression that have their antiquity alone to commend them. If this is the case, then artists and scientists alike are justified in criticizing historians, *not because they study the past*, but because they are studying it with *bad* science and *bad* art. The “badness” of these hoary conceptions of science and art is contained above all in the outmoded conceptions of objectivity which characterize them.⁶

The problem of historical study is then one of affecting an objective, totalizing take on a subject or situation. Complementary with this “outmoded conception” of the nature of reality, is a similarly outmoded means of expression. As White notes,

⁴[219] White, Hayden V. *Metahistory: The Historical Imagination in Nineteenth-Century Europe*. Baltimore: Johns Hopkins University Press, 1973. White, Hayden V. *The Practical Past*. FlashPoints. Evanston, Illinois: Northwestern University Press, 2014. [220] —. *Tropics of Discourse: Essays in Cultural Criticism*. Baltimore: Johns Hopkins University Press, 1978.

⁵Peter Burke, in *Social History of Knowledge Part II* [46], concurs, pointing to the “crisis of sciences” around that time resulting from the new philosophical positions of phenomenology (Husserl [94]) and others. The positivist world view buckled under new, relativistic insights from many different disciplines.

⁶[220] pg. 43

“there have been no significant attempts at surrealistic, expressionistic, or existentialist historiography in this [the 20th] century. . . It is almost as if historians believed that the *sole possible form* of historical narration was that used in the English novel as it developed in the late nineteenth century.”⁷

The resolution proposed by White is to make any history not a totalizing narrative of a specific event or object, but an *admittedly* partial, and therefore incidentally biased, argument based on a particular reading and interpretation of historical sources. He draws this out by referring to the efforts of philosophers of science in “working toward a better understanding of the similarities between scientific statements on the one hand and artistic statements on the other.”⁸ That is, in examining the underlying philosophy and practice of science, these scholars “have modulated the harsh distinctions originally drawn by positivists between scientific and metaphysical statements” and have significantly legitimized the latter. By weakening positivist beliefs about the objective nature of reality, and singular, “correct” ways of viewing it, the philosophy of science has paved the way for a new consideration of history. One that no longer should, or actually can, display an objective account of its subjects.

Thus envisaged, the governing metaphor of an historical account could be treated as a *heuristic rule which self-consciously eliminates certain kinds of data from consideration as evidence*. The historian operating under such a conception could thus be viewed as one who, like the modern artist and scientist, seeks to exploit a certain perspective on the world that does not pretend to exhaust description or analysis of all of the data in the entire phenomenal field but rather offers itself *as one way among many* of disclosing certain aspects of the field. (Emphasis in original, conveniently.)⁹

White’s point is that historical study had limited itself through an attachment to modes of expression that were no longer commensurate with modern philosophy,

⁷[220] pg. 43

⁸[220] pg. 46

⁹[220] pg. 46

or even the basic experiences of people operating in modern society. Furthermore, that history should be seen as a more interpretive — and emphatically biased — art also called for its modes of expression to change in line with modern artistic sensibilities. Perhaps certain histories can only be expressed through particular artistic constellations or a particular style of representation. “To recognize that there is no such thing as a *single* correct view of any object under study but that there are *many* correct views,” that are each dependent on particular types and classes of sources, and can only be coherently represented through varying artistic and discursive strategies.¹⁰

In returning to the production of this thesis, most specifically the previous chapter on citation and discursive presentation, the ability to conduct a history of games and software is dependent on the types of data collected and available for their study. It is also dependent on the modes of expression that try to capture software’s phenomenal and aesthetic experience. And here we are not only talking about the player experience, but also that of game designers and the tacit practice and technical implementations they undertake. Since White’s work functions in a meta-historical frame that attempts to call out the various narrative and literary tropes of historical presentation (he is a literary critic after all), we take White’s legitimation of multiple expressions of and views on history and combine it with the terminology of John Law, a historian of technology.

The “fractal” of this section’s “fractal history” is borrowed from John Law’s work *Aircraft Stories*, which presents a set of interweaving perspectives on the British TSR2 strike and reconnaissance warplane and its representation and interpretation through various evidential vectors. Each avenue of inquiry added more to an alignment of multiple views that resulted in a “fractal coherence” of the object as situated in history and through its various documentary sources.

¹⁰[220] pg. 47

This metaphor of fractal coherence is therefore a potential solution to the multiplicity of histories noted by White. In moving to juxtapose multiple accounts, Law is overcoming some of the deficiencies of singularized ones in hopes that they will “cohere” into a greater truth.¹¹ He is also analyzing a technical and engineered object in his work, which is closer, ontologically, to our target of game software. Law intentionally frames his contribution as an extension from the post-structuralist claims on historical narrative put forward by White, to the history of objects.

Knowing subjects . . . are not coherent wholes. Instead they are multiple, assemblages. This has been said about subjects of action, of emotion, and of desire in many ways, and is often, to be sure, a post-structuralist claim. But I argue in this book that the same holds true for objects too. But it also reveals multiplicity — for instance in wing shape, speed, military roles, and political attributes. I am saying, then, that an object such as an aircraft — an “individual” and “specific” aircraft — comes in different versions. It has no single center. It is multiple. And yet these various versions also interfere with one another and shuffle themselves together to make a singular aircraft. They make what I call singularities, or singular objects out of their multiplicity. In short, they make objects that cohere.¹²

The hope is that in presenting multiple views, multiple types of history about a subject (object) that we end up “drawing things together without centering them,” that we get a glimpse of their totality through the fragments of our analysis and its presentation.¹³ This “totality” however, is explicitly non-prescriptive, in that it still rests on whatever collection of histories we choose, and is never an attempt at a full or complete presentation of historical “truth.” A way to visualize this notion

¹¹The juxtaposition also helps alleviate some of the strain imposed by post-modern critiques of historical inquiry. Namely, that it becomes more difficult to make any emphatic statements about history when having to consider the various subjectivities at work in both the historian’s prose and their sources of argumentation. By placing multiple framings along side one another, we can begin to make out a new historical form that coalesces in the gaps between and betwixt accounts. More on this below.

¹²[120] Law, John. *Aircraft Stories: Decentering the Object in Technoscience*. Duke University Press, 2002. pg. 2

¹³[120] pg. 2

of coherence is shown in 6.1 below. The left side of the figure presents a false, positivistically derived notion of fractal coherence that we explicitly wish to avoid, while the right falls in line with the spirit of Law’s (and many other philosophers of science’s) claims about the problems of an “objective” reality.¹⁴ In the “incorrect” coherence portrait, we see an object sliced in various ways that align with specific historical presentations. This is a false view because it pre-supposes an already existential object about which we are doing the slicing. In the post-structuralist view, and emphatically in Law’s quote above, there is no center, no external “objective” thing about which we are constructing a history. Instead, as shown on the right, our layers of investigating build to reveal the contours of something greater, but that remains permanently out of reach. Adding further perspectives only provides deeper contours, exposing them *fractally* since each new cut only extends and complicates the picture while also adding to it. We are not attempting to *fill in* all the slices, as the “false” coherence on the left assumes, but *filling it out* for further inquiry. Its shape changing, expanding and contorting to fit with each new view or analysis.

We see shades of an argument toward fractal coherence taking form at earlier points in this thesis. Both the discussions of ontological enactment — the reconciliation of the different configurative constellations that support any ontological subject — and constitutive intertextuality — the combinatorial fabric of expressive forms that loom new discursive surfaces — function as ways to provide new views on software objects, and means for the representation of new histories. Any new piece of documentary evidence or additional vector for exploration provided by new tools or discursive forms supports a further (fractal) addition — *not divi-*

¹⁴Recall the discussion of computer game platform in Description about the nature of a platform’s ontological position, and how it shifts based on who is using it and to what end. Others in the history of science to confront this post-structuralist arrangement include, [60], [154], [155], and [230].

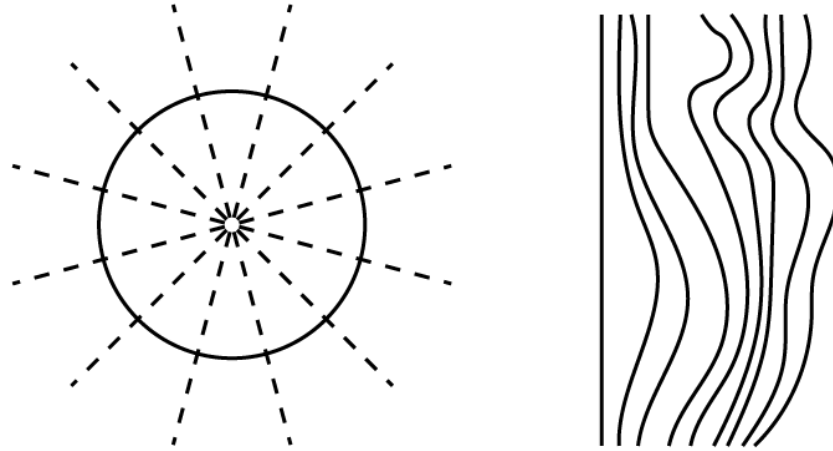


Figure 6.1: Two versions of analysis. “Incorrect” slicing of pre-defined object on the left, topographical, additive contours on the right.

sion — to the object question. The task before us is how to approach the multiple possible histories of computer game software, and the different views one can take on a software object.

6.3 A Model of Game Software Historical Study

Our model arose from experiences with the historical documentation available about game history, and as a reaction to various schemes proposed by game scholars and computer scientists attempting to analyze the meaning and effects of games and other software. This section will describe the model and its implications in light of the above notion of fractal coherence for a historical object. When convenient, it will also contrast the historical goals of the model with the analysis provided by other layered models used in game studies. These “other” models, as we will see, are distinctly ahistorical, derived as they are from literary criticism and other disciplines concerned with textual meaning making and hermeneutics — as opposed to the diachronic employment of historical narrative.

The model's intended effect is to bring attention to ways we could be presenting or approaching the history of software that are currently ignored, or unavailable due to a lack of historical foresight by practitioners and therefore, lack of access for scholars. Finally, the model below, in pulling from our notion of a fractal organization of historical viewpoints, is not intended to be THE model of game historical investigation but one of many. It intends to open up a conversation about the historical discussions (in our case mostly technical ones) that we are not having about games because of a lack of resources and scholarly focus.

Since the birth of game studies, there has been a distinct fascination with the ways that games express meaning. This is usually confronted through a consideration of the how the underlying technical constraints of a system limit the aesthetic and game play possibilities of a finished game object. In 2002, Lars Konzack published an article outlining seven potential layers of game analysis.¹⁵ While the organization of these layers and their basic ontology are a bit confused, they do begin the construction of a view of games as the result of different layers of abstraction building up from a base "platform" and arriving, at the top, as a "socio-cultural" context. Although the relationship between the layers, or why they are distinct or necessary, is left out of the discussion, their basic organization positions the technical infrastructure below the generation of "meaning" which arises at a higher point in the stack. This vertical take on the analysis of com-

¹⁵[110] Konzack, Lars. "Computer Game Criticism: A Method for Computer Game Analysis." In CGDC Conf., 2002. Konzack presents a seven layer schema: "hardware," "program code," "functionality," "gameplay," "meaning," "referentiality," and "socio-culture." This organization is ontologically muddle, and presumes that the meaning of a game exists at a specific strata of analysis, one that is also tied to the material foundation of the apparatus, its code, and its socio-cultural context. Jamming all of this into a hierarchical model is very limiting because it appears to argue that the meaning of a game exists below its socio-cultural context, and also masks the contributions of lower levels, like platform and code, to higher level structures. Each level is imbricated with socio-cultural context and each has specific contributions that contribute to referential structures, functionality, and gameplay. This hierarchy, as a layered structure, does not hold together, but as shown, loops back on and into itself.

puter games, as a holistic consideration of “all” the intermediate layers between platform and culture is not particularly convincing, but did begin the ball rolling towards similar layerings with the same goal. A goal that, if we consider the above discussion of post-structuralist description, is not actually reachable.

Nick Montfort, in an early work of what would become the platform studies field, takes Konzack’s framework and cleans it up.¹⁶ He removes numerous intermediate layers describing the meaning generation of games (“functionality,” “game play,” “meaning,” and “referentiality”) and collapses them into a “game form.”¹⁷ The “interface” and “reception and operation” layers sit atop it. Critically, he retains Konzack’s base layers of “platform” and “game code”, with the latter collapsing both the “code” and “program” into a single unit. This is immediately problematic, because a program is not its code, and from the perspective of our coming historical investigations, code and program have ontologically distinct documentation and evidence. Regardless, the other, perhaps more important move in both of these frameworks (and one we will be avoiding) is the shift from the underlying technical layers, through a “meaning” or “game form” that is then the game “concept” through which a player interfaces or receives it, and through which socio-cultural considerations come into play.¹⁸ To be fair, Montfort’s hierarchy is couched in a desire to look into the context of each layer, and to point out that “certain levels that critics have neglected or glossed over can be important to understanding games, and [those] levels can be usefully explored by scholars.”

These hierarchical views of game analysis cash out into the “platform studies”

¹⁶[157] Montfort, Nick. “Combat in Context.” *Game Studies* 6, no. 1 (December 2006). <http://gamestudies.org/0601/articles/montfort>.

¹⁷We do not go into detail about Konzack’s layers for reasons that will be made apparent below. Additionally, the organization of layers in Konzack’s work appears to be rather arbitrary and is not worth further scrutiny beyond their initial presentation of the vertical abstraction of game meaning and analysis.

¹⁸This top layer is potentially commensurate with our notion of the pre-supposed concept of a game highlighted in the Citation chapter.

series, which are framed as historical works, but rarely engage in diachronic discussion of how the specific platforms — the base layer from which all meaning arises per Montfort’s model — developed. Nathan Altice’s work is a notable exception here, as it mildly attempts to relate the development of the NES’s hardware to historic developments like the company ethos of Nintendo, and the economics of contemporary hardware markets. However, in general, because platform studies, and even software studies in general, are derived from literary criticism, narratology, and fields focused on divining and explaining “meaning” without recourse to historical narratives, most “historical” discussions of games end up focusing on vertical, synchronic analysis instead of on diachronic change or documentation.

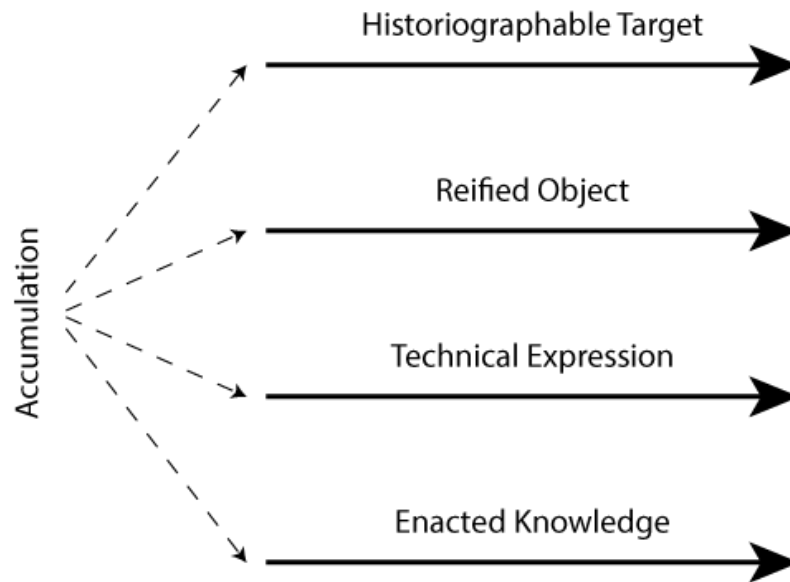


Figure 6.2: Basic model with four layers: historiographable target, reified object, technical expression, and enacted knowledge

Our model acknowledges this vertical, synchronic approach toward game analysis, but embeds it in historical processes of diachronic change and documentary accumulation. We also avoid the sticky notion of “meaning” or “form” arising at

any particular layer, because from the perspective of our model, all of that stuff begins (and derives from) the surface atop a game’s technical infrastructure.¹⁹ Someone needs to be able to play a game for it to have intentional meaning. The top level of our model is the “historiographable target” that *is* most analysis of games. This is the presupposed concept of the game when mentioned in passing, or functioning at a point in a particular chronology of game history. It is what you might imagine when asked to think about playing the game *Pac-Man*. You are probably not thinking about *Pac-Man* on a particular type of arcade machine or home console, but of a general form of *Pac-Man* abstracted out from a particular, historical context of play. However, even if you are thinking about playing the game on your mobile phone; another “you” might be imagining *Pac-Man* beaming from a television foregrounded by an Atari VCS in your living room in 1982. Both instances are *Pac-Man*, and many discussions of game design, game play, and intentional meaning might function exactly the same when applied to either of them, but they are definitely different historical objects. The historiographable layer of our model simply points out this general lack of acknowledgment of the difference between a game as a concept that functions in history, and the game as a particular instance encountered by the historical researcher.²⁰

Below the historiographable layer is the reified object. Again, this layer is derived from a class of documentation consisting of game programs and the collection of files and assets that compromise their “closed” and distributed states. Here we encounter the different versions of particular games before they are subsumed into

¹⁹This is not saying that lowers levels of technical apparatus, like code and platform do not contribute to meaning or should not be studied, but that assigning this meaning is only something one can do retroactively after documentation of the game, or experience with the game itself becomes possible.

²⁰Cf. [125] Link, David. *Archaeology of Algorithmic Artefacts*. First edition. Minneapolis: Univocal Publishing, 2016, for David Link’s discussion of the encounter between technical reproductions of cryptographic ciphers versus their narrativization from documentary sources that have not been re-enacted.

the general objects of the historiographable layer. The objects are “reified” in that they are the result of processes at lower levels. This is an acknowledgement of the transition that occurs between the apparatus and outputs of development activity, and compiled (and playable) game objects. When dealing with, as we have in previous chapters, the complex nature of identifying and tracking different versions of games, this layer actually becomes the primary one from an archivist’s point of view. The reified objects are the things that actually sit in waiting for historians in the future, and that are the target for various reproduction schemes like emulation or expression in original physical hardware.²¹ Importantly, this layer also focuses on the explicit configuration of these different objects. That is, their file organization and structure, and how they express their differences in execution, both of which are paramount to archival and stabilization concerns. That prominent historical research related to versioning differences arises from archival research projects is then understandable. The most notable example is probably the Preserving Virtual Worlds project, which explored the version histories and multiple manifestations of numerous games, including a particularly deep look at the genesis of one of the earliest text adventures, *Adventure*.²²

²¹One potential question for this model is where to place the typical, experiential differences of games as manifest in their embodied play. That is, where are platforms, peripherals, and the like? Well, when discussing a particular reified version of a game, those things are implicit to it. The reified work is both a collection of data marked as a playable, distributed experience, and the apparatus required to play them. Also, given our fractally additive view of things, we cannot argue against including more granular distinctions between versions and their expressive apparatus as they would just be one more angle of inquiry.

²²The PVW team worked with Dennis Jerz to uncover significant new intermediary versions of *Adventure*, created by Don Woods and Will Crowther in the mid-to-late 1970s. For more on this work see [99] Jerz, Dennis G. “Somewhere Nearby Is Colossal Cave: Examining Will Crowther’s Original ‘Adventure’ in Code and in Kentucky” 1, no. 2 (2007). <http://www.digitalhumanities.org/dhq/vol/001/2/000009/000009.html> and [148] McDonough, Jerome, Matthew Kirschenbaum, Doug Reside, Neil Fraistat, and Dennis Jerz. “Twisty Little Passages Almost All Alike: Applying the FRBR Model to a Classic Computer Game” 4, no. 2 (2010). <http://www.digitalhumanities.org/dhq/vol/4/2/000089/000089.html>. Another example of an explicit focus on diachronic version change is found in Altice’s *I AM ERROR* [25], in the discussion of the improvements and upgrades to NES cartridges capabilities and its evolution as a platform over time. His analysis on this note in a platform studies work

As “reified” objects exist on their own strata, the things they are reified from must sit somewhere below. We refer to these things as “technical expressions”. This layer is the place for all of the productive outputs of a group or individuals engaged in the construction of software before it coalesces into a reified object. The outputs are the evidence and products of development, like source code, development tools, and development documentation. Basically, anything that one would find in the appraisal of a development studio’s digital and physical records.²³ Technical expressions are the evidence of process, and the first site where the experience and actions of a developer manifest what will become a game.

The base layer of the model is “enacted and tacit knowledge”. This is the set of *a priori* experiences and personal knowledge that people bring to the craft of software creation. Before interacting with or developing on a platform, before writing any code, and before ideation of new objects begins, we simply have the accumulated experiences and knowledge of the people who will take action. It is also the place where ideas about computation and game design first arise. For example, if one wants to program a binary search of game entities, or organize a collision detection routine, they must draw on previous knowledge that those algorithms and approaches exist. Or conversely, if they invent or re-invent some approach, that invention is still the product of their accumulated knowledge on the subject. At this level we can also analyze the implementation of these ideas and how they end up as tangible technical expressions. By “tangible” we mean things that can be cataloged and recorded as historical evidence and then retrieved or reproduced in the future. Evidence at this layer can also be recorded, but it will be of a more general sort. Instead of the technical expressions that lead to a software

is actually unique, since the other volumes are less concerned with diachronic process.

²³See the Appraisal chapter for a recounting of these effects, along with its complementary report, [102] *A Unified Approach to Cultural Software Objects and Their Development Histories*.

object, the evidence would be the other sources of inspiration and learning that contributed to a game’s ideation.

Tacit knowledge is a significant topic in the history of science and technology, and discussed in the introduction to this thesis as it relates to technological “black boxes,” the set of objects or basic scientific and technical assumptions that pass without question into a technical discourse. In the model, these black boxes are both the “reified objects” that use compilation to hide the technical expressions used in their creation, and also the tacit knowledge — the unacknowledged assumptions of practice — incumbent to game development and creation that are often left out of discussion.²⁴

Pulling an example from the *Doom* discussion below, we can take the development of *Doom*’s spatial rendering algorithm, which relied on a technique known as binary space partitioning (BSP). The inner workings of the algorithm with respect to *Doom* will be addressed below. Here, the only thing to know is that it was an approach to real-time graphics implemented by John Carmack, the *Doom* engine’s developer. Carmack drew upon knowledge from the graphics community of the day and found a way to implement BSP tree-based rendering by combining those approaches with his own thorough knowledge of MS-DOS based machines’ memory architecture. When Carmack programmed his BSP renderer and its attendant BSP compiler in C, he then manifested his previous knowledge into a technical expression of it that functioned in service of his particular needs for

²⁴Work on the revelation of the tacit and implicit assumptions of technological and scientific practitioners is of extreme importance when trying to figure out the “why” and “how” of a field’s development and change. See [54] Collins, Harry. *Changing Order: Replication and Induction in Scientific Practice*. Reprint edition. Chicago: University Of Chicago Press, 1992. [55] —. *Tacit and Explicit Knowledge*. Reprint edition. Chicago; London: University Of Chicago Press, 2012. for more discussion of the role of tacit knowledge in the efforts of science and technology. Another good discussion can be found in [169] Polanyi, Michael. *The Tacit Dimension*. Terry Lectures 1962. Garden City, N.Y: Anchor Books, 1967, who describes tacit knowledge as what “we know but cannot tell.”

