Aftermarket, a Game Design Philosophy

by

Patrick LeMieux

Department of Art, Art History, and Visual Studies
Duke University

Date:_______________________
Approved:

___________________________
William Seaman, Supervisor

___________________________
Mark B. N. Hansen

___________________________
N. Katherine Hayles

___________________________
Timothy Lenoir

Dissertation submitted in partial fulfillment of
the requirements for the degree of Doctor of Philosophy
in the Department of Art, Art History, and Visual Studies
in the Graduate School of Duke University

2015
ABSTRACT

Aftermarket, a Game Design Philosophy

by

Patrick LeMieux

Department of Art, Art History, and Visual Studies
Duke University

Date:_______________________
Approved:

___________________________
William Seaman, Supervisor

___________________________
Mark B. N. Hansen

___________________________
N. Katherine Hayles

___________________________
Timothy Lenoir

An abstract of a dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Art, Art History, and Visual Studies in the Graduate School of Duke University

2015
Abstract

Aftermarket, a Game Design Philosophy documents the community histories and material practices of players who, over the last decade, have transformed videogames from “entertainment systems” into instruments, equipment, tools, and toys for playing, thinking, and making in the aftermarket of the videogame industry. Through a close investigation of the hardware, software, and code enabling tool-assisted speedruns, real time attacks, and ROM hacks, Aftermarket explores how play can become a form of game design located between human experience and the speeds and scales of digital media. Beyond documenting how these different groups convert packaged products into open platforms for critical making, Aftermarket both argues for and enacts a model of game design as a critical practice in which playing, making, and thinking about videogames occurs within the same act—a true game design philosophy. Focusing on the material properties, technical capacities, and social play around a single game, Nintendo’s Super Mario Bros., this “close playing” and “platformer study” does not seek to reify Miyamoto, Tezuka, Kondo, and Nagako’s game, but appropriates, manipulates, duplicates, perforates, aggregates, and dissipates Super Mario Bros. into a different kind of “Mario Paint”—a medium for making art.
Contents

Abstract........................................................................................................................................... iv

Contents................................................................................................................................................ v

List of Figures ......................................................................................................................................... vii

1. Introduction ..................................................................................................................................... 1

   1.1. Aftermarket ............................................................................................................................... 4

   1.2. Game Design Philosophy ......................................................................................................... 10

   1.3. Chapter Descriptions .............................................................................................................. 16

2. NES-4021 to moSMB3.wmv: Speedrunning the Serial Interface ................................................. 24

   2.1. NES-4021 ................................................................................................................................. 27

   2.2. moSMB3.wmv ........................................................................................................................... 34

   2.3. Tool-Assisted Speedrunning ................................................................................................. 42

   2.4. Speedrunning the Serial Interface ......................................................................................... 51

3. 4:57: The Phenomenology of a Frame ......................................................................................... 56

   3.1. TASing Super Mario Bros. ...................................................................................................... 59

   3.2. Lip Jumps, Fireworks Skips, and the 21-Frame Rule .............................................................. 64

   3.3. Wall Jumps and Wrong Warps .............................................................................................. 67

   3.4. Vine Wraps, Wall Clips, and Left + Right ............................................................................ 74

   3.5. Flag Skips and Optimization ................................................................................................. 78

   3.6. Frame 16539 ............................................................................................................................ 80

   3.7. Real Time Attacks .................................................................................................................. 85

4. This is Not Mario: ROM Hacking Super Mario Clouds ............................................................... 92
List of Figures

Figure 1. R.O.B., the Robotic Operating Buddy, was packaged with the Nintendo Entertainment System in order to market the videogame console as a children’s toy. ................................................. 3

Figure 2. The 4021 parallel-to-serial shift register in a Nintendo Entertainment System controller (left) converts the collective state of eight buttons to a serial stream of electrical pulses transmitted over a single yellow wire (right). ................................................................. 29

Figure 3. On eBaum’s World, Morimoto’s “Hilarious Video” was rehosted without attribution and with a broadband warning. .......................................................................................... 36

Figure 4. Screenshots from moSMB3.wmv reveal the low resolution and high compression of Morimoto’s original movie file, adapted for the speed and scale of file sharing in the early 2000s. ................................................................. 39

Figure 5. This sixteen-digit binary to decimal to hexadecimal conversion chart details half a byte of data. .................................................................................................................... 48

Figure 6. This conversion chart illustrates how the eight buttons of an NES controller are represented as two-digit hexadecimal numbers in Famtasia’s .FMV file. ........................................... 48

Figure 7. A brief comparison of the hexadecimal representation of Morimoto’s .FMV to the visualization of his .WMV reveals a synchronic serial interface—a serial database detailing every button press. ................................................................. 49

Figure 8. In a little over a decade of speedrunning (red) and tool-assisted speedrunning (blue), players in both communities are approaching the end of games such as Super Mario Bros. In this case, optimized routes and perfectly performed exploits converge on a common limit: 4 minutes and 57 seconds. ................................................................. 55

Figure 9. For the past ten years, two communities have continued to play Super Mario Bros., discovering new game mechanics and extending aftermarket play through the application of a voluntary speed constraint. ................................................................................. 59

Figure 10. Due to two “secret” warp zones, speedrunning Super Mario Bros. only requires the player navigate eight of thirty-two levels. ................................................................. 63

Figure 11. Created by Blubbler to quell the flood of questions and accusations of cheating, this visualization of lip jumps from Level 1-2, 4-1, 4-2, and 8-1 demonstrates the disparity between graphic display and the collision detection algorithms of Super Mario Bros. ......................................................................................................................... 65
Figure 12. Both the Japanese and North American release of Super Mario Bros. carefully outline the intended mechanics of the game but, as players soon discovered, not all dynamics can be imagined or completely accounted for in advance......................... 68

Figure 13. Through the confluence of Super Mario Bros.’ collision detection and the wall’s ejection algorithms, a pixel- and frame-perfect window opens in which Mario may wall jump. Although the wall jump is only a latent mechanic in Super Mario Bros., it became a paradigmatic component of 3D platform games thanks to Super Mario 64. 71

Figure 14. By recalibrating or misaligning Mario’s relationship to the camera, the alternate pipe glitch or wrong warp manipulates the pointer value specifying which area loads next. ............................................................... 73

Figure 15. Because of a process by which Mario’s body snaps to the vine, placing it slightly offscreen makes Mario’s horizontal position roll over. There are no negative coordinates so anything below 0x00 loops back to 0xFF on the other side of the screen. 75

Figure 16. A comparison between frames 16532 to 16549 in klmz and HappyLee’s tool-assisted speedruns reveals a temporal discrepancy starting at frame 16539. This single frame of input was all it took for HappyLee to take the record on January 1, 2011. ...... 82

Figure 17. The 76 minute, silent video The Making of Super Mario Clouds operates as both a DIY, instructional video for would-be ROM hackers and documentation for the 2004 Whitney Biennial in New York................................................................. 93

Figure 18. Electronic Arts Intermix sells copies of The Making of Super Mario Clouds for thousands of dollars. ................................................................. 95

Figure 19. Super Mario Clouds by Cory Arcangel has been exhibited at the Whitney Museum of American Art since 2004. ........................................................................................................ 97

Figure 20. The Critical Code Studies Working Group is a biannual online workshop dedicated to the close reading of code................................................................. 99

Figure 21. Both the color and composition of Cory Arcangel’s clouds (right) do not match Nintendo’s clouds (left)................................................................................. 101

Figure 22. Super Mario Clouds continuously scrolls two screen widths of clouds. ...... 109

Figure 23. As detailed by Brian Parker (2008b) in his tutorial on backgrounds, the PPU can only assign a total of four colors to each 2 x 2 tile array in the background.......... 110
Figure 24. Looking at clouds.hex reveals not only the position of the clouds, but the specific error in Arcangel’s composition: a repeating “37” instead of a “38” on the red squares. ............................................................ 112

Figure 25. The FCEUX emulator is not only good for creating tool-assisted speedruns, but also has tools for visualizing operations of the Picture Processing Unit. Here Super Mario Bros. palette (top) is compared with Super Mario Clouds (bottom). ............... 114

Figure 26. In “The Making of Super Mario Clouds” and on his personal website Arcangel details the physical process of ROM hacking a NES Cartridge. .................. 116

Figure 27. Attempting to “erase everything but the clouds” and remake Super Mario Clouds (left) results in erasing everything but the coin. In Coin Heaven (right), an original ROM hack made for Aftermarket, a lonely coin remains blinking in the blue sky. ........................................................................... 118

Figure 28. Three different “coin heavens” appear at the top of vines in World 3-1, 4-1, 4-2, 5-1, 5-2, 6-2, 7-1, 8-1, and 8-2 of Super Mario Bros. ................................................................. 119

Figure 29. Platform Games advertising (2015) by Patrick LeMieux. Solo exhibition at Babycastles at 137 W. 14th St., New York, NY from May 7 to May 17, 2015. ............... 126

Figure 30. Platform Games (2015) by Patrick LeMieux. Solo exhibition at Babycastles at 137 W. 14th St., New York, NY from May 7–17, 2015. ......................................................... 127

Figure 31. Fucked Marios (2015) by Patrick LeMieux. Twelve laminated LaserJet prints on foamcore; 12 x 12 inches each. ................................................................. 128

Figure 32. Fucked Marios detail (2015) by Patrick LeMieux. Twelve laminated LaserJet prints on foamcore; 12 x 12 inches each. ......................................................... 129

Figure 33. Brothers (2013) by Patrick LeMieux. 11 CRT televisions, 2 Nintendo Entertainment Systems, 2 ROM hacked cartridges, Arduino; size variable. ............. 131

Figure 34. Brothers detail (2013) by Patrick LeMieux. 11 CRT televisions, 2 Nintendo Entertainment Systems, 2 ROM hacked cartridges, Arduino; size variable. .......... 132

Figure 35. Conflict Free (2015) by Patrick LeMieux. EPROM memory; 1 3/8 x 1/2 inches. ...................................................................................................................... 134

Figure 36. Conflict Free detail (2015) by Patrick LeMieux. EPROM memory; 1 3/8 x 1/2 inches. ...................................................................................................................... 135
Figure 37. Coin Heaven (2013) by Patrick LeMieux. 2 projectors, 2 Nintendo Entertainment System, 2 ROM hacked cartridges; size variable. ........................................ 137

Figure 38. Coin Heaven detail (2013) by Patrick LeMieux. 2 projectors, 2 Nintendo Entertainment System, 2 ROM hacked cartridges; size variable. ........................................ 138

Figure 39. RGB-SMB-TAB-SSS detail (2015) by Patrick LeMieux Fender Stratocaster, Arduino, 6 Servos, 800 LasterJet prints; 43 x 6 x 17 1/2 inches, 8 1/2 x 11 x 6 inches. ...... 140

Figure 40. RGB-SMB-TAB-SSS (2015) by Patrick LeMieux Fender Stratocaster, Arduino, 6 Servos, 800 LasterJet prints; 43 x 6 x 17 1/2 inches, 8 1/2 x 11 x 6 inches. ...................... 141

Figure 41. RGB-SMB-TAB-SSS detail (2015) by Patrick LeMieux Fender Stratocaster, Arduino, 6 Servos, 800 LasterJet prints; 43 x 6 x 17 1/2 inches, 8 1/2 x 11 x 6 inches. ...... 142

Figure 42. One Switch (2015) by Patrick LeMieux. Custom controller, Nintendo Entertainment System, CRT screen; 5 x 2 x 2/3 inches. .................................................... 143

Figure 43. One Switch (2015) detail by Patrick LeMieux. Custom controller, Nintendo Entertainment System, CRT screen; 5 x 2 x 2/3 inches. .................................................... 144

Figure 44. Mario Paint detail (2015) by Patrick LeMieux. 10 gallons of paint, unrefined coltan ore; 32 x 26 x 6.5 inches. ................................................................. 146

Figure 45. Mario Paint (2015) by Patrick LeMieux. 10 gallons of paint, unrefined coltan ore; 32 x 26 x 6.5 inches. ................................................................. 147

Figure 46. Mario Paint detail (2015) by Patrick LeMieux. 10 gallons of paint, unrefined coltan ore; 32 x 26 x 6.5 inches. ................................................................. 148

Figure 47. 99 Exercises in Play (2012-) by Patrick LeMieux. 4 Unity Games, 4 Nintendo Controllers, 4 USB Dongles, 4 Arcade Cabinets; size variable. ......................... 149

Figure 48. 99 Exercises in Play detail (2012-) by Patrick LeMieux. 4 Unity Games, 4 Nintendo Controllers, 4 USB Dongles, 4 Arcade Cabinets; size variable. ................. 150

Figure 49. 99 Exercises in Play detail (2012-) by Patrick LeMieux. 4 Unity Games, 4 Nintendo Controllers, 4 USB Dongles, 4 Arcade Cabinets; size variable. ......................... 151

Figure 50. 4:57 (2013) by Patrick LeMieux. Single channel color video, stereo sound; size variable, 6 minutes. ............................................................................. 152
Figure 51. *Magic Circle* (2015) by Patrick LeMieux. Single channel color video, stereo audio; size variable, 2 hours and 9 minutes.
1. Introduction

“This game requires R.O.B.™, your Robotic Operating Buddy.”
--Nintendo of America Inc., n.d., 1985

In the dark age of digital games, located between the labs and arcades of the 1960s and 1970s and the networked communities of the mid to late 1990s web, single-system software was designed for private play. Game designers mistook the one-player game for the medium and players forgot that videogames were also instruments, equipment, tools, and toys for making metagames. Phantom author functions were spoon-fed to consumers in the form of industry-produced guides, official hotlines, and advertising magazines that simultaneously taught consumers the “right” way to play while strategically masking the fact that videogames are agnostic to how they are played. Industry certification, rights management, security bits, anti-piracy chips, and other technologies of enclosure became part of the production and consumption of videogames as a mass medium and reduced the rules of the game to the mechanics of the system. In the time between the oral communities organized around public arcades and PDP mainframes and the text-based collectives internet relay chats, bulletin boards, and forums of the early internet, the greatest trick the videogame industry ever pulled was convincing the world that videogames were games in the first place. In the twenty-first century, videogames operate as the ideological avatar of play, replacing leisure by
first short-circuiting phenomenal experience through the multi-scalar and multi-temporal operations of technical media, refreshing their own cultural history through the logic of what Terry Harpold (2009, 3) calls the “upgrade path,” supplementing gameplay with an unknowable form of labor, and replacing the promises of radical play with serial consumption.

After the videogame crash of 1983, the distribution and sale of the Nintendo Entertainment System (NES) in American toy stores rebranded the medium as a children’s entertainment in contrast to Atari’s “Video Computer System” (Bogost and Montfort 2009, 134). Rather than courting computer hobbyists, the NES was packaged with R.O.B., the Robot Operating Buddy, and became one of the first videogame systems marketed and purchased for rather than by their new target demographic: children (see Figure 1). Unlike its Japanese counterpart, the Family Computer Disk System or “Famicom,” the NES was skeuomorphically designed to ape the cartridge-based, frontloading VHS decks central to the American den in order to embed itself within the media ecology of 1980s and 90s living rooms. To further control and capitalize on these domestic spaces Nintendo invented and implemented proprietary anti-piracy technologies that allowed them to “certify” what type of content appeared on their closed system. Moreover, play was more furtively regulated through didactic and pedagogical documents published in the form of gaming magazines and tip hotlines—each of which were, by the mid-1980s, funded by game development companies.
themselves. Since 1985, the videogame industry has continued to enclose games, converting players into “gamers” and transforming games into a specific set of packaged products that replaced leisure with a covert form of consumption and affective labor. 

When did the term game become synonymous with hardware warranties, packaged products, intellectual property, copyrighted code, end user licenses, and digital rights management? When did rules become conflated with the physical, mechanical, electrical, and computational operations of technical media? When did the term player become a code word for consumer? And, after the market, what kind of play remains in the wake of the videogame industry?

Figure 1. R.O.B., the Robotic Operating Buddy, was packaged with the Nintendo Entertainment System in order to market the videogame console as a children’s toy.
1.1. Aftermarket

Given the environmental costs, labor issues, intellectual property struggles, and ever-increasing financialization that attends videogames as technical media, numerous scholars have attempted to theorize the complex relationship between games and capital. Alexander Galloway’s “countergaming,” McKenzie Wark’s “gamer theory,” Mary Flanagan’s “critical play,” and Nick Dyer-Witherford and Greig de Peuter’s “games of multitude” each suggest a more radical possibility for videogames in terms of their practical application, philosophical experimentation, cultural critique, and political action. In Gaming: Essays on Algorithmic Culture, Alexander Galloway (2006, 109) applies the term “countergaming” to the work of artists like Cory Arcangel, Tom Betts, Brody Condon, Joan Heemskerk and Dirk Paesmans (Jodi), Joan Leandre (retroYou), Anne-Marie Schleiner, and Eddo Stern. In the same way that French countercinema contrasted Hollywood narrative, countergames “conflict violently with the mainstream gaming industry’s expectations for how games should be designed . . . [and] often defy the industry’s design style point-for-point, with the goal of disrupting the intuitive flow of gameplay” (Galloway 2006, 108). But despite the fact that Galloway (2006, 125) calls for “a critique of gameplay itself. . . [that redefines] play itself and thereby realiz[es] its true potential as a political and cultural avant-garde,” he admits that countergaming is an “unrealized project” which “has yet to flourish.” Countergaming is relegated to the status of a speculative rather than actually existing practice.
In Gamer Theory McKenzie Wark (2007, 023) similarly argues “[g]amespace needs theorists—but also a new kind of practice. One that can break down the line that divides gamer from designer, to redeploy the digital so that it makes this very distinction arbitrary.” In the tradition of the Guy Debord’s dérive and Walter Benjamin’s flâneur, Wark’s gamer theorist is a “trifler” or “archaeologist” who is “not out to break the game” but “wants to hack or ‘mod’ the game . . . to play even more intimately within it” and “make the now rather familiar world of the digital game strange again” (2007, 021, 022, 225). Also inspired by the aesthetic experiments of twentieth century artists, in Critical Play: Radical Game Design, Mary Flanagan (2009, 6, 261) calls for “a careful examination of social, cultural, political, or even personal themes that function as alternates to popular play spaces” and, at the end of the book, imagines the possible future of this kind of critical play as “a new discipline of theory and practice” and “a tool for future game makers, game designers, and scholars” that would “instill the ability to think critically during and after play.” Flanagan’s (2007, 261-2) book concludes with a chapter called “Designing for Critical Play” in which she urges game designers to “unplay, reskin, and rewrite the hidden transcripts so tenaciously rooted in the systems of our world” in order to “manifest a different future” through “interventions at the level of popular culture.”

Finally, Dyer-Witheford and de Peuter (2009, xv) borrow Michael Hardt and Antonio Negri’s theory of the multitude in order to characterize videogames as their
“paradigmatic media of Empire.” Embedded firmly within the dialectical processes of capitalism, Dyer-Witherford and de Peuter (2009, 213) also see the potential for videogames to offer resistance through “counterplay, dissonant development, tactical games, polity simulators, self-organized worlds, and software commons.” These ludic practices comprise what they see as “games of multitude,” yet “the play of the multitude still remains locked inside games of Empire” (2009, 213). Despite the fact that the videogame industry exemplifies the capacity of informatic capitalism to appropriate “so many apparently iconoclastic and utopian ideas” and that “[g]ame capital, rushing to take on team production, modding, machinima artists, MMO populations, digital distributions, and peer-to-peer networks, is a good exemplar of this process in Empire,” Dyer-Witherford and de Peuter (2009, 228) remain hopeful that “in this process of cooptation, Empire cultivates capacities that might exceed its grasp.”

From countergames to gamer theorists and from critical play to games of multitude, each of these concepts has something in common: the differentiation between videogames as platforms for creative practice, philosophical experimentation, cultural critique, and political action and videogames as a mass medium and commodity. However, each concept is figured as an unrealized horizon and a speculative rather than actual practice. Where are the gamer theorists making countergames? When will we critically play games of multitude? Aftermarket, A Game Design Philosophy does not seek to merely speculate on the possibilities for videogames in an imagined future, but
undertakes a media archaeology, media theory, and media practice of those practices that are already taking place. Aftermarket is a history of those players that transformed videogames from “entertainment systems” into open platforms for reverse engineering, modding, hacking, reproducing, playing, critiquing, and making metagames in the aftermarket of the videogame industry.

After the home-console era in the 80s and 90s and especially with the rise of social media and video sharing services, players have begun to play in the aftermarket and, as a result, games have changed. Yet, the desire for an autonomous, ahistorical, and authored experience—what Katie Salen Tekinbaş and Eric Zimmerman (2004, 450) name the “immersive fallacy”—continues to propel the games industry. Although Nintendo persists in distributing decades-old ROM dumps to their various virtual consoles, the Super Mario Bros. (1985) running on contemporary consoles is not the same game that played almost thirty years ago. The intimate, serialized experience of private play has been repurposed and reinvented through both the physical network and the networked subjectivities of contemporary players. Once players began distributing ROM hacks online, building multiplayer interfaces in LUA, competing in telematic races on IRC, and imagining the game in terms of a cultural history of play rather than software and hardware, Super Mario Bros. mutates from intellectual property to a cultural platform and a medium for making metagames: “the games we play in, on, around, and through other games.” (Boluk and LeMieux, Forthcoming, 4).
In *Metagaming: Videogames and the Practice of Play*, Stephanie Boluk and I (Forthcoming, 5) argue that:

From the most complex house rules, arcade cultures, competitive tournaments, and virtual economies to the simple decision to press start, pass the controller, use a player’s guide, or even purchase a game in the first place, for all intents and purposes metagames are the only kind of games that we play. And although the term *metagame* has been used for decades in both Nigel Howard’s mathematical game theory and in Richard Garfield’s game design, the concept has taken on renewed importance and political urgency in a media landscape in which videogames have not only colonized and enclosed the very concept of games, play, and leisure but ideologically conflate the creativity, criticality, and craft of play with the act of consumption.

The relationship between games and the act of making metagames (consciously and unconsciously) recalls Catherine Malabou’s theory of brain plasticity in *What Should We Do with our Brain?* (2008).

In *What Should We Do with Our Brain?*, a pithy and polemical essay on the relationship between neuroscience, phenomenology, and neoliberal capitalism, Malabou’s (2008, 12) asks a single, urgent question: “[w]hat should we do so that consciousness of the brain does not purely and simply coincide with the spirit of capitalism?” Echoing Marx’s declaration that “[h]umans make their own history, but they do not know that they make it,” Malabou’s (2008, 1) answer is to become aware of the fact that “[h]umans make their own brain, but they do not know that they make it.” In other words, we must become aware of the brain’s plasticity. While Malabou (2008, 5) deploys the term *plasticity* to discuss to the brain’s capacity to be both ‘‘formable,’ and formative at the same time,” she also invokes more radical connotations of annihilation
as with plastic explosives like *plastique*--a different kind of grey matter made from the combination of nitroglycerine and nitrocellulose. For Malabou, because the brain is constitutive of human experience, one can only speculate on the nature of the experience and history of the brain itself. In 2015, seven billion absolutely unique and unimaginably dense moving sculptures reflect both the microhistory of neuronal processes and the macrohistory of ideology in the twenty-first century. We make our own brain, but we do not know we make it.