Doom. This BSP approach was then compiled into the *Doom* executable — as a reified object — and distributed to the world — as a historiographable target. The accumulations at each layer would then be: for enacted knowledge, records of contemporary publications in BSPs and DOS memory architecture; for technical expression, the source code of the game implementing BSPs; for the reified object, the *Doom* executable making use of BSP; and for the historiographable target, records of that reified object as interpreted and experienced by players.

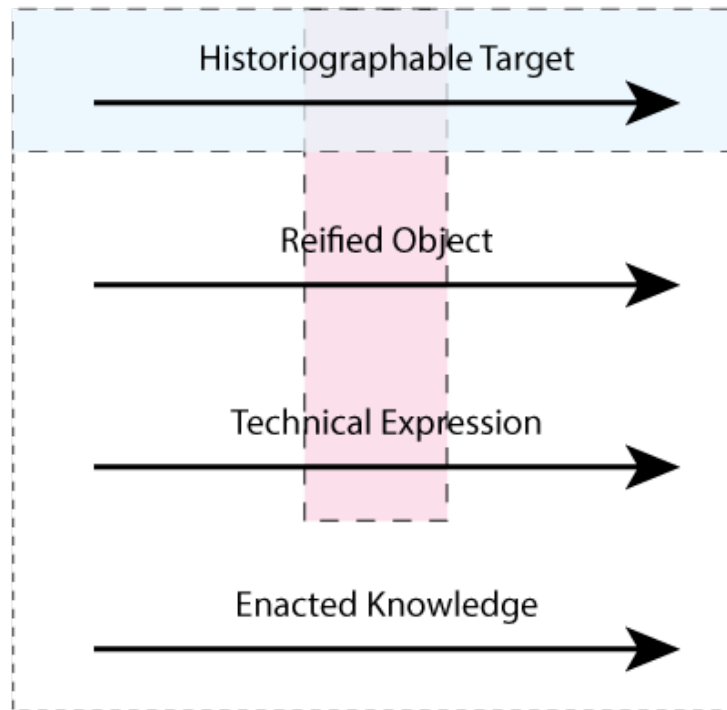


Figure 6.3: “T” in model, most practice is in blue top layer, with synchronic investigations in pink. The white space is still waiting to be filled in more thoroughly.

The model also aims to point out the gaps in the record and study of games. Each layer can be thought of as a diachronic progression from left to right. For

the topmost layer, this is representative of most game historical discussions that focus on games as conceptual historical entities. The horizontal box in the image below highlights this region of current game historical investigation. Likewise, deeper, synchronic, vertical studies of games that attempt to derive and explain meaning from base principles through critical reception appear as vertical boxes in the model. The striated spaces outside of the “T” formed by both common types of analysis are the spaces we want to point out and populate. This is the diachronic history of each lower layer, like (1) the progression of versions and file structural changes of reified software objects; (2) the changes incurred in the source code and tools used over the course of development; and (3) the history of algorithms, architectures, platforms, and knowledge propagation in the development community. Also significant is that these layers are not enforced with regards to the exact types of documentation, but with their function toward the object or conception of an object in question. For example, we could shift the discussion from *Doom*, to the idBSP program — Doom’s BSP compiler. In making this shift, idBSP becomes the object of study instead of historical evidence of *Doom*’s technical expression. Consequently, the Doom program is now positioned as an *a priori* form of enacted knowledge functioning on the creation of idBSP. The history of *Doom* then becomes important to the process of idBSP’s creation instead of the other way around. We could then use *Doom*’s slow speed before version 0.5 as the impetus of idBSP’s ideation and development, as opposed to positioning idBSP as a critical development in the history of *Doom*’s technical expressions and reified objective versions.

Finally, and complementarily, historical revelations in the lower layers cannot be adequately explored without new organizational and analytical approaches, and cannot be stabilized without the hard, meticulous work of description and

cataloging. Luckily, for many popular games, like *Doom*, the community has taken up these organizational tasks, and worked to maintain the evidence of its various layers. This is a major reason that *Doom* was chosen for analysis in the next section. Community-collected evidence about *Doom*'s creation, development and evolution makes it possible to have a discussion about the progression of *Doom*'s history at each layer, and to support arguments about them.

Regarding the exploration of the documentation at each layer, we are still in the early stages of robust approaches to the analysis of software executable, source code, and other evidentiary tracks that contribute to a historical coherence of software. Some of the previous work in this dissertation is a step in that direction, but much more work needs to be done. In framing software history through the above model, we hope it can be a guide to different sites of intervention for computational tools, or more considered methodologies for the evidence of software history.

6.4 *Doom* in Fractal Coherence

This section presents a small collection of different historical arguments about *Doom* motivated by the layers — and lacunae — highlighted in the model above. Each argument articulates the practices and material-methodological considerations of its respective fractal layer, and serves as an example of the larger class of historical perspectives that can fill out the spaces outside of the “T” of current software histories. Recall that in describing the model we noted that most historical study of game software is preoccupied with a “T” shaped area of historical concerns. The horizontal top of the “T” being general histories of games unconcerned with treating them as anything other than “closed” conceptual objects and the vertical body of the “T” occupied with synchronic studies of the layers

of system abstraction. The intention, in these rather busy sections, is to portray a smaller example of a fractal perspective on *Doom*, and to embed our practical methodological findings right in line with the historical investigation undertaken. Therefore, each discussion is as much a historical argument about *Doom* as it is an argument about uncovering and recovering *Doom*'s history. This secondary focus on “uncovering and recovering” aligns with the model's express focus on the accumulated — and often invisible — conditions of each layer; for one cannot describe, reference or retrieve a history if it's not there in the first place.

The model lays out four different layers for material argumentation and investigation: enacted and tacit knowledge, technical expression, reified object, and historiographable target. Clothing these layers in the vestments of *Doom*'s particular accumulations, we end up with the following alignments:

1. Enacted and tacit knowledge → Previous knowledge of game design and development including, critically, the *a priori* knowledge of algorithms and platforms in enough detail to create new, previously unseen game forms
2. Technical expression → The material supports for *Doom*'s creation and proliferation over the last three decades, the journey of its assets and source code
3. Reified object → The game object itself, the executable organization of data that players interact with, and how that object functioned and morphed over time
4. Historiographable target → The game *Doom* as referenced in histories or most recollections. This is the only layer to have already been exposed to historical scrutiny

As a single chapter is not nearly enough space to cover all of these topics in complete detail, we will only address a brief cross section of the history and evidence at each layer. We then reduce each one to more specific contributions:

1. Enacted and tacit knowledge → *Doom*'s effect on and positioning in discussions of game programming during and after release
2. Technical expression → The procession of *Doom*'s source code as a means for defining the contextual needs for source documentation
3. Reified object → A tour of *Doom* versions and modifications in MS-DOS that points toward more careful considerations of what file contents and structure can reveal about development history
4. Historiographable target → *Doom*'s current historiography, its main sources, limitations, and accumulations

For those unfamiliar with *Doom*, it was a computer game first released in version 0.99 on December 10th, 1993 by id Software, a small company working, at the time, out of a Mesquite, Texas office suite with 8 employees (John Carmack, John Romero, Shawn Green, Kevin Cloud, Adrian Carmack, Jay Wilbur, Dave Taylor, and Sandy Petersen). The game is considered a progenitor of the first person shooter genre.²⁵ It also had a significant amount of features that are now commonly associated with that genre and games as a whole. These included local area network (LAN) multiplayer, downloadable content, company sanctioned user modifications (commonly referred to as “mods”), and wide spread Internet distribution. *Doom* is played from the first-person perspective of a space marine

²⁵[207] Therrien, Carl. “Inspecting Video Game Historiography Through Critical Lens: Etymology of the First-Person Shooter Genre.” *Game Studies* 15, no. 2 (December 2015). <http://gamestudies.org/1502/articles/therrien>.

fighting demonic entities first on a space station, and then in a version of hell. The game was initially released as an Internet-distribution “episode” of nine levels. Two continuing episodes, also of nine levels, could be ordered directly from id Software.²⁶

The remaining sections below point toward new approaches and arguments that can develop by looking into the material conditions of each layer of the fractal model presented above.

6.4.1 Historiographable Target

In order to show how our model extends the historical space of *Doom*, we first need to recount the work that has been done, and how it fits in and fills out the model. In the case of *Doom* monographs, only three books spend a significant amount of time with the game. Of those, two — King and Borland’s *Dungeons and Dreamers* and David Kushner’s *Masters of Doom* — are journalistic accounts of the game’s development that predominantly focus on the character (and “caricatures”) of its lead developers. The other, Dan Pinchbeck’s *DOOM: SCARY-DARKFAST*, is a more vertical study of *Doom*’s gameplay and level design. In all cases, these works rely, primarily, on either an analysis of *Doom* itself — with only Pinchbeck’s work mentioning the particularities of *Doom*’s versions — or on interviews with the game’s developers at id Software. They are not particularly deep histories. The journalistic accounts eschew interpretive analysis for more sensationalist claims about the game’s impact and importance, and Pinchbeck’s work straddling the line between academic treatise and fan appreciation.

Now, in fairness, none of these texts claim to be rigorous historical works, but in the larger community they are interpreted as such. A look at the Wikipedia

²⁶The “nine” levels include each episode’s secret bonus levels: E1M9: Military Base, E2M9: Fortress of Mystery, and E3M9: Warrens.



Figure 6.4: Opening of *Doom*

page devoted to *Doom*'s development history ²⁷ is almost entirely sourced from Kushner's *Masters of Doom*, which is problematic because many of his claims about the game's popularity, reception, and effects on larger game culture are presented without much in the way of sourcing, and in some cases apparently willful inaccuracies.²⁸ *Dungeons and Dreamers* is primarily focused on game developer Richard Garriott and his *Ultima* series of games, but does feature two intermezzo chapters devoted to id Software.²⁹ Pinchbeck's work is a bit more puzzling, given that its central thesis — essentially that *Doom* is “awesome” and groundbreaking — is under cut by a final chapter that in trying to position *Doom* as a significant prototype for the first person shooter ends up convincing itself

²⁷[19] “Development of Doom.” Wikipedia, June 6, 2017. https://en.wikipedia.org/w/index.php?title=Development_of_Doom&oldid=784107014.

²⁸In one particularly egregious example, Kushner seems to misquote a magazine article to support his assertion that the gaming press immediately identified *Doom* as a significant historical development, which as we will briefly discuss below, was not the case. Quoting Kushner, “Early reviews echoed the gamers’s glee. PC Week called *Doom* a ‘3-D tour de force.’ Compute said it signaled a new era in computer gaming: ‘The once-dull PC now bursts with power. . . . For the first time, arcade games are hot on the PC. . . . the floodgates are now open.’” [113] pg.161 The quote in COMPUTE! is an introduction for a section about action games on the PC and not directly about *Doom* or its specific influence. *Doom* is mentioned in passing after a description of *Wolfenstein 3D* as “a game that features more involved play mechanics and dazzling special effects.” [140] However, there is no further mention of *Doom* aside from that sentence, and nothing noting it being more significant than any other game listed.

²⁹While there is a mild relationship between the two endeavors, with *Doom*'s texture mapping work reputedly motivated by *Ultima Underworld*, a game technically more “advanced” than *Doom* but released before it. The inclusion of these two chapters is a bit puzzling, and apparently unexpected since they are not mentioned as sources for *Doom*'s development history on the DoomWiki, which is otherwise impressively comprehensive.

otherwise.³⁰

The more important point to make about *Doom*'s historiography is not that efforts to date have a litany of shortcomings, but that extensive material exists to build a much more nuanced and detailed history. Regardless of the level of rigor, the three monographs mentioned do contain a majority of the first-hand information available about *Doom*'s development.³¹ All three relied, in part, on extensive interviews with the development team, with Pinchbeck's explicitly including and highlighting the contributions of id developers besides the two more famous Johns. What is not significantly included in any of the works is the material outlined in the brief layered case studies below, nor any real analysis divining the reasons behind *Doom*'s longevity as both a game and a community building object.³²

The rest of this section will lay out a brief history of the history of *Doom*, and speculate on the reasons behind its notoriety and continuing significance. Historical material about the game is primarily available online, and split into different sources based on the state of information technology available to the community at various points in time. That is, the growth of historical information about *Doom* also follows the growth of the Internet as both an archive and mediator of

³⁰Quote from *Doom:Scarydarkfast*[168] pg. 155-156:

Interestingly, in DOOM, we see the core affordances laid down for the genre (or reaffirmed really, as the template was essentially there from Maze War), but in terms of the experience of gameplay, all of these small incremental nods to strategy and planning mean that the vast majority of contemporary FPS games deviate from DOOM's flavor in subtle but significant ways. Despite DOOM's position as the mother lode of first-person shooters, when we consider the impact of these additional affordances as having the effect of making play slower and more complex, the spirit of Ultima Underworld: The Stygian Abyss rears its head once more.

³¹Except for id Software's own *Book of Id*, released as a companion volume to 1996's id Anthology PC collection release [15].

³²Pinchbeck does talk at length about *Doom*'s level design and gameplay innovations, including how *Doom* influenced a raft of successor first person shooters, but he's does not mention much in the way of historical methodology or touch on much of the information elaborated on below.

historical information. In *Doom*'s case, the major sources for its history are linked to specific epochs of online information dissemination, namely: Usenet and BBS systems, web “portals” and fan sites, and most recently, unified, user editable wikis.

Doom's Release and Newsgroups

Even before *Doom*'s release in late 1993, excitement about the game had been growing since its announcement in a press release on November 5th 1992.³³ Tom Hall, one of id's game designers, had written the text of the press release in a design document known as the “*Doom Bible*”. Though most of the information presented in the *Doom Bible* did not make it into the final game, it is apparent that the ambition at work inside id was impressive. Hall noted that they expected *Doom* to be “the number one cause of lost productivity in the world” when the game was released, and routinely referred to *Doom* as, “heralding another technical revolution in PC programming.” The press release is interesting in that it announced a significant number of features and goals for the game that, by that point, had not remotely been implemented. Hall notes that the game's texture mapping, “a technique that allows the program to place fully-drawn art on the walls of a 3-D maze”, would greatly surpass that of *Wolfenstein 3D*, a previous id game. Hall, it seems, did not anticipate the coming complexity of *Doom*'s level design, or rather, its attempts to “break out” of *Wolfenstein 3D*'s design as later noted by John Romero. Although *Doom* would have a maze-like feel, the vari-

³³Confusingly, this announcement is noted in multiple places as occurring on January 1st, 1993. Checking out the various PC game USENET groups at the time, we can easily locate the announcement from November 5th, 1992 by id's Jay Wilbur on comp.sys.ibm.pc.games [224]. The earliest mention of *Doom* actually predates that announcement by a month and a half when Jay Wilbur commented on *Doom*'s apparent multiplayer functionality [225]. Based on various other postings around the time [1], *Doom* appeared as a scheduled release from Apogee Software before its official announcement, leading to speculation, and presumably some responses from Wilbur to quell inaccurate rumors.

ety of its environments, including what may be the first outdoor arena-like 3D level, show that the maze metaphor would give way sometime between the game’s announcement and release.³⁴

The game’s community existed primarily based on good will from previous id releases. Starting with *Commander Keen* in 1989, id had consistently delivered some of the fastest gaming experiences available on DOS-based computers.³⁵ At the time, Keen’s tile-mapped side scrolling engine provided for gameplay more akin to contemporary console and arcade titles.³⁶ id produced seven Keen games before branching out into 3D. John Carmack, the driving force behind id’s game engine technology, sought to bring arcade-like speed and viscerality to a platform mostly used for contemplative, turn-based role playing games. After id’s founding in February of 1991, the team continued making games on contract for their former employer SoftDisk. While the majority of the team worked on sequels to older titles, or new games built on the Keen and other legacy codebases, Carmack spent February and March of that year building id’s first 3D engine.

The resulting game, *HoverTank 3D* (Figure 6.5) employed a basic ray-casting technique to render three-dimensional walls from an underlying two-dimensional grid. Carmack significantly improved on his engine’s rendering technology in id’s further three-dimensional efforts *Catacomb 3D* and *Wolfenstein 3D*. Another company, Raven Software, also contracted John Carmack to help with an improved

³⁴Episode 3 Mission 6 E3M6 “Mt. Erebus” is noted by Pinchbeck [168] as possibly the first outdoor arena-like level in shooter.

³⁵DOS is the Microsoft Disk Operating System (MS-DOS). While the developers and players at the time referred to *Doom* as a “DOS”, an “IBM PC” or simply a “PC” game, we need to be a bit more careful. Any reference to PC or DOS in this section is then explicitly referring to versions of MS-DOS available on compatible x86 architecture available between 1992 and 1994.

³⁶The first use of the Keen sidescrolling technique was in *Super Mario Bros. 3* knock off called “Dangerous Dave in Copyright Infringement”, made to impress John Romero at SoftDisk before id’s founding. “Dangerous Dave” was a game character John had created as a teenager, and the character had featured in numerous early collaborations between Hall, Carmack, Romero, and (Adrian) Carmack.



Figure 6.5: Opening of *Hovertank 3D*

Wolfenstein engine for their game Shadowcaster. That game featured fully texture mapped floors and ceilings (something *Catacomb 3D* and *Wolfenstein 3D* lacked), as well as sloped terrain (which did not even make it into *Doom*'s engine). All this is to say, that by the time of *Doom*'s announcement, many were familiar with id's 3D development chops, and excited for their next game.

After *Doom*'s initial announcement, information trickled out from id and the press slowly over the next year. The first alpha of the game was ready sometime before February 4th, 1993. Other early versions circulated amongst id's friends and acquaintances followed by a release of *Doom*'s version 0.4 to various press in April 1993.³⁷ A further press release beta circulated in October and was subsequently leaked onto the Internet. Immediately, various community members began reverse engineering its files, including the WAD file format that stored and indexed most of the game's level data.

The alt.games.doom newsgroups appeared in early January of 1994 — following *Doom*'s early December 1993 release — as a means of consolidating the significant amount of *Doom* posts filling up others, like comp.sys.ibm.pc.games and bit.listserv.games-l.³⁸ Topics included issues installing the game, finding all of its secret areas, recordings of gameplay (through .lmp files) and picking apart *Doom*'s engine and data structures. Additionally, the team at id Software consistently updated *Doom* over the course of 1994. The newsgroups functioned as a clearinghouse for info on the latest versions of the game, and on the progress of community modding tools.

Alongside the newsgroups, a proliferation of *Doom* and id focused File-Transfer

³⁷While not explicitly stated as version 0.4, a preview of the game in the July 1993 issue of Computer Gaming World [126] features screenshots of the game that only appear in the 0.4 alpha.

³⁸comp.sys.ibm.pc.games also had a sub-group devoted to “action” games, which is apparently what most people classified *Doom* as at the time.



Figure 6.6: Opening of *Catacomb 3D*



Figure 6.7: Opening of *Wolfenstein 3D*

Protocol (FTP) sites came online. The initial release of *Doom* significantly taxed the University of Wisconsin-Parkside's server, and numerous mirrors appeared with hours of *Doom*'s release to help alleviate the congestion caused by thousands of people trying to download *Doom*'s initial shareware version. In April 1994, after the first raft of *Doom* design tools arrived — and with them a flood of new *Doom* levels — Barry Bloom posted to alt.games.doom that he was starting a unified ftp server for all community outputs.³⁹ In short order, this initial ftp server evolved into the idgames archive, a community driven and maintained archival collection of all id Software-related materials floating around the Internet. Although the level of technical ingenuity and commitment that followed the release of *Doom* might appear odd, one needs to remember that at the time, the primary group with access to early Internet connections were students and professionals at universities. Most of the messages on *Doom*'s forums and the ftp servers hosting *Doom*'s content were in the .edu namespace.⁴⁰ These were computer scientists, electrical engineers, and programmers who appreciated the technical feat of *Doom*, and who also possessed a deep connection to the networked environment that spawned it. Bear in mind that *Doom* never had an official, commercially boxed release, but was distributed as a shareware demo. id gave other companies explicit permission to package and resell their shareware release at a profit because it drove players toward ordering the rest of the game's levels directly from id.

The idgames archive slowly expanded over the next two years, with an announcement in mid 1996 by one of its archivists, Frans de Vries, that it had collected over one gigabyte of id related materials.⁴¹ The archive originally consisted

³⁹FTP consolidation is discussed in this thread from April of 1994 <https://groups.google.com/forum/#!msg/alt.games.doom/Wjp0gf-zBf8/YIYsTQyD9F4J>

⁴⁰David Datta posted a list of all *Doom* mirrors shortly after the games release. [61] Datta, David. "Doom Upload Site List (Final Version) - Google Groups." Accessed June 10, 2017. <https://groups.google.com/forum/#!search/david%20datta%20doom%7Csort:relevance/comp.sys.ibm.pc.games.action/tT3ptNF4qN8/nkrU0is1GjEJ>.

⁴¹Frans de Vries outlined the history of the archive in an announcement posting to the

mostly of modified *Doom* WAD files and demo replay LMP files, but expanded to accept content from all *Doom* engine games in February 1995 after the release of the John Romero produced, Raven Software developed game Heretic. The /idgames root folder name derives from this expansion. The archive also features historical versions of *Doom* and related games in the “historic” folder of the archive. Here community members have collected numerous, previously private alphas of *Doom*, along with historically significant tools and documentation from *Doom*’s release. The archive is maintained through a collection of mirrors, and still sees community activity. Members even managed to locate a “lost” *Doom* alpha, version 0.3, in August of 2015.⁴²

Along with the collection of ftp servers that would become the idgames archive, a community of fan web sites began appearing in mid-1994. The largest at the time was Piotr Kapiszewski’s DoomGate web portal. It hosted links to other sites in the DoomWeb circle and provided downloads for popular Doom mods. DoomGate (now hosted at www.gamers.org) was also the web outlet for the Doom Help service, a community collection of FAQs and resources originally linked to the alt.games.doom.newplayers newsgroup.⁴³ The alt.games.doom newsgroups then re-consolidated, moving to rec.computer.games.doom in late 1994 and taking over the newsgroup end of the service. DoomGate is also the oldest remaining mirror of the idgames archive, having taken it over in 2001 after Walnut Creek CDROM (an earlier shareware and ftp distribution company) decided to stop hosting community archives on their servers.⁴⁴

The early activity of the *Doom* community, specifically in regards to its focus

rec.computer.games.doom lists [64].

⁴²[5] “DOOM Alpha v0.3 (aka ‘DOOM Pre-Alpha’) - Doomworld /Idgames Database Frontend.” Accessed June 10, 2017. https://www.doomworld.com/idgames/historic/doom0_3.

⁴³As noted, the DoomGate (www.gamers.org) website is still online, though it has been superseded by other community sites, mostly DoomWorld, which we will note in the next section.

⁴⁴[63]

on historical documentation, is a boon to historical researchers. Whereas most other games from the period exist solely as executable files with some additional physical documentation, the evidence from the *Doom* community’s effort makes the below source code and game version reconstructions not only possible, but potentially playable.

Source Code and DoomWorld

The next phase of *Doom*’s historical accumulation begins with the release of *Doom*’s source code by id Software on December 23rd, 1997. Researcher Bernd Kreimeier, having provided one of the first academic analyses of *Doom*’s engine architecture decided to begin research on a full length monograph.⁴⁵ He approached id asking for a public release of the code to complement his book. id allowed him access and he worked to tidy up the code for release. This was not an insignificant effort as *Doom*’s original source tree contained numerous versions of files over a mess of folders, including some copyrighted sound driver code that had to be replaced entirely.⁴⁶ While in the midst of the clean up his publisher cancelled the project, citing *Doom*’s age — an eternal three years old at the time — in doubt’s that there would still be profitable interest. Regardless, Bernd decided to finish the code clean up with John Carmack’s help and released it.

The source code’s availability triggered a flurry of programming activity, as players worked to port the game’s code to a large number of platforms. Since the game’s source was released for the Linux operating system, it required that the player programmers change the code to make it “port”able and allow for execution on different system. The first port back to *Doom*’s original DOS platform,

⁴⁵[77] Forsman, Robert and Bernd Kreimeier, “A Brief Summary of DOOM-style Rendering.” 1996.

⁴⁶At the time id did not use a version control system for their files, preferring to just make new copies or overwrite older work. (Correspondence from John Romero.)

DOSDoom 0.1 by Chi Hoang, appeared within 24 hours. This formed the base for an extensive number of DOS ports, some of which slowly migrated to multi-platform Microsoft Windows, Linux, and Apple Mac OS ports. The source code's expansion will be described in more detail below.

Shortly after the source release, in March 1998, the DoomWorld site and forums came online, and became the dominant location for *Doom* community news. DoomWorld also hosted a significant collection of *Doom* source ports and design tools for a variety of architectures. Shortly after launch the site ran a 5th anniversary *Doom* retrospective. The celebration saw the online publication of Tom Hall's original *Doom Bible* (re-written in HTML), and a host of interviews including some of *Doom*'s developers, the manager of the idgames archive, and prominent modders and players. DoomWorld also served as a unified portal for member sites devoted to different platform versions of *Doom*, as well as a hosting location for in-process source port projects. The site, in celebration of *Doom*'s 10th anniversary in 2004, began the annual Cacoawards — named after *Doom*'s Cacodemon monster — that highlighted significant mods, demos, and community service contributions.⁴⁷ DoomWorld's forums absorbed most of the traffic from the older Usenet newsgroups, and became the predominant meeting place for fans of the game as well as contributors to the modding and porting communities. DoomWorld's denizens have launched multiple collaborative probes searching for parts of *Doom* history that are known to be missing. Various earlier source ports like *DOSDoom 0.2*, the progenitor for most DOS source ports, were lost, and triggered a search for other significant versions of source ports and sites that might be in trouble.

The other major entrant into *Doom* historical recording in this period was

⁴⁷[6] "Doomworld – The 11th Annual Cacoawards." Accessed August 24, 2016. <http://www.doomworld.com/11years/>.

Doom's developer John Romero. Responsible for *Doom*'s level design and development tool programming, John started an online presence in late 1999 at www.planetromero.com. At the time he was running an internal team at Ion Storm working on the first-person shooter *Daikatana*. He had left id Software after the completion of *Quake* in March 1996, and again partnered with Tom Hall. Hall had left id following the completion of *Doom*, and developed *Rise of the Triad* for Apogee. The game was originally intended to be *Wolfenstein 3D 2* and was based on a heavily modified version its predecessor's engine.⁴⁸ Romero's early site recounted his work at Ion Storm, and organized his personal game development history. This included synopses of every game he had made since childhood, along with collections of photos recounting those games and most of his professional career.⁴⁹ The site was migrated to rome.ro in the late 2000s, and then retooled as a personal blog, with www.romero.com now hosting his consulting business. Romero lent significant support to the growing *Doom* community, frequently popping in on the *DoomWorld* forums to offer commentary. His photo galleries on his main sites, and complementary collections on smugmug.com — a photo sharing site — and Facebook helped round out id's development history.⁵⁰ They provide a brief glimpse into the development world of id, and early 1990s software development in general. More recently, Romero donated most of his development records, including his original Apple II computer, to the Strong Museum of Play.⁵¹

⁴⁸Originally a collaboration with id Software, *Wolfenstein 3D 2* was renamed *Rise of the Triad* and stripped of potentially infringing content after id revoked Apogee's license to the *Wolfenstein* property.

⁴⁹<https://web.archive.org/web/20040103083931/http://planetromero.com:80/>

⁵⁰Romero's collections on SmugMug (romero.smugmug.com) and Facebook (<https://www.facebook.com/theromero>), while extensive, are also a prime example of an unstabilized resource. Both collections are dependent on for profit corporations without a significant archival focus. The Facebook collections are particularly frustrating, since they are view limited to non-Facebook users.

⁵¹[68] Dyson, Jon-Paul C. "CHEGheads Blog — Preserving John Romero's First Computer at ICHEG | International Center for the History of Electronic Games." Accessed June 10, 2017. <http://www.museumofplay.org/blog/chegheads/2014/08/>

Ironically, he has recently posted about trying to recover his own web pages from planetromero.com from the Internet Archive’s Wayback Machine, which happened to index the site many times before he took it offline in mid-2015.⁵²

DoomWiki

The last major addition to *Doom*’s historical sourcing, DoomWiki, was founded in 2005 by Fredrik Johansson. Housing over 4500 individual pages, the DoomWiki provides information on the original game, all of its sequels and most games based on extensions of *Doom*’s various engines.⁵³ Since the wiki is a community maintained resource, much of its information can be treated as primary source material. Source port creators frequently contribute to articles describing their own efforts, which then indirectly provide authoritative dates through page revision histories for any post-2005 community events.

Originally hosted as a part of the for-profit — but free to create — Wikia wiki network, DoomWiki’s maintainer’s became dissatisfied with the service as “an increasing focus on advertising and social networking made the site difficult to use as a knowledge source.”⁵⁴ Two prominent community members, James Haley (Quasar) and Mike Lightner (Manc), agreed to help migrate the site to a new domain. DoomWiki’s data was migrated from Wikia’s proprietary system to MediaWiki, the open source wiki software that powers Wikipedia. The full migration took a year, from October 2010 to September 2011, and the community then abandoned the old Wikia site. Although it has not been updated since 2011,

[preserving-john-romeros-first-computer-at-icheg/](#).

⁵²Romero spent the end of 2016 tracking down and recovering content from his development blog. He now provides a searchable index of his posts dating to 2002 at <http://rome.ro>.

⁵³The original *Doom* engine is retroactively known as id Tech 1 after the id Tech 5 engine release revealed an internal versioning that was then applied to previous engines.

⁵⁴[3] DoomWiki, “Doom_Wiki: Departure from Wikia,” https://doomwiki.org/wiki/Doom_Wiki:Departure_from_Wikia (accessed 26 Mar. 2017)

the DoomWiki on Wikia still benefits from Wikia’s search engine optimization measures and thus appears before the current DoomWiki in most Google searches. This is an issue as the most *prominent* historical information about *Doom* online is also deprecated and potentially outdated.⁵⁵

Conclusions

The history of *Doom*, its development and community, is still ripe for a significant, considered and critical history. The remaining three sections will partially analyze some of the sources described above. The existence of these historical sources is testament to *Doom*’s developers approach to documentation and community engagement. John Carmack and John Romero’s support of community modding, source code dissemination, and development interviews — in many cases against the wishes of id’s business development team — enabled the community archives and activities we make use of below. Romero, especially within the last five years, has devoted significant effort to promoting *Doom*’s history, appearing at multiple anniversary events, hosting a *Doom* historical postmortem with Tom Hall at the 2011 Game Developer’s Conference, and appearing in numerous YouTube playthroughs of the game. Regardless, while *Doom*’s historical sources are still available, they might not remain so forever. Much more work is needed to develop more advanced strategies for stabilizing the heterogeneous collections of documentation outlined above.

⁵⁵Dan Pinchbeck’s DOOM:SCARYDARKFAST [168], published in 2012, makes repeated references to the DoomWiki on Wikia, meaning that he was unwittingly relying on potentially outdated information. However, given the age of the original *Doom*, its unlikely that any significant information he cited was incorrect.

6.4.2 Reified Object

This section provides a brief object study of *Doom*'s executable, the relationship of that executable to other files included with the game, and the changes to that organization over *Doom*'s development history. Rarely are the versions of a game made apparent in historical discussions about it. Even rarer still is discussion of the underlying organization of the files in those versions. By tracing the addition of component files to the various versions of *Doom*, we also learn a bit about the history of its community and development. Furthermore, as the *Doom* executable is actually an abstracted game engine that processes separate game data provided through WAD files, we further problematize the discretion of versioning as a means for game identification.

Scope and “Object”

Before diving into our brief tour of the *Doom* object's history, we need clarify our scope and focus. When we mention the “*Doom* object” we are referring to a particular constellation of files that were compiled and released with the intention of being played by others. During development — as we saw briefly in the Appraisal chapter with *Prom Week* — things change quickly and break often. Many subtle changes will occur in code and assets that spend even a day with an active development team. Those pieces will coalesce into working builds — each technically a distinct version — that form a train from one released version to the next. We focus on these releases for practical reasons — primarily since intermediary development builds are not recorded and discretized — and because we want to analyze changes significant enough to warrant a version label from the developers. If something was not different enough to be noted by its creators, it is probably not different enough to affect diachronic analysis.

The “*Doom* object” is then a self-contained release of the game, either to testers or the public; one that the developers intended to share, and, in turn, to receive feedback on. That so many releases of *Doom* are recoverable is a testament to the community efforts outlined in the previous section. The idgames archive and a network of fan sites provide preserved and downloadable copies of most *Doom* versions. Their efforts made the comparative discussion below possible. The “*Doom* object” can generally be split into four distinct categories of files: (1) the *Doom* executable program — usually DOOM.EXE — responsible for running *Doom*’s engine code and displaying the game’s visuals; (2) *Doom* WAD⁵⁶ files that function as a database for creative assets (level data, sound files and bitmapped art); (3) supporting tools and programs, usually networking programs, meant to augment the game play experience; and (4) game use documentation, like manuals, patch notes, and FAQs. Each *Doom* object is then a particular collection of files stabilized by a distinct version number and the file hashes associated with it.⁵⁷

One final note on scope is that all discussion below is directed toward versions of *Doom* distributed online. *Doom*’s online releases are all shareware versions of the game. While it may sound as though these versions are deprecated or deficient in some way, they simply lack some of the complete game’s level data and art assets. The WAD file associated with the commercial shareware distributions, DOOM.WAD, only contains one episode with nine individual levels. The full game WAD — inconveniently also DOOM.WAD — includes the shareware episode along with two more, making the full experience three episodes and twenty seven levels. Including *Doom*’s various physical media distributions to analyze their contents

⁵⁶Retroactively referred to as “Where’s all the data?” files. id originally considered them a “wad” of data.

⁵⁷*Doom*, as a primarily online distributed work, also has validated hashes for the compressed archives (.zip files) enclosing its different versions. For example, one of the first distributions of *Doom*, DOOM10S.ZIP (Doom Version 1.0 Shareware), is regarded as a unique file in community hash indexes, along with its contained .EXE and .WAD files.

and production timeline would be a natural extension of this research.⁵⁸

Why Analyze Contents and Structure?

The internal structural changes to the files that compose a piece of software receive scant analytical attention. So, why is this information historically important and what can it tell us about a program and its development? For one, the secondary contents included with the executable proper contribute to our understanding of the whole by providing context for how to interact with the software, and noting potential, critical changes from previous versions. Another reason is that the organization of files themselves may reveal insights into the functions of the program and the labors of the development team.

In regards to *Doom* specifically, the presence of two separate game files, one for the engine and one for the game data, represents the terminus of a game asset consolidation process. id's earlier titles up through *Hovertank 3D* broke out level data into individual files, and included separate sound and art assets as well. In Table 6.1 below we see the directory listings of *Rescue Rover*, *Hovertank 3D*, *Catacomb 3D*, *Wolfenstein 3D*, and *Doom*. The files progress towards more data consolidation and lower numbers. *Rescue Rover* and *Hovertank 3D* each contain explicit files for every individual level, while *Catacomb 3D* and *Wolfenstein 3D* consolidate the map data into single collections. Previous id efforts, including most of their games for SoftDisk and the Commander Keen series, also broke out individual level and asset files. *Doom*'s consolidation represents a distinct change in game data packaging. Having a single file makes the interaction between the engine executable and game data less confusing from a management point of view.

⁵⁸Various physical copies of *Doom* are more difficult to acquire, and would probably require a separate, dedicated research effort to track down. Luckily, we assume, the *Doom* engine and WAD information on those disks is identical to the known versions of the game (the exact filenames and document structure may differ however).

Every piece of pertinent game data could be fed to a WAD builder program — wadlink.exe — that then created a single archive, which was easier to distribute and validate. While compressing the files into a zip archive is also possible — id used this scheme for later Quake distributions — the singular file format also provided a nice template for further modifications and easier versioning.⁵⁹

Table 6.1: Comparison of id Games’ File Structures

Rescue Rover	HoverTank 3D	Catacomb 3D	Wolfenstein 3D	Doom
CTLPANEL.ROV	DOCSHELL.EXE	AUDIO.C3D	AUDIOHED.WL6	DOOM.EXE
DOCSHELL.EXE	DSOUND.HOV	CAT3D.EXE	AUDIOT.WL6	DOOM1.WAD
EGABTILE.ROV	EGAGRAPH.HOV	CONFIG.C3D	CONFIG.WL6	LICENSE.DOC
EGAFONT0.ROV	EGAHEAD.HOV	DOCSHELL.EXE	GAMEMAPS.WL6	ORDER.FRM
EGAHEAD.ROV	HOVER.EXE	EGAGRAPH.C3D	MAPHEAD.WL6	README.EXE
EGAPLANE.ROV	INSTRUCT.TXT	GAMEMAPS.C3D	VGADICT.WL6	SETUP.EXE
EGATILES.ROV	LEVEL01.HOV	INSTRUCT.TXT	VGAGRAPH.WL6	VENDOR.DOC
ENDPIC.ROV	LEVEL02.HOV	MENU.SHL	VGAHEAD.WL6	
HINTS.TXT	LEVEL03.HOV	QUICK.TXT	VSWAP.WL6	
INSTRUCT.TXT	LEVEL04.HOV	README.BAT	WOLF3D.EXE	
LEVEL01.ROV	LEVEL05.HOV	README1.TXT		
LEVEL02.ROV	LEVEL06.HOV	README2.TXT		
LEVEL03.ROV	LEVEL07.HOV	README3.TXT		
LEVEL04.ROV	LEVEL08.HOV	START.BAT		
LEVEL05.ROV	LEVEL08.HOV			
LEVEL06.ROV	LEVEL08.HOV			
LEVEL07.ROV	LEVEL09.HOV			
LEVEL08.ROV	LEVEL10.HOV			
LEVEL09.ROV	LEVEL11.HOV			
LEVEL10.ROV	LEVEL12.HOV			

Continued on next page

⁵⁹John Romero recently revealed the purpose of wadlink.exe, and showed how its use on a collection of files in id’s development folder constructed the Ultimate Doom version of the game. [179] John Romero, “Happy 23rd Birthday, DOOM!,” Rome.ro <http://rome.ro/news/2016/12/10/happy-23rd-birthday-doom> (accessed 26 Mar. 2017)

Continued from previous page

Rescue Rover	HoverTank 3D	Catacomb 3D	Wolfenstein 3D	Doom
LEVEL11.ROV	LEVEL13.HOV			
LEVEL12.ROV	LEVEL14.HOV			
LEVEL13.ROV	LEVEL15.HOV			
LEVEL14.ROV	LEVEL16.HOV			
LEVEL15.ROV	LEVEL17.HOV			
LEVEL16.ROV	LEVEL18.HOV			
LEVEL17.ROV	LEVEL19.HOV			
LEVEL18.ROV	LEVEL20.HOV			
LEVEL19.ROV	MENU.SHL			
LEVEL21.ROV	QUICK.TXT			
LEVEL22.ROV	README.BAT			
LEVEL23.ROV	README1.TXT			
LEVEL24.ROV	README2.TXT			
LEVEL25.ROV	README3.TXT			
LEVEL26.ROV	SOUNDS.HOV			
LEVEL27.ROV	START.BAT			
LEVEL28.ROV				
LEVEL29.ROV				
LEVEL30.ROV				
MENU.SHL				
MENUPIC.ROV				
QUICK.TXT				
README.BAT				
README1.TXT				
README2.TXT				
README3.TXT				
ROOMPIC.ROV				
ROVER.EXE				
S_DEMO.ROV				

Continued on next page

Continued from previous page

Rescue Rover	HoverTank 3D	Catacomb 3D	Wolfenstein 3D	Doom
S_PLAY.ROV				
SAVEGAME.ROV				
SOFTPIC.ROV				
SOUNDS.ROV				
START.BAT				
TILEINFO.ROV				
TITLEPIC.ROV				

WAD files are a general archive file format. Each WAD is simply a structured collection of data “lumps” packaged according to a specific indexing scheme.⁶⁰ In the case of a *Doom* WAD, the WAD file’s dictionary — the indexing table that describes the data it holds — need only contain a certain types of data that corresponds to *Doom*’s map, asset, and sound formats. *Doom*’s executable is therefore a WAD processing engine, since the executable would have nothing to render without WADed data. This already problematizes the official versioning scheme presented in the next section, since each version of *Doom*’s engine expects different internal features of complementary WAD files. Two separate version trees result from decoupling *Doom*’s engine and data, one for the *Doom* executable and the other for the progression of WADs internal contents. As we will see below, this manifests in the subtle distinction between *Doom* and *Doom II*, both of which are based on the same engine but played with different WADs.⁶¹ Interestingly, this

⁶⁰The term “lump” is the technical name for the data entries in a WAD file. The file’s index points to specifically named “lumps” of data, like TEXTURES, LINEDEFS, and NODES, which the engine then uses to construct and render a level. Any other arbitrary data can also be put into a WAD with differently named lumps. This allowed enhanced versions of the engine, either for commercial pursuits — *Heretic* and *Hexen* — or user modifications to easily include and parse additional data for new features.

⁶¹This also leads to slightly confusing naming conventions for WADs, since the original DOOM.WAD is now retroactively named and distributed as DOOM1.WAD in some cases, despite being identical to the original release IWADs.

separation also allowed a for common WAD format for all versions of *Doom* on all platforms. For instance, the DOOM.WAD file found in the Sony PlayStation release can be interpreted by the same WAD analysis tools as the version for Microsoft MS-DOS or the Microsoft Xbox 360. id, internally, conceived of the *Doom* engine as more of a software platform for WAD files than as a game executable. This is evident in the first release of the game, which actually referred to itself as the “Doom Operating System Version 0.99” upon start up.⁶²

The Version Tree of *Doom*

Figure 6.8 represents a complete tree of the known versions of *Doom* across all platforms. Only versions of *Doom* are listed, with the exception of pointers to important derivations and *Doom*’s sequel, *Doom II*. We include *Doom II* because its engine, beginning with version 1.8, is the same one that powered *Doom*. *Doom II* is really any version of the *Doom* engine greater than 1.666, since those versions of the engine support the additional features included in *Doom II* WAD files. Therefore any version of the *Doom II* engine, released with *Doom II* in its physical packaging, is compatible with *Doom*’s earlier WADs.

The progression of *Doom* versions operates in two distinct phases demarcated by the source code release in late 1997. Up until that point, all versions of *Doom* were explicitly licensed to third-party developers for other platform. After the source became available, many fan driven source ports arose — and will be covered in the next section.

The version progression begins with the five recovered alphas of *Doom* released to id’s testers, friends and various press outlets over the course of 1993. The press release beta is a time locked demo distributed in October of 1993, and the first

⁶²This was changed to simply “Doom Version X.X” in subsequent releases, perhaps due to players’ confusion about the difference between an operating system and a game. For us, however, it is a delightful reminder of the fluidity of the classification of different types of software.

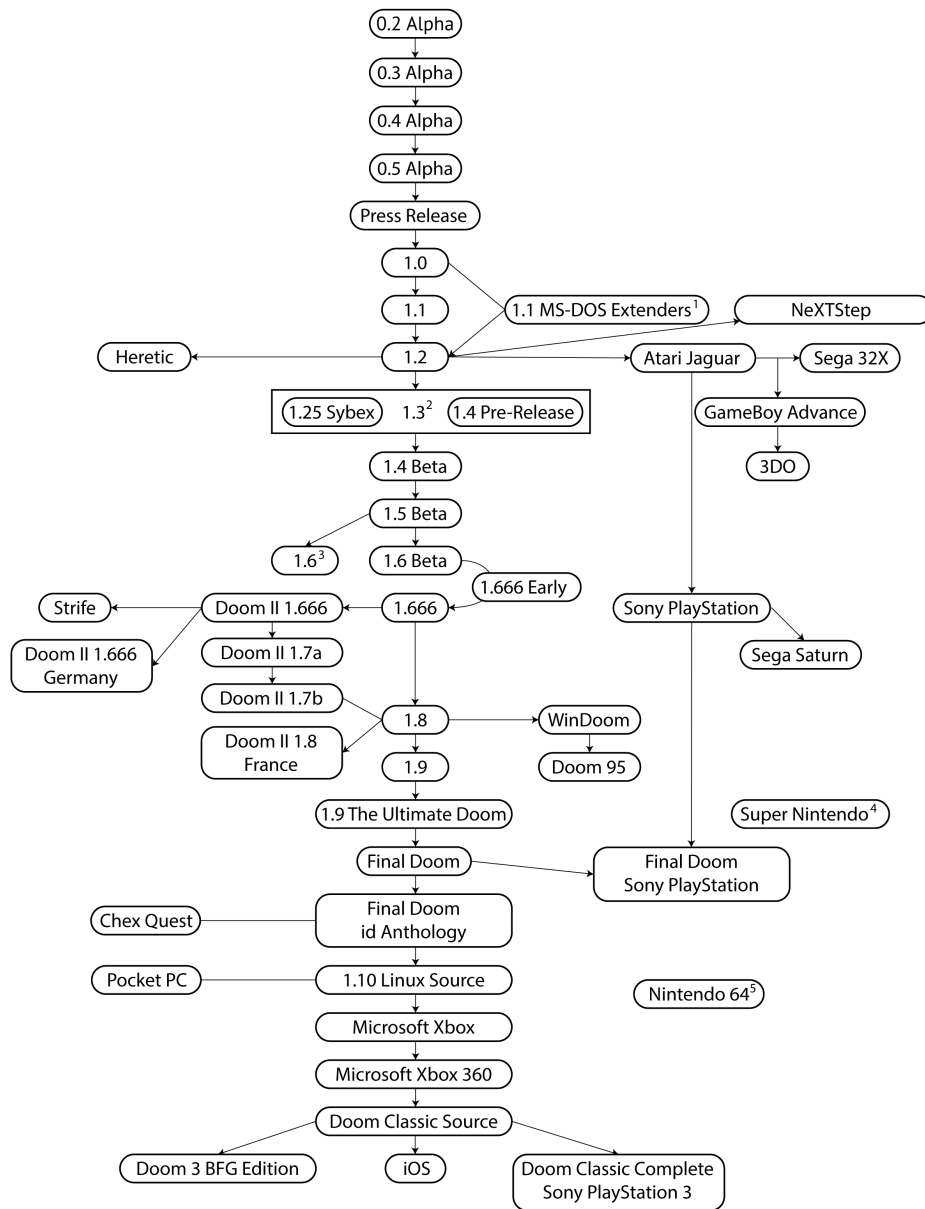


Figure 6.8: Versions of *Doom*. (1) MS-DOS memory extension incorporated into Version 1.1; (2) Version 1.3 is actually a combination of different releases, there was no “official” Version 1.3; (3) Version 1.6 is a hacked leak; (4) the Super Nintendo Version is not based on *Doom* engine code; (5) *Doom 64* is not based on *Doom* WADs, but was a totally new game.

version of the game leaked online.⁶³ Members of the community scouring this version encountered the WAD file for the first time, and proceeded to tear it apart. The timing of the leak allowed an extra month of lead time in developing modification tools, which became available in early January 1994 with the release of the *Doom Editing Utilities* (DEU) package.