Following Derrida’s reading of Hegel, Malabou’s concept of plasticity could go by another name: *play*. After all, humans also make their own *games*, but they do not know they make them. Players throughout the history of videogaming constantly (and most of the time unconsciously) make their own metagames but the logic of the marketplace obfuscates this form of critical practice. Consider, for example, the ways in which the concept of plasticity is co-opted by contemporary cognitive capitalism to naturalize models of economic precarity in the form of flexible, contingent labor. Noting this tendency, Malabou (2008, 46) writes, “[i]f I insist on how close certain managerial discourses are to neuroscientific discourses, this is because it seems to me that the phenomenon called ‘brain plasticity’ is in reality more often described in terms of an economy of flexibility.” Flexibility, she declares, “is the ideological avatar of plasticity” (Malabou 2008, 12). Everywhere the brain is in chains and plasticity opens a path to freedom. At the same time, plasticity--like play--is always at risk of being employed as
the “biological justification of a type of economic, political, and social organization in which all that matters is the result of action as such: efficacy, adaptability—unfailing flexibility” (Malabou 2008, 31). Flexibility is an accessible, “vague notion, without tradition and without history,” whereas plasticity remains hidden yet promises “neuronal liberation” in which consciousness of the brain itself becomes a vehicle for “producing the conditions of possibility for a new world of questioning” (Malabou 2008, 13). If flexibility is the “ideological avatar of plasticity,” then, by extension, videogames are the ideological avatar of play (Malabou 2008, 12).

1.2. Game Design Philosophy

Against this ideological avatar of play, Aftermarket, a Game Design Philosophy attempts to uncover alternative histories of play defined not by the code, commerce, and computation of videogames but by the diverse practices and material discontinuities that emerge between the human experience and the nonhuman operations of technical media. The videogame conceived as a copyrighted piece of code, a patented set of processes, or protected intellectual properties is not, in fact, what is at play in front of television screens. Just as finance capital trades on the pure de-referentialized abstractions of the market, videogames mobilize abstract rules in terms of mathematical formulae, computational processes, and intellectual property apart from their practical implementation. McKenzie Wark determines, “[w]ithout exception, [games] all come
down to a strict decision: out or in, foul or fair, goal or no goal. Anything else is just ‘play’” (2007, 79). As Boluk and I (Forthcoming, 13) argue,

The metagame, then, emerges as the material trace of the discontinuity between the phenomenal experience of play and the abstract rules of digital games. From player-created modifications to ironic parodies and from fan-fiction and forum discussion to the latest trends made popular by professional players, metagaming turns autonomous and abstract pieces of software into human forms of play and turn player into game designer.

Much of these practices take place within an after-market economy that is completely independent of the traditional economic cycle of mainstream games.

*Aftermarket* is an attempt to document these practices in the form of a true game design philosophy—a critical practice in which playing, making, and thinking about videogames occurs within the same act. This call to collectively and concurrently play, think, and make is an attempt to account for the material complexity of all media in twenty-first century—from digital games to print manuscripts. As such, a game design philosophy based on metagaming integrates making and critique in the hopes of adequately theorizing the complexities of technical media while simultaneously experimenting with new forms of creative practice.

As N. Katherine Hayles and Jessica Pressman (2013, xvii) argue in *Comparative Textual Media*, “[w]ithout theorizing, practice can be reduced to technical skills and seamless interpolation into capitalist regimes; without practice, theorizing is deprived of the hands-on experience to guide it and develop robust intuitions about the implications of digital technologies.” The very concept of *metagaming* decenters and
reconceptualizes the production of both hardware and software within the entertainment industry as well as scholarly research within the academy. As Hayles and Jessica (2013, xix) suggest, this comingled form theory and practice “explores the possibilities for cultural, social, economic, and theoretical transformation not only by tearing down but also by building up, thereby opening new horizons of understanding and alternative practices.” In academia, the hope is to “catalyze new kinds of research questions, attract students, reconceptualize curricula, and energize faculty” whereas the industry should benefit from a deeper engagement with the phenomenology, history, and materiality of play (Hayles and Pressman 2013, xxi). Aftermarket participates in this new form of scholarship. This research would not be possible without a sustained engagement with the material specificity of videogame platform, software, code, and cultural practices.

One of the most popular platforms for producing metagames is Nintendo’s Super Mario franchise. Just as videogames function as the ideological avatar of play, Mario has become the avatar of videogames in general. Jeremy Parish (2014, 2), for example, writes “1985’s Super Mario Bros. may actually be the single most influential video game ever—certainly it rivals Pac-Man, Tetris, Doom, and Space Invaders in terms of importance. In any case, it has exerted tremendous influence over any game revolving around jumping, both 2D and 3D, which makes for an awful lot of descendants.” And,
in his painstaking platform study of the Nintendo Entertainment System, Nathan Altice (160) argues

Super Mario Bros, was Nintendo’s breakout console hit, spawning one of the most popular and influential videogame franchises of all time and spurring Nintendo’s acceptance in the U.S. videogame market—in fact, saving that market from imminent collapse. Super Mario Bros. helped make the company’s name synonymous with videogames. In the U.S., “playing Nintendo” replaced “playing Atari” as the linguistic metonym for playing any videogame, not just software exclusive to Nintendo’s console. Super Mario Bros. also entrenched the platformer as the NES genre par excellence, spawning a chain of look- and play-alikes that continue even today.

Because of Super Mario Bros. ubiquity, the game operates as a control variable when exploring aftermarket game design philosophy. Between 2004 and 2014, many diverse communities of players appropriated the platformer game as their platform to experiment with new forms of play.

Alongside the original Super Mario Bros. trilogy on the Nintendo Entertainment System and the ever-expanding Super Mario Land, World, Sunshine, and Galaxy titles on later consoles, there are now a host of player-created variations that circulate on the Internet. Beyond simply building modifications or “mods” using classic Mario game engines or completing Mario games in record times, these experimental gaming practices stretch Mario to his limits. There have been “Mario Quadruns,” in which the first four Super Mario titles are played simultaneously and “Mario Mashups” starring Mario in other 8-bit era videogames. There are “Mario Speedruns” exploiting glitches to complete games in seconds while “Infinite Mario,” a Java application featuring randomly generated levels, stretches out as far as the thumb can play. Dozens of
automatic “Mario Sequencers” convert custom Super Mario World levels into j-pop beat machines, automatically propelling Mario across each level like the proverbial bouncing ball set to synthesized music while “Asshole Mario” mods make it difficult for the player to do anything at all. Finally, there is “Quantum Mario,” an emulator used to play with the Everett-Wheeler “Many Worlds Interpretation” of quantum physics and “Mario AI” competitions have been held at the Ph.D. level since 2009 in many computer science departments across the globe. Even Cory Arcangel’s Super Mario Clouds (2002), one of the first game mods exhibited in the Whitney Museum of American Art, modifies Super Mario Bros., and attempts to evacuate the game of all content except the blue, monochromatic sky and a few blocky, leftward floating cloud forms. From remakes of ROM hacks to speedruns of sequencers, Mario has been manipulated, duplicated, generated, appropriated, and aggregated. These metagaming practices have developed as players grow bored with standard challenges and begin to metagame the limits of the hardware, software, and even player community in the aftermarket.

Beginning with the history of how videogames became the ideological avatar of play in the twenty-first century, Aftermarket investigates three communities that metagame the phenomenal, historical, economic, and practical possibilities of games. Through a close investigation of real-time attacks, tool-assisted speedruns, ROM hacking, and hardware reproduction, this dissertation examines how play in the aftermarket becomes a form of game design. Chapter 2 examines the deep history and
infinite possibility of play obfuscated by the upgrade path and recovered through the platform-based practices of tool-assisted speedrunners. Chapter 3 focuses on the problem of experiencing digital play in the speedrunning community, a group of virtuosic performers and electronic athletes who complete games as quickly as possible and in the process perform a kind of software studies that attempt to address the microtemporal operations of videogames at the level of a single frame. Chapter 4 considers ROM hacking as a critical practice in which software and hardware, code and platform, are entangled despite the utopian desire for immaterial, autonomous, and escapist videogames. Finally, Chapter 5 documents an art exhibition developed alongside *Aftermarket* and featuring after-aftermarket games that extend the practices of each community.

These chapters are organized around *Super Mario Bros*. This NES game figures as both the paradigmatic videogame in terms of code, software, platform, intellectual property, and capital and, on account of its massive sales and cultural ubiquity, has become a prolific medium for making metagames and a programming platform in its own right. As a methodology for engaging both the theory and practice of videogames, each chapter of *Aftermarket, a Game Design Philosophy* follows a specific subset of media theory—from platform studies to software studies to critical code studies—in order to combine historical and practical research terminating in a form of critical artmaking. By deploying this method, the dissertation not only performs a platform study and
media archeology of hardware and software packaged for consumption, but also attempts to reverse engineer new metagames from the platform of cultural play--after-aftermarket games. What follows is a short summary of the chapters of the dissertation.

1.3. Chapter Descriptions

<table>
<thead>
<tr>
<th>Chapter Descriptions</th>
<th>Chapter 2</th>
<th>Chapter 3</th>
<th>Chapter 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Community History (2004-2014)</td>
<td>Tool-Assisted Speedrunning</td>
<td>Real Time Attacks</td>
<td>ROM Hacking</td>
</tr>
<tr>
<td>Primary Actors (Players)</td>
<td>Morimoto and Joel Yliluoma</td>
<td>Andrew Gardikis and Happy Lee</td>
<td>Cory Arcangel</td>
</tr>
<tr>
<td>Primary Objects (Games)</td>
<td>moSMB3.wmv</td>
<td>4:57 Record Time</td>
<td>Super Mario Clouds</td>
</tr>
<tr>
<td>Material Technology (Super Mario Bros.)</td>
<td>NES 4021 Bit Shift Register (Input)</td>
<td>Ricoh 2A03 Micro-Processor (RAM)</td>
<td>CMOS Mask ROM (ROM)</td>
</tr>
<tr>
<td>Academic Discourse (Media Studies)</td>
<td>Platform Studies</td>
<td>Software Studies</td>
<td>Critical Code Studies</td>
</tr>
<tr>
<td>Critical Intervention (Media Theory)</td>
<td>Technology and Experience</td>
<td>Probability and Play</td>
<td>Capital and Materiality</td>
</tr>
<tr>
<td>Discoveries (Historical Revisions)</td>
<td>Morimoto’s moSMB3.wmv is the genealogical origin of both speedrunning and tool-assisted speedrunning on consoles.</td>
<td>The history of real time attacks is not propelled by individual skill but by the discovery of new exploits that break the metagame.</td>
<td>Super Mario Clouds is not a ROM hack and obfuscates this fact to promote a narrative of erasure as a conceptual gesture in art.</td>
</tr>
<tr>
<td>Artistic Practice (Solo Exhibition)</td>
<td>RSG-SMB-TAB-SSS, One Switch, Brothers</td>
<td>4:57, 99 Exercises in Play</td>
<td>Coin Heaven, Conflict Free, Mario Paint</td>
</tr>
</tbody>
</table>

Although play is irreducible, games repeat. Beyond the serial repetition that characterizes industrial forms of mechanical reproduction like newspapers, comics, novels, and films, in the case of videogames the microtemporal speed of serial interfaces
and massive scale of serial distribution operate both below and above the horizon of conscious experience. As such, the serial operations of videogames structure, enclose, and ultimately alienate the technical processes of play from the conscious knowledge of the player. In “Digital Seriality: On the Serial Aesthetics and Practice of Digital Games,” Shane Denson and Andreas Jahn-Sudmann (2014) characterize the diachronic sequencing of serial interfaces and synchronic consumption of videogames as “digital seriality.” Chapter 2 explores digital seriality through the history and practice of tool-assisted speedrunning, a form of metagaming that stages an intervention at the level of serial interfacing and subsequently disrupts the collective serialization of videogames as a mass medium. From the operations of the NES-4021, a parallel-to-serial shift register that governs controller input in Nintendo Entertainment System, to the history of moSMB.wmv, an early speedrunning video by Morimoto that went viral in 2003, tool-assisted speedruns transform twitch-based platform games into turn-based puzzles and single player experiences into massively multiplayer online games by playing the serial interface.

Unlike speed runs or real time attacks, tool-assisted speedruns allow for slow-motion key logging, instant replay, re-sequencing, and even artificially intelligent agents to produce superhuman play. The first tool-assisted speedruns appeared in the 1990s after advancements in open-source software enabled Doom (1993) players to program
“speed demos” or “build runs” which could be autonomously executed by computers.

No player necessary. As the community at TASvideos.org states (TASVideos 2014):

[W]e strive to push games to their limits. The emulators we use allow for undoing mistakes, slow-motion gameplay, and even in some cases utilizing robots to do our bidding. Using these tools, we overcome human limitations to complete games with extremely high precision, entertaining our viewers as our players tear through games at seemingly impossible speeds. The end result of this process is simply a series of key-presses which may be performed on the original hardware.

In logging the controller input for every frame of a given playthrough, tool-assisted speedruns can predict overall RAM states to produce a deep history of specific instances of play. There is no need for decoding video and counting frames because a TAS operates at the level of the frame in the first place.

Following the early history of tool-assisted speedrunning, Chapter 3 inspects the phenomenology of a single frame. Super Mario Bros. is designed, first and foremost, around the fantasy of direct control. Press “A” and Mario jumps. Press “right” and Mario moves. Reach the flagpole and the level is over. The time and scale of action represented in the game is formatted specifically for the CRT television screens of an international audience in the mid 1980s. However, despite the best attempts to naturalize the control scheme and onscreen interface to human play, the increased fidelity of motion on the NES also increases variability. Chapter 3 questions the correlation between human play and digital games by analyzing the performances of speedrunners and their real time attacks (RTA). A speedrun is the attempt to complete a goal within a videogame as quickly as possible by human players on original hardware.
Published by the Speed Demos Archive (SDA) and competitively raced live at Speed Runs Live (SRL), speedrunners are “expected to use every method at their disposal, including glitches, to minimize time [while] side issues such as entertainment are secondary” (Speed Demos Archive). Though playing a game quickly seems obvious and is clearly suggested in the design of videogames in the home console era, speedrunning, takes this logic to its extreme, and, in doing so, challenges the very notion of what a videogame is and can do. Speedrunners do not merely choose to play games fast but, in the act of applying an unnecessary obstacle or speed constraint to a set of invariable mechanics, invent new aftermarket games by comparing play and producing communities outside and apart from these console games’ original rulesets. One of the first and most famous games to speedrun is Nintendo’s Super Mario Bros, and Andrew Gardikis held the world record in this game for seven years, from 2007 to 2014.

For almost a decade, Gardikis has continued to work on his record, whittling down his time frame by frame. His final goal is to trim 8 frames off his current time of 4:58:09 for a 4:57. The number 4:57 represents one horizon of possibility for human play within Super Mario Bros, is the ultimate goal of Gardikis--what he calls his “gaming masterpiece.” In a practically Duchampian gesture, Gardikis has turned a game of skill into a game of luck by partitioning play into more and more granular units beyond the scale of human perception. At this threshold of performance that, a skill-based game is transmuted into a luck-based game. What distinguishes Gardikis from other players is
not his talent, but capacity to engage in thousands of hours of repetition in order to wait for all the elements to align. Each playthrough is now more a dice roll than the demonstration of virtuosic play—Gardikis’ body is turned into a computer processing the statistics of a random number generator.

In order to assemble these frame-perfect microhistories, tool-assisted speedrunners must hack. ROM hacking, a practice intimately related to both speedrunning and tool-assisted speedrunning represents the next phase of *Aftermarket*, a Game Design Philosophy and the site of initial contact with the electronics driving digital games. Whereas the mechanics of tool-assisted speedrunning described in Chapter 2 foreground the impossibility of ever fully accessing the nonhuman temporal domains of videogames and the phenomenal experience represented in Gardikis’ speedrunning in Chapter 3 produces a disconnect between embodied knowledge and technical media, Chapter 4 demonstrates the indexical relation between 6502 assembly code and the NES’ electronic components while acknowledging the industry’s desire for videogames to stand apart from their platforms. This chapter begins with a critical code and software study of *Super Mario Bros.* source code and culminates with an analysis and remake of Cory Arcangel’s *Super Mario Clouds*. Despite much publicity and scholarship on Arcangel’s landmark artgame, there is no published critique of the fact that none of the code from *Super Mario Bros.* is used in the mod. Arcangel’s piece neglects the specificity of the Nintendo Entertainment System and operates instead
according to an economy of desire based on a utopian fantasy at the heart of both games and art.

The practitioners of ROMhacking.net, a conglomerate of many smaller hacker communities, believe that hacking:

Compris[es] both the analysis and manipulation of data, [and] can appeal to the spirit of exploration, creative problem solving, engineering, and creativity. Thanks to the subject’s breadth and its propagation through the Internet, ROM hacking has become an art form. (GuyInSummers 2005)

Arguably one of the most famous ROM hacks is Cory Arcangel’s Super Mario Clouds (2003). Exhibited at the Whitney in 2004, Arcangel claims that “Super Mario Clouds is an old Mario Brothers cartridge which I modified to erase everything but the clouds.” After interviewing the artist for the New Yorker a few years ago, Andrea K. Scott (2011) writes that Arcangel’s “idea was as simple as silk-screening soup cans: take the code to the classic 1985 Nintendo cartridge and erase everything but the clouds, which typically drift behind the action.” Scott is more right than she knows. As Foucault (1983, 54) suggests at the end of his book on René Magritte, “A day will come when, by means of similitude relayed indefinitely along the length of a series, the image itself, along with the name it bears, will lose its identity. Campbell, Campbell, Campbell, Campbell.”

Following Foucault, in this chapter I argue that—in the same way that is not a pipe and that is not soup—this is not Super Mario Bros. Arcangel’s Super Mario Clouds does not actually contain any source code from the PRG ROM driving Nintendo’s original game.
Arcangel embraces the hacker ethos and has open sourced much of his artwork. Even with such transparency, the fact that Super Mario Clouds is not running any of Mario’s original code goes unnoticed in the literature surrounding the project. Arcangel even notes on github, “to compile this code requires Bob Rost’s NBASIC, and the 6502 compiler NESASM,” a clear sign that the program was written in BASIC, not assembly. Thus Super Mario Clouds is less like a hack and more a homebrew—a new piece of software developed in the aftermarket economy that surrounds a few of these now vintage systems. The pixel-wise movement, randomization, and even coloring do not match Shigeru Miyamoto’s game. Alongside this media archeology and critical code study of Super Mario Clouds, a close analysis of race, gender, and political economy in Arcangel’s art reveals the ways in which the utopian ideology of the “white cube” and the “magic circle” overlap.

Aftermarket culminates in a chapter on Platform Games, an art exhibit designed in tandem with this book and installed at Babycastles from May 7 to 17, 2015. Featuring original ROM hacks and EPROM poetry, a speedrunning documentary and tool-assisted tablature on a self-playing guitar, a one-switch controller and a network of autonomous Nintendos, as well as an arcade of Mario metagames and a bucket of unrefined coltan ore, Platform Games did not attempt to reify Nintendo’s game as a iconic or ideal piece of software, but appropriates, manipulates, perforates, duplicates, aggregates, and dissipates videogames into a different kind of “Mario Paint”—a historically specific
medium for making metagames and media art. If the MN4021B shift register is Mario Paint and the Ricoh 6502 processor is Mario Paint and the PPU graphics chip is Mario Paint and the APU sound card is Mario Paint the CMOS ROM masks storing Super Mario Bros. are Mario Paint then conflict minerals, slave labor, and e-waste are also colors in Mario’s crayon box. Platform Games painted a picture that disrupts the cultural logic of an immersive and escapist magic circle, black box, and white cube in which the phenomenal, political, economic, and material history of play are erased. Chapter 5 documents the exhibit and details each of the artworks designed as an “afterword” to Aftermarket and also operating within the genre of art “after” other artists. Citing the practices of players like Morimoto, Joel Yliluoma, Happy Lee, Andrew Gardikis, Cory Arcangel, Liz Ryerson, Peter Greenwood, Alexander Galloway, and even Emily Dickenson, Platform Games attempts to articulate the history of the aftermarket by using those metagames invented by its players as platforms for making art.
2. NES-4021 to moSMB3.wmv: Speedrunning the Serial Interface

“A game is input->output. Repeat. Each cycle of this costs 1 unit time. A game also has a state. The game state is the memory. All of the variables. One game state may be observed and agreed upon as the ‘start.’ Another may be observed and agreed upon as the ‘end.’ Measuring the number of input->output cycles that occur between the ‘start’ and ‘end’ is a speedrun. The goal of a speedrun is to lower this as much as possible.”
--Cosmo Wright; July 24, 2014

Input 1 then Input 2 then Input 3 in Frame 1. Frame 1 then Frame 2 then Frame 3 in Level 1. Level 1 then Level 2 then Level 3 in World 1. World 1 then World 2 then World 3 in Game 1. Game 1 then Game 2 then Game 3 in Franchise 1. Franchise 1 then Franchise 2 then Franchise 3 in Platform 1. Platform 1 then Platform 2 then Platform 3 in the serial history of videogames. Although phenomenal, material, and historical instances of play are irreducible, games repeat. Alongside the mechanical reproduction of newspapers, magazines, comics, novels, records, television programs, film, and even websites, industrial forms of seriality structure the diachronic production and synchronic consumption of videogames as a mass medium. Whether taking the form of a series of actions, a series levels, or a series of games, the term seriality can refer to both

1 The first long distance telegraph message, serially transmitted by Samuel Morse from a hearing in the chamber of the Supreme Court in Washington, DC to Albert Vail at the Mount Clair depot in Baltimore and back, reads “WHAT HATH GOD WROUGHT.”
the temporal relation between two or more objects in time (e.g., sequences) or the similitude between two or more objects in space (e.g., copies)—twin effects of mechanical production. Produced in mass yet consumed individually, the repeating elements of serial media like Nintendo’s Super Mario Bros. both reproduce and reduce specific instances of play to generalizable sequences of serial pulses sent both to and from standard controllers. Aside from the haptic sensation of fingers pressing plastic and the physicality of phosphors glowing on the surfaces of CRT screens, games repeat. From subsecond sampling and sequencing of button combinations to the conspicuous consumption surrounding videogames as commodities, serial games operate both under and over individual experience, abstracting the specific phenomenal, material, and historical qualities of play into abstract quantities that repeat in time and space. As Jean-Paul Sartre (2004, 262) writes, these serial forms of industrial culture “are lived separately as identical instances of the same act.”

In “Digital Seriality: On the Serial Aesthetics and Practice of Digital Games,” Shane Denson and Andreas Jahn-Sudmann differentiate three forms of “ludic seriality” based on three scales of industrial repetition in games. Using Super Mario Bros. as an example, their tripartite schema includes “intra-ludic seriality, which manifests itself

---

2 In Sartre’s Critique of Dialectical Reason, seriality becomes a means of expressing contemporary urban anomie. For Sartre, seriality is the process by which individuals are organized and arranged in relation to one another through a social and technical apparatus. Yet membership within the series goes unacknowledged as each member remains oblivious to their larger systemic relations. Sartre (2004, xxviii) contrasts seriality to what he calls “group formation,” the process of becoming conscious of one’s relationship to a group identity—which subsequently enables the possibility for political action.
within games” (e.g., the sequence of stages, levels, and worlds inside a given game like Level 1-1, 1-2, and 1-3); “inter-ludic seriality, which emerges between games” (e.g., the relationship of sequels, prequels, and remakes which extend a series of games like Super Mario Bros. 1, 2, and 3); and “para-ludic seriality, which is constituted outside of the actual games” (e.g., the transmedial and largely fan-produced works that exceed the artificial corporate boundaries of a game as an autonomous, consumer object not unlike Gérard Genette’s concept of “paratext”) (Denson and Jahn-Sudmann 2013, 11). Although these forms of ludic seriality are not unique to Super Mario Bros. or digital media more generally, the incredible speeds of serial interfaces and widespread distribution of videogames transform industrial modes of spatial and temporal reproduction into what Denson and Jahn-Sudmann characterize as “digital seriality.”