Between December 10, 1993 and late 1996, 20 versions of *Doom* are available and recoverable for MS-DOS alone. All engine changes and software fixes along the way are therefore analyzable. At various points, a version of the game would be licensed with a third-party developer, with those efforts taking on a life of their own in the case of the port for the Atari Jaguar. That version provided the base for the GameBoy Advance, 3DO, Sega 32X and Playstation ports. The Atari Jaguar and Sega 32X ports are very similar, as John Carmack ported them both simultaneously in mid-to-late 1994. The release of the Atari Jaguar version's source code in April 2003 verified this similarity. Compiler preprocessor flags in the code pointed to both Atari Jaguar and Sega 32X release targets. All of the console ports of *Doom* suffered from some form of degradation as a result of the specific limitations of their architecture. Designed initially for the particular memory constraints of MS-DOS architecture, other versions had to make significant concessions in the amount of levels, speed of play and viewable area to make game play possible. The least successful is probably the 3DO version, which was ported in six weeks to an alpha level of quality before being released for the holidays in 1996.⁶⁴

⁶³This notion of time locked execution, a form of temporal digital rights management, is important to consider in archival contexts. The demo checks the current system time. If it is not sometime in October 1993 the game will refuse to load. Lee Killough, a prominent *Doom* source port figure, released a hack to specifically disable this DRM, called "fakedate". For our purposes, we would need to modify the emulator's system time to play the game, something that it currently does not support.

⁶⁴The development process of the 3DO port is actually highly entertaining, and recounted by its developer Rebecca Heineman on her YouTube channel and in various interviews.

John Carmack, owing to a hacker mentality and a distinct lack of concern for common business ethics, sent source code to interested developers willing to put time into a port. This led to partially completed ports for the BeOS operating system, and further work on a proper NeXTStep port that had since been abandoned by the id team. Michael Abrash — a well-known graphics programmer working with Carmack on Quake’s engine — once asked Carmack if, after completing a successful project, he ever doubted he’d be able to do it again. Carmack was apparently puzzled by the question, since failing to top himself had not occurred to him.⁶⁵ It is likely then, that Carmack probably lost interest in *Doom*’s engine and code as the challenge of Quake loomed ahead, which partially explains the laxity regarding *Doom*’s source dissemination.

Returning to an earlier point, the presence of *Doom II* on this tree is due to it being merely an extension to *Doom*’s engine instead of a completely new one. id had begun working on *Doom II* shortly after *Doom*’s release. Some of the intermediate beta versions listed between 1.2 and 1.666 actually support some features of *Doom II*’s WAD file, with version 1.666 being the version of the engine that shipped in *Doom II*’s physical packaging. The beta versions feature explicit documentation in their README files of the advances made to the engine along with requests to not disseminate them publicly. The latter directive was certainly ignored, given that a version 1.6 of *Doom* that was compatible with the registered WAD file included hacked credits making fun of John Romero. The split from version 1.666 to 1.7a|b was a series of patches to the commercial release of *Doom II* that did not affect *Doom*’s original engine functionality. After version 1.8, both

[173] Rebecca Heineman, “Burgertime 7/12/2015: DOOM 3DO,” <https://www.youtube.com/watch?v=rBbIi12HPSU> (accessed 26 Mar. 2017). [17] Don’t Die, “Rebecca Heineman,” <http://www.nodontdie.com/rebecca-heineman/> (accessed 26 Mar. 2017)) This version was so rushed that upon completion of the game, players are presented with a black screen with congratulatory text and then punted to an unusable command prompt.

⁶⁵[113] pg. 190

Doom and *Doom II* unified in supporting the WAD files for each game.

The Linux 1.10 release and *Doom Classic* are both virtual releases, since they were not directly attached to a specific version of the game but are derived from or the basis of other versions. The cleaned up Linux source code supported two Xbox Live Arcade releases, followed by John Carmack's efforts to get *Doom* running on the newly released iPhone in 2007. Fittingly, the *Doom Classic* port to iOS is written in Objective-C, which was first developed for the NeXTStep operating system on which Carmack and the team first programmed *Doom*. Essentially, *Doom* was originally written in Objective-C frameworks that allowed for cross-compilation to DOS, and all of *Doom*'s development tools were Objective-C applications written by either Romero or Carmack.

Finally, the Super Nintendo (SNES) version of *Doom* is represented as a satellite node because its engine was not written by id. Programmer Randy Linden wrote his own Rage / Reality engine to process DOOM.WAD's data and render it on the SNES.⁶⁶ Apparently, Linden, then at the company Sculpted Software, programmed SNES *Doom* in secret and presented it, in a completed state, to id Software.⁶⁷ *Doom 64* is also included on this list mainly due to its dependence on the Sony PlayStation port. The game features a significant modified engine, and does not share any levels in common with any other version of *Doom*. It is therefore *Doom*-like but not technically a version of the original game.

⁶⁶As discovered by the community (and easily verified with a hex editor) the SNES *Doom* ROM file actually contains the name of the custom *Doom* engine at location 10F. It reads, "Rage / Reality Engine written by Randy Linden. Special thanks to my loving wife, Jodi Harvey."

⁶⁷John Romero revealed this secret production in a blog post. [180] John Romero, "Doom History 1994," www.planetromero.com, <https://web.archive.org/web/20150108101506/http://planetromero.com/2009/01/doom-history-1994> (accessed 26 Mar. 2017)

Doom's Objectivity

Simply referring to “Doom” as a unified signifier is a fraught archival and historiological proposal. No only are there many versions of Doom across varied execution contexts, but even the internal structure of those versions have specific sub-dependencies. The separation between *Doom*'s engine executable and its game data necessitates a level of description *below* versioning, one more like a descriptive *constellation* of the various files that make up any presupposed *Doom* version.⁶⁸ Treating Doom as a unit supposes that one has fulfilled all of its dependencies and that those dependencies will manage to accompany it forward in time. In the below table outlaying the file contents of each version of *Doom*'s shareware, we see complementary programs, like DOS4GW.EXE — a program enabling easier DOS memory management — or DEICE.EXE — a decompression program used to allow *Doom*'s data to shrink in accommodation to early 1990s bandwidth limitations — that are instrumental in retrieving and running a version of the game. Furthermore, this ontological quagmire is not directly addressed in most software studies works, or even explicit platform studies, which are pervaded by the assumption that having a version of a piece of software is the ultimate historical unit of analysis. More work is needed to codify the distinction between a software object, its versions and how its file contents and dependencies reflect upon and constrain those higher structures.

⁶⁸Recall in the Appraisal chapter, our discussion of the need for connecting a file's creation to the specific dependent programs that created it. This sub-versioning is another example of that issue.

Table 6.2: Comparison of Doom Image Contents 0.2 to 1.25

0.2	0.3	0.4	0.5	Press	1.0	1.1	1.2	1.25
doom.exe	DOOM.EXE	DOOM.EXE	CONGIF.LMP	DEICE.EXE	DOOM.EXE	DOOM.EXE	DEFAULT.CFG	DEICE.EXE
doom.wad	DOOM.WAD	ALPHREAD.ME	DOOM.EXE	DEMO.1	DOOM1.WAD	DOOM1.WAD	DOOM.EXE	DOOM1_25.1
	README.TXT	CONFIG.LMP	DOOM.WAD	DEMO.2	LICENSE.DOC	LICENSE.DOC	DOOM1.WAD	DOOM1_25.2
		DOOM.WAD	DOS4GW.EXE	DEMO.DAT	ORDER.FRM	ORDER.FRM	IPXSETUP.EXE	DOOM1_25.DAT
		DOS4GW.EXE		INSTALL.BAT	README.EXE	README.EXE	LICENSE.DOC	INSTALL.TXT
				README.TXT	SETUP.EXE	SETUP.EXE	MODEM.CFG	
				SCNSHOTS.ZIP	VENDOR.DOC	VENDOR.DOC	ORDER.FRM	
							README.EXE	
							SERSETUP.EXE	
							SETUP.EXE	

Additionally, there is a history inside these computational objects, since they are a reflection of their creators' constitutive processes and actions. Taking tables 6.2 and 6.3 as an example, we find a diversity of file organization schemes linked to methods of distribution and the influence of the community. The unofficial *Doom* FAQ, a monstrous document organized by Hans Leukert from 1994-1996 became official in the version 1.4 beta release of *Doom*, superseding id's own documentation (which was itself locked inside "README.EXE", a separate DOS program written by John Romero). The files DOOM1_A.ZIP, DOOM1_B.ZIP, and DOOM1_C.ZIP, reflect the storage and bandwidth limitations of the day. Each compressed ZIP file being around 700k, which was small enough to fit on 720kb disks for local distribution or take-home copies.⁶⁹ Since many users could not connect to various FTP servers for long enough to download the full DOOM1_0.ZIP release, they acquired chunks at a time based on the limitations of their service providers. This also reveals a potential hidden economic externality that aided *Doom*. Since many initial downloads were to Internet connected users at universities and tech corporations, those larger structures effectively subsidized *Doom*'s dissemination to its audience at a time when online distribution was still relatively expensive.

Benefits of Object-Level Analysis

Doom as a reified software object exists in a multitude of forms across a range of platforms. It is also part of a lineage of previous constellations of files and reified objects produced by id Software between its founding in 1991 and *Doom*'s release in 1993. Recalling the earlier historical model, it should be apparent that there is much inside the object that can still support, enlighten and modify

⁶⁹Kirk Membry, "doom1_a.zip ?," comp.sys.ibm.pc.games.action, https://groups.google.com/forum/#!search/doom1_a/comp.sys.ibm.pc.games.action/K6Prpc7QR0M/WBvLukT2i9wJ (accessed 26 Mar. 2017)

understandings of the game’s development history. Already, from the brief look in this section, a more complicate image of *Doom*’s existance has emerged. id’s efforts at file consolidation and its effect on re-organizing their level structures toward the creation of a new, easily disseminated and encoded WAD file become apparent simply from the comparative availability of previous works. *Doom*’s separation of game engine and game data, allowed for easier development of new versions of the game, since they were basically intepreters for a common WAD format. As Lowood notes, id pioneered the notion of a “game engine” — most likely even coining the term “engine” based on John Carmack’s side interest in car maintenance. The benefits of that approach shine through in the progress of *Doom*’s modding community, and the format’s adoption by other companies, like Raven Software’s *Heretic* and *Hexen*, and Bungie’s *Marathon* series.⁷⁰

The availability of *Doom*’s various objective works also, again, points to the benefits that community engagement and archiving can reap for the history of computer games. Without the meticulous early archiving efforts of the *Doom* community — as noted in the historiography section above — none of assertions of this section would be possible.

6.4.3 Technical Expression

In discussing the history of *Doom*’s source code, and its travels from id Software out the community in the late 1990s, we again encountered issues of boundaries and historical discretation. Much like the items described in the last section, each instance of *Doom*’s source code is a complex collection of files, tools, and other technical assets that coalesce into a “singular” playable object that can be

⁷⁰Since *Marathon*’s object is also available, a look at its file structure revealed a similar WAD structure. This again shows how a simple consideration of file contents can reveal latent chronologies upon which to build more informed histories of games.

referred to as a computer game. Source code analysis presents significant problems for the historical researcher. Old code is by definition outdated, so tropes and contemporaneous strategies for program creation need to be recovered to make any sense of it. Additionally, production source code is rarely neat and orderly. As mentioned in the Appraisal chapter, the outputs of production processes, the technical expressions that eventually make up a finished object, are varied, diverse and usually haphazardly distributed. This means that in the analysis of source code, we also confront the limitations of that study, and how the contextual information surrounding the code might be as — or perhaps *more* — important than the code itself for specific historical questions.

In *Doom*'s case, the “official” source code released to the community is not actually a true example of production code. Cleaned up and extracted from the mess of id's internal servers, it represents a more idealized version of the code compiled into earlier *Doom* versions. This means that a certain part of id's internal processes — like their management and development strategies — are not actually found in the official source code, only the shadow of their programming ingenuity and problem solving skills remain intact. This leads to two conclusions about the historical usefulness of source materials that are at odds with some prevailing views. First, the assumption that obtaining source code is a “holy grail” for the comprehension and reproduction of a system's history is misguided. Source is an important part of the equation, but it is a highly contingent resource — dependent on the remaining collective knowledge about its context and use — that does not reveal itself easily. *Doom* is lucky, in that most of the systems used to develop it are still historically available. Furthermore, it was written in dialects of the languages “C” and “Objective-C” that are still in widespread use and were already standardized at the time of its creation. So source code's historical usefulness is

very dependent on the context of its creation and resources available to decode it. Second, the idea that one can simply “read” a program and understand what it is doing, how it will act in run-time, and divine the primacy of its creation is false. We feel that the literary minded “reading” source code⁷¹ is actually of limited value without positioning it within a more “naturalist” and exploratory paradigm of scholarly investigation.⁷² This again cuts against arguments for the value of source code as a primary resource.

Now, this is not to say that source code should not be considered an incredibly valuable piece of the game historical and preservationist puzzle. Without it, certain practical reconstructive efforts would not be possible, and system analysis might not be able to proceed at all. In this section’s examination of *Doom*, we will see how the sustained efforts of community developers and interested researchers have preserved and organized the history of *Doom*’s source code. This will illuminate the requirements that need to be in place for a robust history of *Doom*’s technical expressions.

Doom’s Source History

The history of *Doom*’s source code is demarcated by its public release in December of 1997. Before that, all source access to *Doom* was given out piecemeal based on licensing agreements with id, and as mentioned above, in certain instances where John Carmack was feeling particularly charitable.⁷³ Unique to

⁷¹[37] Black, Maurice. “The Art of Code.” Dissertation, University of Pennsylvania, 2002.

⁷²[192] Seibel, Peter. “Code Is Not Literature,” 2014. <http://www.gigamonkeys.com/code-reading/>. presents the “naturalist” metaphor. [216] Wardrip-Fruin, Noah. Expressive Processing. Cambridge (Mass.): MIT Press, 2009. also describes the pitfalls of applying certain forms of close-reading to code that was not constructed as literature in the first place.

⁷³In addition to the source code shared for the BeOS port mentioned in “Reified Object,” Carmack also gave the code to the OMNI group for free in order to pursue a completed port for NeXTStep. It is likely that there are other instances of pseudo commercial sharing of this kind, but these are the only two verified ones. <http://blog.wilshiple.com/2013/12/my-doom-20th-anniversary-stories.html>

Doom, and really to the id in general, is that most of id's source code from the company's founding in 1991 through 2008 has been open sourced.⁷⁴ The Commander Keen series, HoverTank 3D, *Catacomb 3D*, and *Wolfenstein 3D* all have available source, which allows for a diachronic analysis of id's programming development that is probably unrivaled for early game companies. Complementarily, after *Doom*'s source code release, and especially after its re-licensing in GPL in October of 1999, most of its continued community development source is also available.⁷⁵ This enables a look at not only id's programming progression, but also that of larger enthusiast community in taking id's ball and running with it to new technical heights.

Doom's technical achievements, mainly the speed and fluidity of its three-dimensional, first person viewpoint, created a bit of a stir in the technical game community after its release. However, the game visual fidelity was not *that* revelatory in comparison with other contemporary games. Ultima Underworld featured fully texture-mapped surfaces, and 360-degrees of camera movement. Raven's *Shadowcaster*, using John Carmack's intermediary engine between *Wolfenstein* and *Doom*, supported sloped surfaces. And much of the attention at the time was devoted to the approaching importance of the CD-ROM format in allowing for games with full motion video. Early reviews of *Doom* in the game and computing press did not usually mention it as a graphical powerhouse, and its coverage was overshadowed by other releases, including, ironically, *Shadowcaster*. What *Doom* did have, to those who understood the inherent technical complexities of VGA rendering on 386 and 486 IBM PC compatibles running MS-DOS,

⁷⁴After the release of *Rage* in 2008, Bethesda Softworks purchased id Software, and eventually stopped releasing id's engine source code.

⁷⁵The original source release had a restrictive commercial license attached. The Gnu General Public License (GPL) is an open free-software license that makes community code easier to share and reuse.

was amazing speed. Every other game mentioned in this paragraph might have had more advanced features on paper, but none of them were particularly fast. *Ultima Underworld* required button clicks to move forward and change view points. CD-ROM games presented stunning but rather static visuals that required significant loading times to get into memory. *Doom* somehow managed to bring what appeared to be three-dimensional geometry, complex lighting effects, and superior sound design to a table set for slower, more contemplative tastes. Its technical virtuosity was not lost on game developers or graphics researchers however, and it was one of these researchers that partially encouraged *Doom*'s source release.

In the section on *Doom*'s historical accumulation, we outlined the basic history of *Doom*'s source release. Bernd Kreimeier, a software engineer and author, approached id about writing a technical overview of *Doom*'s algorithms in 1996. By 1997, the book deal had fallen through — considered as too dated by the publisher — but Kreimeier had already been working with id to clean up the code. That code became the base for a large number of “source ports” — modifications of the source code to allow for compilation on other systems. The term “source port” itself is attributed to the first port of *Doom*'s source code back to MS-DOS — entitled “*DOSDoom*” — by Chi Hoang at the University of Calgary.⁷⁶ In order to get *DOSDoom* running, Hoang notes that he had to incorporate files from *Wolfenstein 3D* (for ASCII text conversion tables) and *Quake* (for CWSDPMI.EXE, an interface for DOS protected mode), and thus also substantiates the value of having access to a company's other technical efforts (as they are most likely related to one another).⁷⁷

⁷⁶This is a “fact” of the community, as Hoang's documentation just refers to his work as a “port” of the “unix” source code and not a “source port”. It is possible that the now lost *DOSDoom* Version 0.2 might have referred to itself as a “source port” in its similarly lost documentation.

⁷⁷CWSDPMI.EXE stands for Charles W. Sandmann's DOS Protected Mode Interface, and was not a file unique to *Quake* or id software (though the version included with *Quake* was

The release of the *Doom* source code was not id's first code release. Although we mentioned that *Hovortank 3D*, *Catacomb 3D*, and *Wolfenstein 3D* also have available source code, only *Wolfenstein 3D*'s was released before *Doom*'s (on July 21, 1995). A significant note on the conceptual framing of source code releases is found in John Carmack's RELEASE.TXT file, where he states that,

We are releasing this code for the entertainment of the user community. We don't guarentee (sic) that anything even builds in here. Projects just seem to rot when you leave them alone for long periods of time.

This is all the source we have relating to the original PC wolfenstein (sic) 3D project. We haven't looked at this stuff in years, and I would probably be horribly embarassed (sic) to dig through my old code, so please don't ask any questions about it. The original project was built in borland c++ 3.0. I think some minor changes were required for later versions.⁷⁸

The complexity of treating source code as a stable historical object is encapsulated in this brief note. First, "this is all the source we have relating to" *Wolfenstein 3D* highlights that the game's code needed to be recovered internally and did not exist as a discrete set of technical expressions ready for dissemination. The source is "related" to the technical efforts of id during the Wolfenstein "project" and not tied to the notion of a specific version of the game. The source is then more amorphous than its release would assume, and encounters some of the similar issues we have stressed earlier in this thesis of the problems with divining the boundaries between technical works. This disambiguation of code is even more apparent with *Doom*. Based on our access to a snapshot of *Doom*'s original "unreleased" code, it is difficult to locate the ultimate collection of source

specifically modified for it). The Wolfenstein 3D addition is a collection of DOS keyboard scan code values taken from the ID_IN.H header file for *Wolfenstein 3D*'s input manager. id's engines do share significant similarities in structure, so it is not particularly surprising that files and constant definitions used for one of their DOS games would also show up in another.

⁷⁸RELEASE.TXT as found on <https://github.com/id-Software/wolf3d>. Although the GitHub repository is from 2012, the actual source code release from 1995 appears to be identical.

files “related” to its release at that time.⁷⁹ The issue, technically, is that source code is not a static, unchanging set of documentation while it is under development. Therefore, its internal structure, how files link to each other, and even what the code does when executed are highly contingent and idiosyncratic to the state of development at any given moment. Teams create new files, decide not to use them, move things around, and delete them in a steady churn of activity. Approaching this mess from the outside means that one needs to reconstruct the context of these activities in order to figure out which files are useful, and part of the actual source documentation for the given version of the software at the time one is looking.

Second, Carmack remarks that they do not know if the code still compiles, and simply recommends the original compiler used for the project “Borland C++ 3.0”. At this point in 1995, that compiler was already four years old, and as Carmack notes “rot” has already set in. This addresses one of the major points of this layer of the model, in that the availability and presence of source code is not a guarantee for recovery of the original program. Luckily, again due to the popularity of C/C++ and the DOS platform, copies of Borland C++ 3.0 are not difficult to acquire. With the aid of DOSBox it is currently possible to compile the original *Wolfenstein 3D* source (albeit with a bit of tinkering).⁸⁰

Lastly, the source for *Wolfenstein 3D* and *Doom* both contain only files relevant to the compilation of a binary executable. They do not feature any of the copyrighted art content for the game, nor any of the numerous tools and systems developed by id to manage map creation and editing, preprocessing of level ge-

⁷⁹The impression of the id source directories, graciously provided by John Romero — but also illegal to distribute — reflect the state of *Doom*’s development around the release of the id Anthology Series version of Final Doom. This is not immediately apparent given the inherent mess of the directories and files in the directories.

⁸⁰The full compilation instructions are available at: http://fabiensanglard.net/Compile_Like_Its_1992/index.php.

ometry or creation of archived game data files like WADs. They contain only source that is “instrumental” to a specific compilation target, one that still manages to hide the greater messiness of its original creation. This makes sense in light of Carmack’s goals for the release, programmers would not appreciate having to wade through a collection of source documents to find the pieces needed for modding and future development. However, for historians of technology and software engineering, that mess is important to the understanding of historical development practice.

Understanding the technical expressions that coalesce into a piece of software, and comprehending how those expressions affect the final expression of the software object is a highly contingent process. As we have seen, the source code from *Doom* is only comprehensible due to a collection of specific, and fortunate historical conditions. But even then, making sense of all its technical expressions is still a daunting task. *Doom* had the benefit of being written in a standardized, and still popular language; the support of an active development community obsessed with explaining and modifying its inner workings, and probably most importantly, developers willing to share their code with that community. As noted by John Carmack, the maintenance of knowledge about *Wolfenstein 3D*’s compilation evaporated when he moved onto the next project, and it was probably a similar situation with *Doom*. Relying on source code alone then does not provide, in many cases, adequate context for a program’s recovery, even, after a time, for the developer of that program! Further work into tying the technical expressions that constitute game software to the organization, actions, and processes of their developers is then imperative for future comprehension of past works. Trying to read programs *as* literature implies that they were written *as* literature, which excluding some well intentioned projects by famous computer scientists, is gener-

ally not the case.⁸¹ Additionally, as we saw in the preceding chapter on Citation, many kinds of system interactions and run-time behaviors benefit from explicitly non-literate presentations like system visualization images or dynamic and executable content. Source is an important piece of the software historical puzzle, but it is still only a piece. Again, without the initial model pointing toward the analytical potential of technical expressions, as analyzed over time and enacted on by historical processes, none of these practical considerations about source code become visible. Clearly, a focus on the method of historical investigation can also help to reveal new areas of study.

6.4.4 Enacted and Tacit Knowledge

As mentioned above, the floor for historical analysis of software — when approached by software and platform studies — tends to stop at formation and analysis of technical expressions. In this section the focus goes a bit further into the psychic and somatic territory of the tacit and enacted knowledge that makes technical expressions — and therefore code, assets and what have you — possible in the first place. This type of knowledge is probably the most difficult to nail down and describe most specifically because it arises out of the creative process and the accretion of failures and successes built up over periods of time. The best that can be done is to try and find the sources of technical inspiration for game software, and link those sources to the creative outputs of the programmers and

⁸¹See [108] Knuth, Don, and Doug McIlroy. “A Literate Program.” *Communications of the ACM* 29, no. 6 (1986) for an example of “literate programming” in the creation of a “hash-trie” to form a concordance for arbitrary textual inputs. McIlroy notes, in his criticism, that sometimes an image could do more than a literary exposition, which cuts to the heart of our discussion of the over reliance on source code itself for comprehending a program’s meaning. Knuth’s work is the extreme form of programming as a literary construct, and while it is impressive and useful, its immediate flaws also show the limits of the paradigm. For an even more fun “literate programming” example, see [107] Knuth, Don. “Adventure,” 1998. <http://www.literateprogramming.com/adventure.pdf>. in which he rewrites the source code for *Adventure* in a literate fashion.

designers that relied on that information.

As Michael Polanyi notes, tacit knowledge is “a knowledge that we cannot tell”, and therefore that which can only be partially revealed through the replaying of the processes, either the re-enactment of knowledge through new tools of expression, or literal mimetic reproduction of an activity.⁸² Polanyi’s main thrust is that the revelation of process must start with an awareness of its indistinct and non-historicizable nature. There is no material “evidence” of someone’s technical skill separate from the products of that effort. Furthermore, a practitioner might not be able to explain how they engage with an activity on an explicit, intellectually parsable level. In these cases, the recover of process, and the recovery of evidence of those processes functions as a deficient but informative proxy. Work by Henry Collins, most notably *Changing Order*, finds the historian working with his research subjects in the construction of their technical apparatus in an attempt to tease out the practitioners’ implicit assumptions — along with their sources.⁸³ The main note is that production knowledge comes from somewhere before being internalized and deployed through constitutive acts — like software production and game development. The remainder of this section applies this perspective to the knowledge sharing about graphical techniques at work in game development around and within *Doom*.

In Fabien Sanglard’s analysis of the game engine for *Trespasser* — a first-person shooter situated in the Jurassic Park universe — he quotes one of that game’s developers, Seamus Blackley, describing the process of constructing the game’s physics engine in the mid-1990s.

In the 90s, it was tough. The real issue, honestly, was that it was too much for me to do physics and also be in charge, and I never figured

⁸²[169] pg. 5

⁸³Latour and other ethnographers of technology take this embedded approach to knowledge comprehension and relay.

out how to fix that. Back then, there were no books or libraries on game physics, it was all research, and it was really, really hard.⁸⁴

Blackley had to re-invent certain routines and approaches, in this case copying rendering algorithms straight from textbooks and hand-rolling a rigid body physics engine, because that information was not otherwise available. Game Developer Magazine would have been about three years old, GDC five,⁸⁵ and the available publications on game programming typically focused on introductory development (and therefore algorithms and approaches well out of the date). Blackley would have liked access to information on modern physics programming research, but the now common networks trafficking in research PDFs did not exist.⁸⁶ The progress that was made in this era relied more on happenstance and continuous re-invention than on a foundational basis of knowledge labeled “3D game engine programming”. No general tools existed for most games’ constructions. Companies could license some pre-existing engines, id Software’s own DOOM and Wolfenstein 3D engines spawned numerous titles, as did Ken Silverman’s Duke Nukem 3D engine (BUILD), but the dissemination of knowledge about programming techniques fundamentally relied on game developers’ previous experiences, haphazard exposure to others’ code, and community networks spread across early publications and Internet resources. That is, most of the ability to create games did not rely on codified, explicit knowledge of game programming or game software design, but on piecemeal collections of methods and techniques compiled from a variety of sources. John Carmack noted the lack of available advanced programming knowledge, similar to Blackley’s above, in an interview with Dan Pinchbeck,

⁸⁴[191] Sanglard, Fabien. “Jurassic Park: Trespasser CG Source Code Review.” Accessed June 10, 2017. <http://fabiensanglard.net/trespasser/>.

⁸⁵The Game Developer’s Conference

⁸⁶It should be noted that Blackley did have a physics degree and experience with graduate physics research, so he was not completely in the dark.

It's interesting to talk to people about the old days. Of course, you've got the Internet now. You can find anything nowadays. But back then, it was really something to get reprints of old academic papers. There were some clearinghouses I used to use: you'd pay twenty-five dollars or whatever, and they'd mail you xeroxes of old research papers. It was just a very, very different world. I learned most of my programming when I had a grand total of like three reference books. . . So I'd be all proud of myself for having figured something out, and then I'd find it was just classic method and they did it better than I did.⁸⁷

The lack of resources for development information, and the absence of any unified programs for the “field” of game development, meant that new innovations and techniques appeared through the games themselves. New releases heralded improvements in graphics and experiences that showed what was possible and commercially viable. The progression of the technical state of the art became effectively a form of show and tell. For id Software, this progression came to define the company. *Commander Keen* showed the viability of smooth, console-like sidescrolling on the PC; *Wolfenstein 3D* the capacity for smooth 3D-like graphics and texture mapped mazes. And *Doom* foreshadowed the possibility for the fluid 3D networked gameplay delivered by *Quake*. Because of id's games popularity — and their perceived technical achievements — it is possible to construct a historical sketch of both the sources of their technical knowledge, and how that knowledge percolated through to the larger field of game development. We look at the material available for a consideration of the enacted knowledge embedded in id's work, and show how that work connected to knowledge dissemination structures in the game development community of the time. This study primarily focuses on *Doom*, and the basis for its rendering on the MS-DOS machines it targeted in 1993.⁸⁸

⁸⁷[168] pg.43

⁸⁸More specifically, it targeted MS-DOS 6.22 running on 386 architectures. This is an important point as 386 machines did not support floating point arithmetic, which made 3D graphical applications difficult to implement.

Foundational Knowledge

In an email to Stanford’s Game Business list, a young developer remarking on the current (2016) state of virtual reality research flippantly commented that the last time VR had the spotlight, “computers could support *Doom* . . . and that was about it.” This comment referred to the primitive nature of 3D graphics available to the average user in the early 1990s, and alluded that that limitation was a primary reason for the failure of the early VR industry.⁸⁹ In response to this comment, Bryan Salt, a developer active during the first VR boom took umbrage with the younger’s empathic (and apparently obvious) statement about *Doom*’s support.

Computers didn’t support Doom, some very creative people worked past the limitations of the time. . . when we saw Wolfenstein we were not too phased, but when Doom came out we did a collective look around the room as if to say “shit, why did we even bother” . . . It’s easy to look back and think, well that was an obvious way of doing things, that Doom etc. was inevitable, but it needed a radical shift in perspective (no pun intended).⁹⁰

So what was this shift in perspective, and how were did id envision and pull it off? First and foremost, the basic technical strength of the team was steeped in significant achievements dating back to early childhood. Both *Doom*’s engine programmer, John Carmack, and development tools programmer, John Romero, had histories of programming achievement dating back to early childhood. Romero kept fastidious records of his early games, and his development notebooks are riddled with juvenile drawings and scribbling surrounding Apple II 6502 assembly code. Many of Romero’s games featured on the cover of Uptime, a hobbyist programming magazine, while he was a teenager. He even published a book on

⁸⁹For a fantastic overview of the hype and hyperbolic prognostications of the early 1990s VR boom, see [175] Rheingold, Howard. *Virtual Reality*. New York, N.Y.: Simon & Schuster, 1992.

⁹⁰Salt, Bryan. “John Carmack says VR devs are ‘coasting on novelty’”, sent to gsb-videogames@lists.stanford.edu on October 10, 2016.

Apple II arcade game programming while still in high school.⁹¹ By the time of id's official formation, Romero had already programmed, designed, or produced multiple games a year for nearly a decade.

In a similar vein, Carmack's childhood programming efforts found him reverse engineering Richard Garriott's *Ultima* code and immersing himself in advanced graphics programming.⁹² Both Johns' expertise in programming resulted from significant auto-didactic efforts, and those experiences set the foundation for their eventual pioneering work on *Wolfenstein 3D* and *Doom*. Pioneering in this instance is not in the development of wholly unknown algorithms, but in the combination and application of expertise in multiple areas that allowed for the creation of both games. The progression toward id's 3D games can be seen in the team's preceding work for Apogee software and Softdisk. As noted in the object and technical expressions sections above, file organization strategies — like separate level and engine data — and algorithms — like zone memory management and raycast rendering — are apparent in id's earlier 3D works, and even some of their grid-based 2D games.

A Shift in Perspective

The advancements of *Wolfenstein 3D* and the *Doom* are built on the confluence of two major undercurrents in id's development work. The team's earlier successes with *Commander Keen* provided financial support for both a new round of top-of-the-line equipment — NeXTStep workstations — allowing for the development of a robust development pipeline and, probably more importantly, the time to

⁹¹Uptime covers, and article on game programming are available on Romero's SmugMug account <https://romero.smugmug.com/>.

⁹²Carmack recently remarked that he spent a lot of time trying to recreate the three-dimensional rendering on the cover of James Foley's and Andres Van Dam's *Fundamentals of Interactive Computer Graphics*, probably *the* primary source for college and professional level graphics algorithms and approaches in the 1980s.

work through the technical advances needed for Carmack's 3D engine designs. Up until *Wolfenstein 3D*, most of id's games, through their contracts with Apogee and Softdisk, were designed and engineered inside a month or two. id's deal with Softdisk at the time explicitly hinged on the monthly publication schedule for the Gamer's Edge. When more development time began to be necessary for id's 3D games, the team either secretly offloaded the work to Apogee, or relied on previous game engines that did not need further technical modifications. This freed up Carmack's time for research. During *Hovortank 3D*'s development, engine work took nearly six weeks, leading Apogee to develop Scuba Steve for Softdisk as a way to take some pressure off of id.⁹³

According to *Masters of Doom*'s narrative, Carmack looked around for 3D approaches at the beginning of *Hovortank 3D*'s development, and finding nothing suitable, rolled his own approach. Given that we know that Carmack had exposure to graphics programming books and even research papers before this period, and that *Hovortank 3D*'s engine is based on raycasting (a diminutive form of ray tracing), this narrative is probably a simplification. Foley et al's *Computer Graphics: Principles and Practices* from 1990 devotes some coverage to ray-casting as a form of visible surface rendering, and Carmack is noted in Pinchbeck's work as having used that book's notes on binary space partitioning in the construction of later *Wolfenstein 3D* ports and as the basis for *Doom*'s rendering engine. Regardless, returning to the dual contributions fostered at id that lead to the advancements of *Doom*, we have Carmack's dedicated engine development time — 6 weeks for *Hovortank 3D*, 6 months for *Wolfenstein 3D* and nearly a year for *Doom* — and Romero's contributions to development tools and workflow as the key ingredients for *Doom*'s shift in perspective for the game development community.

⁹³id's Commander Keen series was lucrative enough that Scott Adams at Apogee figured whatever the next thing coming out of id was, it was worth supporting.

The team’s NeXTStep development process included a full suite of tools for level creation (DoomEd(itor)), sprite illustration (Fuzzy Pumper Palette Studio), and cross-compilation to DOS execution. NeXTStep’s windowed interface management allowed the game’s designers, Romero, Hall, and Peterson, to effectively live edit changes in the *Doom* level editor and then quickly load them up and play through them in a separate window on the same machine.⁹⁴ This process allowed for quick iteration on level designs for *Doom*, which differed significantly from *Wolfenstein 3D*’s efforts due to flexibility provided by variable heights for platforms and adjacent sectors of the game map. *Doom*’s designers had to break from the conceptual space of 2D mazes that had characterized their earlier efforts and articulate a new design language for the levels featuring arbitrary angles for walls and elevation.

id’s games rested on the backbone of Carmack’s engines, and the advancements of *Wolfenstein 3D* and *Doom* built on his familiarity with modern graphics techniques, and ways of getting them running on home computers. The movement to *Wolfenstein 3D* from its predecessors mainly relied on its ability to improve the speed of the game through low-level memory operations. As alluded to in Bryan Salt’s quote above, *Wolfenstein 3D*’s raycasting engine appeared similar to other contemporaneous advances at the time. DoomWiki describes *Wolfenstein 3D*’s engine as a “fairly textbook ray casting engine”, and the game’s visuals did not move too far afield from other titles using similar techniques. Carmack most likely relied on his years of experience with optimizing assembly graphics code, and mentions works in that vein, like Michael Abrash’s Zen of Assembly Language, and

⁹⁴It is unclear whether this level editing framework was in place for *Wolfenstein 3D*, given that the game’s levels were comparatively much simpler (grid-based), and did not include differing elevations or the potentially complex line of sight issues that windows and the like afforded *Doom*’s levels. *Doom*’s editing interface is described in the first issue of Game Developer Magazine [27] in January 1994.

Power Graphics Programming, as being influential to his understanding of such optimizations. A focus on speed drove the efforts of *Wolfenstein 3D* and then *Doom*, and it is that speed that surprised other developers at the time of *Doom*'s release.

The main advancement for *Doom*'s engine, though according to Carmack also a bit overblown,⁹⁵ was its use of binary space partitioning (BSP) coupled with the memory management advances leveraged from *Wolfenstein 3D*. A major challenge in any 3D engine is to ensure that the computer only spends time and uses memory for exactly what is has to show at that current moment. As Michael Abrash remarked at the time, 'programming is a process of caching', in that most effort to get things running was in finding ways to reduce the amount of work done by the machine. In many cases this meant finding ways to avoid repeating work and storing the results of complex computations before making use of them at runtime. *Wolfenstein 3D*'s engine, for example, relied on projecting rays from the player position at a fixed number of angles and then computing their intersection with walls drawn on the 2D grid representing each level. To cut down on computation time, many of these angular calculations were pre-computed and stored in a look up table, which provided a cheaper computational means of resolving the intersection equations. Similarly, though more complexly, *Doom*'s BSP approach provided a way to quickly look up the ordering of walls displayed in front of the player, and to remove anything not viewable.⁹⁶

Doom's BSP algorithm pre-organized all the walls in map into a binary tree by choosing a starting wall, and then recursive adding other walls as nodes that were either in front of or behind a given root node. By locating the player at a

⁹⁵[168] pg. 43

⁹⁶This process is known as frustum culling and *Doom*'s BSP trees, due to the way they partitioned each level's spaces, could also identify and drop whole sets of walls that would not appear in the player's view.

specific coordinate in the map, the engine could quickly ascertain which walls were behind the player and immediately ignore their rendering. Additionally, for those areas appearing in front of the player, the engine could then find the closest wall and traverse the BSP tree to render everything behind it, all without doing any significant ordering calculations. The trade off in processing occurred because the map data could be pre-computed and embedded with the BSP sectors and their relative positioning. Lee Killough, a prominent *Doom* modder in the late 1990s, spent a significant amount of time further improving id’s own BSP compilation engine since it formed the most severe development bottleneck wherever BSP nodes needed to be recalculated.

Carmack most likely borrowed this approach by combining work done by Bruce Naylor — the inventor of BSP trees — and research into applications of BSP trees in front-to-back rendering schemes. Killough, who also maintained a *Doom* historical website in the late 1990s, believed that Carmack’s implementation most resembled work published by Dan Gordon and Chen Shuhong in the September 1991 issue of *IEEE Computer Graphics*.⁹⁷ The “shift” in *Doom* was then not a significant advance in the state of the art in graphics algorithms, but in the ability to get those optimizations to run on a 386 machine without proper floating point support. Additionally, the use of BSP in 3D real-time game engines was also new, and led to a rash of research after *Doom*’s release into “Doom Style Rendering” and the use of BSPs for world generation.

Another cautionary note about the need for further research into the underlying algorithms and approaches of game development is the still common misconception that *Doom* made use of a raycasting engine. In order to optimize texture drawing, *Doom* still rendered its walls using columns of pixels that allowed for

⁹⁷Carmack had apparently been searching for ways to get *Wolfenstein 3D* running at an acceptable speed on the Super Nintendo Entertainment System, and began experimenting with BSPs at that point.

memory accesses to texture files to be loaded in order from memory, each column being rotated 90 degrees to align with left-to-right orientation of memory access. This caused many to assume that Doom used column-based raycasting because the walls appeared similar to those in *Wolfenstein 3D*, and upon Doom’s release the initial discussion on programming listservs focused on how it was possible to get a raycaster (like *Doom* “obviously was”) to draw with the speed and complexity shown in play.⁹⁸ This form of confusion, termed the “Talespin Effect” by Noah Wardrip-Fruin, also mucks up certain theories of technological progress narratives.⁹⁹ One example for Doom is Dominic Arsenault et al.’s discussion of technical innovation, that incorrectly lumped *Wolfenstein 3D* and *Doom* together technologically, whereas in actuality Doom was a significant breakthrough in Carmack’s implementation of 3D engines.¹⁰⁰

Resources

The basic assertion of this section, that *Doom*’s shift of perspective was based on previous expertise and a connection between difficult to access academic knowledge and a forward thinking notion to implement it on common consumer hardware, deserves significantly more historical attention. *Doom*’s graphical engine influenced the development of further 3D works, with the BUILD engine borrowing *Doom*’s notion of sector-based level structure (as opposed to grids), and numerous game programming sites devoting significant time to BSP rendering techniques in the mid-1990s. The PC Game Programmers Encyclopedia, a collection of 3D

⁹⁸[4] “DOOM 3D Engine Analysis - Google Groups.” Accessed December 27, 2016. <https://groups.google.com/forum/#!topic/rec.games.programmer/e0n1umKpuUA>.

⁹⁹[216] Wardrip-Fruin, Noah. *Expressive Processing*. Cambridge (Mass.): MIT Press, 2009. pg. 115

¹⁰⁰[29] Arsenault, Dominic, Pierre-Marc Coté, Audrey Larochelle, and Sacha Lebel. “Graphical Technologies, Innovation and Aesthetics in the Video Game Industry: A Case Study of the Shift from 2D to 3D Graphics in the 1990s.” *G|A|M|E Games as Art, Media, Entertainment* 1, no. 2 (2013).

engine programming techniques compiled online in 1994, featured contributions by many community members who had helped to reverse engineer *Doom*'s engine structures.¹⁰¹ The flows of information from these efforts and around the development communities of the time, as encapsulated in forums, online guides, and other released games, could provide a rich base for investigating knowledge transfer in the early 3D engine community. After the release of *Wolfenstein 3D*, and certainly after *Doom*, the basic thought that engines like these were possible at arcade speeds led to increased academic interest in BSP rendering, and even convinced IBM to develop WebView3D, a BSP-based *Doom* engine clone as an internal research project.¹⁰² Surely, *Doom* is important, but it might be more useful as a means to pry open the networks of expertise and communication that existed at the time, and that form the base for most modern 3D engines and game worlds.

6.5 Conclusion

The model of *Doom* presented and analyzed above would not be possible without the insights and practical experience gained through the earlier work in this thesis. By taking a deep look through the documentation of *Prom Week* in Appraisal, it became apparent that significant segments of documentation about games and their development was being overlooked and under-collected. Divining out the various, fractal documentary vectors of *Doom*'s software development only occurred after working through, with practical methodologies, the constraints of *Prom Week*'s accumulation. Above, we aligned the model's layers: historiography,

¹⁰¹[11] "PC-GPE on the Web." PC-GPE on the Web. Accessed December 27, 2016. <http://bespin.org/~qz/pc-gpe/>.

¹⁰²[197] Stephens, Philip J. "Writing a Fast 3D Graphics Engine," September 8, 1995. <http://www.gamers.org/dEngine/doom/papers/webview.ps.gz>.

reified object, technical expression, and enacted and tacit knowledge, with historical and methodological arguments aimed at revealing more potential histories for *Doom*.

1. *Historiography* laid out the terrain of *Doom* source documentation, and revealed the lengths that *Doom*'s community has gone to to preserve its history. These investigations also reveal the depth of documentation available and waiting for exploration by historians, along with the limitations of current narratives.
2. *Reified Object* argued that historical analysis of the form and content of executable objects can reveal new insights and pose new questions about the development and organization of software programs and the people behind them. In highlighting the version progression of *Doom*'s executable data, and noting the evolution of file organization in id Software's early releases, we showed that the game was not a static, singularly historicizable object, but a multiplicity of continuing relationships between the object, practitioner and player. Further, the implicit use of the GISST toolset in helping with comparative analysis made this work easier, and time-permitting, future work on this layer will make extensive use of it.
3. *Technical Expression* presented the history of *Doom*'s source code, and its release to the wider community. It also showed that the conditions of *Doom*'s source code and its contextual documentation are what made it possible to maintain an active 25-year long development community. The conditions of source code are contingent on other historical and social factors. In isolation, program code may not be a readable or recoverable object without complementary knowledge of its creation (and potential standardization efforts).