Digital seriality hinges on “the microtemporal scale of individual players’ encounters with algorithmic computation processes (the speed of which escapes direct human perception and is measurable only by technological means)“--or “serial interfacing“--and “the macrotemporal (more properly ‘historical’) level of collective brokerings of political, cultural, and social identities in the digital age“--or “collective serialization” (Denson and Jahn-Sudmann 2013, 1). Operating both below and above the bandwidth of conscious experience, the serial interfacing and collective serialization of videogames structure and enclose play without the player’s explicit knowledge--a form of alienation that Bernard Stiegler (2010, 45) calls “systemic stupidity.“ From the
operations of the NES-4021 shift register to Morimoto’s viral video, moSMB.wmv, this chapter explores the history and practice of tool-assisted speedrunning, a form of aftermarket game design that plays the serial interface.

2.1. NES-4021

Press “START.” Sixty times a second3 an electrical impulse is sent from the Nintendo Entertainment System (NES)4 to the sixth pin of its first controller port. From port to plug to cord to controller, the signal travels down one of five colored wires to the NES-4021, an 8-bit, parallel-to-serial shift register housed within a standard controller (see Figure 2). After receiving a high pulse for 12 microseconds5 from the orange wire connected to pin six, the 4021 “latches” the state of the controller’s eight buttons and immediately sends a single pulse of electricity back to the NES along the yellow wire, pin seven. This pulse represents a single bit of serial data. An absent or “low” current pulse (0V), is interpreted as a 0 by the NES’s central processing unit (CPU)—a modified

3 Outside of those regions that implement PAL or SECAM encoding standards for color television (e.g., Europe, Africa, Australia, most of Asia, and half of South America), the Nintendo Entertainment System (NES) operates at about 60 Hz. In Japan and North America, for example, the NES’s central processing unit (CPU) runs at 1.789773 MHz. Its picture processing unit (PPU), the Ricoh RP2C02, requires 1 CPU cycle for every 3 “dots” it renders. There are 341 dots in a single scanline and 262 scanlines in a single frame. Therefore if the 1.789773 MHz refresh rate of the CPU is multiplied by 3 PPU cycles then divided by the 341 dots in each of the 262 scanlines, the system can update the screen about 60.0988 times a second or 60.0988 Hz, very close to the NTSC encoding standard. This standard refresh rate limits observable output. Although the states of the buttons could be (and in some cases are) sampled numerous times per frame, this essay assumes a standard of 60 inputs every second for the sake of clarity when describing serial input (especially in the case of NES emulators).

4 Although there are major design differences and minor technical differences between the North American Nintendo Entertainment System (NES) and the original Japanese Famicom (FC), for the purposes of simplicity this essay refers to both systems as the NES.

5 A microsecond (µs) is one millionth of a second.
version of MOS Technology’s popular 6502 processor called the Ricoh 2A03. A “high” current pulse (+5V) is registered as a 1.

Although digital media are never quite digital, the infinitely individuated physical attributes of electrical current are measured, sampled, and abstracted into serial units by mechanisms like the flip-flop circuits of the 4021 and the semiconductor arrays of the 6502’s input/output registers.6 As Matthew Kirschenbaum (2008, 61) confirms in Mechanisms: New Media and the Forensic Imagination, “while bits are the smallest symbolic units of computation, they are not the smallest inscribed unit.”7 Serial communication, however, privileges the discrete, repeatable bit over the continuous flows of electricity. Surges, spikes, static, and other forms of interference are either ignored by the processor, translated into bits by the processor, or crash the processor—there is no middle ground.8 Press “START.” A continuum of analog phenomena is reduced to the discrete differences that characterize digital media. In the case of this input in this frame of this game on this platform, if the first electrical current measured

---

6 In How We Think: Digital Media and Contemporary Technogenesis, N. Katherine Hayles (2012, 91) argues that both formal and forensic “[m]ateriality comes into existence […] when attention fuses with physicality to identify and isolate some particular attribute (or attributes) of interest.” Importantly, as is the case with shift registers and CPUs, “attention” and “interest” are not necessarily human attributes as a wide variety of mechanisms can observe, identify, and isolate patterns.
7 Although Kirschenbaum focuses on how magnetic inscriptions of data are stored and retrieved on hard disks and drives, his distinction between the formal materiality of bits and the forensic materiality of physical media also applies to the so-called “volatile” memory stored in the semiconductors of RAM or CPU chips. Kirschenbaum (2009, 50) notes “even the popular myth that RAM is always absolutely volatile, gone forever at the flip of a switch, proves false; there are at least experimental techniques for recovering data from RAM semiconductor memory.”
8 As McKenzie Wark (2007, 023) claims in Gamer Theory, “[t]he real violence of gamespace is its dicing of everything analog into the digital, cutting continuums into bits.”
by the CPU after “latching” is above a certain threshold, the 6502 registers a 1 at memory address $4016$. Sent one at a time, always in the same order, the CPU registers the states of each button starting with “A.” If the last bit of memory at $4016$ is a 1, the “A” button was not pressed.⁹

![Diagram of NES controller and serial shift register]

**Figure 2.** The 4021 parallel-to-serial shift register in a Nintendo Entertainment System controller (left) converts the collective state of eight buttons to a serial stream of electrical pulses transmitted over a single yellow wire (right).

Press “START.” Exactly 6 microseconds after the “A” value was registered by the Nintendo’s CPU, there are still seven values “latched” to the shift register. The remaining buttons are relayed in order, one at a time, according to a series of short pulses (interleaved with pauses) that the shift register receives from the fifth pin, the red wire. For each 6-microsecond tick of this “clock” the next bit of data is shifted forward then serially sent to the NES along the yellow wire. Tick, tock, tick, tock, tick, tock, tick.

---

⁹ Unlike the other seven buttons connected to the NES-4021, the state of the “A” button transmits to the NES’s CPU directly after latching. The other seven values are shifted forward and relayed only after receiving signal from the “clock” or “pulse” line.
High, high, low, high, high, high. 1, 1, 0, 1, 1, 1, 1. “B” was not pressed. “SELECT” was not pressed. “START” was pressed. “UP” was not pressed. “DOWN” not pressed. “LEFT” not pressed. “RIGHT” not pressed. A final “tick” is sent to complete the sequence. In 108 microseconds (about one ten-thousandth of a second), one “latch” and eight “clocks” were serially “written” to the controller and, in return, eight bits of data were serially “read” by the NES. Every sixtieth of a second, eight more bits representing the states of eight buttons are polled by the NES-4021 then sent serially from the controller to the 6502 processor. Again and again, the states of eight simple contact switches are relayed in sequence at the last bit of memory at CPU address $4016$. These values are (typically) inverted and accumulated into a single byte of RAM for future reference. The future, in this case, is very short. If $4016$ reads 1, 1, 1, 0, 1, 1, 1, 1, and those values are stored as 00010000 (written as 10 in hexadecimal), then “START” was pressed. If “START” was pressed, perhaps a software routine will begin the game. The sweaty palms, particular grip, and proprioceptive experiences of the player must be translated into digital data. The undifferentiated physicality of play must be sampled, serialized, and stored as bits before it can impact the operation of the videogame. Within the sixtieth of a second between pressing start and the game starting, serial operations unfold at scales and speeds outside the consciousness of the button masher.

Press “START.” On the Nintendo Entertainment System, the interface between player and game, between pressing start and the game starting, is serial. Whereas
parallel communication requires synchronizing or “linking” two or more communication channels in order to send multiple bits of information at the same time, “serial communication consists in transmitting a single bit of data at a time, sequentially, forming a serial data stream” (Advantech 2012). From smoke signals and distant drums to transatlantic telegraph cables and RS-232 standard in computing, long-distance communication is often mediated by a serial interface. Beyond the mere presence or absence of smoke, sound, or signal (which is nevertheless a historically important, albeit rudimentary form of serial data), a second order of complexity emerges in serial communication with time. Morse code, for example, is not only based on repeating tones, but repeating times. The difference between a “dot” and a “dash” is not only defined in terms of the presence or absence of a signal but the division of duration into repeating, increments. Just as a bit is produced by switching mechanisms such as transistors and shift registers, time is produced by strobing mechanisms such as resonating crystal oscillators. Although a single plume of smoke on the horizon transmits information, the production of temporal patterns within a continuous signal (like covering and uncovering a signal fire with a wet blanket), can expand the message from either 0 or 1 to a series 0s and 1s able to encode a variety of formal symbols.

In order to be decoded on the receiving end, serial communication requires temporal standardization to synchronize not only bits between storage media, but also “ticks.” In 1962, the Electronic Industries Association (EIA) published the “Interface
between Data Terminal Equipment and Data Communication Equipment Employing Serial Binary Data Interchange” otherwise known as the “RS-232 Standard” that defines

1. the electrical signal characteristics
2. the interface mechanical characteristics
3. a functional description of the interchange circuits
4. standard interfaces for selected communication system configurations (Wood 1981, 301)

Originally implemented to connect teletype machines to early modems, the RS-232 Standard also informed the design of the serial port and cable shipped with the original IBM PC, the PS/2 interface for mice and keyboards, Firewire and USB data transfer, Ethernet networking, and, of course, most videogame controllers.

The NES controller is both a serial interface and an example of Denson and Jahn-Sudmann’s concept of “serial interfacing.” For Denson and Jahn-Sudmann (2013, 11), serial interfacing qualitatively describes the “processes of temporal-serial experience that transpire at the interface between humans and digital technologies.” This essay explores at greater length how forms of digital seriality also occur between nonhuman actors. Between the temporary storage of bits “latched” to shift register and the bits registered in the CPU’s memory (two, independent forms of synchronic seriality), there also lies the diachronic, serial interfacing of pulsing currents. These nonhuman serialities provide technical scaffolding for the emergence of interactions between human players and networked and programmable media. Following Henri Bergson’s distinction between temporality as process and temporality as measured, N. Katherine Hayles (2012, 86) asks:
Along what time scales do interactions occur between humans and technical objects, specifically networked and programmable machines? What are the implications of concatenating processual and measured time together in the context of digital technologies? What artistic and literary strategies explore this concatenation, and how does their mediation through networked and programmable machines affect the concatenation?

Perhaps nowhere in the culture around videogames are these questions of both human and nonhuman temporality better explored than in the relatively new (and intertwined) practices of speedrunning and tool-assisted speedrunning.

Related to both Henry Lowood’s (2006, 26) “high-performance play” (i.e., “play as performance, modification of content, and community-based tools and content development”) and James Newman’s (2008, 124) “superplay” (i.e., “the use of the knowledge and techniques uncovered and laid out in Game Guides, the exploitation of the structures, [non-] linearity and limitations of videogames as designs as well as the harnessing of glitches in game code”), speedrunning and tool-assisted speedrunning are community-based practices which add voluntary constraints to the involuntary mechanics driving digital games, convert videogames from games in and of themselves into equipment for playing metagames, and turn play into a game design practice. These games played in, on, around, and through other games are metagames (Boluk and LeMieux Forthcoming, 5). While digital seriality articulates the character of videogames as a technical and mass medium, metagames locate the specific, material histories of twenty-first century play. Extending Richard Garfield’s definition of metagames from “how games interface with life” to “how games serially interface with life,” tool-assisted
speedrunning exemplifies how community practice can both identify and challenge the serial structures governing videogames (Garfield qtd. in Boluk and LeMieux Forthcoming, 21).10 Metagames like tool-assisted speedrunning remake and remediate videogames according to the conscious choices of small communities rather than the rules of serial consumption. In the case of the Super Mario Bros. franchise, the history of speedrunning and tool-assisted speedrunning console games began with a viral video that began circulating in 2003.

2.2. moSMB3.wmv

Not long after .GIFs of dancing babies (and dancing bananas [and dancing hamsters]) colonized Geocities websites in the late nineties and “all your base belonged to us” in 2001, a serial Mario meme spread across the Internet alongside Star Wars Kids and Badger Badger Badgers in 2003. Titled moSMB3.wmv, the 18.4-megabyte Windows Media Video file was traded via torrents, uploaded to university accounts, and, of course, exhibited on eBaum’s World as “Super Mario 3 beat in 11 minutes.” Before Leroy Jenkins and lonelygirl15 were streaming videos on YouTube in 2006, eBaum’s World

---

10 Garfield’s definition of metagaming closely mirrors Mark B. N. Hansen’s articulation of media as “an environment for life” (Hansen, 297). Following Marshall McLuhan’s definition of media as the extensions of man and Bernard Stiegler’s concept of epiphylogenesis, in his essay, “Media Theory,” Hansen argues that “my interrogation ultimately conceptualizes the medium as an environment for life: by giving concrete form to ‘epiphylogenesis’ (the exteriorization of human evolution), concrete media find their most ‘originary’ function not as artifacts but through their participation in human technogenesis (i.e., our co-evolution with technics)” (2006, 297). Similarly, one might argue games do not function as autonomous objects or abstract rules but require participation by players (be they human or nonhuman). If media is an environment for life, then metagames are an environment for games.
was one of the earliest entertainment aggregators that hosted--and on more than one occasion claimed to author--much of the early history of Internet memes. Posted in the “Hilarious Video” section of the website on December 3, the .WMV came with a warning: “[t]his video is a very large file and will take 3 hours to download if you are on a slow dial up internet connection. Broadband users only!” (eBaum’s World 2003) (See Figure 3). Although moSMB3.wmv is a derivative work based on the third entry of a widely disseminated videogame franchise, these aspects of industrial production and distribution are not unique to digital seriality.

11 Since eBaum’s World incorporated at the end of 2002 it has been the subject of multiple lawsuits and copyright claims by corporate entities, other websites, internet communities, and individual users. Perhaps the company’s best known infringement, aside from Viacom’s cease and desist letters targeting an interactive soundboard based on audio clips of Howard Stern, was the misappropriation and watermarking of the “Lindsay Lohan does not change facial expressions” or “Lohan Facial” .GIF originally released on YTMND.com by Derek Lutz (Kushner 2006). “The face that started a war” was finally removed from the site after multiple denial-of-service (DOS) attacks, on-site delivery pranks, and other threats from anonymous Internet users (Lutz 2005).
Figure 3. On eBaum’s World, Morimoto’s “Hilarious Video” was rehosted without attribution and with a broadband warning.

Downloaded bit by bit from a central server, the speed and scale of TCP/IP protocol exemplifies one aspect of digital seriality: the diachronic repetition of modular units, rapidly transmitted in vast sequences of 1s and 0s which characterize serial interfacing. Once magnetically etched into the spinning platter of a hard drive client side, moSMB3.wmv enacts a second, related aspect of digital seriality: the synchronous storage of copied data accessed simultaneously on separate screens in the form of collective serialization. From the speed and scale of information exchange to the private experience of consuming media, Internet memes are perhaps one of the best examples of the diverse spatial and temporal modes in which seriality operates. Without the crowd
sourced “liking,” “upvoting,” or “retweeting” now common to the production and consumption of memes within social media networks such as Facebook, Reddit, and Twitter, file trading produced an audience of individuals, separated by the industrial conditions structuring the transmission of files. Beyond the serial production and networked consumption of viral video, moSMB3.wmv, a speedrun of Nintendo’s Super Mario Bros. 3 performed by a Japanese player known as Morimoto (もりもと), reveals the alienating effects of digital seriality and dramatizes the distinction between human and machine-scales of temporality.

After downloading moSMB3.wmv (for what might have been hours) and getting the video to play (after also downloading the proper codec), Morimoto’s speedrun begins with the faint sound of the start screen of Super Mario Bros. 3 (1998) accompanied by a white overlay with two lines of green, right-justified text: “super mario bross3 [sic] / time attack video” (see Figure 4). As the overlay dissolves into the World 1 map screen, the first stage is quickly selected and Mario glides through Level 1-1 with mechanical precision. The tiny, four-tone sprite scrolls right at a constant rate of 3.5 pixels per frame, effortlessly avoiding obstacles and bouncing off enemy after

---

12 The history of media players is remarkably difficult to trace due to the separation between discussions of video content and video players on online forums. Since the Windows Media Video file format was new in 2003, only certain video players included codecs able to interpret .WMVs such as Windows Media Player, RealPlayer, Media Player Classic, and VLC. Because of the large file size and new format, some users were not able to see Morimoto’s Mario 3: “:tear:: I couldn’t get it to open. I guess too many hits to the page. But just hearing you guys talk about it... wow. That’s both impressive and frightening” (mikomonk, 2004).

13 Like the original Super Mario Bros., movement speed and collision detection in Super Mario Bros. 3 are based on manipulating Mario’s position on the screen. Position is calculated at a “subpixel” scale stored
enemy before reaching his goal in the first three stages of Grass Land. After two minutes of gameplay and two warp whistles, Mario has already entered World 8 and is nearing the end of his quest to rescue the princess. Instantly transported from the pastoral fields and benign obstacles of World 1 to the dark, industrial hell of what is ostensibly Bowser’s home turf, Mario starts to really show off. The final world of Mario 3 begins three “autoscrollers”—levels in which the screen moves at a fixed rate to simulate the procession of wooden tanks, ships, and planes that make up the Koopa King’s army. Since speed is constrained to the slow panning of the stage, rather than simply pressing his nose against the rightmost pixel of the frame, Mario bides his time by bouncing acrobatically from bob-omb to bob-omb to cannonball and back, racking up thousands of points and extra lives. Whereas the streamlined speedrun through the first three levels seemed practiced, Mario’s death-defying antics and carefree hot-dogging at the end of the game are downright superhuman. The first stage of the Dark Land is completed with 79 lives. The entire game is over in eleven minutes, three seconds, and ninety five milliseconds—exactly 39,837 frames—and fades to another white slide with

within a single byte of RAM at memory address 0x074D. Despite the fact that a byte can represent 256 discrete values, in Mario 3, the byte at 0x074D is incremented by 16, thus subdividing subpixel into sixteenths of a pixel (adelikat et al. 2014). This value is updated sixty times a second, according to the speed of both the Nintendo Entertainment System’s PPU and the refresh rate of CRT television screens that comply to the NTSC color encoding standard.

14 In their “Workers’ History of Videogames,” Nick Dyer-Witheford and Greig de Peuter frame Mario as a “working-class hero,” “an overall-clad, cloth-capped industrial artisan who liberates Princess Toadstool by overcoming a series of bosses” (Dyer-Witheford and de Peuter 2009, 3).

Figure 4. Screenshots from moSMB3.wmv reveal the low resolution and high compression of Morimoto’s original movie file, adapted for the speed and scale of file sharing in the early 2000s.

When Morimoto first released the video in late November 2003, direct links from the popular Japanese bulletin board 2channel (http://2ch.net) quickly exceeded the bandwidth limits of his personal website. Scrambling to subsidize server costs with online advertisements on the 28th, Morimoto lamented, “the end of the world is near” (Morimoto 2003). Five days later, on the other side of the world, moSMB3.wmv was

---

15 Bored while on exchange at the University of Central Arkansas in 1999, Hiroyuki Nishimura built a text-based bulletin board with a twist: anonymity. Over the next decade, 2channel became one of the most popular websites in Japan and, in 2006, was followed by Nishimura’s Nico Nico Douga, a video portal (originally a Japanese frontend to YouTube) that allowed users to anonymously overlay comments on top of video (Katayama 2008). 2channel, abbreviated as 2ch, is a subtle reference to radio frequency modulators—a standard video and audio interface between videogame consoles like the Famicom and channel 2 of most Japanese televisions.
rehosted on eBaum’s world and downloaded thousands of times. Although consumed privately, the collective serialization was captured across various web forums and online communities. Initially viewers were dumbfounded by the spectacle of such play:

OH MY GOD... i cant beleive what i just saw.. this guy is awesome and at the same time SCARY.. i cant put this into words only one WOW!!!! (Yessie 2003)


If I didn’t know any better I’d say he knew the position of every gumba, cannonball, and man-eater plant int he whole game!! (5stringdna 2003a)

The jumping was amazing. He finished with 99 lives . . . I considered myself a Mario nerd, but I never knew about the first whistle he gets, the walk-through-the-wall thing he does on the last level, or the fact that you can kill Bowser by jumping on him enough times. (Erroneus 2003)

h4x!!!!!! (Sonic 2003)

As is already evident in the preceding posts, there was a pervasive sense of disbelief that morphed into a kind of antipathy towards a player who performed such virtuosic, yet “alien” play:

:shock: that’s inhuman... that’s just wrong..... (nublu01 2003)

I think some things are better left alone or untold so they don’t shatter peoples’ hopes and dreams. I cried tears of joy when I saw that. (fizzlephox 2003)

Morimoto-sama, I bow before thee... even though to get this good, you must have wasted an incredible amount of your life. ::mourns this Morimoto person:: tongue.gif (Spectrum 2003)

Ok, so if this guy spent an equal amount of time doing something like, kung-fu or whatever, he coudl be the grand-mster mac-daddy, but instead he’s decided to master an 8-bit nintendo game? (5stringdna 2003b)

---

16 The spelling, punctuation, and typography in this passage and all subsequent quotations from online forums have been transcribed exactly as they were found.
*Sigh* "No life!" (Toddathon 2003)

Skepticism and sarcasm gave way to overt racism:

I think the guy that did that is japenese or chinese, notice its not the english version of the game. Either that or it's done on an emulator and he didn't get the english rom. (Vamp 2003)

:eek: :wtf: :eek: :wtf: :eek: That was 100% FLAWLESS. As I watched that, I asked myself... “what human being possesses such robotic precision?” Then I saw the Japanese writing at the end, and it all made sense... (Bluefire 2003)

It’s about an extremely lonely man who had nothing but his Nintendo, a 2 liter, and a desire to be the best at his craft. . . . With a little determination and love (and absolutely no desire to ever leave his home prefecture), this wacky asian man was able to conquer Super Mario 3 in 11 minutes and some odd seconds. . . . Easy peezy Japaneezy. (Snaggletooth 2003)

teh japanese are :alien: (Ilitirit 2003a)

The attitudes surrounding an “alien” invasion onto the site of traditional play are in keeping with the projection of techno-Orientalist anxieties common to science fiction like William Gibson’s Neuromancer (1984) and Ridley Scott’s Blade Runner (1982). In adjacent posts, Morimoto is described as a “cybernetic soldier,” “made in germany and exported to japan,” and “designed on a genetic level to destroy joysticks” as posters channel racist characterizations that “[t]he Japanese are unfeeling aliens . . . cyborgs and replicants” (PhazeORage 2003; Iliterit 2003b; Riot999 2003; Morley and Robins qtd. in Sohn 2008, 7). As Stephen Hong Sohn (2008, 6) points out, “the connection between the Asian American and the alien other still remains a force to draw upon to allegorize racial tension and exclusion.” Analyzing how both “alien” and “Asian” are entangled concepts within the contemporary Western imaginary, Sohn (2008, 6) further emphasizes,
“Alien/Asian does invoke conceptions of its homonymic counterparts, alienation and alien-nation.” In terms of Sartre’s theory of seriality, alienation is the effect that both distances producer from produced (or player from play) and separates individuals from community group formation—literally forming alien nations behind the folds of newspapers (or perhaps in today’s context, the glowing screens of smartphones). MoSMB3.wmv continues this tradition digitally. Both the spectacular play and the hostile spectatorship function together to allegorize the alien and alienating effects of serial interfacing as speculations about Morimoto’s racial identity ultimately stand in for the technological apparatus of tool-assisted speedrunning, a way of playing the serial interface at human conscious timescales.17