4. *Enacted and Tacit Knowledge* brought together some theories about knowledge production as it relates to narratives of progress and individual historical contributions. In the main, the section showed that *Doom*'s graphical achievements telegraphed a shift in the application of more advanced graphical techniques toward home computers. It also pointed out that most of the work on *Doom* had precedents, in that other contemporaneous titles also made use of similar techniques. Understanding the lineage and influence of technical knowledge, and its diffusion is important to understanding the modern world. As noted by Michael Mahoney, the “operational models” embedded into software systems come from somewhere, and need to be nailed down, and dissected to trace their influences.¹⁰³

The purpose of this chapter essentially boils down to locating some holes, and trying to shove some initial pebbles and debris into them. In looking at the historical model, the point is to realize that many areas of software history are still not being explored and that they lack the documentary support for fully realized analysis. The earlier chapters on methodology help in this regard. Appraisal formed the seed for the *Doom* investigations, while Description provided a rationale for the enactment of different executable objects and the needs for their disambiguation. Citation introduced a means for the use of more active and engaging references to software objects, and a system, GISST, that directly addressed issues in the “object” layer of the model. The Discovery chapter's connection is more tenuous, but the sheer amount of development information available — *Prom Week* development had about 17,000 — the use of machine learning techniques for the exploration and organization of historical content is an obvious new direction.

¹⁰³Another notion related to this is “operational logics” proposed by Wardrip-Fruin and Mateas [139], which are in some sense how an “operational model” is enacted in software and how those enactments function as a fulcrum for computational meaning-making.

Doom's documentation also represents a new formation, in that its progression mirrors that of the growing dissemination network of the early Internet. Most technical objects nowadays will have similar documentary profiles to *Doom* — albeit with shorter chronologies. Leveraging and managing these various resources will continue to occupy a significant amount of an historian's time and efforts. The approach to and model of *Doom* above is therefore easily transferrable to similarly organized and networked objects.

In closing, and continuing with remarks on the “network” of *Doom*'s documentation, there are significant areas of *Doom*'s historical production that did not make it into this analysis. Namely, *Doom*'s development as an e-sport and the online culture dedicated to its competitive play is a significant historical condition deserving of more scrutiny. Luckily, the same apparatus we used to look into *Doom*'s more singular existence and material can easier expand to this competitive arena. The sources of documentation are virtually the same, save the gameplay recordings of *Doom*, the LMP files that replay its engine inputs. Beneficially, the model and tools we've developed in this thesis are flexible enough to accommodate. GISST can learn a new import format and run *Doom* versions with demo playback, and the changes to *Doom*'s object become even more apparent when considering the modifications necessary to improve and maintain its multiplayer apparatus. There is definitely still more work to do, and more holes to fill.

Chapter 7

Conclusion

This conclusion has two express goals. The first is to reiterate and tie together the work presented in the thesis at large, and the second is to extend the themes of this work into the future. Attending to the former requires a basic summation of the work of each chapter (and is accomplished in the next section). The latter, as an extension of this thesis' focus on the material and methodological concerns of software historical study, is a bit more fraught. While the work presented above attempted to provide a basis for intervention into — and improvement of — various aspects of software scholarship, those very interventions are tied to our current historical moment, one that is rapidly shifting and evolving as regards the place and process of future historical studies.

Following the summation of completed work, the next section will describe how all of the efforts in this thesis are attached to important historical contingencies. Contingencies that need to be further described and considered if historical study is going to progress at all. These contingencies all involve the ever going shift from a pre-networked to a post-networked culture that happens to align quite well with the rise of computer games and software. In one sense, this contingency is obvious, given that that very software is responsible for the creation of networked culture.

However, it is also apparent that current cultural and historical structures are still not comprehending or adequately preparing for the effects of the networked world on most forms of records, not only software ones. The “network” contingency operates on all the points of intervention highlighted in the introduction to this thesis, affecting the accumulation of historical records, how they can be explored, and what it is possible to express about them. This contingency also functions on software and games as well, since they are becoming more and more of the network with each passing year. In this sense, all the above work in this thesis is playing catch up to an older paradigm of historical methodologies and studies, and there probably will not be time to solve all the problems presented by non-networked software before needing to address the coming networked deluge.

Obviously, since this is the conclusion to a work, there is not space to solve — or really even fully address — the coming contingencies so much as to gesture at their implications in light of the research presented above. The concluding section of this chapter, and this thesis in general, presents a brief description of the current “standard” model of historical software, its documentation, and the management of its objects. Following that, is a discussion of how the imposition of networked technologies onto this model has significance methodological and material consequences for future scholarship. Both the meta-methodological and meta-documentary concerns that arise from networks are then folded back onto the work of this thesis to locate potential seeds that can use its methods as a fertile basin for the sprouts of new, network contingent approaches.

7.1 The Conditions of a History for Games and Software

The purpose of the first four chapters in the Appraisal, Description, Citation, and Discovery of computer game software and its records was to start to piece together a broad set of applied methods to various intervening points in a model of scholarly process. Recall, in the introduction to this thesis, the simple model of scholarly process consisting of three nodes: practitioner, scholar, and audience, and two edges: historical accumulation, and exploration / expression. Each chapter intended to show how steps could be made into the improvement of each edge to the benefit of software history. The first two chapters, Appraisal and Description, sought to stabilize the accumulated records of software through the adaptation of structures and systems from library and information science. Citation and Discovery, complementarily, attempted to stage similar interventions into the exploration of stabilized records, and their use in historical expressions — or argumentations. The fifth chapter, with its focus on *Doom* aimed to combine some of the insights of the previous chapters to show how new things could be said about the game that would not have been possible without those chapters insights. This led to the formation of a new historical model for the investigation of computer game software, based in part on the requirements made manifest in the Appraisal chapter, described in Description, and enacted through Citation. The needs of divining and locating the historically salient records of *Doom* (appraisal) led to a theorization of those records descriptions and historical conditions as stored material (description), and the need to find ways to leverage and retrieve those records for later discussion (citation).

The model of intervention, and the model of historical software study from *A*

Model of Doom, both point toward the holes in the current historical practices around software. As interventions, much more work needs to be done in stabilizing and accessing historical game records, and as material resources, much more effort is required to acquire and retain records of game objects, their technical expressions — source code and the like — and the tacit knowledge required for their creation. The work above took some basic steps along both of these paths:

1. *Appraisal* adapted methodologies from historians and librarians of science and applied them to the development documentation of the computer game *Prom Week*. This prefaced the model of historical software (in *A Model of Doom*) by pointing out the types and kinds of records that, hitherto, have not received much scrutiny from archives and collections.
2. *Description* presented a theory of the contextual definitions that are required for the accurate description of computer game platform and media format records. It also pointed out the need for better descriptive practice and showed how those practices can benefit from an understanding of complementary structures in library and information science works.
3. *Citation* introduced the Game and Software Scholarship Toolkit (GISST) as a way to operationalize (in scholarly argumentation) the records of software objects and their data that may someday be stored, en masse, in institutional repositories.¹ It also worked to show how new forms of records about games, and even the games themselves might be leveraged to fill in the holes in the software historical model, and enable new forms of intervention into historical expression.

¹As noted in the Appraisal chapter, the treatment and storage of *Prom Week*'s born-digital records is very uncommon. Additionally, while some forward thinking institutions are beginning to store and describe historical software data for academic use, the practice is not very widespread and does not currently have any standardized and adopted solutions.

4. *Discovery* applied statistical natural language processing techniques to the location and comparison of historical game records. The approach showed that new methods from computer science, in textual analysis and information retrieval, could be applied to and benefit the historical study of computer games and software. An intention of the discovery work was to literally visualize the missing areas in the history of games, and point toward new potential spaces of study.

In *A Model of Doom*, these threads coalesced into a fractal, multi-vectored investigation into different aspects of *Doom*'s history. Starting with *Doom* historiography, and the sources available to construct histories of the game, the model argues for a deeper look into the material conditions of software history and the benefit of looking past more superficial and chronicle-like historical discussions of the game. The model places these historiographic assumptions at the top of a layered model of software history, where other perspectives exist, deserving of further scrutiny. Below the historical object, there are its various versions, the literal executable objects of discussion. And below those, the technical expressions that compile into playable games. Deeper still is the bedrock of tacit knowledge shared among *Doom*'s development team, and based on their previous experience and the technical information available to them, that provided the means for the creation of *Doom*'s systems and initial conceptualization.

In order to leverage all the layers of this model, the progress started in the earlier methodological chapters must be expanded. GISST's existence helps with the organization of the *Doom*'s objects and the appraisal strategies point toward areas of development and documentation to look for while also showing new locations specific to *Doom*'s long history. The next section on the influence of networks on the historical study of software is already present in the analysis of *Doom*. As

an early, network-disseminated piece of software, *Doom*'s documentation contains within it the genesis of both networked game objects, and the networked sources of information that slowly accreted into the online communities of the 21st century.

7.2 The Network Contingency and its Implications on Practice

In Vernor Vinge's novel *A Deepness in The Sky* humanity's interstellar existence eclipses a vast swath of the galaxy.² An enormous trade empire, the Qeng Ho, operates over this territory as a capitalist, networked collective of mercantile families all united by a common set of practices, and a common base of advanced technologies and software. The Qeng Ho's technological stack is a complex, and confusingly intricate, accretion of centuries of software upgrades and modifications. In fact, most of their programming efforts — aimed at keeping expansive network of spaceships adrift — involve a good deal of archaeology. The needs of any specific voyage cannot be known in advance. They assume that as the programmatic requirements change upon encountering new phenomena that suitable solutions can be imported from previous efforts. As a result, the Qeng Ho stack is a library of programs stretching all the way back to the Unix operating system, a complete accumulation of all previous software. When a need for new programs arise a class of archaeologist-programmers dive into the networked past of all previous Qeng Ho outputs to piece together new solutions from older parts. The success of their civilization then depends on their capabilities for networked information retrieval and re-appropriation. They need to find, and understand older software quickly in order to stay ahead of the trials of space travel. For

²[215]

them, information and computer science methods enable their entire civilization.

Now while this dependence on, and deep acknowledgement of, knowledge management issues is professionally edifying it is also unsurprising given Vinge's background — he was a practicing computer scientist after all. However, his perspective on the future's dependence on software maintenance is not far off the mark. The problems of the future are the same as the problems of understanding that future's history. Namely, how to find and make use of things born of, and enmeshed in, the network long after their introduction to it. To bring this discussion back down to the historical scope of this thesis, the rest of this section will elaborate on the implications of the network contingency on the work conducted above, and on the future of the historical study of software.

In enlisting methods from information and computer science, we also enlist certain assumptions about the objects those methods seek to evaluate. For instance, in the Appraisal chapter, we took guidance from older reports by historians of technology seeking to better describe what at the time were the completely new paradigms of digital information and long-term research data storage. The preservation of software is an afterthought in most of those accounts, and all the documentary guidance is mainly directed towards physical records. The notion of a network of data, or any kind of computational network structure at all is not present. As such, in the appraisal of *Prom Week* and the translations it necessitated for applying those older paradigms to newer born-digital documentation, some structures had no significant analog. *Prom Week*'s team made extensive use of cloud storage, and networked version control management systems. This created numerous issues for the accessibility and reconstruction of the game's development documentation. Furthermore, *Prom Week* itself is a network-disseminated executable, designed to be embedded with the frameworks of other websites well

outside of the appraisal's scope.

Prom Week, through its executable, also straddles the divide between singular game play experiences and networked play architectures. It mostly belongs to what will be termed the *standard* model of game historical study. This *standard* applies to executable software that does not rely or required a network connection for replay or re-experience. It is not *network contingent*, in that future preservation and historical efforts will not need to take the network into account. The lack of network dependence is implicit to the methods of this thesis, and needs to be acknowledge as a significant next step for future work. The vocabularies in the Description chapter, and the records they attend to, all assume a singular, fixed object instead of a constellated one. Similarly, the objects referenced in Citation, specifically the games under management by GISST, are all singular experiences that can be expressed through isolated, non-networked emulations. While accounting for networked state and other more complex data management schemes is possible, there are other significant difficulties that arise from network contingent artifacts. Difficulties that must be addressed in further *network contingent* extensions of research into *standard* practices that constitute the majority of this thesis.

7.2.1 Problems of the Network Contingency

Computer games are getting more technologically complex. Major game development studios are much larger than they used to be, and teams for so-called AAA titles are incredibly large. Over 1000 people made Grand Theft Auto V, in comparison to the 10 or so that worked on *Doom*. Modern development records are then dependent on toolsets and technologies that are significantly more complex and sprawling than those of even a decade ago. And, as shown above, even

a small development effort results is a mess of shared documentation and support programs also enmeshed in network practices and contingencies.³

Most modern games are also distributed over a network, do not have physical dissemination of any kind, and require some form of network connection for play or patch updates. When a system is dependent on multiple parts that are asymmetrically disseminated, reconstructing the object and its played experience become much more difficult. “Asymmetrical dissemination” means that the player of a game does not actually have all the data needed to run it without some form of network transaction; there is an asymmetry of access. In the past, all distribution was symmetric by necessity. Without a networked data source, all the required resources to run a game needed to be include with the distributed media format or the game would not function. Historical methodologies must account for this asymmetry going forward. If the foreign server turns off at some point, that part of the game system is most likely irrecoverable. If that server is gone, the game client will no longer function. Its history is in jeopardy.

A potential route out of this problem is to emulate the entire network responsible for the game. Get all the server and client software functioning through emulation and then hopefully recover the gameplay. On a technical level, while this might be feasible, you have the issue that any significantly networked game or activity probably also had a contemporaneous community associated with it — recall the elaborate *Doom* community described above — and a social space with the game that cannot be recovered. This phenomenon is most striking in massively multiplayer online games, where the game is essentially an operationalization of a social world. If you recover the whole system, you really do not get back much of the experience. This lack of recovery is a reason for the inclusion

³This section is based on text from the blog post Kaltman, Eric. “Current Game Preservation in Not Enough.” *How They Got Game*, June 2016. <https://web.stanford.edu/group/htgg/cgi-bin/drupal/?q=node/1211>. [101].

of “performances” in GISST. If a game disappears, at least a video recording is better than nothing. James Newman, in *Best Before*, argues that in many cases game play recording is superior in practical archival methodological terms, since the network contingencies do not affect static video and its preservation characteristics are well understood.

The major problem with full network emulation is access to the information necessary to recreate a system or network of systems. Most game servers are proprietary, considered corporate secrets, and will never be disseminated in a manner that allows for their emulation. That is not to say that with significant reverse engineering effort you cannot create something that imitates a proprietary service or server, but that the effort required is incommensurate with any reasonable allocation of institutional resources. To save a network contingent game is to save its whole network.

Another pressing issue is the coming scale of the issue. In the far future, looking back at the past like those Qeng Ho archaeo-programmers, what will it look like? Based on the current progress of networked culture, a majority of all software and games will be in the network-contingent category. (In fact, as the statistics below will show, a case can be made that this is already the case.) In the figure below, the problematic trend of network contingent game production becomes apparent.

Backing this claim up with some (albeit slightly problematic) data is not difficult. MobyGames, a website that tries to account for all historical game titles, lists around 55,000 in their database. Of those titles, around 20,000 are for some version of Microsoft Windows. Steam, a popular computer game distribution service has a total of 8,900 games available for play on the current version of Windows.⁴ Therefore, roughly half of all the game created for Windows platforms,

⁴This is a bit more complex, since Microsoft Windows 10 is compatible with many earlier ver-

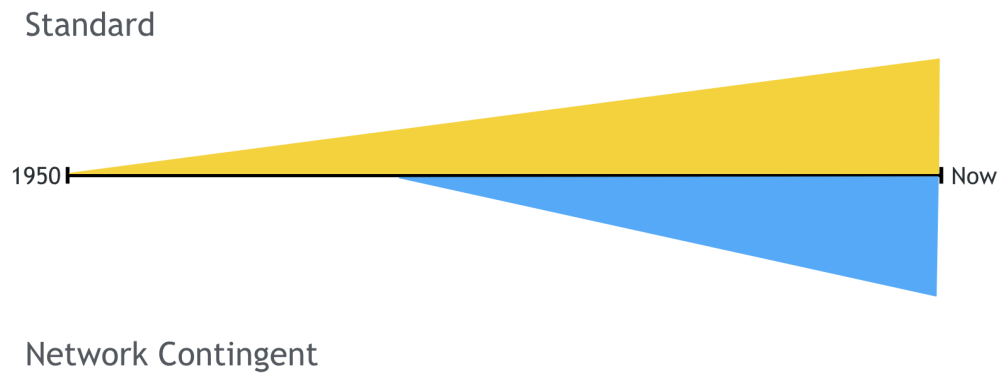


Figure 7.1: Growth of Network Contingent Games 1950 to Now

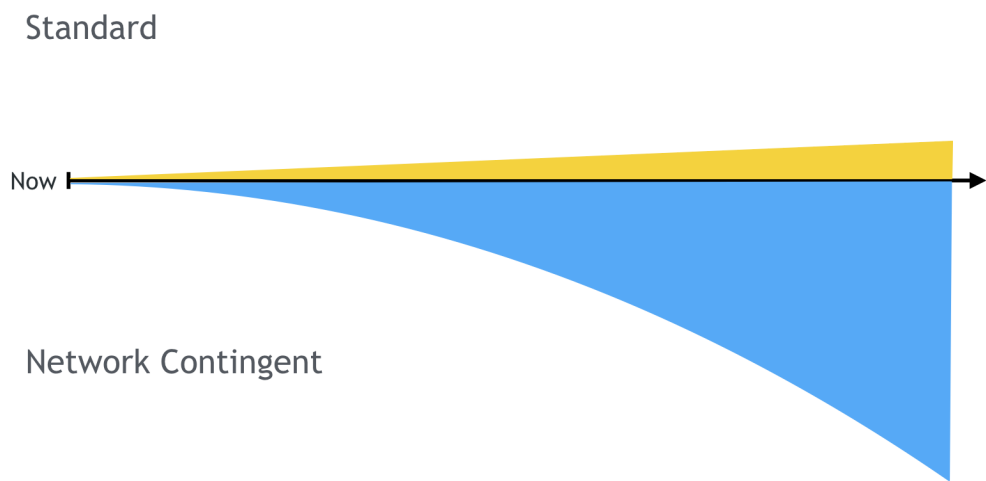


Figure 7.2: Growth of Network Contingent Games Into Future

since the platforms inception in 1985, are currently available for Windows 10, network-contingent, and network distributed through a single service.

The situation gets more drastic when considering mobile game titles. Games are the largest segment of applications available through Apple's App Store for iOS are games. According to recent numbers from Pocket Gamer, an app tracking and analytics web site, there are currently 776,661 active games available for download. Therefore, based on MobyGames's numbers, there are currently an order of magnitude more games on iOS then have been created for the entire history of computer games. The scale of production is immense, and none of those games fit directly into the methodologies proposed above for more standard game objects. Now, most of these games are probably not historically significant, but as noted in the Appraisal chapter, that is not an easy or simple prescription to make.

The next two figures highlight another important implication of the network, which is that most future games will be disseminated without a physical form.

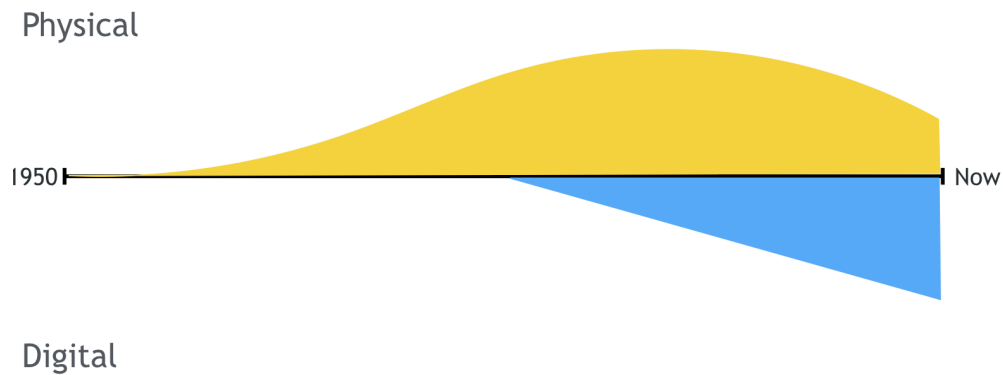


Figure 7.3: Network Distribution 1950 to Now

sions. Therefore, these 8,900 are not exclusively for a specific version of the Microsoft Windows platform.

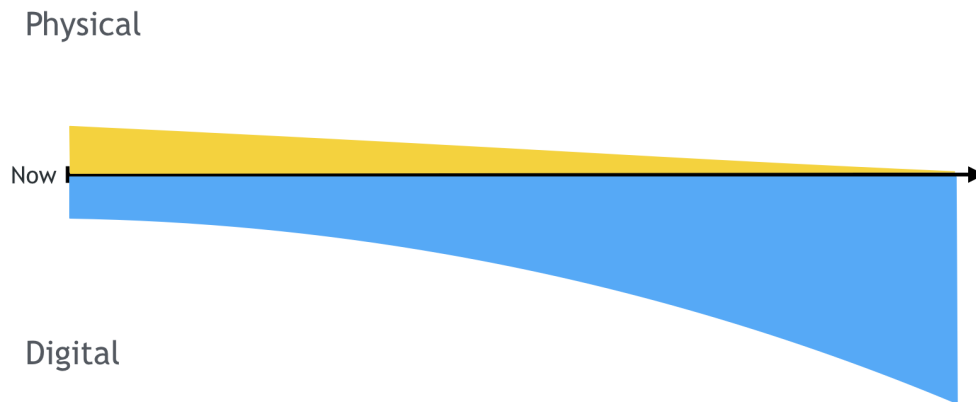


Figure 7.4: Network Distribution Now Into Future

The incredible production rates of current games, and the inability to currently preserve them all will lead to a situation where predominantly single player, standard games will be overrepresented in the “playable” record. If developers and designers disseminate more information about their development practices, or take on a standardized framework for the dissemination of legacy source code and development documentation, then those breadcrumbs could help prevent a significant loss of more network contingent objects. One cannot reconstruct an entire loaf from the crumbs, but you can gain a sense of its texture, ingredients, and baking process. And some sustenance for a hungry historical scholar.

Bibliography

- [1] Apogee - What's Wrong With Them?!?!? - Google Groups, . URL [https://groups.google.com/forum/#!searchin/comp.sys.ibm.pc.games/doom\\$20before\\$3A1992\\$2F12\\$2F01%7Csort:date/comp.sys.ibm.pc.games/pyKjy2UjBHU/w_brunBkYoUJ](https://groups.google.com/forum/#!searchin/comp.sys.ibm.pc.games/doom$20before$3A1992$2F12$2F01%7Csort:date/comp.sys.ibm.pc.games/pyKjy2UjBHU/w_brunBkYoUJ).
- [2] BIBFRAME - Bibliographic Framework Initiative (Library of Congress), . URL <https://www.loc.gov/bibframe/>.
- [3] Departure from Wikia, . URL https://doomwiki.org/wiki/Doom_Wiki:Departure_from_Wikia.
- [4] DOOM 3d engine analysis - Google Groups, . URL <https://groups.google.com/forum/#!topic/rec.games.programmer/eOnlumKPuUA>.
- [5] DOOM Alpha v0.3 (aka "DOOM Pre-Alpha") - Doomworld /idgames database frontend, . URL https://www.doomworld.com/idgames/historic/doom0_3.
- [6] Doomworld – The 11th Annual Cacowards, . URL <http://www.doomworld.com/11years/>.
- [7] GAMECIP - Game Metadata and Citation Project, . URL <https://gamecip.soe.ucsc.edu/>.
- [8] NES Cafe - Nintendo - NES Emulators - Zophar's Domain, . URL <https://www.zophar.net/java/nes/nescafe.html>.
- [9] OpenCyc, . URL <http://www.opencyc.org/>.
- [10] PBCore - Public Broadcasting Metadata Dictionary Project, . URL <http://pbcore.org/>.
- [11] PC-GPE on the Web, . URL <http://bespin.org/~qz/pc-gpe/>.
- [12] PREMIS: Preservation Metadata Maintenance Activity (Library of Congress), . URL <https://www.loc.gov/standards/premis/>.

- [13] The Suggested Upper Merged Ontology (SUMO) - Ontology Portal, . URL <http://www.adampease.org/OP/>.
- [14] Zotero | Home, . URL <https://www.zotero.org/>.
- [15] *Book of Id.* 1996.
- [16] ANSI/NISO Z39.19-2005 (R2010) Guidelines for the Construction, Format, and Management of Monolingual Controlled Vocabularies - National Information Standards Organization. Technical report, National Information Standards Organization, 2005.
- [17] rebecca heineman, March 2016. URL <http://www.nodontdie.com/rebecca-heineman/>.
- [18] Computing platform, March 2017. URL https://en.wikipedia.org/w/index.php?title=Computing_platform&oldid=772442103. Page Version ID: 772442103.
- [19] Development of *Doom*, June 2017. URL https://en.wikipedia.org/w/index.php?title=Development_of_Doom&oldid=784107014. Page Version ID: 784107014.
- [20] List of video games considered the best, June 2017. URL https://en.wikipedia.org/w/index.php?title=List_of_video_games_considered_the_best&oldid=783239311. Page Version ID: 783239311.
- [21] Espen Aarseth and Gordon Calleja. The Word Game: The Ontology of an Undefinable Object. In *The Philosophy of Computer Games Conference*, 2009.
- [22] Espen Aarseth, Solveig Marie Smedstad, and Lise Sunnan\aa. 3. A Multi-Dimensional Typology of Games. 2003.
- [23] Espen Aarseth, Lev Manovich, Frans Mäyrä, Katie Salen, and Mark JP Wolf. Define real, moron! *Some remarks on game ontologies*. In *Stephan Günzel, Michael Liebe & Dieter Mersch (Eds.), DIGAREC Series*, 6:50–69, 2011.
- [24] Moshe Adler. Stardom and talent. *The American economic review*, 75(1): 208–212, 1985.
- [25] Nathan Altice. *I am error: the Nintendo family computer/entertainment system platform*. Platform studies. The MIT Press, Cambridge, Massachusetts, 2015. ISBN 978-0-262-02877-6.

- [26] American Association for State and Local History and Oral History Association. *Oral history: an interdisciplinary anthology*. American Association for State and Local History book series. AltaMira Press, Walnut Creek, 2nd ed edition, 1996. ISBN 0-7619-9189-1.
- [27] Alexander Antoniadis. The Game Developer Archives: 'Monsters From the Id: The Making of Doom '. URL http://www.gamasutra.com/view/news/112355/The_Game_Developer_Archives_Monsters_From_the_Id_The_Making_of_Doom.php.
- [28] J. Scott Armstrong. Unintelligible management research and academic prestige. *Interfaces*, 10(2):80–86, 1980.
- [29] Dominic Arsenault, Pierre-Marc Coté, Audrey Larochelle, and Sacha Lebel. Graphical technologies, innovation and aesthetics in the video game industry: a case study of the shift from 2d to 3d graphics in the 1990s. *G/A/M/E Games as Art, Media, Entertainment*, 1(2), 2013. ISSN 2280-7705.
- [30] Willa K. Baum. *Transcribing and editing oral history*. American Association for State and Local History, Nashville, 1977. ISBN 0-910050-26-0.
- [31] Willa K Baum. *Oral history for the local historical society*. American Association for State and Local History by special arrangement with the Conference of California Historical Societies, Nashville, Tenn., 1987. ISBN 0-910050-87-2 978-0-910050-87-6.
- [32] Caetlin Benson-Allott. 40 Platform. In *Debugging Game History: A Critical Lexicon*, pages 343–349. 2016.
- [33] John Berger, Sven Blomberg, Chris Fox, Michael Dibb, and Richard Hollis. *Ways of seeing*. 1973. ISBN 978-0-14-021631-8 978-0-14-013515-2 978-0-563-12244-9. OCLC: 632700.
- [34] Wiebe E Bijker. *Of bicycles, bakelites, and bulbs: toward a theory of sociotechnical change*. MIT Press, Cambridge, Mass., 1995. ISBN 978-0-262-02376-4. OCLC: 31659485.
- [35] Steven Bird. NLTK: the natural language toolkit. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, pages 69–72. Association for Computational Linguistics, 2006.
- [36] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data-the story so far. *International journal on semantic web and information systems*, 5(3):1–22, 2009.

- [37] Maurice Black. *The Art of Code*. Dissertation, University of Pennsylvania, 2002.
- [38] Hans Booms. Society and the formation of a documentary heritage: issues in the appraisal of archival sources. *Archivaria*, 24(3):69–107, 1987.
- [39] Christine L. Borgman. *Scholarship in the digital age: information, infrastructure, and the Internet*. MIT Press, Cambridge, Mass, 2007. ISBN 978-0-262-02619-2.
- [40] Geoffrey Bowker and Susan Leigh Star. *Sorting Things Out: Classification and its Consequences*. Massachusetts Institute of Technology, Cambridge (Mass.), 1999. ISBN 0-262-02461-6.
- [41] Roger B. Bradford. An empirical study of required dimensionality for large-scale latent semantic indexing applications. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 153–162. ACM, 2008.
- [42] Julian Brooke and Matthew Hurst. Patterns in the stream: Exploring the interaction of polarity, topic, and discourse in a large opinion corpus. In *Proceedings of the 1st international CIKM workshop on Topic-sentiment analysis for mass opinion*, pages 1–8. ACM, 2009.
- [43] Bruce Bruemmer and Sheldon Hochheiser. *The high-technology company: a historical research and archival guide*. Charles Babbage Institute, Center for the History of Information Processing, University of Minnesota, 1989.
- [44] Alexander Budanitsky and Graeme Hirst. Evaluating wordnet-based measures of lexical semantic relatedness. *Computational Linguistics*, 32(1):13–47, 2006.
- [45] Anne Burdick, editor. *Digital humanities*. MIT Press, Cambridge, MA, 2012. ISBN 978-0-262-01847-0. OCLC: 793581385.
- [46] Peter Burke. *A social history of knowledge II: from the encyclopaedia to Wikipedia*. Polity, Cambridge, 2011. ISBN 978-0-7456-5042-5 978-0-7456-5043-2.
- [47] Roger Caillois. *Man, play, and games*. Free Press of Glencoe, New York, 1961.
- [48] Michel Callon, John Law, and Arie Rip. *Mapping the dynamics of science and technology: sociology of science in the real world*. Macmillan, Basingstoke, 1986. ISBN 978-0-333-37223-4. OCLC: 13159883.

- [49] Simon Carless. Why game discovery is vital - introducing Games We Care About., June 2014. URL http://www.gamasutra.com/blogs/SimonCarless/20140606/218988/Why_game_discovery_is_vital_introducing_Games_We_Care_About.php.
- [50] Laura Catalá, Vicente Julián, and José-Antonio Gil-Gómez. A cbr-based game recommender for rehabilitation videogames in social networks. In *International Conference on Intelligent Data Engineering and Automated Learning*, pages 370–377. Springer, 2014.
- [51] Chaomei Chen and Les Carr. Trailblazing the literature of hypertext: author co-citation analysis (1989–1998). In *Proceedings of the tenth ACM Conference on Hypertext and hypermedia: returning to our diverse roots: returning to our diverse roots*, pages 51–60. ACM, 1999.
- [52] Chaochang Chiu, Re-Jiau Sung, Yu-Ren Chen, and Chih-Hao Hsiao. App Review Analytics of Free Games Listed on Google Play.
- [53] Meri Coleman and Ta Lin Liau. A computer readability formula designed for machine scoring. *Journal of Applied Psychology*, 60(2):283, 1975.
- [54] Harry Collins. *Changing Order: Replication and Induction in Scientific Practice*. University Of Chicago Press, Chicago, reprint edition edition, June 1992. ISBN 978-0-226-11376-0.
- [55] Harry Collins. *Tacit and Explicit Knowledge*. University Of Chicago Press, Chicago; London, reprint edition edition, December 2012. ISBN 978-0-226-00421-1.
- [56] Nicole Convery. From reactive to proactive appraisal. *Archives and Manuscripts*, 42(2):158–160, May 2014. ISSN 0157-6895. doi: 10.1080/01576895.2014.911676.
- [57] Terry Cook. 'We Are What We Keep; We Keep What We Are': Archival Appraisal Past, Present and Future. *Journal of the Society of Archivists*, 32(2):173–189, October 2011. ISSN 0037-9816. doi: 10.1080/00379816.2011.619688.
- [58] Trevor F. Cox and Michael AA Cox. *Multidimensional scaling*. CRC press, 2000.
- [59] Kate Cumming and Anne Picot. Reinventing appraisal. *Archives and Manuscripts*, 42(2):133–145, May 2014. ISSN 0157-6895. doi: 10.1080/01576895.2014.926824.

- [60] Charis Cussins. Ontological choreography: Agency through objectification in infertility clinics. *Social studies of science*, 26(3):575–610, 1996.
- [61] David Datta. Doom upload site list (final version) - Google Groups. URL <https://groups.google.com/forum/#!search/david%20datta%20doom%7Csort:relevance/comp.sys.ibm.pc.games.action/tT3ptNF4qN8/nkrU0islGjEJ>.
- [62] Greta De groat. A History of Video Game Cataloging in U.S. Libraries. *Cataloging & Classification Quarterly*, 53(2):135–156, February 2015. ISSN 0163-9374. doi: 10.1080/01639374.2014.954297.
- [63] Frans de Vries. Gamers.Org - Frans P. de Vries, . URL <http://www.gamers.org/~fpv/>.
- [64] Frans P. de Vries. The idgames Archive hits 1 Gigabyte, . URL <http://www.gamers.org/pub/idgames/docs/misc/milestone.txt>.
- [65] Shane Denson. Digital Seriality. URL http://shanedenson.com/stuff/visualizing_digital_seriality/digital-seriality.html.
- [66] Luciana Duranti. The concept of appraisal and archival theory. *American Archivist*, 57(2):328–344, 1994.
- [67] Émile Durkheim, Marcel Mauss, and Rodney Needham. *Primitive classification*. Routledge paperbacks. Cohen & West, London, reprinted and first published as a routledge paperback edition, 1970. ISBN 978-0-7100-3362-8 978-0-7100-6891-0. OCLC: 248017790.
- [68] Jon-Paul C. Dyson. CHEGheads Blog » Preserving John Romero’s First Computer at ICHEG | International Center for the History of Electronic Games. URL <http://www.museumofplay.org/blog/chegheads/2014/08/preserving-john-romeros-first-computer-at-icheg/>.
- [69] Umberto Eco. *How to Write a Thesis*. MIT Press, Cambridge, MA; London, 2015.
- [70] Clark A. Elliott. *Understanding progress as process: documentation of the history of post-war science and technology in the United States*. Society of Amer Archivists, 1983.
- [71] Albert Endres. Commentary on James E. Tomayko. In *History of Computing: Software Issues*, pages 77–82. Springer-Verlag, Berlin; Heidelberg; New York, 2002.
- [72] Norman Fairclough. *Discourse and social change*. Polity Press, Cambridge, Mass, 1992. ISBN 978-0-7456-0674-3.

- [73] Norman Fairclough. *Critical discourse analysis: the critical study of language*. Language in social life series. Longman, London ; New York, 1995. ISBN 978-0-582-21980-9 978-0-582-21984-7.
- [74] Clara Fernández-Vara. *Introduction to Game Analysis*. Routledge, New York, 1 edition edition, July 2014. ISBN 978-0-415-70327-7.
- [75] Paul Feyerabend and Ian Hacking. *Against Method*. Verso, London ; New York, fourth edition edition edition, May 2010. ISBN 978-1-84467-442-8.
- [76] Dr Andrew Flinn. Community Histories, Community Archives: Some Opportunities and Challenges. *Journal of the Society of Archivists*, 28(2):151–176, October 2007. ISSN 0037-9816. doi: 10.1080/00379810701611936.
- [77] Robert Forsman and Bernd Kreimeier. A Brief Summary of DOOM-Style Rendering. July 1996.
- [78] Tracy Fullerton, Chris Swain, and Steven Hoffman. *Game design workshop: Designing, prototyping, & playtesting games*. CRC Press, 2004.
- [79] Chaim Ophir Gingold. Play Design. *eScholarship*, January 2016. URL <http://escholarship.org/uc/item/8qr533m2>.
- [80] Asunción Gómez-Pérez, Mariano Fernández-López, and Oscar Corcho. *Ontological engineering*, volume 139. Springer Heidelberg, 2004.
- [81] Dan Gordon and Shuhong Chen. Front-to-back display of BSP trees. *IEEE computer Graphics and Applications*, 11(5):79–85, 1991.
- [82] Lindsay D. Grace. A linguistic analysis of mobile games: Verbs and nouns for content estimation. *Proc. FDG*, 2014.
- [83] Anthony Grafton. *The footnote: a curious history*. Harvard University Press, Cambridge, Mass, revised edition edition, 1997. ISBN 978-0-674-90215-2.
- [84] Thomas R. Gruber. Toward principles for the design of ontologies used for knowledge sharing? *International Journal of Human-Computer Studies*, 43(5):907–928, November 1995. ISSN 1071-5819. doi: 10.1006/ijhc.1995.1081.
- [85] Raiford Guins. 8 Console. In *Debugging Game History: A Critical Lexicon*, pages 63–79. 2016.
- [86] Raiford Guins and Henry Lowood, editors. *Debugging game history: a critical lexicon*. Game histories. The MIT Press, Cambridge, Massachusetts, 2016. ISBN 978-0-262-03419-7.

- [87] Joan K. Haas, Helen Willa Samuels, and Barbara Trippel Simmons. *Appraising the records of modern science and technology: a guide*. Massachusetts Institute of Technology Cambridge, MA, 1985.
- [88] Carl Hamacher, Zvonko Vranesic, Safwat Zaky, and Naraig Manjikian. *Computer Organization and Embedded Systems*. McGraw-Hill, New York, NY, 2012.
- [89] Harry H. Harman. Modern factor analysis. 1960.
- [90] Vi Hart and Nicky Case. Parable of the Polygons. URL <http://ncase.me/polygons>.
- [91] John L. Hennessy and David A. Patterson. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann Publishers, San Francisco, CA, 4th edition edition, 2007.
- [92] John L. Hennessy and David A. Patterson. *Computer Organization and Design: The Hardware / Software Interface*. Morgan Kaufmann Publishers, San Francisco, CA, 4th edition edition, 2009.
- [93] Johan Huizinga. *Homo ludens ; a study of the play-element in culture*. Number 15 in Beacon paperbacks. Beacon Press, Boston, first beacon paperback edition edition, 1955. ISBN 978-0-8070-4681-4.
- [94] Edmund Husserl. *The crisis of European sciences and transcendental phenomenology ; an introduction to phenomenological philosophy*. Northwestern University studies in phenomenology & existential philosophy. Northwestern University Press, Evanston, 1970. ISBN 978-0-8101-0255-2.
- [95] Ken Hyland. Academic attribution: Citation and the construction of disciplinary knowledge. *Applied linguistics*, 20(3):341–367, 1999.
- [96] Ken Hyland. *Disciplinary discourses: social interactions in academic writing*. Applied linguistics and language study. Longman, Harlow ; New York, 2000. ISBN 978-0-582-41904-9.
- [97] International Conference on the History of Computing, Ulf Hashagen, Reinhard Keil-Slawik, Arthur L Norberg, and Heinz Nixdorf MuseumsForum, editors. *History of computing: software issues : International Conference on the History of Computing, ICHC 2000, April 5-7, 2000, Heinz Nixdorf MuseumsForum, Paderborn, Germany*. Springer, Berlin; New York, 2002. ISBN 978-3-540-42664-6. OCLC: 49649935.

- [98] Dietmar Jannach, Lukas Lerche, Fatih Gedikli, and Geoffray Bonnin. What recommenders recommend - an analysis of accuracy, popularity, and sales diversity effects. In *International Conference on User Modeling, Adaptation, and Personalization*, pages 25–37. Springer, 2013.
- [99] Dennis G. Jerz. Somewhere Nearby is Colossal Cave: Examining Will Crowther’s Original "Adventure" in Code and in Kentucky. 1 (2), 2007. URL <http://www.digitalhumanities.org/dhq/vol/001/2/000009/000009.html>.
- [100] Elizabeth Johnson. Our archives, our selves: documentation strategy and the re-appraisal of professional identity. *The American Archivist*, 71(1): 190–202, 2008.
- [101] Eric Kaltman. Current Game Preservation in Not Enough, June 2016. URL <https://web.stanford.edu/group/htgg/cgi-bin/drupal/?q=node/1211>.
- [102] Eric Kaltman, Noah Wardrip-Fruin, Henry Lowood, and Christy Caldwell. A Unified Approach to Preserving Cultural Software Objects and their Development Histories. 2014. URL <http://escholarship.org/uc/item/0wg4w6b9.pdf>.
- [103] Eric Kaltman, Noah Wardrip-Fruin, Henry Lowood, and Christy Caldwell. Methods and Recommendations for Archival Records of Game Development: The Case of Academic Games. *Proceedings of the 10th International Conference on the Foundations of Digital Games*, 2015.
- [104] Eric Kaltman, Noah Wardrip-fruin, Mitch Mastroni, Henry Lowood, Greta De groat, Glynn Edwards, Marcia Barrett, and Christy Caldwell. Implementing Controlled Vocabularies for Computer Game Platforms and Media Formats in SKOS. *Journal of Library Metadata*, 16(1):1–22, January 2016. ISSN 1938-6389, 1937-5034. doi: 10.1080/19386389.2016.1167494.
- [105] Matthew G. Kirschenbaum. *Mechanisms: New Media and the Forensic Imagination*. The MIT Press, Cambridge, Mass.; London, January 2012. ISBN 978-0-262-51740-9.
- [106] Steven A. Knowlton. Three decades since prejudices and antipathies: a study of changes in the library of congress subject headings. *Cataloging & Classification Quarterly*, 40(2):123–145, 2005.
- [107] Don Knuth. Adventure, 1998. URL <http://www.literateprogramming.com/adventure.pdf>.

- [108] Don Knuth and Doug McIlroy. A Literate Program. *Communications of the ACM*, 29(6), 1986.
- [109] Donald Ervin Knuth. Literate programming. *The Computer Journal*, 27(2): 97–111, 1984.
- [110] Lars Konzack. Computer Game Criticism: A Method for Computer Game Analysis. In *CGDC Conf.*, 2002.
- [111] Wessel Kraaij and Wilfried Post. Task based evaluation of exploratory search systems. In *Proc. of SIGIR 2006 Workshop, Evaluation Exploratory Search Systems, Seattle, USA*, pages 24–27, 2006.
- [112] Thomas S. Kuhn. *The Structure of Scientific Revolutions*. University of Chicago Press, Chicago, IL, 3rd edition edition, December 1996. ISBN 978-0-226-45808-3.
- [113] David Kushner. *Masters of doom: how two guys created an empire and transformed pop culture*. Random House, New York, 1st ed edition, 2003. ISBN 0-375-50524-5.
- [114] George Lakoff. *Women, fire, and dangerous things: what categories reveal about the mind*. University of Chicago Press, Chicago, 1987. ISBN 0-226-46803-8.
- [115] George Lakoff and Mark Johnson. *Metaphors we live by*. University of Chicago Press, Chicago, 1980. ISBN 978-0-226-46801-3 978-0-226-46800-6. OCLC: 6042798.
- [116] Shyong K. Lam and John Riedl. Shilling recommender systems for fun and profit. In *Proceedings of the 13th international conference on World Wide Web*, pages 393–402. ACM, 2004.
- [117] Ora Lassila and Deborah McGuinness. The role of frame-based representation on the semantic web. *Linköping Electronic Articles in Computer and Information Science*, 6(5):2001, 2001.
- [118] Bruno Latour. *Science in action: how to follow scientists and engineers through society*. Harvard University Press, Cambridge, Mass., 1987. ISBN 0-674-79290-4 978-0-674-79290-6 0-674-79291-2 978-0-674-79291-3.
- [119] Bruno Latour and Steve Woolgar. *Laboratory Life: The Construction of Scientific Facts*. Princeton University Press, April 2013. ISBN 1-4008-2041-3.
- [120] John Law. *Aircraft stories: Decentering the object in technoscience*. Duke University Press, 2002.

- [121] Dylan Lederle-Ensign and Noah Wardrip-Fruin. What is strafe jumping? idtech3 and the game engine as software platform. *Transactions of the Digital Games Research Association*, 2(2), 2016. URL <http://todigra.org/index.php/todigra/article/view/35>.
- [122] Jin Ha Lee, Joseph T. Tennis, Rachel Ivy Clarke, and Michael Carpenter. Developing a video game metadata schema for the Seattle Interactive Media Museum. *International Journal on Digital Libraries*, 13(2):105–117, March 2013. ISSN 1432-5012, 1432-1300. doi: 10.1007/s00799-013-0103-x.
- [123] Jin Ha Lee, Sungsoo Ray Hong, Hyerim Cho, and Yea-Seul Kim. Vizmo game browser: Accessing video games by visual style and mood. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 149–152. ACM, 2015.
- [124] Chong-U. Lim and D. Fox Harrell. Revealing social identity phenomena in videogames with archetypal analysis. In *Proceedings of the 6th International AISB Symposium on AI and Games*, 2015.
- [125] David Link. *Archaeology of Algorithmic Artefacts*. Univocal Publishing, Minneapolis, first edition edition, 2016. ISBN 978-1-937561-04-8.
- [126] Chris Lombardi. They’re Going to Hell for This One! *Computer Gaming World*, (108):104–105, July 1993. ISSN 0744-6667.
- [127] Henry Lowood. The Hard Work of Software History. *RBM: A Journal of Rare Books, Manuscripts and Cultural Heritage*, 2(2):141–160, 2001.
- [128] Henry Lowood. Perfect Capture: Three Takes on Replay, Machinima and the History of Virtual Worlds. *Journal of Visual Culture*, 10(1):113–124, April 2011. ISSN 1470-4129, 1741-2994. doi: 10.1177/1470412910391578.
- [129] Michael Lynch. The externalized retina: Selection and mathematization in the visual documentation of objects in the life sciences. *Human studies*, 11(2):201–234, 1988.
- [130] James MacQueen and others. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA., 1967.
- [131] Jimmy Maher. *The future was here: the Commodore Amiga*. Platform studies. MIT Press, Cambridge, Mass, 2012. ISBN 978-0-262-01720-6.
- [132] Michael S. Mahoney. The history of computing in the history of technology. *Annals of the History of Computing*, 10(2):113–125, 1988.