2.3. Tool-Assisted Speedrunning

In Alien Phenomenology, or What It’s Like to Be a Thing (2013), Ian Bogost deploys the figure of the alien to different effect. Proposing an anti-correlationist, flat ontology based on the philosophical work of Graham Harman and Levi Bryant, Bogost (2013, 34) argues that “[t]he true alien recedes interminably even as it surrounds us completely. It is not hidden in the darkness of the outer cosmos or in the deep-sea shelf

17 In “Race and/as Technology” Wendy Chun (2012, 38) asks, “[t]o what degree are race and technology intertwined? To what extent can race be considered a technology and mode of mediatization, that is, not only a mechanism, but also a practical or industrial art?” In this particular case, the comments of the posters—a mixture of aversion and envy—conflate the machinic apparatus and the objectified, racialized body of the imagined Japanese player on to which the posters project their fantasies about both the content and production of Morimoto’s video. The “tool-assisted speedrun” becomes a “race-assisted speedrun” as race itself functions as a technology through which forum posters interface with Morimoto’s playthrough of Super Mario Bros. 3.
but in plain sight, everywhere, in everything.” Although this claim privileges chips and circuits as much as coffee cups and ice cream cones, videogames like Atari’s *E.T. the Extra-Terrestrial* (1982) feature heavily in Bogost’s book. For Bogost (2013, 17-8), *E.T.* is “8 kilobytes of 6402 opcodes and operands,” “a flow of RF modulations,” “a molded plastic cartridge,” “a consumer good,” “a system of rules or mechanics,” “an interactive experience,” “a unit of intellectual property,” and “a sign that depicts the circumstances surrounding the videogame crash of 1983.” A similarly irreducible, alien assemblage of technical media and cultural significance, in the early 2000s Internet users confronted the profound disconnect between their own, lived experiences of *Super Mario Bros. 3* as a consumer good and its uncanny technical capacities. After initial reactions of stunned disbelief and racist speculation, forum threads settled into a more analytic approach to the video as almost all posters were learning about tool-assisted speedrunning for the first time:

> . . . there are some things that i don’t think add up in the video, but i’m not sure if i should start listing, since there might be a chance no one really played the game here. Are there any experienced mario 3 players out there that think there's something’s wrong with the video? (KQX 2003)

> I’ve not done any reading on the matter but I am quite convinced (having played a LOT of mario 3 myself) that this was not done on an actual NES console. (Algorithm 2003)

> Does it seem odd to anyone else that this video is encoded in wmv? I didn’t think Microsoft had a big presence over there. Wouldn't it be easier to encode it as a avi? (cletus 2003)

> . . . he did it frame by frame on an emulator and it took two years to complete. Sorta like putting a cartoon together. (Tremmie 2003)
Oh, after examining the [emulator] file closer, I have realized that Famtasia allows you to save state in the middle of a movie and rerecord from this state. Thus, the 11 minute video was actually interrupted and restarted over 40,000 times... more than the number of frames in the movie! Also, apparently the guy took a couple years to do it, according to his website. Thus, while no hacks were used, it was far, far from being ‘real.’ (MEGA¥TE 2003)

Doesn’t seem quite so impressive, now :/ (Axyon 2003)

Although there was initial confusion among the global audience over what exactly they were watching, the small green links at the end of the video linked to two Japanese websites, authored by Morimoto, that explained in detail the process of making the video--the inaugural tool-assisted speedrun.18

The first person to carry out a full investigation of Morimoto’s work outside of Japan was Joel “Bisqwit” Yliluoma, the Finnish software engineer and computer hobbyist who would go on to found TASvideos.org in 2006. TASvideos is currently the largest community dedicated to making and publishing tool-assisted speedruns. As the community at TASvideos.org (2014) states:

[W]e strive to push games to their limits. The emulators we use allow for undoing mistakes, slow-motion gameplay, and even in some cases utilizing robots to do our bidding. Using these tools, we overcome human limitations to

---

18 Although Morimoto’s Super Mario Bros. 3 video was a popular and widespread example of tool-assisted speedrunning, similar techniques had been used to play both Doom (1993) and earlier Super Mario Bros. titles. As the founder of TASVideos.org, Joel “Bisqwit” Yliluoma, notes in a 2005 interview, “[w]hile Morimoto’s movie was the igniter for the western world to start making tool-assisted emulator movies using the feature called ‘re-recording’, Morimoto’s movie wasn’t the first of its kind. According to my knowledge, another Japanese person called Tokushin made also a Super Mario Bros 3 movie in 2001, and yet another Japanese person called Yy made a tool-assisted Super Mario Bros movie in 2000... Although thus the exact origin of tool-assisted speedruns (emulator or not) is not known, it’s clear that it was Morimoto’s movie in November 2003 that made the phenomenon world-wide famous” (Yliluoma 2005). Although the early history of tool-assisted speedruns of PC games in the form of “speed demos” is well documented in the work of Henry Lowood and James Newman, Morimoto’s video represents a major turning point in the history of the practice as applied to console games.
complete games with extremely high precision, entertaining our viewers as our players tear through games at seemingly impossible speeds. The end result of this process is simply a series of key-presses which may be performed on the original hardware.

By the second week of December 2003, Yliluoma had translated a rough version of Morimoto’s documentation and produced his own website, “Bisqwit’s NES bittorrent video downloads” (later known simply as NESvideos), which attempted to dispel rumors and popularize Morimoto’s methods by detailing exactly how moSMB3.wmv was made (Yliluoma 2004).^{19}

According to Morimoto’s documentation, moSMB3.wmv was not an inimitable, real-time performance of Super Mario Bros. 3 on the NES, but a carefully constructed sequence of serial inputs planned in advanced, manipulated during production, and executed according to the affordances of NES emulators for the personal computer. Morimoto’s .WMV was not a recording of real-time, human play but visual evidence of another kind of play altogether: an .FMV, or Famtasia Movie Capture file, detailing a linear sequence of inputs which could be replayed and edited in the NES emulator, Famtasia, named after the Japanese Family Computer or Famicom. In computing, an emulator is a piece of hardware or software that implements the functions and operations of another piece of hardware or software. As Nathan Altice (2011) nicely summarizes

Console emulation mimics a target platform on another, typically more powerful, platform, ideally permitting users to play game software with the closest approximation to the original experience as possible. Accuracy is a key constraint and never perfect. Emulation is not solely a matter of replicating the target console’s CPU, but also any additional co-processors, input/output devices, lower level instruction sets, and so on. The NES had dedicated Picture and Audio Processing Units (PPU and APU) in addition to its 6502-based CPU.

Released by nori and taka2 in 1999, Famtasia attempts to recreate the operations of Nintendo’s first videogame platform on the personal computer.

Ian Bogost and Nick Montfort define “platform” as “an abstraction, a particular standard or specification before any particular implementation of it” and, like the NES or Famicom, Famtasia supports a wide array of software (Bogost and Montfort 2009, 2). Whereas the operations of the original system were constrained by its CPU, PPU, and APU (which were, in turn, constrained mostly by money), emulators like Famtasia support additional features like save states, slow down, instant replay, and cutting/pasting—functions which allow players to access, rearrange, re-record, and write the units of serial communication that structure the interface between controller and game. No longer bound to the temporality of the physical platform, Famtasia allows for multiple approaches to playing the serial interface out of order and out of time.

Although the player can still steer Mario with a controller (or keyboard), the emulator also allows for play at the level of bit-wise manipulation of the numbers.

Morimoto’s .FMV file begins with a 144-byte header that Famtasia identifies based on a simple, four-byte file extension: “46 4D 56 1A.” Each cluster of hexadecimal characters represents a byte of data. Unlike binary notation in which only two digits--
and 1--are used to signify arrangements of bits, hexadecimal notation uses sixteen characters: 0 through F (see Figure 5). Like most file headers, the first four bytes of Morimoto’s movie declares the file extension “FMV” followed by an escape character interpreted as “/x1a.” The remainder of the header contains metadata such as the title (“Morimoto’s SMB3 speedrun/recorded by Bisqwit”), how the movie was constructed (resets or save states), and the number of total rerecords (40268 times). After the header, beginning at address 0x0090 (i.e., byte 144), every subsequent byte in the .FMV stands in for an instance of serial input. A byte can be described numerically as a series of eight binary digits (e.g., 00110001), three decimal digital (e.g. 049), or two hexadecimal digits (e.g., 31) that stand in for controller input (see Figure 6). The data temporarily “latched” to the 4021 shift register signifies the state of all eight buttons on the NES controller and Famtasia’s movie files store a sequence of these states. 60 inputs per second become 60 bytes per second. From 00 to FF, all 256 possible combinations of controller input can be represented. For example, exactly 9.85 seconds into Morimoto’s video, on frame 591, the byte stored at address 0x02E0 (i.e., byte 736) of the .FMV file is 00100000, which represents pressing the “A” button the controller and, when played back, selects Level 1-1 from the Grass Land map screen of Super Mario Bros. 3. After pressing “A” (20 in hex

20 Since Morimoto’s original movie file was created before TASvideos.org was established, it was reproduced and republished by Joel “Bisquit” Yliluoma for archival purposes in the fall of 2006.
notation) for 5 frames to select the level, Morimoto makes his way through Level 1, running (11) and jumping (31) over goombas and green pipes (see Figure 7).

<table>
<thead>
<tr>
<th>Binary (Base 2)</th>
<th>Decimal (Base 10)</th>
<th>Hexadecimal (Base 16)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0001</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0010</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>0011</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>0100</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>0101</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>0110</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>0111</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>1000</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>1001</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>1010</td>
<td>10</td>
<td>A</td>
</tr>
<tr>
<td>1011</td>
<td>11</td>
<td>B</td>
</tr>
<tr>
<td>1100</td>
<td>12</td>
<td>C</td>
</tr>
<tr>
<td>1101</td>
<td>13</td>
<td>D</td>
</tr>
<tr>
<td>1110</td>
<td>14</td>
<td>E</td>
</tr>
<tr>
<td>1111</td>
<td>15</td>
<td>F</td>
</tr>
</tbody>
</table>

Figure 5. This sixteen-digit binary to decimal to hexadecimal conversion chart details half a byte of data.

<table>
<thead>
<tr>
<th>Button</th>
<th>Binary</th>
<th>Decimal</th>
<th>Hexadecimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>NONE</td>
<td>00000000</td>
<td>0</td>
<td>00</td>
</tr>
<tr>
<td>RIGHT</td>
<td>00000001</td>
<td>1</td>
<td>01</td>
</tr>
<tr>
<td>LEFT</td>
<td>00000010</td>
<td>2</td>
<td>02</td>
</tr>
<tr>
<td>UP</td>
<td>00000100</td>
<td>4</td>
<td>04</td>
</tr>
<tr>
<td>DOWN</td>
<td>00010000</td>
<td>8</td>
<td>08</td>
</tr>
<tr>
<td>B</td>
<td>00100000</td>
<td>16</td>
<td>10</td>
</tr>
<tr>
<td>A</td>
<td>01000000</td>
<td>32</td>
<td>20</td>
</tr>
<tr>
<td>SELECT</td>
<td>01000000</td>
<td>64</td>
<td>40</td>
</tr>
<tr>
<td>START</td>
<td>10000000</td>
<td>128</td>
<td>80</td>
</tr>
<tr>
<td>ALL</td>
<td>11111111</td>
<td>256</td>
<td>FF</td>
</tr>
</tbody>
</table>

Figure 6. This conversion chart illustrates how the eight buttons of an NES controller are represented as two-digit hexadecimal numbers in Famtasia’s .FMV file.
Figure 7. A brief comparison of the hexadecimal representation of Morimoto’s .FMV to the visualization of his .WMV reveals a synchronic serial interface—a serial database detailing every button press.

On the hardware platform, the temporality of controller input is synchronized with the speed at which the NES’s Picture Processing Unit (PPU) can update 262 scanlines on a CRT screen. Because the PPU interfaces with CRTs about 60 times a second, the CPU also samples the 4021 about 60 times a second. In the same way a sequence of cinematic stills rolling from reel to reel in a movie projector produce a form of diachronic, industrial seriality, the stream of bit-wise data sent from the 4021 to the Nintendo Entertainment System is an example of the accelerated temporality of digital seriality. However, once the serialized bits sent from controller to console can be represented in the bytes of Famtasia’s “movie” files, the diachronic temporality of serial interfacing is put on pause. Rather than real-time streaming, the serial interface is collected and spatially distributed, articulating previously out of reach microgestures in a synchronic, serial database. Like a film reel that has been linearly laid out for editing,
Famtasia’s .FMV files store every frame of input and can be manipulated by hand in the non-linear interface of a hexadecimal editor. Rather than the serial operations of a time-based movie, it is a manipulable movie file. Tool-assisted speedrunning is not a movie in the sense of a frame-by-frame animation of Mario’s sprite within the Mushroom Kingdom, but simply the record of a series of inputs which, when replayed, drive a new instance of the game.

Crucially, this kind of play depends on a deterministic relationship between input and output. As Cosmo Wright (2014) curtly states, “[a] game is input->output. Repeat.” On videogame platforms like the Nintendo Entertainment System, a series of button presses performed at the same place and at the same time in the same game will always yield the same output. Cycling timers and random variables stored within the system’s RAM are reinitialized upon reset so rebooting the game not only restarts the various software routines and processes, but also starts them from the same predictable state. Timed button presses not only determine Mario’s movement but, by extension, the value of every single variable in the NES’s CPU. As long as the temporal relation between input and output remains consistent, enemies will appear and act predictably and physics will adhere to deterministic patterns. Importantly, slowing down the speed of the CPU and serial input in tandem does not change the mechanics of the game, just their duration. Twitch-based platform games like Super Mario Bros. become turn-based
puzzle games as tool-assisted speedrunners like Morimoto consider the best strategy
frame-by-frame by playing the serial interface.

2.4. Speedrunning the Serial Interface

The NES-4021 shift register, Morimoto’s moSMB3.wmv viral video, and the
larger practice of tool-assisted speedrunning provide examples of playing the times and
scales constitutive of what Denson and Jahn-Sudmann (2013, 13) call “serial interfacing”
and “collective serialization.” Whether in the context of literature, comics, film, or
videogames, discussions of seriality in digital media often overlook the underlying
processes of sampling, sending, and storing sequences of data over a single channel.
Serial communication, however, is the durational substrate that ultimately governs all
forms of digital seriality. From the intra-ludic levels and worlds to the inter-ludic series
and sequels to the para-ludic transmedia of convergence culture, modes of seriality are
contingent on these atomistic forms of serial communication that network
computational media. Although both human and nonhuman players are oblivious to
these underlying serial operations--what Denson and Jahn-Sudmann (2013, 15) call
“blindness to computational temporality”--these forms of serial interfacing still structure
the phenomenal, affective, cognitive experience of play within the collective serialization
of consumer culture. While serial communication is the medium through which
videogame play occurs, metagaming practices like tool-assisted speedrunning do not
simply play videogames but attempt to play the serial interface itself.
Understanding the industrial design, electrical engineering, and computational logic of a given process or platform (via tool-assisted speedrunning) does not necessarily provide conscious access to the alien territories and alienating effects of digital seriality. However, playing the serial interface both dramatizes the difference between the duration of human experience and that of videogame consoles like the Nintendo Entertainment System and offers a possible strategy for intervening in the operations of serial media. Bogost (2013, 10), for his part, acknowledges that “[a]s operators or engineers, we may be able to describe how such objects and assemblages work. But what do they experience?” (emphasis original). Similarly, Hayles (2012, 86) suggests, “there can be no account of how duration is experienced by objects.” On the other hand, in Feed Forward, Hansen (2015, 25) suggests it is possible to gain “digital insight” by “strategically deploying technical intervention to modulate the inaccessible operational present of sensibility.” Hansen (2015, 43) calls for “a supplementary layer of mediation between technical recording and human experience” and claims “where mediation once named the technical inscription of human experience, today mediation must be redirected to the task of composing relations between technical circuits and human experience” (emphasis original). Tool-assisted speedrunning, as it plays the serial interface and disrupts the expectations of videogame play, offers one possible example of this form of mediation—a supplementary interface between technical circuits and human experience.
Moving from serial communication to playing the serial interface, Morimoto’s video also exemplifies how online culture at the turn of the millennium (a time when modem speeds were measured in Kbps, not Mbps) produced specific forms of serial collectivity. In this sense, tool-assisted speedrunning not only transforms platformers into puzzles, but it also converts single-player software designed for the home console into massively multi-player online games as networked communities collaborate to discover new ways to play and compete with each other for the fastest time. Initially misinterpreted as a virtuosic performance, Morimoto’s video inspired players to begin performing “real-time attacks” (RTAs) of console games like Super Mario Bros. (discussed extensively in the next chapter). It is no coincidence that Speed Demos Archive, a clearinghouse for collecting Quake demos created by Nolan “Radix” Pflug began accepting real-time attacks of console games performed live by human players around the time Joel Yliluoma created NESvideos (the precursor of TASvideos). While not the first tool-assisted speedrun, Morimoto’s video signals the origin of these two very different metagaming communities who have subsequently changed the way games are played.21 As one prescient forum poster noted after watching Morimoto’s 21 At Awesome Games Done Quick 2014, a week-long charity event which raised over a million dollars for Médecins Sans Frontières, members of speedrunning community were joined for the first time by tool-assisted speedrunners who demonstrated their practice live and on physical hardware by electrically manipulating the NES-4021 through a custom controller. Automated controllers, colloquially referred to as “NESbots,” are able to play the serial interface in real-time, verifying that input designed for digital emulators like Famtasia can reliably reproduce the output patterns of tool-assisted speedruns when synced with the speed of the Nintendo Entertainment System and translated into electrical currents running through the NES-4021 shift register.
video in 2003, “[i]t seems like this is turning into a new genre of gaming: seeing how fast you can beat the game. And not just running through it fast, literally training to the point that you know where everything in the game is at all times” (CAPiTA 2003). Input 1 then Input 2 then Input 3 in Frame 1. Frame 1 then Frame 2 then Frame 3 in Game 1. Game 1 then Game 2 then Game 3 in Platform 1. Platform 1 then Platform 2 then Platform 3 in a decade of speedrunning the serial interface. From minutes to seconds to milliseconds, both speedrunning and tool-assisted speedrunning asymptotically approach the limits of serial play (see Figure 8). 22

22 Because of a discrepancy in standard timing between the tool-assisted speedrunning community on TASvideos.org and the speedrunning community on SpeedDemosArchive.com, these records appear closer than they actually are. In general, speedrunners begin their timers when they first gain control of the game (i.e., at the beginning of Level 1-1 in Super Mario Bros.) whereas tool-assisted speedrunners begin timing when the system is turned on. If the tool-assisted speedruns (blue) were retimed according to RTA standards (red), they would each be approximately 3.27 seconds faster.
Figure 8. In a little over a decade of speedrunning (red) and tool-assisted speedrunning (blue), players in both communities are approaching the end of games such as Super Mario Bros. In this case, optimized routes and perfectly performed exploits converge on a common limit: 4 minutes and 57 seconds.
3. 4:57: The Phenomenology of a Frame

It’s delicious monotony without the least emotion. . . . You see I haven’t quit being a painter, now I’m sketching on chance.
--Marcel Duchamp; n.d., 1924

Andrew Gardikis does not know. Sitting alone in the bedroom of his childhood home in Quincy, Massachusetts and livestreaming Super Mario Bros. to a modest audience on Twitch.tv, Gardikis does not know because, in the moment, he cannot know. Is it a 4:57? Is it even possible to beat Super Mario Bros. that quickly? If so, how would he even know? The temporal precision of speedrunning outpaces the manual dexterity necessary to accurately hit a split or stop a watch (even as such precision is required to play the game itself). Caught in a cybernetic loop between the glow of a CRT television screen and the electrical pulses of the controller’s 4021 bit shift register, knowledge surfaces in retrospect. After each (increasingly rare) viable attempt to beat the world record Gardikis must decode and retime his DVR recording frame by frame in

---

1 Joel Ylihuoma’s (2006) amateur translation of Morimoto’s reason for producing his famous Super Mario Bros. 3 (1988) tool-assisted speedrun reads “[m]oreover, as for me, I only wanted to see a perfect Mario game completion, even if [it] needs to be made with an emulator in order to accomplish it.”
video editing software like VirtualDub—a process that takes substantially longer than completing Super Mario Bros, itself. Leaping past Bowser’s barrage of flying hammers on October 7, 2013, Gardikis exclaimed “Oh my gosh! I don’t even know! Oh my gosh. Oh my gosh! What the? What!? [Laughs] I don’t even know. [Laughs] Ahhh! It’s gonna be like, it’s gonna be like 4:58.03 or something. I’m gonna die. I’m going to die.”

For seven years between 2007 and 2014, Gardikis held the world record for completing Super Mario Bros. in the shortest amount of time. Unlike tool-assisted speedrunning, discussed in the previous chapter, Gardikis attempts to play videogames quickly in real time using Nintendo’s standard controller interface and the limitations of human perception, performance, and most of all, patience. Although videogames are produced and consumed according to the ideological desire to beat, complete, master, or simply finish the game, in the aftermarket, it seems Mario can never be completed nor wholly consumed. After the initial thrill of exploring the Super Mario Bros.’s thirty-two

---

2 Although a DVR or digital video recording could be timed in any video editing suite, VirtualDub is the popular, open source, software most Speed Demos Archive uses to time their publications. In a publication on Twin Galaxies in 2011, Gardikis (2011) mentions that his records were “measured with virtualdub, recorded with DVD.” Although both Twin Galaxies and Speed Demos Archive have traditionally relied on timing in whole seconds, the competition around games like Super Mario Bros. requires finer measurements. Due to the vagaries of video and timing standards, the German speedrunner and current world record holder, Blubbler, has invented an ingenious timing method in which a single screenshot of Mario’s position in relation to the enemies in the last room of the last level is enough to calculate the exact length of an optimized speed run. Blubbler (2014) notes:

[Random number generation (RNG)] in this game is determined by what frame you enter a screen on. Now you can enter the last room on a number of different frames on an emulator, look at the pattern of the hammer bro, the two fireballs and Bowser’s movement and hammers, compare that with the video and hit the axe on the same position. If you know what framerate the run was started on (on the title screen) and that the runner held B and right for the duration of the last room, you can get the frame-precise timing of the run.
levels is over and after the game has disappeared from store shelves and the pages of strategy guides, is it game over? How do you beat Super Mario Bros.? And why is 4:57 the endgame of both real-time attacks (RTA) and tool-assisted speedrunning (TAS)?

Before Gardikis could even dream of a 4:57, the number had to first be discovered through the emergence and exploitation of game mechanics not immediately sensible to human players (see Figure 9). Mechanical exploits like “wall jumping,” “wrong warping,” “vine wrapping,” “wall clipping,” and “flag skipping” changed the way the game was played only after players added an additional speed constraint. In this sense, speedrunning itself operates as an aftermarket game design philosophy and a form of software studies that uncovers the algorithmic logic of the game not immediately accessible to the player. Beginning with the decade-long history of playing, researching, and reinventing Super Mario Bros. through tool-assisted speed running, this chapter investigates the phenomenology and uncertainty of Andrew Gardikis’ quest to achieve a 4:57 in realtime.
Figure 9. For the past ten years, two communities have continued to play Super Mario Bros., discovering new game mechanics and extending aftermarket play through the application of a voluntary speed constraint.