- [133] Michael S. Mahoney. Issues in the History of Computing. In *History of programming languages-II*, pages 772–781. ACM, 1996.
- [134] Frank Manchel. *Film study: an analytical bibliography*. Fairleigh Dickinson University Press ; Associated University Presses, Rutherford : London, 1990. ISBN 978-0-8386-3186-7.
- [135] Gary Marchionini. Exploratory search: from finding to understanding. *Communications of the ACM*, 49(4):41–46, 2006.
- [136] Michael Mateas. AI and Interactive Storytelling: How We Can Help Each Other, March 2010.
- [137] Michael Mateas and Andrew Stern. A behavior language for story-based believable agents. *Intelligent Systems, IEEE*, 17(4):39–47, 2002.
- [138] Michael Mateas and Andrew Stern. Façade: An experiment in building a fully-realized interactive drama. In *Game Developers Conference, Game Design track*, volume 2, page 82, 2003.
- [139] Michael Mateas and Noah Wardrip-Fruin. Defining operational logics. *Digital Games Research Association (DiGRA)*, 2009.
- [140] Scott A. May. The Best in Arcade Game Software. *COMPUTE! Magazine*, 16(160):S–1, January 1994. ISSN 0194-357X.
- [141] G. Harry Mc Laughlin. SMOG grading-a new readability formula. *Journal of reading*, 12(8):639–646, 1969.
- [142] Josh McCoy, Mike Treanor, Ben Samuel, Michael Mateas, and Noah Wardrip-Fruin. Prom Week: social physics as gameplay. In *Proceedings of the 6th International Conference on Foundations of Digital Games*, pages 319–321, 2011.
- [143] Josh McCoy, Mike Treanor, Ben Samuel, Aaron A. Reed, Michael Mateas, and Noah Wardrip-Fruin. Prom Week: Designing past the game/story dilemma. *Proceedings of the 8th International Conference on Foundations of Digital Games*, 2013.
- [144] Joshua McCoy. *All the World’s a Stage: A Playable Model of Social Interaction Inspired by Dramaturgical Analysis*. PhD thesis, University of California, Santa Cruz, June 2012.
- [145] Joshua McCoy, Mike Treanor, Ben Samuel, Brandon Robert Tearse, Michael Mateas, and Noah Wardrip-Fruin. The Prom: An Example of Socially-Oriented Gameplay. In *AIIDE*, 2010.

- [146] Joshua McCoy, Mike Treanor, Ben Samuel, Noah Wardrip-Fruin, and Michael Mateas. Comme il Faut: A System for Authoring Playable Social Models. In *AIIDE*, 2011.
- [147] Robert R. McCrae and Paul T. Costa. Validation of the five-factor model of personality across instruments and observers. *Journal of personality and social psychology*, 52(1):81, 1987.
- [148] Jerome McDonough, Matthew Kirschenbaum, Doug Reside, Neil Fraistat, and Dennis Jerz. Twisty Little Passages Almost All Alike: Applying the FRBR Model to a Classic Computer Game. 4(2), 2010. URL <http://www.digitalhumanities.org/dhq/vol/4/2/000089/000089.html>.
- [149] Jerome P. McDonough. *Preserving virtual worlds: Final Report*. Graduate School of Library and Information Science, University of Illinois at Urbana-Champaign, 2010.
- [150] mediaXstanford. A Tale of Two Jousts: Multimedia, Game Feel, and Imagination. URL <https://www.youtube.com/watch?v=JkbCNMAS0qI&feature=youtu.be>.
- [151] Angelika Menne-Haritz. Appraisal or documentation: can we appraise archives by selecting content? *The American Archivist*, 57(3):528–542, 1994.
- [152] George A. Miller. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41, 1995.
- [153] Modern Language Association of America, editor. *MLA handbook*. The Modern Language Association of America, New York, eighth edition edition, 2016. ISBN 978-1-60329-262-7.
- [154] Annemarie Mol. Ontological politics. A word and some questions. *The Sociological Review*, 46(S):74–89, 1998.
- [155] Annemarie Mol. *The body multiple: Ontology in medical practice*. Duke University Press, 2002.
- [156] Nick Montfort. Cybertext Killed the Hypertext Star | Electronic Book Review, December 2000. URL <http://www.electronicbookreview.com/thread/electropoetics/cyberdebates>.
- [157] Nick Montfort. Combat in Context. *Game Studies*, 6(1), December 2006. ISSN 1604-7982. URL <http://gamestudies.org/0601/articles/montfort>.

- [158] Nick Montfort and Ian Bogost. *Racing the beam the Atari video computer system*. MIT Press, Cambridge, Mass., 2009. ISBN 978-0-262-25493-9 0-262-25493-X.
- [159] Theodor H. Nelson. *Computer lib: Dream machines*. Tempus Books of Microsoft Press Redmond, 1987.
- [160] James Newman. *Best before: Videogames, supersession and obsolescence*. Routledge, 2012.
- [161] James Newman. Ports and patches: Digital games as unstable objects. *Convergence: The International Journal of Research into New Media Technologies*, 18(2):135–142, 2012.
- [162] Noam Nisan and Shimon Schocken. *The Elements of Computing Systems: Building a Modern Computer from First Principles*. Mit Press Cambridge, MA, 2005.
- [163] Bethany Paige Nowviskie. speculative collections, October 2016. URL <http://nowviskie.org/2016/speculative-collections/>.
- [164] Hope A. Olson, John J. Boll, and Rao Aluri. *Subject analysis in online catalogs*. Libraries Unlimited, Englewood, Colo, 2nd ed edition, 2001. ISBN 978-1-56308-800-1.
- [165] Seymour Papert. *Mindstorms: Children, computers, and powerful ideas*. Basic Books, Inc., 1980.
- [166] Deuk Hee Park, Hyea Kyeong Kim, Il Young Choi, and Jae Kyeong Kim. A literature review and classification of recommender systems research. *Expert Systems with Applications*, 39(11):10059–10072, 2012.
- [167] Tim Parks. References, Please. URL <http://www.nybooks.com/daily/2014/09/13/references-please/>.
- [168] Daniel Pinchbeck. *Doom: scarydarkfast*. Landmark video games. University of Michigan Press, Ann Arbor, 2013. ISBN 978-0-472-07191-3.
- [169] Michael Polanyi. *The tacit dimension*. Number 1962 in Terry lectures. Anchor Books, Garden City, N.Y, 1967.
- [170] Tammera M. Race, M. P. Popp, and D. Dallis. Resource discovery tools: Supporting serendipity. *Planning and implementing resource discovery tools in academic libraries*, pages 139–152, 2012.

- [171] Kevin Raison, Noriko Tomuro, Steve Lytinen, and Jose P. Zagal. Extraction of user opinions by adjective-context co-clustering for game review texts. In *Advances in Natural Language Processing*, pages 289–299. Springer, 2012.
- [172] Stephen Ramsay. *Reading machines: toward an algorithmic criticism*. Topics in the digital humanities. University of Illinois Press, Urbana, 2011. ISBN 978-0-252-03641-5.
- [173] Rebecca Heineman. Burgertime 7/12/2015: DOOM 3do. URL <https://www.youtube.com/watch?v=rBbIi12HPSU>.
- [174] Mitch Resnick and Brian Silverman. Active Essays. URL <http://www.rupert.id.au/microworlds/circles/active-essay.html>.
- [175] Howard Rheingold. *Virtual reality*. Simon & Schuster, New York, N.Y., 1992. ISBN 0-671-69363-8 978-0-671-69363-3 0-671-77897-8 978-0-671-77897-2.
- [176] Francesco Ricci, Lior Rokach, and Bracha Shapira. *Introduction to recommender systems handbook*. Springer, 2011.
- [177] Richard Rinehart and Jon Ippolito, editors. *Re-collection: art, new media, and social memory*. Leonardo. The MIT Press, Cambridge, Massachusetts, 2014. ISBN 978-0-262-02700-7. OCLC: 858975361.
- [178] Alex Roland. What Hath Kranzberg Wrought? Or, Does the History of Technology Matter? *Technology and Culture*, 38(3):697, July 1997. ISSN 0040165X. doi: 10.2307/3106860.
- [179] John Romero. Happy 23rd Birthday, DOOM! URL <http://rome.ro/news/2016/12/10/happy-23rd-birthday-doom>.
- [180] John Romero. Doom History 1994. planet romero, January 2015. URL <https://web.archive.org/web/20150108101506/http://planetromero.com/2009/01/doom-history-1994>.
- [181] Eleanor Rosch, Carolyn B Mervis, Wayne D Gray, David M Johnson, and Penny Boyes-Braem. Basic objects in natural categories. *Cognitive Psychology*, 8(3):382–439, July 1976. ISSN 0010-0285. doi: 10.1016/0010-0285(76)90013-X.
- [182] Nathan Rosenberg. *Exploring the black box: technology, economics, and history*. Cambridge University Press, Cambridge [England] ; New York, 1994. ISBN 978-0-521-45270-0 978-0-521-45955-6.
- [183] David SH Rosenthal. Emulation & Virtualization as Preservation Strategies. 2015.

- [184] Jeff Rothenberg. Ensuring the longevity of digital documents. *Scientific American*, 272(1):42–47, 1995.
- [185] Jeff Rothenberg. Ensuring the longevity of digital information. *Int’l. J. Legal Info.*, 26:1, 1998.
- [186] James Owen Ryan, Eric Kaltman, Andrew Max Fisher, Timothy Hong, Taylor Owen-Milner, Michael Mateas, and Noah Wardrip-Fruin. Large-Scale Interactive Visualizations of Nearly 12,000 Digital Games. *Proceedings of the 10th International Conference on the Foundations of Digital Games*, 2015.
- [187] James Owen Ryan, Eric Kaltman, Timothy Hong, Michael Mateas, and Noah Wardrip-Fruin. People Tend to Like Related Games. *Proceedings of the 10th International Conference on the Foundations of Digital Games*, 2015.
- [188] James Owen Ryan, Eric Kaltman, Michael Mateas, and Noah Wardrip-Fruin. Tools for Videogame Discovery Built Using Latent Semantic Analysis. *Proceedings of the 10th International Conference on the Foundations of Digital Games*, 2015.
- [189] James Owen Ryan, Eric Kaltman, Michael Mateas, and Noah Wardrip-Fruin. What We Talk About When We Talk About Games: Bottom-Up Game Studies Using Natural Language Processing. *Proceedings of the 10th International Conference on the Foundations of Digital Games*, 2015.
- [190] James Owen Ryan, Eric Kaltman, Timothy Hong, Katherine Isbister, Michael Mateas, and Noah Wardrip-Fruin. GameNet and GameSage: Videogame Discovery as Design Insight. San Jose, CA, May 2016.
- [191] Fabien Sanglard. Jurassic Park: Trespasser CG Source Code Review. URL <http://fabiensanglard.net/trespasser/>.
- [192] Peter Seibel. Code is not literature, 2014. URL <http://www.gigamonkeys.com/code-reading/>.
- [193] Raymond George Siemens and David Moorman, editors. *Mind technologies: humanities computing and the Canadian academic community*. University of Calgary Press, Calgary, 2006. ISBN 1-55238-172-2.
- [194] Brian Cantwell Smith. *On the origin of objects*. MIT Press, Cambridge, Mass, 1996. ISBN 0-262-19363-9.
- [195] Silvia B. Southwick. A Guide for Transforming Digital Collections Metadata into Linked Data Using Open Source Technologies. *Journal of Library Metadata*, 15(1):1–35, January 2015. ISSN 1938-6389, 1937-5034. doi: 10.1080/19386389.2015.1007009.

- [196] David J. Staley. *Computers, visualization, and history: how new technology will transform our understanding of the past*. History, the humanities, and the new technology. Routledge, Abingdon, second edition edition, 2015. ISBN 978-1-317-50740-6.
- [197] Philip J. Stephens. Writing a fast 3d graphics engine. September 1995. URL <http://www.gamers.org/dEngine/doom/papers/webview.ps.gz>.
- [198] Mary Stevens, Andrew Flinn, and Elizabeth Shepherd. New frameworks for community engagement in the archive sector: from handing over to handing on. *International Journal of Heritage Studies*, 16(1-2):59–76, January 2010. ISSN 1352-7258. doi: 10.1080/13527250903441770.
- [199] Alva T. Stone. The LCSH century: a brief history of the Library of Congress subject headings, and introduction to the centennial essays. *Cataloging & Classification Quarterly*, 29(1-2):1–15, 2000.
- [200] C. A. Sula and M. Miller. Citations, contexts, and humanistic discourse: Toward automatic extraction and classification. *Literary and Linguistic Computing*, 29(3):452–464, September 2014. ISSN 0268-1145, 1477-4615. doi: 10.1093/lc/fqu019.
- [201] Brian Sutton-Smith. *The ambiguity of play*. Harvard University Press, Cambridge, Mass, 1997. ISBN 978-0-674-01733-7.
- [202] Patrik Svensson and David Theo Goldberg, editors. *Between humanities and the digital*. The MIT Press, Cambridge, Massachusetts, 2015. ISBN 978-0-262-02868-4.
- [203] Steve Swink. *Game Feel: A Game Designer’s Guide to Virtual Sensation*. Morgan Kaufmann, Burlington, MA, 2009.
- [204] Arlene G Taylor. *The organization of information*. Libraries Unlimited, Westport, Conn., 2004. ISBN 1-56308-976-9 978-1-56308-976-3 1-56308-969-6 978-1-56308-969-5.
- [205] Katie Salen Tekinbaş and Eric Zimmerman. *Rules of play: game design fundamentals*. MIT Press, Cambridge, Mass, 2004. ISBN 978-0-262-24045-1.
- [206] Katie Salen Tekinbaş and Eric Zimmerman, editors. *The Game design reader: a rules of play anthology*. MIT Press, Cambridge, Mass, 2005. ISBN 978-0-262-19536-2.

- [207] Carl Therrien. Inspecting Video Game Historiography Through Critical Lens: Etymology of the First-Person Shooter Genre. *Game Studies*, 15(2), December 2015. ISSN 1604-7982. URL <http://gamestudies.org/1502/articles/therrien>.
- [208] J. E. Tomayko. *Computers take flight: a history of NASA's pioneering digital fly-by-wire project*. Number 2000-4224 in NASA history series. National Aeronautics and Space Administration, NASA Office of Policy and Plans, NASA History Office, Washington, D.C, 2000. ISBN 978-0-16-059053-5.
- [209] J. E Tomayko, United States, National Aeronautics and Space Administration, and Scientific and Technical Information Division. *Computers in spaceflight: the NASA experience*. National Aeronautics and Space Administration, Scientific and Technical Information Division, Washington, D.C., 1988. OCLC: 68711652.
- [210] James E. Tomayko. Software as Engineering. In *History of Computing: Software Issues*, pages 65–76. Springer-Verlag, Berlin; Heidelberg; New York, 2002.
- [211] Mike Treanor. *Investigating Procedural Expression and Interpretation in Videogames*. PhD thesis, University of California, Santa Cruz, June 2013.
- [212] Bret Victor. The Ladder of Abstraction. URL <http://worrydream.com/#!2/LadderOfAbstraction>.
- [213] Bret Victor. Ten Brighter Ideas? An Explorable Explanation, 2010. URL <http://worrydream.com/TenBrighterIdeas/>.
- [214] Bret Victor. Explorable Explanations, 2011. URL <http://worrydream.com/#!/ExplorableExplanations>.
- [215] Vernor Vinge. *A Deepness in the Sky*. Tor Books, New York, reprint edition edition, January 2000. ISBN 978-0-8125-3635-5.
- [216] Noah Wardrip-Fruin. *Expressive processing*. MIT Press, Cambridge (Mass.), 2009. ISBN 978-0-262-01343-7 0-262-01343-6.
- [217] Claire Warwick, Melissa M. Terras, and Julianne Nyhan, editors. *Digital humanities in practice*. Facet Publishing in association with UCL Centre for Digital Humanities, London, 2012. ISBN 978-1-85604-766-1.
- [218] David Weinberger. *Too big to know: rethinking knowledge now that the facts aren't the facts, experts are everywhere, and the smartest person in the room is the room*. Basic Books, New York, 2011. ISBN 978-0-465-02142-0 978-0-465-02813-9. OCLC: 701015486.

- [219] Hayden V White. *Metahistory: the historical imagination in nineteenth-century Europe*. Johns Hopkins University Press, Baltimore, 1973. ISBN 0-8018-1469-3 978-0-8018-1469-3 0-8018-1761-7 978-0-8018-1761-8.
- [220] Hayden V. White. *Tropics of discourse: essays in cultural criticism*. Johns Hopkins University Press, Baltimore, 1978. ISBN 978-0-8018-2127-1.
- [221] Hayden V. White. *The practical past*. FlashPoints. Northwestern University Press, Evanston, Illinois, 2014. ISBN 978-0-8101-3006-7. OCLC: 879583905.
- [222] Ryen W. White and Resa A. Roth. Exploratory search: Beyond the query-response paradigm. *Synthesis Lectures on Information Concepts, Retrieval, and Services*, 1(1):1–98, 2009.
- [223] Ryen W. White, Gary Marchionini, and Gheorghe Muresan. *Evaluating exploratory search systems: Introduction to special topic issue of information processing and management*. Pergamon, 2008.
- [224] Jay Wilbur. Official DOOM information - Google Groups, . URL [https://groups.google.com/forum/#!topicsearchin/comp.sys.ibm.pc.games/doom\\$20AND\\$20authorname\\$3A%22Jay\\$20Wilbur%22\\$20AND\\$20before\\$3A1992\\$2F12\\$2F01/comp.sys.ibm.pc.games/aKeDging-IA](https://groups.google.com/forum/#!topicsearchin/comp.sys.ibm.pc.games/doom$20AND$20authorname$3A%22Jay$20Wilbur%22$20AND$20before$3A1992$2F12$2F01/comp.sys.ibm.pc.games/aKeDging-IA).
- [225] Jay Wilbur. Wolf3d and a Serial Link... - Google Groups, . URL [https://groups.google.com/forum/#!searchin/comp.sys.ibm.pc.games/doom\\$20before\\$3A1992\\$2F12\\$2F01%7Csort:date/comp.sys.ibm.pc.games/f9SqYz0bxtU/ID_wR38QbKAJ](https://groups.google.com/forum/#!searchin/comp.sys.ibm.pc.games/doom$20before$3A1992$2F12$2F01%7Csort:date/comp.sys.ibm.pc.games/f9SqYz0bxtU/ID_wR38QbKAJ).
- [226] Douglas Wilson. A breakdown of 2013's most fascinating video game moment, December 2013. URL <http://www.polygon.com/2013/12/23/5227726/anatomy-of-a-spelunky-miracle-or-how-the-internet-finally-beat>.
- [227] Megan A. Winget and Caitlin Murray. Collecting and preserving videogames and their related materials: A review of current practice, game-related archives and research projects. *Proceedings of the American Society for Information Science and Technology*, 45(1):1–9, January 2008. ISSN 1550-8390. doi: 10.1002/meet.2008.1450450250.
- [228] Megan A. Winget and Wiliam Walker Sampson. Game Development Documentation and Institutional Collection Development Policy. In *Proceedings of the 11th Annual International ACM/IEEE Joint Conference on Digital Libraries*, JCDL '11, pages 29–38, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0744-4. doi: 10.1145/1998076.1998083.

- [229] Steve Woolgar. On the alleged distinction between discourse and praxis. *Social Studies of Science*, pages 309–317, 1986.
- [230] Steve Woolgar and Javier Lezaun. The wrong bin bag: A turn to ontology in science and technology studies? *Social Studies of Science*, 43(3):321–340, June 2013. ISSN 0306-3127, 1460-3659. doi: 10.1177/0306312713488820.
- [231] Takashi Yamamiya, Alessandro Warth, and Ted Kaehler. Active Essays on the Web. In *Creating, Connecting and Collaborating through Computing, 2009. C5'09. Seventh International Conference on*, pages 3–10. IEEE, 2009.
- [232] Chuttur M. Yasser. An Analysis of Problems in Metadata Records. *Journal of Library Metadata*, 11(2):51–62, April 2011. ISSN 1938-6389, 1937-5034. doi: 10.1080/19386389.2011.570654.
- [233] José P. Zagal and Noriko Tomuro. The aesthetics of gameplay: a lexical approach. In *Proceedings of the 14th International Academic MindTrek Conference: Envisioning Future Media Environments*, pages 9–16. ACM, 2010.
- [234] José P. Zagal, Michael Mateas, Clara Fernández-Vara, Brian Hochhalter, and Nolan Lichti. 2. Towards an Ontological Language for Game Analysis. *Worlds in Play: International Perspectives on Digital Games Research*, 21: 21, 2007.
- [235] José P. Zagal, Noriko Tomuro, and Andriy Shepitsen. Natural language processing in game studies research: An overview. *Simulation & Gaming*, 43(3):356–373, 2012.
- [236] José Pablo Zagal and Noriko Tomuro. Cultural differences in game appreciation: A study of player game reviews. In *FDG*, pages 86–93, 2013.
- [237] Miaoqi Zhu and Xiaowen Fang. Developing playability heuristics for computer games from online reviews. In *International Conference of Design, User Experience, and Usability*, pages 496–505. Springer, 2014.
- [238] Miaoqi Zhu and Xiaowen Fang. Introducing a revised lexical approach to study user experience in game play by analyzing online reviews. In *Proceedings of the 2014 Conference on Interactive Entertainment*, pages 1–8. ACM, 2014.
- [239] Miaoqi Zhu and Xiaowen Fang. What nouns and adjectives in online game reviews can tell us about player experience? In *Proceedings of the extended abstracts of the 32nd annual ACM conference on Human factors in computing systems*, pages 1471–1476. ACM, 2014.

- [240] Miaoqi Zhu and Xiaowen Feng. Using lexicons obtained from online reviews to classify computer games. 2013.
- [241] Miaoqi Zhu, Xiaowen Fang, Susy S. Chan, and Jacek Brzezinski. Building a dictionary of game-descriptive words to study playability. In *CHI'13 Extended Abstracts on Human Factors in Computing Systems*, pages 1077–1082. ACM, 2013.