3.1. TASing Super Mario Bros.

The history of tool-assisted speedrunning Super Mario Bros. ends with a change to frame 16539, but, almost a decade before this number had any significance, dozens of players collaborated to design the tools and discover the techniques that would eventually shave off 18.34 seconds—an eternity within a competition usually measured in frames—from Joel “Biqwit” Yliuoma’s first attempt, a 5:15.65 on December 6, 2003. After watching Morimoto’s viral moSMB3.wmv in late November 2003, Yliuoma began to play Mario with a few additional constraints. Following Morimoto’s lead (and that of
lesser known Japanese tool-assisted speedrunners such as Yy), Yliluoma booted the
ROM from Nintendo’s (then) best-selling game in Famtasia, an early NES emulator that
allowed save states, slow down, input recording, and nonlinear editing. Yliluoma’s goal
was not simply to play the game fast, but to discover how to play Super Mario Bros. the
fastest. As detailed in Chapter 2, a tool-assisted speedrun, or TAS, is a sequence of
controller input designed to play a videogame as quickly. Although the goals and
methods of tool-assisted speedrunning developed over time and hundreds of variations
exist, the defining characteristic of this type of gameplay is a time limit or speed
constraint.

As described by TASVideos.org (2014), a collection of the fastest known tool-
assisted speedruns and a community hub dedicated to “push[ing] games to their limits”:

> The emulators we use allow for undoing mistakes, slow-motion gameplay, and
even in some cases utilizing robots to do our bidding. Using these tools, we
overcome human limitations to complete games with extremely high precision,
entertaining our viewers as our players tear through games at seemingly
impossible speeds. The end result of this process is simply a series of key-presses
which may be performed on the original hardware. . . . Some of these tricks we

---

3 Yy is responsible for the first known run of Super Mario Bros, that also happens to be the oldest known
tool assisted speedrun, dated June 9, 2000. Yy’s video, however, was a “warpless” run of the game. Rather
than attempting to beat the game in the shortest amount of time by taking advantage of the warp zones in 1-
2 and 4-2, Yy added an additional constraint on top of speed: playing through every level. Without
employing any major exploits, Yy’s early efforts resulted in a 19:26.6. As of January 1, 2015, the current TAS
record for Super Mario Bros. “warpless” is 18:32.22 by HappyLee (February 1, 2012) and the RTA record is
19:12 by Andrew Gardikis (November 11, 2013).
4 After the runaway success of the Wii U in 2006, Super Mario Bros, longstanding sales record of 40.24
million units was finally displaced by Nintendo’s 82.54 million sales of Wii Sports (2006).
5 Nonlinear editing in Famtasia was the product of manipulating the hexadecimal data of the resulting .FMV
file.
6 Although Yliluoma stepped away from managing TASVideos.org in late 2006, the community website
originally began as NESVideos and was originally located at http://bisqwit.iki.fi/nesvideos/.
use make the games look broken. But we are not breaking the games, we are just breaking your notion on them.

Whereas ROM hacking (discussed at length in Chapter 4) entails the manipulation of the copyrighted code etched in the integrated circuitry of the mask ROMs sold in Nintendo cartridges, TASing operates within the idiom of play. As TASVideos (2014a) confidently declares, “we are not breaking the games, we are just breaking your notion on them.”

Player input is limited to a serial sequence of button presses but and prepared according to the output and analysis provided by digital tools and techniques. After Famtasia waned in popularity in late 2004, emulators like FCEU allowed players to watch the RAM, frame advance, and use non-linear editing. Unlike modders who work to re-program the game’s ROM, the challenge of a TAS is that it exclusively exploits the specific constraints and affordances of the given software in order to produce a different kind of game by playing in a way that exceeds the capacity of human dexterity.

However, every tool-assisted speedrun has to start somewhere and Yliuoma’s Super Mario Bros. begins with a precise playthrough of the most obvious route.

Super Mario Bros. has thirty-two levels divided between eight worlds. Luckily, Level 1-2 and Level 4-2 contain “warp zones” that instantly teleport Mario three worlds forward, from 1-2 to 4-1 and from 4-2 to 8-1. Taking these warps into consideration, the fastest route is obvious. Overground, underground, warp; overground, underground, warp; then travel through the four longest and most difficult levels in the game to rescue the princess. In the end, only eight levels must be traversed: 1-1, 1-2, 4-1, 4-2, 8-1, 8-2, 8-3,
8-4 (see Figure 10). Ignoring, for the moment, that there might be some kind of wrong warp or memory manipulation that would make the process of traveling from 1-1 to 8-4 trivial, the next question is how best to navigate each level? Despite his familiarity with both videogame logic and emulation, Yliluoma’s first TAS attempts take the form of more or less a clean playthrough of the game that follows the same path that a highly proficient player would take.

As Russian TASer, ROM hacker, and FCEUX admin Andrew “AnS” Strebkov7 (2011) writes in the TAS Editor 1.01 Manual,

> [e]arly TASing, just as regular speedrunning, was not too far away from the ‘normal’ gaming process. TASer simply launched an emulator, switched on buttons logging and played a game, saving and loading often enough to fix the most obvious mistakes, slowing down the gameplay at the most intense moments, thus compensating for the slow reaction of the human organism.

In the same way early television relied on the conventions of radio prior to exploiting its medium-specific potentials, Yliluoma’s early speedruns still mimic human modes of play and interaction. Nonetheless, these first TAS attempts laid the foundation for an alternative history of play to emerge around Super Mario Bros. that, throughout 2004, quickly departed from any resemblance to a standard input. Aside from taking the fastest route and maintaining speed through precision platforming, Yliluoma showcased some of the games’ strange behaviors that became standard practice for speedrunners.

________________________________________

7 Romanization of Андрей Стребков.
Figure 10. Due to two “secret” warp zones, speedrunning Super Mario Bros. only requires the player navigate eight of thirty-two levels.

In this chapter, tool-assisted speedrunners pave the way for real time attacks by studying the strange and subtle behaviors of Super Mario Bros. Over the past decade players have discovered dozens of exploits and invented new ways to play that ended up dictating the history of world record speedruns. This process of forming assumptions based on the empirical experience of playing Super Mario Bros., experimenting with the physics of the Mushroom Kingdom, reverse engineering the rules of the game, and inventing new ways to play operates a kind of software studies in which players black box the material processes and source code of Nintendo’s game by looking at how the input of the NES 4021 influences the output of the 6052 processor’s RAM. Moving from serial input to programmatic output, speedrunners redefine the physics of platform games and institute new moves like “lip jumps,” “fireworks skips,” and the “21-frame rule.”
3.2. Lip Jumps, Fireworks Skips, and the 21-Frame Rule

First performed in Level 2-1 of Yy’s ancient “warpless” speedrun from 2000, lip jumps appear in every “warped” run that follows. Halfway through Level 1-2, the fastest method of avoiding three goombas while also sailing over three deadly piranha plants is for Mario to bound off the few pixels of green pipe on either side of a snapping enemy. Although it appears as though the plumber is colliding with the enemy when executing such a jump, Mario’s hitbox never intersects with that of the plant (see Figure 11). Interactions with enemies in *Super Mario Bros.* do not occur according to graphics of each sprite, but an invisible hitbox. Mario’s hitbox is stored in four bytes beginning at memory address 0x04AC that represent the X and Y coordinates of two points. Every frame the area of this invisible rectangle is compared to the hitboxes of five enemies stored in a sequence of 20 bytes at 0x04B0. If the hitboxes intersect, depending on the plumber’s vertical trajectory, the enemy is either squashed or Mario is damaged. In *Super Mario Bros.*, graphics actually accentuate enemy and environmental danger and obfuscate that some obstacles are more forgiving than they appear to the eye. The result is that the precise player must learn the scale and dimensions of this hitbox, and not the enemy graphic, in order to perform actions that are not necessarily mimetically or naturalistically depicted on screen. The hitbox foregrounds the disconnect between the

---

8 Because damage is calculated by vertical speed, “mustache stomps” and the like are possible if Mario’s head intersects an enemy while traveling downward. There is no check for Mario’s feet when calculating collision with enemies.
representation of the action on the screen and the procedural operations governing play. The TASer and RTAer alike must both learn how to navigate within this procedural space that exceeds pixelated representation.

Figure 11. Created by Blubbler to quell the flood of questions and accusations of cheating, this visualization of lip jumps from Level 1-2, 4-1, 4-2, and 8-1 demonstrates the disparity between graphic display and the collision detection algorithms of Super Mario Bros.

After vaulting over plants and avoiding invisible hitboxes, Mario must make sure to avoid the fireworks which appear to “reward” the player if the flagpole at the end of each level is touched when the in-game timer’s last digit is 1, 3, or 6. The speedrunner must plan to either save enough time (or in some levels like 8-3, lose enough time) to avoid a shower of fireworks.⁹ Beyond the fireworks timer, at the end of each level the overall time of each level is rounded up to the nearest 21 frames. The black menu screen that partitions each level in Super Mario Bros. loads the subsequent screen according to the behavior of a 21-frame “interval timer.” This counter, located at  

⁹ Although these fireworks were implemented in the game as an easter egg or pseudo-random reward for the player who finished on 1, 3, or 6, like many of the bonus rounds and perks of other games that take up time between levels, this fanfare becomes the TAS and speedrunner’s bane as it adds precious frames to the brief cutscene that takes place between each level.
memory address $077F$, decrements continually on every frame of the game. Upon
reaching 0x00, it resets to 0x14 (hexadecimal for 21) and begins the countdown again.

Integral to the behavior of multiple timers (and probably implemented so that certain
processes would not need to be run every frame), this behavior determines when each
level loads. As such, a speedrun of Super Mario Bros. resolves to 21-frame units of time.

If each level is completed in:

1 frame, the total time will be N+21 frames
15 frames, the total time will be N+21 frames
20 frames, the total time will be N+21 frames
21 frames, the total time will be N+21 frames
22 frames, the total time will be N+42 frames
30 frames, the total time will be N+42 frames
42 frames, the total time will be N+42 frames
43 frames, the total time will be N+63 frames (TAS Videos 2014b)

And so on, where N is a constant minimum length of the black screen. As Andrew
Gardikis (2013) explains:

The game is always running a 21 frame long (0.35 seconds) cycle from the
moment you power on the NES. In each stage, you can only gain or lose time in
increments of 21 frames. At the end of each stage, the game determines which
frame rule cycle you are in and rounds the time to the beginning of the 21 frame
cycle. The only stage where this doesn’t apply is 8-4 since the time ends at
Bowser’s axe before the rule can take place.

Because of this coarse metric that reduces the passage of time to 21-frame chunks rather
than the more granular unit of the single frame, Super Mario Bros. not only becomes
more easily solvable, but allows for some slight human errors (or entertaining antics) as
long as each frame window is met. Although this may seem a small technical detail, the
organization of the game into 21-frame clusters has massive implications for both
TASing and RTAing: the result being that the human players are offered some wiggle room when attempting to complete a “perfect” run. While this prevents tiny infelicities from affecting the overall run, at the same time this frame rule shuts down the possibilities for optimization that tool-assisted speedrunning supports. Thus the metagame of playing Super Mario Bros. as fast as possible appears to have a solution and attainable time ceiling: 4:57. Of course, it would take almost exactly seven years for this ceiling to be reached in the tool-assisted speedrunning community (beginning with Yliluoma’s 5:15.65, 5:15.6, and 5:13.05 in the winter of 2003) and even longer in realtime.

3.3. Wall Jumps and Wrong Warps

Shortly following Yliluoma’s first three Super Mario Bros. speedruns between December 6, 2003 and January 14, 2004, a prolific player named Michael Fried revealed that Mario had more mechanics than was reported in manuals and player guides. The videogames original manual only depicts the discrete effects of eight buttons: up, down, left, right, select, start, A, and B (see Figure 12). As described in Chapter 2, Super Mario Bros. only registers eight states of eight switches serially sent from the NES-4021 chip to memory addresses 0x06FC and 0x074A (among other places like the button flags stored at 0x000A and 0x000B). Recombinations thereof reveal a series of unexpected exploits that propel Mario level to level. Only two weeks after Yliluoma’s attempts, Fried changed the way the game would be played through the discovery and exploitation of two exploits: the wall jump and the wrong warp.
Both the Japanese and North American release of Super Mario Bros. carefully outline the intended mechanics of the game but, as players soon discovered, not all dynamics can be imagined or completely accounted for in advance.

Whereas glitches like static shocks, tilted cartridges, and even the oxidized copper contacts on the cartridge inject random, analog effects into an otherwise hermetic system and disrupt the intended operations of the game, unintended exploits like wall jumping, on the other hand, are simply features of the software which do not neatly correlate to either the producers’ or consumers’ expectations and assumptions about videogames. Mia Consalvo (2009, 114) discusses the concept of exploits in Cheating: Gaining Advantage in Videogames, where she notes “[e]xploits don’t involve a player actively changing code in a game or deceiving other players; instead, they are ‘found’
actions or items that accelerate or improve a player’s skills, actions, or abilities in some way that the designer did not originally intend, yet in a manner that does not actively change code or involve deceiving others.” In the same vein, Speed Runs Live (2014) allows all glitches because “[t]he game merely executes the code in the way it was programmed to do. The game is the law. If you start trying to get at ‘developer intentions,’ then you start a game of guesswork trying to figure out what exactly was intended or not.” In speedrunning, there are no glitches, there is no cheating, and everything is fair game within the mechanics of the game. The manifesto on SpeedRunsLive emphasizes the disconnect between the phenomenology of the player and that of the operations of the game. From the perspective of software, there is no such thing as a glitch. And speedrunning practices are indifferent to the way in which the market initially framed the possibilities for play. In the aftermarket, the metagame of speedrunning decouples the rules of the game from the mechanics of the videogame and transforms software back into toys.

Although wall jumping was first officially introduced in Super Mario 64 (1996), becoming one of the most iconic features of 3D Mario movement, it may come as a surprise to learn that Mario can also wall jump in Super Mario Bros. Neither an intended mechanic nor an outright glitch, wall jumping emerges at the interstice of two correctly functioning systems: collision detection and wall ejection. Throughout the game Mario’s animation state is stored at RAM address 0x00001D. The byte oscillates between four
values depending on the position of Mario’s sprite in relation to the various obstacles that compose each level’s design:

- 0x00 - Standing on solid ground
- 0x01 - Airborn by jumping
- 0x02 - Airborn by walking off a ledge
- 0x03 - Sliding down the flagpole (Data Crystal 2014)

Before Mario can jump the RAM address 0x00001D must be reset to the hexadecimal value 0x00 via the sprite’s collision with the ground.

However, due to the way that collision is detected and calculated in many NES-era games, players can exploit the time it takes to perform collision check. For example, in *Super Mario Bros.*, Mario does not stop because he has run into an obstacle. Technically, Mario stops because the object is “ejecting” him, frame by frame, when his hitbox collides with the wall. Mario’s position is being modulated by a single pixel. So, if Mario’s feet intersect a wall at any horizontal seam, for a single frame he is registered as standing, allowing an additional jump (see Figure 13). Literally occurring at one pixel for one frame, wall jumping is an emergent property of two distinct processes interacting with each other and it is a paradigmatic example of twitch-based precision necessary speedrunning. Although extremely difficult, wall jumping is not impossible for human players to accomplish in real time due to carefully prepared, metric motion (e.g., the predictable parabola of a full speed, full height jumps) and anticipatory timing for the frame-perfect button press. There is a difference between a random reflex actions and the anticipation of a gesture through audio and video cues. On January 28, 2004,
Fried implemented the first wall jump in the history of tool-assisted speedrunning and followed it with a “wrong warp.”

![Image](https://via.placeholder.com/150)

**Figure 13.** Through the confluence of *Super Mario Bros.*’ collision detection and the wall’s ejection algorithms, a pixel- and frame-perfect window opens in which Mario may wall jump. Although the wall jump is only a latent mechanic in *Super Mario Bros.*, it became a paradigmatic component of 3D platform games thanks to *Super Mario 64*.

In *Super Mario Bros.*, there is a single RAM address dedicated to storing the next location Mario will visit given he enters a pipe or climbs a vine. This identification number is loaded according to the in-game camera’s horizontal position in the level. However, due to the fact that players can exploit this system to decouple the camera from Mario’s position, if there are two exits in close proximity to one another it is relatively easy to exchange their addresses and produce a wrong warp. For example, positioning Mario ahead of or to the right of the camera by executing a series of carefully interrupted jumps makes the pointer outdated by a few pixels, allowing Mario to warp via an outdated pipe. Positioning Mario to the left is simple considering the camera
never scrolls left so after walking forward to load a new pointer, Mario can double back and use an updated pipe. Both of these gestures are useful when speedrunning Super Mario Bros.

After executing the wall jump in his early 2004 video, in the next room of Bowser’s castle Fried travels just far enough in the room to change RAM address $0000750$ and $0000751$, the pointer responsible for storing each successive area, from the normal values 0x65 and 0x01 to updated values, 0x02 and 0x00. After a sharp about face, Mario dives into the pipe behind him, rather than continue on in the castle. Because of the wrong warp, Mario is warped underwater in the aquatic penultimate screen rather than looping back to the beginning of Bowser’s palace (an algorithm discussed further in Chapter 4) (see Figure 14).

As a result, the time for Super Mario Bros. dropped 5.1 seconds from Yliuoma’s initial time of 5:13.05 to 05:07.95--18477 frames. Fried’s submissions in late January 2004 not only implemented the wall jump and wrong warp for the first time and anchored both techniques as a permanent part of the Mario metagame, but also made it possible to imagine future exploits based on similar principles. As Strebkov (2011) writes “[i]llusions of the game realm now confine his mind much less than before.” Rather than exploiting collision detection systems or the limited memory capacity of Super Mario Bros., alongside Yliuoma and Fried, new runners like Ryosuke, Mana, Phil Cote,
Genisto, and Pom researched and applied techniques like “vine wrapping,” “wall clipping,” and “left + right” to further speed up the game throughout 2004.

Figure 14. By recalibrating or misaligning Mario’s relationship to the camera, the alternate pipe glitch or wrong warp manipulates the pointer value specifying which area loads next.
3.4. Vine Wraps, Wall Clips, and Left + Right

A few months later, on April 20, 2004, Ryosuke published a speedrun verifying that it is 11 frames faster for Mario to hit the top of the flagpole at the end of 1-1, 4-1, 8-1, 8-2, and 8-3 rather than the bottom. Regardless of whether Mario latches on to the top, middle, or bottom, it takes the small white flag the same amount of time to slide to the bottom of the screen. However, if Mario leaps to the top, scoring 5000 points, he accelerates faster after leaving the pole. Obviously spurred by the new world record, Michael Fried returned and submitted two runs over the next day that not only optimized his previous attempts but also implemented a new exploit: the vine wrap.

Alongside level 8-4, level 4-2 is the most technical part of speedrunning Super Mario Bros. because of the warp zone that can only be accessed by climbing a vine. In previous attempts to complete level 4-2 as quickly as possible, tool-assisted speedrunners simply follow the most obvious route: leap off the last elevator to hit the block with the vine then, after touchdown, jump to reveal a hidden coin block before using it to access the warp zone. In the same way the wall jump exploits two independent systems (i.e., collision detection and wall ejection), the vine wrap emerges by disrupting one process with another, higher priority one. In this case, Mario cannot normally go beyond the leftmost edge of the camera. However, by positioning the center of the vine offscreen before triggering Mario’s climbing animation, the horizontal position rolls over from 0x00 to 0xFF, instantly wrapping Mario to the other side of the...
screen (see Figure 15).\textsuperscript{10} Offsetting Mario’s horizontal position in relation to the in-game camera makes it so the pointer storing the value for the next location does not update as Mario enters a nearby pipe. By vine wrapping then wrong warping, Fried not only skips the climbing animation, but the small cutscene that begins the bonus area.

\textbf{Figure 15. Because of a process by which Mario’s body snaps to the vine, placing it slightly offscreen makes Mario’s horizontal position roll over. There are no negative coordinates so anything below 0x00 loops back to 0xFF on the other side of the screen.}

Fried’s two April records indulge in secondary, purely aesthetic constraints like killing enemies, collecting coins, increasing the highscore, and making the shortest jumps possible to avoid obstacles and enemies. Because of the 21-frame rule, Fried is able to work within time constraints while adding stylistic flourishes to make a more entertaining movie file. Although the speed constraint is still of primary value, as the history of tool-assisted speedrunning progresses, spectacle becomes a valued metric for justifying the publication of speedruns--particularly for those that are either conducted in unknown videogames or make use of arbitrary constraints (i.e., button limitations,\textsuperscript{10})

\textsuperscript{10} The same wrapping process is possible in many NES games and can even be achieved on the first screen of \textit{The Legend of Zelda} (1986). By positing Link one pixel away from the collision trigger for the screen transition, then manipulating Link’s position by inputting up or down for a single frame, the character sprite will “snap” to an 8 x 8 pixel subgrid, essentially “jumping” over the screen transition trigger and enabling Link to walk completely off the side of the screen only to appear on the opposite side due to the position variable rolling over from 256 to 0 or vice versa.
alternate endings, etc.). The alienating aesthetic of this kind of machinic play signifies the moment in which the techniques used to speedrun *Super Mario Bros.* began to surpass the possibilities of human performance.

Following Fried’s vine wrap, on November 23, another speedrunner nicknamed Mana saved 224 frames and lowered the record from 5:03.73 to 5:01.27. Just as Fried’s vine wrap extends the wrong warp from 8-4 to 4-2 (and exploits the conditions of nonhuman play), on November 11 Mana discovered how to extend the wall jump into a full wall clip. Now Mario can walk through walls. In World 1-2, the use of clipping makes it possible to reach the Warp Zone faster by avoiding the long drop from the top of the screen (although immediately entering a pipe, before scrolling the screen transfers Mario to “Minus World.”). Then, in World 4-2, walking through the wall eliminates the need for the vine glitch, since Mario’s position on screen is displaced to the right enough to outdate the area location pointer such that the wrong warp can be performed immediately.

Between Mana’s important contribution on November 23 and his friend R. “Pom.” Yoshizawa’s careful optimizations on December 16, Phil Côté and Genisto again changed the way *Super Mario Bros.* was played. In their November 26 publication, the duo saved time on every level by discovering a faster way for Mario to accelerate from his standstill loading position at the beginning of each level or bonus zone. By pressing left and right simultaneously for the first frame of each level (a feat that, in real time, not
only requires preternatural timing and remarkable dexterity but also would require a modded controller), Mario accelerates faster than simply running forward. Because Nintendo’s patented D-pad is a solid, hard piece of plastic suspended above four pressure sensitive pads by a single fulcrum, pressing two opposite directions simultaneously (either up and down or left and right) is not possible without applying physical force or somehow permanently partitioning the D-pad down each axis. Because pressing left and right is not a typical use case, programmers designing games for Nintendo’s consoles either overlooked the potential response by their software (e.g., *Zelda 2: The Adventure of Link* [1987]) or used it for debugging purposes (e.g., *Super Mario World 2: Yoshi’s Island* [1995]). Côté and Ginesto’s record only lasted 21 days until Yoshizawa optimized it by applying the duo’s own “L+R” method to turn Mario around before each jump in the game, increasing not only the plumbers acceleration at the start of each level but in-air acceleration generally. Adding the extra keypress by hand in a hex editor, Yoshizawa (2004) writes, that the 19 frames were saved by “mainly applying the acceleration power to back in the air. . . . 80% of this movie is made by [editing] a file.” After that, the record would go untouched for a whole seven months, only updated by Yoshizawa once again that next summer. Importantly, pressing left and right is responsible for the record dipping below the five minute mark for the first time.
3.5. *Flag Skips and Optimization*

Two years passed before klmz combined Mana’s wall clipping with Côté and Ginesto’s l+r technique to implement “flag skipping.” Flag skipping is a method by which Mario approaches the brick at the bottom of the flagpole at a particular angle in order to trigger the cutscene. This technique greatly reduces the time it would take to slide down the flag and, in the case of using a turtle shell or bullet bill to trigger the cutscene, even eliminates the time it takes for Mario to exit the scene. Klmz worked for three years reducing the record from 4:59.6 (17976 frames) on December 14, 2007 to 4:58.18 (17891 frames) on March 19, 2008 and then finally to 4:57.33 (17869 frames) on August 7, 2009.

At this point, it appeared as though *Super Mario Bros.* had been solved. Given the highly-contested records in 2004 and the time spent optimizing the game over the next five years, it did not seem as though the record could be beaten. On TASVideos.org, however, there is a tradition of claiming a “perfect run,” when in fact there is none. Since the beginning of *Super Mario Bros.*, speedrunning, every new record seems perfect—an eschatological dream for the end time. The design of *Super Mario Bros.*, perhaps more than most other videogames, amplifies this desire for perfection. The existence of the 21-frame rule enables human operators to approximate the frame-perfect mechanical precision of computer-assisted play more closely than other games. Thus, the possibility of perfection—and an endgame run—appears to be almost within reach.
After Mana and Pom’s improvements in 2004, the community was shocked that the previous record had been beaten:

Wow. I didn’t think anyone would beat SMB any faster. (feitclub 2004)

It’s so good to see that even the videos that seem 100% flawless can still be improved. (SailorBacon, 2004)

I did not think this game was improvable at all. Great! (Mazzic 2004)

[This]ny further improvement will probably feature some new tricks, which will also make it worth publishing even if the actual difference in time is not really noticeable. (schneelocke 2004)

In 2007, after klmz finally improved Pom’s optimization with his flag skips the forums reacted in a similar way:

Yet another “perfect” movie surpassed! . . . Congratulations on beating a (supposedly) perfect TAS! (Chef Stef 2007)

Entertaining, not to mention that it beats the supposedly [sic] perfect run. (mingle 2007)

I was sure I would never ever see a Super Mario Bros TAS again, because the last was perfect too. ;-) (MattyXB 2007)

1.4 seconds shaved off of a run that was thought to be perfect is amazing. (Gardikis 2007)

By the time klmz improved it again, even Andrew Gardikis (2009) admitted to his serial doubt:

This is one epic improvement. Not sure if we’ll see this improved again. Though, I’ve now said this 3 times, at this point it really does seem perfect. I can’t believe that we’re 1 frame short of improving level 1-1 by 21 frames. And it’s like impossible to improve it. I really doubt anyone will ever save that frame and improve this TAS either. Congrats klmz on a “perfect” TAS.

But other community members were more skeptical: “No TAS is ever perfect. We’ve said that about Rockman 1 HOW many times? And it got improved pretty much each
and every time” (Sir VG 2009). And the Super Mario Bros. record would be improved again--on frame 16539.

3.6. Frame 16539

Klmz made a mistake. Not an immediately obvious mistake nor a particularly large one, but, considering the stakes of the game klmz was playing, a single frame makes all the difference. Frame 16539 went unrecognized for almost two years. Perhaps this was because klmz had held the record for speedrunning Super Mario Bros. over four years since the end of 2007. For most tool-assisted speedrunners, the rush to solve the iconic game after Morimoto’s viral video popularized the practice in 2003 was long over and faster times now required new techniques. Five, then six, then seven years later some serious research was now required to figure out how to improve the game. In practice, tool-assisted speedrunning is not like preparing for a race. Instead, it resolves into a logic puzzle. Given a set of mechanics (e.g., Super Mario Bros.), a set of assumptions or strategies (e.g., the history of exploits), a voluntary constraint (e.g., speed), and a specific goal (e.g., at least one frame faster 17869 frames), there is only so much that can be accomplished without changing one of the gamic ingredients. A specific exploit being outlawed (e.g., Twin Galaxies’s real time attack policies that ban exploits) would produce a radically different metagame. But, in the case of Super

---

11 Twin Galaxies (2014) defines Super Mario Bros. “glitches” as the following:
Mario Bros., since none of the rules changed and new research did not improve the time, the only possibility for change was the discovery of an error in the performance. A 16-year-old student from China named HappyLee was the first to notice klmz’s mistake.

HappyLee discovered an un-optimized jump in the third room of Bowser’s Castle. The mistake had to do with acceleration and deceleration when entering the pipe to wrong warp in 8-4. First discovered by Michael Fried in 2004, it is surprising that the approach to the pipe was not completely mapped out and optimized by the time klmz published the last video in 2009. Nevertheless, halting Mario’s jump one frame earlier allows the plumber just enough height to slip through the corner of the green pipe and enter the aquatic room one frame earlier (see Figure 16). On January 1, 2011, HappyLee published the last tool-assisted speedrun on record to improve the time of Super Mario Bros. Since then, HappyLee has refined his TASing skills, currently holding the record for every competitive category of Super Mario Bros. Given his continued research and experience with the game, it appears unlikely that there will be any further improvement. However, given the history of “perfect” runs on TASVideos.org, it would

[Wall-Jumping is defined as jumping against a wall [or pipe] in such a manner that a pixel or two of Mario is stuck in the wall [or Pipe] and is able to use that to jump again, far higher then can be done normally. Further definition; Mario cannot be made to jump off of a surface that is 90 degrees perpindicular to the ground. Please note this can be further used to also go through solid objects not normally passable as is disallowed.] [The Alternative Pipe usage refers to, for example, going through a lone brick while crouched to go through the wall and access World -1, whereas the pipes would normally be Warp Pipes to Worlds 2-4 Respectively. The doubling back on World 8-4 to get to the underwater area is another example] (emphasis original)
seem hubristic to suggest any videogame has been perfectly optimized. Even at the scale of the frame, the operations of technical media can never precisely correlate to human perception.

![Figure 16](image.png)

Figure 16. A comparison between frames 16532 to 16549 in klmz and HappyLee’s tool-assisted speedruns reveals a temporal discrepancy starting at frame 16539. This single frame of input was all it took for HappyLee to take the record on January 1, 2011.

HappyLee’s endgame of *Super Mario Bros.* was only discovered through the emergence and exploitation of game mechanics not immediately sensible to human players. Tool-assisted speedrunning explored the mechanics, discovered exploits, and,
in the process, transformed the way Super Mario Bros. was played in the aftermarket.

Although HappyLee’s video has not been beaten in years, the TAS community continues to explore Mario through additional constraints such as “walkathons” in which pressing the run-button is disallowed, attempts to complete the game with the fewest button inputs, and low score runs. Whereas the TAS community keep inventing categories to keep playing the game, the speedrunners strive for a time of 4:57. The game ends when the metagame terminates. When there are no glitches left to be discovered, there is no reason to revisit the game. New techniques inspire conversation and the conversation inspires continued play.

Is it possible to beat HappyLee? Although in practice, building artificially-intelligent bots or even brute forcing every combination of the videogame is extremely difficult, there have been speculative discussions of this possibility on TASVideos.org. In 2009, after it appeared that klmz had completed the game, one such discussion took place. Derakon (2009) explains:

Say there’s 8500 frames in the fastest possible movie, and the game has six inputs that matter. So to check every possible combination of inputs, you’d need to do $6^{8500}$ checks. The actual number is smaller (we can assume, for example, that any inputs that result in Mario dying are unproductive, and that input during points where you have no control over Mario don’t count), but even $6^{20}$—every possible combination of the first twenty frames of input—is $3656158440062976$. If you could simulate 100000 frames per second, you’d need 1160 years to check all of those.

Given six inputs per frame, actually checking all of Mario’s possible positions at the end of 17867 frames (one frame shorter than HappyLee’s record) produces a 13904 digit
figure (i.e., $6^{17867}$) representing the total recombinant potential of the game. Even with all the computers and all the time in the world, one comes nowhere close to calculating every outcome. While such enormous figures may exist as numeric expressions, they exceed physical enumeration. “[I]f you don’t mind waiting for the heat death of the universe,” Derakon reminds us, “we can do that.”

Despite the fact that $6^{17867}$ is a discrete number, that each of its 13904 digits are known quantities, and that mathematical operations may be carried out both with and within it, enumeration of such a figure not only outpaces human consciousness but all of known time and space. We cannot count to a duocentillion. We cannot beat Super Mario Bros. As with Carl Sagan’s (2002) demonstration of a “googolplex” in Cosmos (1980), there are not enough atoms in the known universe to account for such numbers. Unlike $10^{10^{100}}$, $6^{17867}$, and even $0$—mathematical notation that substitute symbol for count—brute force methods attempt to render an enormous sum in bit shifts, processor cycles, and liquid crystal refractions (what Matthew Kirschenbaum [2008, 9] might call a “forensic” territory rather than a “formal” map). And like Jorge Luis Borges’ cavalier cartographers from “On Exactitude in Science” or Jonathan Swift’s Lilliputian lecturers in Gulliver’s Travels, anyone attempting to exhaust play within Super Mario Bros. will end up exhausting the limits of human experience. But, in a way, this is what real time attacks become: not a test of skill, but a test of endurance. While enumeration was never the goal of tool-assisted speedrunning, real time attacks require dedicated practice,
disciplined patience, and an abundance of free time in order to enumerate the possibilities of play approaching 4:57.

### 3.7. Real Time Attacks

Still sitting in his room in Quincy, Massachusetts, it is easy to see the effects of time attacks on Andrew Gardikis. When Gardikis first started speedrunning in 2004 and 2005, the community was split between two main organizations: Twin Galaxies, the primary officiator of arcade high scores, and Speed Demos Archive, a newer community that was just beginning to accept speedruns of videogames apart from *Doom* and *Quake*. Much like Joel Yliluoma’s first tool-assisted speedruns in 2003,\(^\text{12}\) the first real time attacks of *Super Mario Bros.* were performed without exploits in 2004 and 2005. According to Twin Galaxies (2014), the rules were:

> Timer starts the instant you hit the Start button and stops the INSTANT you touch the golden axe in the final castle. *Time will be recorded in WHOLE seconds ONLY.* Use of the Wall-Jump tactic is NOT allowed! You MUST access the Underwater area of 8-4 by using the pipe across the lava pit. Doubling back and using the pipe before the pit to do so will equal Disqualification. *Use of ANY Pipe to go somewhere different then the programmers intended is BANNED!* You MUST complete the game or your record/attempt WILL be disqualified! (emphasis original)

The divide between Twin Galaxies and Speed Demos Archive is not simply one of arbitrary preference, but it exemplifies two radically different ideologies towards the software. On the one hand, Twin Galaxies reinforces the model of *Super Mario Bros.* as a

---

\(^\text{12}\) Yliluoma’s “exploit free” runs occurred before Michael Fried introduced the wall jump and wrong warp in early 2004.
gamespace that is fully enclosed by both capital and intellectual property. Twin Galaxies tacitly naturalizes the belief that Nintendo issues the final commands on what constitutes legitimate play. Whereas Speed Demos Archive follows in the footsteps of Roland Barthes’ “death of an author” in order to enact the “death of the developer,” Twin Galaxies modulated their play according to what Michel Foucault might call a “developer function”: playing according to the intention of, if not the actual, historical developers, then an imagined authority figure legitimating only certain kinds of practices (usually in correspondence with the logic of the dominant power, in this case, the market). Even in 2015, this “developer function” continues to enclose the radical potential of play within the limits of capital. And even in the aftermarket practices of speedrunning, its effects continue to ripple through (and produce conflict among) different communities of play.

Gardikis, however, did not obey the rules of any real or discursive developer for long. Trailing behind world record holders such as Scott Kessler and Trevor Seguin as they lowered their times in Super Mario Bros. from 5:11 on February 26, 2004 to 5:05 on September 14, 2006, Gardikis eventually tied their already tied world record (measured in whole seconds) in the spring of 2007. 5:05 was considered another perfect run. But as discussed before, in speedrunning perfection is rarely long-standing. Gardikis’ innovation, much like Michael Fried’s tool-assisted speedruns in early 2004, was to supplement the most efficient route and precision play with two exploits: wall jumps
and wrong warps. By stepping away from Twin Galaxies (at least momentarily), Gardikis was able to implemented the two exploits and achieve a new world record of 5:00.60 on April 14, 2007. After achieving the record, Gardikis continued to whittle down his time from 5:00 to 4:59 to 4:58 at the beginning of 2013. That year Gardikis improved the world record by milliseconds almost every other month: 4:58.78, 4:58.56, 4:58.41, 4:58.34, 4:58.15, 4:58.13, 4:58.09. The closer he got, the further 4:57 seemed--Gardikis was living Zeno’s paradox.

After a year of subsecond improvements, Gardikis could no longer ascertain in the moment whether he had or had not achieved the world record. In a practically Duchampian gesture, speedrunners turn games of virtuosic skill into games of serial luck by partitioning play into more and more granular units until the game takes place beyond the bandwidth of human perception. At odds with the uncertainty of operating within the space that bypass conscious perception (and burnt out on grinding the same game day after day), Gardikis (2014) logged onto Speed Demos Archive to write:

I’m retired from speedrunning as of today. ‘Retired’ until I sort my life out. I’ve said this before, but this time I won’t be playing video games until I figure some things out. I’m in a much worse position than I was previously... I love speedrunning, so I will definitely be back. I don’t know how long I will be gone.

---

13 Gardikis still cares about and competes on Twin Galaxies, recently achieving the world record highscore in Super Mario Bros with 1,434,100 points on January 8, 2015.
14 Gardikis is alluding to the fact that he has a reputation for announcing his permanent retirement in a moment of frustration only to return to his beloved Super Mario Bros shortly thereafter.
One week later, Andrew Gardikis lost the world record to a German speedrunner named Blubbler who, much like Gardikis added wall jumps and wrong warps to beat Scott Kessler’s times, had implemented a third exploit: the flag glitch using a bullet bill in 8-3.

It is ironic that in 2008, only hours before klmz published the discovery of the bullet bill glitch for the first time, Gardikis (2008) made the same discovery:

Man, yesterday I almost decided to try my first TAS, and considering it is smb1, I thought it would be a good idea to do. TASing a simple game would be a good way to start and I know all the glitches used. I can't beleive [sic] I found this and realized it would save time only a few hours prior to it being published. I wish I had stayed up and played for hours attempting to pull off my first ever TAS.

Although it would be unfair, after holding the record for seven years, to say that Gardikis is always the runner-up and not the winner of the race, Blubbler’s 4:57 reproduces the finitude of HappyLee’s final frame permanently leaving both Gardikis and klmz in the runner-up position indefinitely. Whether 4 minutes 57 seconds or 17868 frames, both achievements appear to “beat” their respective game--foreclosing the platform to any kind of future play. Whereas HappyLee’s run operates at the fidelity of a frame, many real time attacks are still judged according to whole seconds. But is 4:57 the endgame of Super Mario Bros, or just the end of human interest in this particular metagame?

Like the progression of world records in physical sports, the asymptotic decrease of Super Mario Bros. times are not based on the talent or tenacity of individual players but index the technical innovations, historical understanding, and communal support of
those players dedicated to performing real time attacks over the past decade. For his part, Gardikis’ innovation was to incorporate techniques that, in 2005, were not based on twitch reflexes or innate skill, but because he decided to employ a suite of exploits previously considered glitches, cheats, or hacks—unfair advantages operating outside the game. Gardikis’ innovation was not a technical exploit, but a social one: ignoring the taboo dictating the state of the metagame in the mid 2000s. Poetically, Gardikis lost his world record to Blubbler because he attempted to perfect a specific rout through the game instead of applying the latest (and riskiest) techniques. If, at any given time, no further innovations are uncovered, then players are left to optimize a single route. Usually, in these cases, the metagame grows stale, community interest wanes, and only the most obsessive (record holders) continue to compete, not in terms of seconds, but frames.

Confronted with the formal and mathematical mechanisms organizing these games of virtual track and field, human play must tarry with the number, the statistic, the score, and the frame that Steven Connor calls ratio in the Philosophy of Sport (2011). Connor (2011, 151) writes that competitions like speedrunning revolve around numbers, or scoring, and “scoring involves a conflict and convergence between two entirely incommensurable orders, the qualitative syntax of bodily motions and actions (kinesis), and the qualitative calculus of number (ratio).” The urge to compare times, to render different moments of play in terms of a general and exchangeable abstract order, is the
motor driving what Roger Caillois (2001) calls agon, or competition. Agon lies at the
heart of the most ancient games and, noncoincidentally, the heart of real time attacks.
With the drama and agony of competition, it is easy to forget that the actions of
individual players and the occasions that generate new records are always abstracted, or
grammatized, into discrete numbers like Happy Lee’s 17868 and Blubbler’s 4:57. In the
process, play becomes precisely the act of negotiating the discontinuity between
phenomenal experience of speedrunning and the mechanics of Super Mario Bros.
Whereas play (like speedrunning) is often theorized as a voluntary and unnecessary
activity that can never be wholly predicted, measured, or even known, games (like
Super Mario Bros.) offer the chance for uncertainty, the units to measure play, and the
conditions for unknowing in the form of rules.

As Andrew Gardikis’ quest for a 4:57 illustrates, the friction between rules and
play is expressed as probabilistic or chance-based mechanics. As Connor (2011, 171)
writes, “[j]ust as probability can neither be distinguished from nor wholly identified
with number, so number can neither be extricated from nor entirely exhaust play.
Probability is the play of number that number itself makes possible.” Probabilistic games
are, in part, structured around this form of playful unknowing and speedrunning, at the
highest levels of play, becomes a game of chance. Through real time attacks, Super
Mario Bros. no longer operates in terms of eye-hand coordination, but muscle memory,
not skill, but chance. Super Mario Bros. becomes a platform for a kind of gambling game
where the outcome can never be known. Recalling Gardikis’ doubt “I don't even know! Oh my gosh. Oh my gosh! . . . I don’t even know.”
4. This is Not Mario: ROM Hacking Super Mario Clouds

“... erase everything but the clouds.”
--Cory Arcangel, n. d., 2005

Roll tape. Single-channel, color video. Shaky cam, no mic. A Marshall amp. A stack of CDs. EPROMs on a VHS cassette. Two Nintendo Entertainment Systems’s on a record player next to a CRT television. A VCR, a hi-fi system, and speakers on a wire stand. The coffee table is cluttered. There are three remotes. Binders and books, pliers and wire cutters, and a pair of helping hands. Nearby a pile of NES cartridges are scattered on the floor. It is 2004 but it looks like the nineties. Bands of magnetic interference scroll down the screen as Cory Arcangel holds up a Super Mario Bros. cartridge he bought for $1.99. Inside the cartridge the circuit board contains two mask ROMs labeled HVC-SM-0 CHR and HVC-SM-0 PRG. These ROMs contain the graphics data and programming data of Super Mario Bros., arranged as a series of 0s and 1s burnt into an integrated circuit of microscopic metal-oxide simiconductors. Though the tool-assisted speedruns and real time attacks discussed in Chapter 2 and Chapter 3 may be performed with ROM files and emulators, these digital abstractions stand in for the electrical devices Nintendo produced and protected with their various anti-piracy technologies. As documented in The Making of Super Mario Clouds (2004), Arcangel is
aiming to replace the PRG or programming ROM with a copy of his own game (see Figure 17).

Figure 17. The 76 minute, silent video *The Making of Super Mario Clouds* operates as both a DIY, instructional video for would-be ROM hackers and documentation for the 2004 Whitney Biennial in New York.

Hands of the animator. The camera rotates 180 degrees. Cory Arcangel is sitting on a tartan couch. The twenty-six-year-old media artist is wearing a loose, button-up shirt and denim jeans. There is a black electric guitar leaning against the sofa.¹ From the objects in the room to the banding on the screen, every affectation is carefully constructed. Arcangel’s video is both a DIY guide for ROM hackers and documentation for the biennial exhibition at the Whitney Museum of American Art in 2004. Both a critique and celebration of concept of the artist’s studio, Arcangel begins removing the

¹ As articulated on the Whitney’s (2015) online “object label” for *Super Mario Clouds*, Arcangel was “trained in classical music, considers computers and video game consoles his instruments, and insists on mastering them prior to creative exploration; he will often learn a new programming language in order to develop a work.”
PRG ROM. He takes the lazy approach, snipping the legs off the chip to separate it from the board. The mask ROM is destroyed in the process—no more Super Mario Bros. It is hard to grab the little legs but after a few snips the chip comes off. Twenty-eight tiny pins remain soldered to the board. Arcangel leaves to get the soldering iron, pliers, and a copper braid. The video continues. One take, 76 minutes.

In Arcangel’s room there is a Mac and there is a PC. It is hard to see. A white pillow obscures the screen. Then the camera focuses somewhere between the dust on its own lens and artist’s silhouette. Everything is a blur; the screen becomes a glow. The faint blue above the desk is a poster of the game Arcangel is making. In “The Making of Super Mario Clouds,” a text that both shares a name and release date with this film, Arcangel (2005, 106) explains “[i]n this work, I [take] an old Super Mario Brothers Nintendo video game and [erase] everything but the clouds. [This] tutorial [explains] my motives for making the work, and [includes] the source code behind the work as well as technical information on how the work was constructed. . . . (people of course [are] free to use these directions to make their own Super Mario Clouds.)” But, as evidenced by the indecipherable bedroom scene, there is something missing. Super Mario Bros., mask ROMs, helping hands, wire cutters. A soldering iron, a copper braid, an EPROM burner, a power drill, some masking tape, and a black ball point pen.

Arcangel intentionally occludes a necessary step from his seemingly open source tutorial in order to further the frame narrative around erasure as a conceptual gesture in the
history of art. Arcangel’s ethics are at odds with his artwork as it costs five hundred dollars to buy a copy of this video for educational purposes (see Figure 18).

Figure 18. Electronic Arts Intermix sells copies of The Making of Super Mario Clouds for thousands of dollars.

As noted on Data Crystal, ROMhacking.net’s wiki, the practitioners of the largest community of ROM hackers believe that hacking:

Comprises both the analysis and manipulation of data, [and] can appeal to the spirit of exploration . . . problem solving, engineering, and creativity. Thanks to the subject’s breadth and its propagation through the Internet, ROM hacking has become an art form. (GuyInSummers, 2005)
Although there are hundreds of hacks hosted at ROMhacking.net, Cory Arcangel’s *Super Mario Clouds* (2002)\(^2\) is arguably one of the most famous ROM hacks and, after being exhibited regularly as part of the Whitney’s permanent collection since 2004,\(^3\) an important touchstone in the history of digital media art. Though exhibited in different formats, *Super Mario Clouds* typically consists of a hacked cartridge playing on a Nintendo Entertainment System outputting a soundless video feed to two projections of varying sizes and a CRT screen (see Figure 19). Upon interviewing the artist for the *New Yorker* after his solo exhibition at the Whitney, Andrea K. Scott (2011) notes that Arcangel’s “idea [is] as simple as silk-screening soup cans: take the code to the classic 1985 Nintendo cartridge and erase everything but the clouds.” And Scott is more right than she knows. As Michel Foucault (1973, 54) wrote at the end of his short book on René Magritte, *This is Not a Pipe* (1973), “[a] day will come when, by means of similitude relayed indefinitely along the length of a series, the image itself, along with the name it bears, will lose its identity. Campbell, Campbell, Campbell, Campbell.” In the same way “this is not a pipe” and “that is not soup,” this is not *Super Mario Bros.*

\(^2\) Since 2002, the spelling of Arcangel’s most well-known piece is sometimes subverted to *Super Mario Cloudz* and occasionally appended with year designations like 2k3 or 2k9 (or version designations like V2k3 and V2k9). Although the differences between the versions of the artwork are significant (and will be discussed later in the chapter), for the sake of clarity the project will be referred to as simply *Super Mario Clouds*.

\(^3\) *Super Mario Clouds* was exhibited at the Whitney Biennial in 2004 and was also included in *Synthetic*, a group showing of the museum’s permanent collection curated by in Carter E. Foster in 2009 and *Pro Tools*, Arcangel’s solo exhibition in 2011. In 2015, the installation can still be viewed daily at the museum’s new location as a part of *America is Hard to See*. 

96
Figure 19. Super Mario Clouds by Cory Arcangel has been exhibited at the Whitney Museum of American Art since 2004.

Embracing an open source, hacker ethos, Arcangel has made much of his artwork available online and in various documentary materials. Despite such transparency, few have investigated what the code reveals about the project—taking it as documentation guaranteeing authenticity. However, even a cursory comparison of Arcangel’s assembly code and Nintendo’s game reveals that Super Mario Clouds does not use any programming data from Super Mario Bros. Arcangel (2005, 109) boasts that “[w]hen making a ‘Super Mario Clouds’ cartridge, I only modify the program chip, and I leave the graphic chip from the original game intact. Therefore . . . the clouds you see are the actual factory soldered clouds that come on the Mario cartridge. There is no generation loss, and no ‘copying’, because I did not even have to make a copy. Wasss up.” He includes shout outs and disses in the comments of his code “aimed at media artists who think they can step to [his] style” but, as seen in the video, the programming ROM is destroyed--not modified and definitely not erased (Arcangel 2005, 106). There is
narrative dissonance between the concept of the piece and the material processes at work in ROM hacking. So *Super Mario Clouds* is less like a ROM hack and more a homebrew—a new piece of software developed in the aftermarket economy surrounding vintage videogame hardware. The pixel-wise scrolling, nametable layout, and even palette colors do not match Miyamoto, Tezuka, Kondo, and Nakago’s game. Neglecting the material specificity of the Nintendo Entertainment System, Arcangel’s artwork operates instead according to an economy of desire based on a utopian fantasy at the heart of games, media, and art. This chapter will first attempt to identify how *Super Mario Clouds* participates in the tripartite ideology of the magic circle, the black box, and white cube before documenting the production of an original ROM hack which attempts to erase everything but the clouds.

4.1. CCSWG 2014

In the Spring of 2014, Jeremy Douglass and Mark Marino convened the third annual Critical Code Studies Working Group, an online collaboration between computer scientists, media scholars, historians of technology, and digital artists aimed at critically reading computer code. As Marino (2006) writes, Critical Code Studies “applies critical hermeneutics to the interpretation of computer code, program architecture, and documentation within a socio-historical context.” While each week revolved a different topic like “Exploratory Programming,” “Feminist Code,” and “Postcolonial Critical Code Studies,” in the final week members of the working group conducted a critical
code study of Arcangel’s original assembly ripped from “The Making of Super Mario Clouds” (see Figure 20). The working group considered the history of erasure in conceptual art like Robert Rauschenberg’s Erased de Kooning Drawing (which, as evidenced by infrared photography, leaves a trace of the original artwork). There was an analysis of race, gender, and sexuality prompted by Arcangel’s adoption of a homophobic epigraph at the beginning his code. And there were even comparisons of other gaming platforms as thread members created their own versions of Super Mario Clouds by modding other games like Flappy Bird and 2048. As the Critical Code Studies Working Group discovered immediately, the clouds are not what they seem

Figure 20. The Critical Code Studies Working Group is a biannual online workshop dedicated to the close reading of code.
The differences between Super Mario Bros. and Super Mario Clouds were obvious. Even a cursory comparison between Nintendo and Arcangel’s code revealed stark differences. Whereas the decompiled assembly code for the original Super Mario Bros. is over 16,000 lines long, the source code of Super Mario Clouds amounts to 290 lines. Without a closer analysis and line-by-line comparisons, however, it is hard to determine the differences between the two ROM files. A visual comparison of the two videogames, however, reveals three major differences (see Figure 21). First, and perhaps most obviously, the colors of Arcangel’s artwork do not appear in the original game. When comparing the palettes of Super Mario Clouds to those in the original Super Mario Bros., Arcangel’s game appears to have swapped the background with the color of the clouds’ interior details. Second, after inspecting the composition of the pixelated graphics, one also begins to question Arcangel’s (2005, 109) claim that “the clouds you see are the actual factory soldered clouds that come on the Mario cartridge. . . . [with] no generation loss, and no ‘copying’” because the rightmost contour of Super Mario Clouds does not match the clouds in Super Mario Bros. Beyond a misplaced 8 x 8 pixel tile that Arcangel eventually fixed in 2009 version of Super Mario Clouds after the Whitney group show, the larger cloud formation itself--how they’re spaced and how often they repeat--never occurs within the original Super Mario Bros. Finally, without attempting to hack Mario’s walk speed, the speed at which the screen scrolls in Super Mario Clouds cannot be easily replicated in the original game.
Figure 21. Both the color and composition of Cory Arcangel’s clouds (right) do not match Nintendo’s clouds (left).

How are these differences produced and why did Arcangel decide to produce an artwork that differs so greatly from the original game (despite the narrative suggesting otherwise)? Is this a case of technical negligence, intentional design, conceptual subversion, or playful provocation? Is Arcangel making a metagame meant to be played within the source code? What follows is an attempt to “play” Super Mario Clouds through a close reading of the code Arcangel published in “The Making of Mario Clouds” in 2004. This “close playing” of a single piece of software operates both in terms of critical code studies and the community practices of ROM hackers who, over the last two decades, have performed this kind of analysis on thousands of games, not to
mention the careful disassembly and reprogramming of Super Mario Bros. Just as some aspects of tool-assisted speedruners resembled platform studies in Chapter 2 and real time attacks required a kind of ad-hoc software studies in Chapter 3, ROM hacking resembles the academic discourse of critical code studies and this chapter will attempt to follow both researchers and players by first reading Super Mario Clouds line by line then attempting to produce an original ROM hack.

4.2. ; “punks jump up to get beat ; down”

The first few lines of Arcangel’s code look like this:

;***************************
; Super Mario Clouds
; BEIGE 2002/03
; Cory Arcangel
;
; www.post-data.org/beige/
; www.beigerecords.com/cory/
; del.icio.us/cory_arcangel/

In assembly programming, nonexecutable code called “comments” begin with ; and are ignored by the 6502 processor. Illegible to the machine, the natural language following each ; is arbitrary—the programmer is free to jot down notes, explain that which follows, label each segment of code, or even include text-based art. These comments can be placed anywhere horizontally but are usually formatted to correspond with the length the longest line of code in order to act as a visual frame. In the case of Super Mario Clouds, Arcangel’s code begins, as many programs do, with metadata. This data includes the title, date, author, and URL of the work alongside two epigraphs:
"you mess with the best, you die like the rest" - Anon

"punks jump up to get beat down" - Brand Nubian

Whereas later comments in Arcangel’s code like `loop forever...` and `vblankzzzzzz` connote ironic hacker lingo or self-effacing leetspeak to frame Arcangel’s b-boy affectations, the attributed epigraphs in the header are more specific and help contextualize the artwork. "you mess with the best, you die like the rest" - Anon is a U.S. special forces motto used here as a reference to the 1995 movie Hackers—a tongue-in-cheek countercultural reference that nevertheless situates the piece in relation to the military industrial complex upon which videogames are founded. The other quotation, "punks jump up to get beat down" - Brand Nubian, is the hook of a track by nineties hip hop group Brand Nubian. Other lyrics on the song include “Though I can freak, fly, flow, fuck up a faggot / Don’t understand their ways I ain’t down with gays,” a couplet that recontextualized the meaning of the chorus.

In an article from 2013, Lord Jamal (qtd. in Barrow 2013), one of the members of Brand Nubian, offered a response to the criticism that “Punks Jump up to Get Beat Down” was homophobic:

When we did shows people would get their asses whipped, but none of them were homosexuals. When that song came on, if you had a grudge against somebody they would wait til that song came on and start whopping they ass. It was the beat down song...
Noting these connections in the Critical Code Studies Working Group, Stephanie Boluk (2014) writes

Despite claims that “faggot” was not being used to signify gay people, Brand Nubian ended up releasing a version in which “sissy” was substituted. “Sissy” allowed the song to circulate on the radio, however both terms are used to shame individuals via the suggestion that they fail to conform to a privileged model of heteronormative masculinity. The erasure of “faggot” does little to challenge the gendered and homophobic connotations implicit in the phrase. . . . [A] similar disavowal might be going on with the erasure of Mario in Super Mario Clouds.

Boluk (2014) further argues

When Arcangel notes in “The Making of Super Mario Clouds” that his epigraphs are “aimed at media artists who think they can step to my style” he is issuing a challenge. Rather than establish an environment of collaboration, he’s setting up one of competition: turning his artwork into the demoscene’s version of flyting and one-upsmanhsip. So what does it mean that a cisgender white man appropriates the braggadocio style of a hip hop “beat up song” as the epigraph to his work? Arcangel is engaged in an act of appropriation and remix that divorces the racial, political, and class history of a song like “Punks Jump Up to Get Beat Down” from its original context. And much like Lord Jamar’s disavowal of the semantic stakes of Brand Nubian’s use of “faggot,” I see Arcangel engaging in a radical software art that is similarly underwritten by its own politics of disavowal. Both economically, culturally, and materially.

Authored alongside this chapter (and discussed in depth below), the production of an original ROM hack in response to Arcangel’s work performs the argument Boluk outlines above. Initially, the idea to create a new piece of software seems less "after Arcangel" and more "vs. Arcangel" as this new game tries to "1-up" the original ROM hack with a more authentic or better crafted version of Super Mario Clouds while simultaneously pointing out certain infelicities through the precise medium and method with which the artist worked. While the original ROM hack was undertaken in order to contrast Arcangel’s piece and engage the material history that Super Mario Clouds is
deliberately obfuscating, it nonetheless participates in a masculinist culture of competition.

4.3. \texttt{incbin "clouds.hex"}

After the opening comments and Arcangel’s epigraphs, the executable code begins with a simple line:

\texttt{asm}

This code signals Bob Rost’s BASIC compiler, nBasic, to interpret the following code as 6502 ASM, a particular flavor of assembly language designed for the 6502 processor. What is significant about this line is that unlike \texttt{Super Mario Bros.}, which is coded in assembly, Arcangel is writing his software in BASIC.

Next, a four-line header is written to give the emulator or console all the information about the game including mapper, graphics mirroring, and PRG and CHR sizes.

\texttt{.inesprg 2 ; 2x 16KB bank of PRG code}
\texttt{.ineschr 1 ; 1x 8KB bank of CHR data}
\texttt{.inesmir 1 ; background mirroring}
\texttt{.inesmap 0 ; mapper 0 = NROM, no bank swapping}

These lines are typically included at the very beginning of an ASM file and are identical to those used at the beginning of \texttt{Super Mario Bros.} Even though \texttt{Super Mario Clouds} does not use 32KB of PRG memory, Arcangel still designates the same parameters as Nintendo’s original game. Instead of \texttt{.inesprg 2 ;}, the first line could be \texttt{.inesprg 1;} and the game would run fine.
Next, directives are commands sent to the assembler to perform functions like locating code in memory. They start with a . and are indented. Arcangel’s first directive is:

.org $8000

Whereas some programmers use tabs or four spaces, Arcangel uses five spaces in Super Mario Clouds. This particular directive, .org $8000, points to the code starting at memory location $8000, the first address inside the game’s ROM. To understand why this particular value is used, it is helpful to know a little bit about the architecture of the processor.

The Ricoh 6502 processor is a modified version of the famous 6502 processor used in the Apple 2 and Atari. The 6502 is an 8 bit processor with a 16 bit address bus that can access 64KB of memory without bank switching. In the NES this memory space is split up into random access memory (RAM), the picture processing unit (PPU), the audio and controller access ports, optional WRAM inside the cartridge, and, finally, ROM:

- $0000-0800 - Internal RAM, 2KB chip in the NES
- $2000-2007 - PPU access ports
- $4000-4017 - Audio and controller access ports
- $6000-7FFF - Optional WRAM inside the game cart
- $8000-FFFF - Game cart ROM

The first directive in Super Mario Clouds, .org $8000, defines the location of the game’s ROM—the same register that Super Mario Bros. and every other Nintendo game begins.
After that, Arcangel adds a label:

clouds_start:

aligned to the far left and ending with a :, labels are used to organize code and make it easier to read. The assembler translates the label into an address. Taken together with the previous line, when the code is compiled, each clouds_start: will be replaced with the address $8000. So, if there was a line of code later that uses a function like STA clouds_start, the compiler will also find and replace the label to produce STA $8000.

Next is a .dw or “define word” directive:

```
.dw clouds_start_addr
clouds_start_addr:
```

Along with .db or “define byte” these pseudo-ops are two-character versions of .byte and .word and are used for storing data (1-2 bytes) at specific memory addresses. The .dw here just pushes the name clouds_start_address into memory so that every time that specific address is loaded it jumps to the following label.

Additional data files are frequently used for graphics data or level data. In the next line Arcangel includes a binary file:

```
.incbin "clouds.hex"
```

clouds.hex contains a hexadecimal representation of the 4-screen layout called a “nametable.” Instead of using a long series of .db arrays, Arcangel built this .incbin in a separate hex editor. The .incbin directive can be used to include external data in the .NES file. This data will not be used yet, but is needed to make the .NES file size match
the header. As noted in a comment at the end of Arcangel's code: .incbin "mario.chr" ;includes 8KB graphics file from SMB1.

Inspecting clouds.hex reveals 4096 bytes of hexadecimal data that simply replace the long .db arrays that would normally be included directly in the ASM. Instead of loading each value of a .db array in his load name tables routine, Arcangel loads clouds.hex from the label start_clouds.

As explained on the NESDev Wiki (2014), the PPU’s nametables in his tutorial on backgrounds:

A nametable is a 1024 byte area of memory used by the PPU to lay out backgrounds. Each byte in the nametable controls one 8x8 pixel character cell, and each nametable has 30 rows of 32 tiles each, for 960 ($3C0) bytes; the rest is used by each nametable’s attribute table. With each tile being 8x8 pixels, this makes a total of 256x240 pixels in one map, the same size as one full screen.

The NES’s PPU can load a total of two name tables, or two screen widths starting at $2000. So pixel-wise scrolling on the NES is always constrained by two screen-widths (see Figure 22).
Figure 22. Super Mario Clouds continuously scrolls two screen widths of clouds.

Following each nametable, a 64-byte attribute table defines which palette is assigned to each part of the background. Each attribute table, starting at $23C0$, $27C0$, $2BC0$, or $2FC0$, is arranged as an 8 by 8 byte array. Since there are many fewer attribute values than titles, each element of the attribute table defines the palette for four tiles (see Figure 23).
Figure 23. As detailed by Brian Parker (2008b) in his tutorial on backgrounds, the PPU can only assign a total of four colors to each 2 x 2 tile array in the background.

In Super Mario Clouds, Arcangel sets every element of each attribute table to $00$ (or 0 0 0 0) because each of the four tiles is colored by palette 0 which consists of the
colors $21$, $30$, $11$, and $0D$. These attribute tables punctuate clouds.hex and are loaded into $23C0$, $27C0$, $2BC0$, and $2FC0$ when "load name tables" runs. The background scrolls as a single unit based on the above composition, rather each individual cloud moving autonomously. Since Arcangel could place the clouds in any arbitrary order, why did he design Super Mario Clouds the way he did? The clouds in the original Super Mario Bros. repeat, but not in Arcangel's order. Looking at clouds.hex in a hex editor, each cloud graphic are represented by numbers other than $24$ and Arcangel mistakenly repeated value $37$ on the edge instead of using as $38$ as is the case in Super Mario Bros (see Figure 24). It’s the telltale typo. As Nathan Altice (2013, 13) writes “[i]t is ironic that Arcangel highlighted the lack of any generational loss in his hack, since the incorrect tile is clear evidence that he hand-assembled the clouds instead of ‘ripping’ them from the game’s pattern tables.”
Figure 24. Looking at clouds.hex reveals not only the position of the clouds, but the specific error in Arcangel’s composition: a repeating “37” instead of a “38” on the red squares.

4.3 *lda #$21; lda #$30; lda #$11; lda #$0D;*

Beyond the contour and layout of the clouds, the colors of Arcangel’s clouds do not match those in *Super Mairo Bros*. The colors are loaded in this sequence:

```asm
; +++++++++++++++++++++++++++
; load palette
; +++++++++++++++++++++++++++
lda #$3F
sta $2006
lda 0
sta $2006
lda #$21 ;background
;(powder blue)
sta $2007
lda #$30 ;cloud inside
;(blue)
sta $2007
lda #$11 ;highlight
;(blue)
sta $2007
lda #$0d ;outline
```
Looking at this code, except for the #30, which represents the clouds’ main, white color, the other values are different in Arcangel’s work (see Figure 25). Arcangel also uses the dreaded “0D,” a value that ROM hacker Brian “Bunny Boy” Parker (2008a) warns “is a bad color and should not be used.” The “NES Tech Frequently Asked Questions” explains

> The official [register value for black that] Nintendo recommends is either $0E or $0F. Palette value $0D is the blackest value of them all . . . known as ‘blacker than black’. In other words, some video monitors or television screens might not accept that colour as a valid NTSC colour value (NES Dev 2002).

The entire “D” column is more of an artifact of the Picture Processing Unit or PPU rather than colors at all. Most aftermarket RGB PPs, for example, do not generate any of the “D” colors. Of all the available blacks, why would Arcangel pick this particular value, an artifact blacker than black?
Figure 25. The FCEUX emulator is not only good for creating tool-assisted speedruns, but also has tools for visualizing operations of the Picture Processing Unit. Here Super Mario Bros. palette (top) is compared with Super Mario Clouds (bottom).

One of the reasons Scott’s earlier remarks about Warhol’s practice is so evocative is that just as Warhol addressed mass commodity through the appropriation of both corporate branding and industrial production--screen printing Brillo or Campbell’s
logos on monochrome canvases in his “art factory”—Arcangel deploys the
representation of Super Mario Bros.’ graphics as an icon, rather than a game. Even as the
distinction between graphics and programming is dramatized by the separate ROM
chips on Nintendo’s original cartridge, these two orders converge in both the way the
NES processes information and in the ways in which games are played. In this sense,
Super Mario Clouds is a game that is being played in galleries and museums, but its
relation to Super Mario Bros. in terms of authenticity is less obvious. Arcangel’s clouds
are, in some ways, as much Super Mario Bros. as Warhol’s cans are filled with broth or
Magritte’s pipe is packed with tobacco. Which is to say, the play in Super Mario Clouds
operates in terms of representation even as it attempts to index the platform-based
processes driving the original game. Perhaps if Warhol screen printed with tomato soup
or Magritte sketched with tobacco smoke, the point would be a bit clearer given the
medium specific and sculptural nature of Arcangel’s exhibits (with their loose cables
and CRTs strewn about). Nevertheless, to echo Foucault, “[a] day will come when . . .
the image itself, along with the name it bears, will lose its identity.” Nintendo, Nintendo,
Nintendo Nintendo.

4.4. Coin Heaven

In order to “erase everything but the clouds,” this chapter concludes with
documentation of an original ROM hack produced by following the directions Arcangel
advertises in his video, essay, and website on “The Making of Super Mario Clouds.” In
2004, Arcangel’s documentation was designed and disseminated strategically to coincide with the exhibition of Super Mario Clouds. As Arcangel discovered, it turns out “erasing everything but the clouds” is harder than it seems. In order to physically reproduce an NES cartridge equipment like an original Super Mario Bros. (complete with its PRG and CHR chips), erasable programmable read only memory or EPROM chips, a universal EPROM burner, and a EPROM eraser (which is just a drawer with a UV light inside) are required to attempt this form of erasure—tools Arcangel must have had to complete his original piece (see Figure 26).

Figure 26. In “The Making of Super Mario Clouds” and on his personal website Arcangel details the physical process of ROM hacking a NES Cartridge.

Unlike the original mask ROM containing the graphics and data for Super Mario Bros., EPROM memory is encased in glass and when exposed to UV, can be reset to a
bank of filled capacitors representing 1s. After resetting a chip, a binary file of 0s and 1s can be “pushed” from capacitor to capacitor, storing a series of data on the chip in perpetuity (or at least until it’s charged through UV light again.) The filaments running from the memory chip to each of the twenty-eight legs, interface with the Nintendo through the the edge connector of the cartridge and the 72-pin connector in the console. But, like Arcangel documents in The Making of Super Mario Clouds, one must first desolder the mask ROMs from the cartridge, added 28-slot EPROM sockets, and cut a window into an original Nintendo cartridge before new, modified software can be played on the system.

To “erase everything but the clouds,” I started with a community disassembled and commented version of Super Mario Bros. original machine code. The first thing I did was redirect all the music and sound effect pointers to the address for silence. Super Mario Clouds should be silent. After that, I needed the game to auto-scroll to the right so I used a level editor to change the order in which Super Mario Bros. progressed. Now the first level loaded would be the short animation sequence between World 1-1 and World 2-2 (where Mario walks from the castle into the green pipe.) In order for this short animation to loop indefinitely, I reduced the layout of the level to a flat, ground plan and then added the “looping” function from Bowser’s mazes. Since Super Mario Bros. only loads two screens worth of data at a time, a few of Bowser’s castles were programmed to reload the same section of the level over and over unless Mario was at
the proper vertical height to trigger the next screen (usually correlated to paths on the bottom, middle, and top of the level). After getting rid of Mario’s music and getting him to constantly scroll the screen, I used a couple different tile editors to “erase” some of the graphics. Importantly, this act of erasure parallels the conversion of 1s to 0s on the ROM chip itself. I defaced the letters in the menu and scrubbed out Mario’s sprites by painting over them with the “invisible” color for transparency. Then I set the brick tiles that make up the ground to the same color as the sky. The result of these many different forms of erasure is a ROM hack in which an invisible, animated Mario who constantly walks forward, looping within two screen widths of blue sky and clouds. Aside from Mario’s walk speed, the looping procedure, and the color, shape, and pattern of the clouds, something is very different from Arcangel’s artwork: a lonely coin (see Figure 27).

Figure 27. Attempting to “erase everything but the clouds” and remake Super Mario Clouds (left) results in erasing everything but the coin. In Coin Heaven (right), an original ROM hack made for Aftermarket, a lonely coin remains blinking in the blue sky.
Scattered across the Mushroom Kingdom in patterned arcs and drawing the player forward in leaps and bounds, coins are a part of the environment in Super Mario Bros. Even on a technical level, coins are background material. Whereas a single row of 16 x 16 pixel coins would easily max out the 8-sprite-per-scan-line limitation of the Nintendo’s Picture Processing Unit, the coins in Super Mario Bros. are rendered using background tiles with a cycling palette. They are literally embedded in the landscape. Like trees of pudding or the pre-butchered pigs inhabiting Brueghel’s Land of Cocaine, money, it would seem, does grow on trees--or at the top of beanstalks. Nowhere is this utopian imagery more obvious than in Super Mario Bros.’ bonus zones. Devoid of enemies and filled with money, these three autoscrollers go by the name of “coin heaven” (see Figure 28).

Figure 28. Three different “coin heavens” appear at the top of vines in World 3-1, 4-1, 4-2, 5-1, 5-2, 6-2, 7-1, 8-1, and 8-2 of Super Mario Bros.
The lone coin flashing on the screen in the new ROM hack is not a standard collectable. Rather than the normal 16 x 16 pixel coins Mario grabs throughout his adventures, this 8 x 8 icon is the last remaining element of the game’s menu. To my chagrin and the chagrin of many ROM hackers, this tiny, blinking coin is also what is known as “Sprite 0,” the first sprite in the Picture Processing Unit’s memory and the only sprite that includes hard-coded collision detection or a “hit flag.” If a nontransparent pixel of Sprite 0 overlaps a nontransparent pixel of the game’s background, the flag is set. In Super Mario Bros this special flag is reset at the beginning of every frame and then used to determine not where but when in a given frame update, the screen should begin scrolling. Time does not move without money and making Sprite 0 invisible freezes the game. Apparently all that is solid does not melt into air as I tried to get past this brick wall and “erase everything but the clouds,” Sprite 0 began to symbolize not the formal autonomy of games, art, and capital—but, like all currency, the impossible desire for a type of utopia in which these practices operate without material base. This coin, then, only appears to offer the player a “coin heaven,” and in doing so reveals the intimate and intractable relation between the scrolling software and the processes of the Nintendo’s PPU chip.

In Signifying Nothing, Brian Rotman (1987, 27) argues that the figure of zero operates as a “meta sign.” For Rotman, zero is the invention of a concept that does not signify the presence of an object but rather its absence. Rotman (1987, 27) argues that the
meta-sign both “initiates the signifying system and participates within it as a constituent sign.” Sprite 0 is a nothing that is still very much a something: it is precisely that constitutive absence which--were you to remove that single, blinking pixel--would crash the entire system. Sprite 0 is the meta-sign of Super Mario Bros. and stands in for the larger metagame that has been erased in Super Mario Clouds. Arcangel’s removal of Sprite 0 is not just a code of omission, but inadvertently reinforces the ideology in which art, games, and technology exist in a world apart. From rare earth mineral mining in the Congo to e-waste landfills in China, the disappearance of Sprite 0 does not just erase the gameplay, but allegorizes the erasure of the historical, material, and economic modes of production through which Super Mario Bros. circulates. While Arcangel’s piece is famous for “erasing everything but the clouds,” it also effects an erasure of the game’s medium specificity, depicting a utopian autonomous zone that renders invisible the game’s history of money and materiality. By contrast, Coin Heaven refuses this portrayal, demonstrating the forensic history that is hard coded into the game’s electrical circuits.
5. **Platform Games: An Exhibition at Babycastles**

“why does it always have to be about fucking videogames? why can’t it be about art? or anything else? . . . FUCK MARIO. fuck mickey mouse. fuck bugs bunny. fuck star wars.”
--Liz Ryerson; March , 2014

From tool-assisted speedrunning to real time attacks to ROM hacking and hardware reproduction, without play, there is no game. Throughout this study, the term *aftermarket* has been used to designate the period of time period in which the life of a videogame is extended beyond the advertisement and sale of new systems and never-ending sequels. In the aftermarket, play ceases to operate as a given--a unconscious way to consume a product--and begins to behave like a form of game design. In the aftermarket, players do not only decide to play, but in doing so completely reinvent the game, transforming software into platforms for new kinds of experiences. Not only would tool-assisted speedrunning, real time attacks, and ROM hacking cease to exist without a decade-long investigation by the communities that continue to play in the aftermarket, but the previous chapters would not be possible without a robust form of material play.

Craigslisting consoles, unscrewing security bits, clipping anti-piracy chips, copying copyrighted code, burning binary data, soldering second-hand integrated circuits, and making metagames are not only the ways people play with *Super Mario*...
Bros., but the way I researched this volume. Produced alongside the chapters of this book were a series of projects. The relationship between Morimoto’s moSMB3.mov and the NES 4021 was explored alongside single-switch controllers and an automatic guitar. The history of speedrunning Super Mario Bros. was inspired by a video of 99 failing Marios and a videogame of 99 modified Mushroom Kingdoms. And the close reading of Cory Arcangel’s Super Mario Clouds source code was performed in tandem with a ROM hacking experiment, a test to see if sunlight could erase an EPROM, and a palette of paints made to match the output of the Nintendo Entertainment System’s PPU. These research projects were then cleaned up and collected for Platform Games, a solo exhibition of original art that was installed at Babycastles in New York City from May 7–17, 2015.

Founded by Kunal Gupta and Syed Salahuddin in 2009, Babycastles (2015) is a “collective with roots in New York’s D.I.Y. culture dedicated to bridging the independent game developer community with the broader New York art community.” One of the only venues in the world dedicated to exploring art and videogames, their alternative community space at 137 W. 14th St. in Manhattan features a gallery, arcade, and event space that is open daily for co-working hours and is host to a constantly evolving program of exhibitions, workshops, presentations, and artist residencies. Situated between Aram Barthol’s Point of View (February 19–April 17, 2015) and Anna Anthropy’s The Road to Empathy (June 20–July 16, 2015), Platform Games attempted to
engage the community histories and material practices of the players discussed in previous chapters who, over the past ten years, transformed videogames from packaged products into open platforms for making their own ludic and aesthetic experiments.

For ten days, Platform Games focused on the physical properties, technical capacities, and social play around a single game: Nintendo’s Super Mario Bros. Featuring original ROM hacks and EPROM poetry, a speedrunning documentary and tool-assisted tablature on a self-playing guitar, a one-switch controller and a network of autonomous Nintendos, as well as an arcade of Mario metagames and a bucket of unrefined coltan ore, this “close playing” or “platformer study” of Super Mario Bros. did not attempt to reify Nintendo’s game as an iconic or ideal piece of software, but appropriates, manipulates, perforates, duplicates, aggregates, and dissipates videogames into a different kind of “Mario Paint”—a historically specific medium for making metagames and media art.

If the MN4021B shift register is Mario Paint and the Ricoh 6502 processor is Mario Paint and the PPU graphics chip is Mario Paint and the APU sound card is Mario Paint the CMOS ROM masks storing Super Mario Bros. are Mario Paint then conflict minerals, slave labor, and e-waste are also colors in Mario’s pallet. Platform Games painted a picture that attempted to disrupt the cultural logic of an immersive and escapist magic circle, black box, and white cube in which the phenomenal, political, economic, and material history of play are erased. Following the discussion of ROM
hacking, critical code studies, and *Coin Heaven* and based on the *Platform Games* catalog, this chapter will feature documentation of the exhibition at Babycastles (see Figure 29 and 30).
Figure 29. Platform Games advertising (2015) by Patrick LeMieux. Solo exhibition at Babycastles at 137 W. 14th St., New York, NY from May 7 to May 17, 2015.
Figure 30. Platform Games (2015) by Patrick LeMieux. Solo exhibition at Babycastles at 137 W. 14th St., New York, NY from May 7–17, 2015.
5.1. *Fucked Marios*

Figure 31. *Fucked Marios* (2015) by Patrick LeMieux. Twelve laminated LaserJet prints on foamcore; 12 x 12 inches each.
Figure 32. *Fucked Marios* detail (2015) by Patrick LeMieux. Twelve laminated LaserJet prints on foamcore; 12 x 12 inches each.
The posters, postcards, invitations, and other announcements advertising *Platform Games* featured the pixelated, full saturation graphics of the exhibition’s most visually captivating work: *Fucked Marios* (see Figure 31 and 32). Upon entering Babycastles and walking past the dripping show title on the left, twelve laminated LaserJet prints mounted on foamcore hung in a four by four grid floating about an inch away from the wall. Both the title and form of the piece are tributes to Liz Ryerson, an artist, musician, and game designer whose prolific writing, making, playing, and presenting function as a primary example of aftermarket game design philosophy. Synthesizing Ryerson’s “Fuck Mario” blog post with her daily Tumblr practice, this series of prints uses a modified version of Michael Brough and Andi McClure’s *BECOME A GREAT ARTIST IN 10 SECONDS* to make “Fucked Marios.”
5.2. *Brothers*

Figure 33. *Brothers* (2013) by Patrick LeMieux. 11 CRT televisions, 2 Nintendo Entertainment Systems, 2 ROM hacked cartridges, Arduino; size variable.
Figure 34. *Brothers* detail (2013) by Patrick LeMieux. 11 CRT televisions, 2 Nintendo Entertainment Systems, 2 ROM hacked cartridges, Arduino; size variable.

Turning clockwise from the twelve prints hanging in the back left side of Babycastles, a stack of CRT televisions displayed the nonhuman play of two networked Nintendo Entertainment Systems (see Figure 33). Each console contained a ROM hack of *Super Mario Bros.* with no music, infinite lives, and fast resets. Chords hang from the audio channel and first player control port of each system, interleaving between the two sets of screens (see Figure 34). In *Brothers*, an electrical signal generated by the audio processing unit of a Nintendo Entertainment System flips a switch. Mario jumps over a pipe and into pit. As his 8-bit body leaves the screen, Koji Kondo’s conciliatory “game over” music signals a second switch. Another brother, running on another platform,
jumps. In the piece, feedback between two networked NESs reveals both the material constraints of Nintendo’s flagship platform and a form of nonhuman play. Asymmetry in the two systems—whether material, temporal, or structural—produces emergent patterns as each plumber leapfrogs over goombas and green pipes according to the thanatopic assistance of the other. The slowly oscillating death drive of two suicidal automata recast Conway’s Game of Life as a Game of Death. Without the player, there is still play and without life, there is still a game.

Although Super Mario Bros. was engineered to showcase the affordances of the Nintendo Entertainment System, the widespread success of the game in the late 1980s naturalized the scaled and speed of Mario’s movement as a design idiom and software genre for a generation of players. Without human intervention, the rhythm and repetition of the platform’s electrical components and speed and seriality of the software’s design produce an emergent, zero-player game that expresses the material play of the Nintendo Entertainment System itself. Driven forever forward by the rightward momentum of Super Mario Bros.’ scrolling mechanic, a million mindless Marios must die to make it through the Mushroom Kingdom—Sisyphean brothers in a Tartarean landscape.
5.3. Conflict Free

Figure 35. Conflict Free (2015) by Patrick LeMieux. EPROM memory; 1 3/8 x 1/2 inches.
Figure 36. Conflict Free detail (2015) by Patrick LeMieux. EPROM memory; 1 3/8 x 1/2 inches.
Directly to the right of Brothers and resembling the dripping contours of the Platform Games logo, a white cloud on a sky-blue accent wall frames a small, memory chip (see Figure 35). The EPROM chip is encoded with the three verses of “‘Heaven’—is what I cannot reach!” by Emily Dickenson:

"Heaven"—is what I cannot reach!
The Apple on the Tree—
Provided it do hopeless—hang—
That—"Heaven" is—to Me!

The Color, on the Cruising Cloud—
The interdicted Land—
Behind the Hill—the House behind—
There—Paradise—is found!

Her teasing Purples—Afternoons—
The credulous—decoy—
Enamored—of the Conjurator—
That spurned us—Yesterday!

‘Heaven’—is what I cannot reach!” not only makes an allusion to the Greek myth of Tantalus, but places his predicament among the “cruising clouds” and interdicted ground” of Super Mario Bros.’ landscape.

Up to his chin in rippling water, down to his nose in ripe fruit, Tantalus can never consume that which he desires most. Just as Sisyphus embodies the very concept of labor, Tantalus is desire. Fused together in the depths of Tartarus, these two figures allegorize the ideology of play in the twenty-first century: an impossible desire for an endless labor. Tantalum, the conflict mineral named after Tantalus, is used to produce the processors, shift registers, and ROM masks that make the technical operations of videogames possible. Doomed to the agony of another kind of Tartarus—a magic circle,
a black box, a white cube—gamers and artists alike often unsuccessfully attempt to
forget the material, economic, and political history of the platforms through which they
play. Exposed to sunlight streaming through the windows in the back of Babycastels, the
poem is slowly erased over the course of ten days (see Figure 36).

5.4. Coin Heaven

Figure 37. Coin Heaven (2013) by Patrick LeMieux. 2 projectors, 2 Nintendo
Entertainment System, 2 ROM hacked cartridges; size variable.
Figure 38. *Coin Heaven* detail (2013) by Patrick LeMieux. 2 projectors, 2 Nintendo Entertainment System, 2 ROM hacked cartridges; size variable.
As discussed in the previous chapter, one of the most famous ROM hacks is Cory Arcangel’s **Super Mario Clouds**. Arcangel claims “Super Mario Clouds is an old Mario Brothers cartridge which I modified to erase everything but the clouds.” And although Arcangel embraces a hacker ethos and has open sourced much of his artwork, attempting to follow his instructions and “erase” **Super Mario Bros.** produces an entirely different game. As the central piece organizing **Platform Games**, in **Coin Heaven** an invisible Mario walks on invisible ground, looping endlessly in a cloudscape where a cinematic sequence once took place between World 1-1 and World 1-2. Beyond the speed that the screen scrolls, the composition of the clouds, and even the color of the game’s palette, something is very different from Arcangel’s well-known artwork: a lonely coin remains, still blinking in the game’s menu.

This coin, known as “Sprite 0,” is the first sprite of the Nintendo’s Picture Processing Unit and the only sprite that includes a hard-coded hit flag. Time does not move without money and making “Sprite 0” invisible freezes the game. This coin then, only appears to offer the player a “heaven” and, in doing so, reveals the intimate and intractable relation between scrolling software and the processes of the PPU chip as well as the larger circuits of global capital through which this game continues to move.
5.5. **RGB-SMB-TAB-SSS**

Figure 39. **RGB-SMB-TAB-SSS** detail (2015) by Patrick LeMieux Fender Stratocaster, Arduino, 6 Servos, 800 LasterJet prints; 43 x 6 x 17 1/2 inches, 8 1/2 x 11 x 6 inches.
Figure 40. RGB-SMB-TAB-SSS (2015) by Patrick LeMieux Fender Stratocaster, Arduino, 6 Servos, 800 LaserJet prints; 43 x 6 x 17 1/2 inches, 8 1/2 x 11 x 6 inches.
On the stage at the front of the gallery a series of six servos plucks an open chord on a pink, “single-single-single” pickup Stratocaster (see Figure 39 and 40). The guitar’s idle strumming is set to the rhythm of Alexander Galloway’s How to Win Super Mario Bros. or RSG-SMB-TAB, a record of controller input from a playthrough of Super Mario Bros. at Eyebeam between March 30th and April 4 in 2003 (see figure 41). Formatted as guitar tablature and hosted alongside “tutorial” videos featuring only Galloway’s hands, RSG-SMB-TAB satirizes text-based strategy guides on websites like GameFAQS by offering specific input instead of general guidelines. Although RSG-SMB-TAB deploys a conceptual aesthetic in which textual instructions take the place of the object or event, RGB-SMB-TAB-SSS makes Galloway’s recording playable.
5.6. *One Switch*

![Image of One Switch](image)

Figure 42. *One Switch* (2015) by Patrick LeMieux. Custom controller, Nintendo Entertainment System, CRT screen; 5 x 2 x 2/3 inches.
Figure 43. One Switch (2015) detail by Patrick LeMieux. Custom controller, Nintendo Entertainment System, CRT screen; 5 x 2 x 2/3 inches.
Backstage, behind the pink guitar and under the screen displaying what looks like standard *Super Mario Bros.* gameplay are two orange vinyl chairs, a CRT television, and a strange controller plugged into a Nintendo Entertainment System (see Figure 42 and 43). As argued in Chapter 2, from Atari’s joysticks to Nintendo’s D-pads, videogame controllers are designed to correlate qualitative play to the binary states of digital buttons. Whereas standardized control standardizes play and imagines normative players, alternative interfaces do not simply make videogames accessible, but radically transform what videogames are and what they can do. Following the practices of players at AbleGamers.com and the design of one-switch games made for persons with limited manual dexterity, this modified Nintendo controller constantly pushes Mario to the right. Without access to any other buttons, the player is left to rhythmically tap jump, transforming *Super Mario Bros.* from a single-player game into a single-switch game for multiple players.
5.7. Mario Paint

Figure 44. Mario Paint detail (2015) by Patrick LeMieux. 10 gallons of paint, unrefined coltan ore; 32 x 26 x 6.5 inches.
Figure 45. Mario Paint (2015) by Patrick LeMieux. 10 gallons of paint, unrefined coltan ore; 32 x 26 x 6.5 inches.
Arriving back at the entrance of Babycastles after circumnavigating the show, an open door next to the Platform Games logo reveals three more projects. The first piece, Mario Paint, features the ten gallons of colorful paint used to paint the show’s logo (see Figure 44 and 45). Each bucket represents a color of Super Mario Bros.’ palette generated by the Nintendo Entertainment System’s Picture Processing Unit. Sitting at the bottom of the empty, sky-blue bucket are chunks of unrefined coltan ore, the primary source of tantalum (see Figure 46). Mirroring the circular window exposing arrays of semiconductors in Conflict Free and the blinking Sprite 0 in Coin Heaven, the coltan floating in a cylinder of sky blue paint articulates the materiality of the games we play.
5.8. 99 Exercises in Play

Figure 47. 99 Exercises in Play (2012-) by Patrick LeMieux. 4 Unity Games, 4 Nintendo Controllers, 4 USB Dongles, 4 Arcade Cabinets; size variable.
Figure 48. *99 Exercises in Play* detail (2012-) by Patrick LeMieux. 4 Unity Games, 4 Nintendo Controllers, 4 USB Dongles, 4 Arcade Cabinets; size variable.
Moving past Mario Paint and into Babycastle’s arcade, a cacophony of color and sound fills the tiny space. Nintendo controllers dangle out of the faces of the gallery’s array of custom arcade cabinets that, for the duration of Platform Games, featured different builds of 99 Exercises in Play (see Figure 47 and 49). Based on the constrained writing practices of the Ouvroir de littérature potentielle (Oulipo) and specifically Raymond Queneau’s Exercises de style, this original game redeployes World 1-1 from Super Mario Bros. as a constraint for producing ninety nine metagames— an “Oujeupo” project that engages the serial histories of a single level (see Figure 48).

Figure 49. 99 Exercises in Play detail (2012-) by Patrick LeMieux. 4 Unity Games, 4 Nintendo Controllers, 4 USB Dongles, 4 Arcade Cabinets; size variable.
Figure 50. 4:57 (2013) by Patrick LeMieux. Single channel color video, stereo sound; size variable, 6 minutes.
Projected from on top of Babycastles’ arcade cabinets, the back wall of the back room is filled with an image of ninety nine Marios bounding in unison through the Mushroom Kingdom while the muddy voices of ninety nine players talk over one another (see Figure 50). This short, five-minute documentary composites ninety nine of Andrew Gardikis’ failed attempts to beat Super Mario Bros. as fast as humanly possible before he achieved a world record speedrun on July 1, 2013. As discussed in Chapter 3, the number 4:57 represents one horizon of possibility for human play within Super Mario Bros. and is Gardikis’ ultimate goal--what he calls his “gaming masterpiece.” At any given frame of the documentary, the cloud of Marios in the middle of the screen is evidence of both Gardikis’ precision as well as the temporal and spatial registrations undergirding videogames as technical media.

At the end of the video, Gardikis shouts “oh my gosh, I don’t even know!” Just as Duchamp transformed chess into a game of chance, for Gardikis speedrunning operates outside of human perception as Super Mario Bros. has ceased to function in terms of human agency. Only after thirty minutes of decoding video and counting frames was this world record discovered in retrospect. Gardikis’ uncertainty articulates a kind of play occurring outside the register of human consciousness and an unknowable, serial history of Super Mario Bros., in which thousands of thumbs have millions of Marios over goombas and green pipes.
5.10. *Magic Circle*

![Image of Magic Circle](image)

**Figure 51. Magic Circle (2015) by Patrick LeMieux.** Single channel color video, stereo audio; size variable, 2 hours and 9 minutes.

After exiting Babycastles’ eclectic arcade and walking by both paint buckets and the Platform Games logo, one might notice a faint green glow behind the gallery door and pick up on the radio chatter of sports announcers quietly commentating a secret game in hushed tones (see Figure 51). Standing in the entryway, with a hand on the doorknob ready to leave the gallery, upon further inspection this last artwork (complete with title card and didactic description) operates as an afterword in contrast to the rest of the exhibition—the rich greens of the baseball diamond contrasting the overwhelming blues of *Super Mario Bros.* cited in almost every other artwork in Platform Games.
On Wednesday, April 29, 2015, the Baltimore Orioles and the Chicago White Sox played a game at Camden Yards. For the first time in the history of Major League Baseball, the event was closed to the public due to the protests, riots, and general unrest in the city after the police murdered Freddie Gray on April 19. Despite setting a record for the lowest attendance in the history of the professional sport, the game between the Orioles and Sox was unremarkable. Safely enclosed within the magic circle provided by the social ritual, formal play, networked media, and global capital of professional baseball, the game proceeded as usual. The crack of the bat and the snap of leather gloves punctuated the silence of the empty stadium as protesters and police faced off outside the gates of Camden Yards.

5.11. Live Events

Beyond each of these attempts to extend the play of fan communities and media artists alike, Platform Games also was designed to operate as a public platform for a series of live events organized explicitly to coincide with the exhibit. These events included an opening with music by Liz Ryerson; a symposium with presentations by Jacob Gaboury, Nick Montfort, Laine Nooney; a ROM hacking workshop; a speedrunning event with performances by Blechy, Cyghfer, Dram, Micro500, Tonic, Wyrm; and a screening of a film by Meghan Gordon.

The opening initiated the exhibition on Thursday, May 7 at 7PM and featured a guided tour and demonstration of all ten artworks, a new mixtape of “fucked”
videogame soundtracks as well as original compositions by Liz Ryerson, and a series of Mario-themed mixdrinks designed by Babycastles resident and game developer, Frank DeMarco.

The second event, "Tables, Teapots, and a Tartan Couch" was an academic symposium on Friday, May 8th at 7PM. The symposium featured a series of talks on the topic of game history, media archeology, and platform studies including “Standard Objects and Speculative Archaeologies in Early Computer Graphics” by Jacob Gaboury, “This is Not Mario: ROM Hacking Cory Arcangel’s Super Mario Clouds” by Patrick LeMieux, “A Pedestal, A Table, A Love Letter: The Archaeology of Gender in Video Game History” by Laine Nooney, and a response by Nick Monfort. Beginning with tables, teapots, and tartan couches, the panel discussed different approaches to studying games and digital media that attempted to disrupt essentialist narratives of control and intent, abstraction and idealism, as well as the “upgrade path” of technological progress. From the cave to the clouds, the speakers spelunked media speleologies, reached toward speculative archaeologies, and played with platformer studies.

The third event, "CLIPPING RIPPING HACKING BURNING" was a workshop on ROM hacking, hardware reproduction, and reverse engineering for the Nintendo Entertainment System on Monday, May 11th at 7PM. Starting with a physical copy of Super Mario Bros., participants clipped the pins of the games’ mask ROM and built a prototype cartridge before they ripped the game’s data using a universal programmer.
From there, the workshop focused on reverse engineering the relationship between the games’ ROM and the Nintendo Entertainment System’s RAM in the 6502 processor to make an original hack. After messing with Mario’s coins, lives, and timer with a hex editor and the FCEUX emulator, participants finally burnt their “babyhacks” onto EPROM memory chips and, using prototype boards from the exhibition, played the games on the stacks of CRT TVs in the back of the gallery space.

The final event, "RTA is anagram for ART," was a speedrunning event on May 13 at 7PM that featured real time attacks by local New York City players Blechy, Cyghfer, Dram, Micro500, Tonic, and Wyrm. The night began with an introduction to speedrunning in which Blechy and Cyghfer found the fastest route through a random co-op game of the audience’s choice (Snow Bros [1990]). Then there was a live performance of the classic 1994 platform game Mega Man X2 by Tonic and a runthrough of the impossible Kaizo Mario World ROM hack by Dram. Micro500 also demonstrated his tool-assisted speed running bot by serially streaming the input necessary to beat both Super Mario 64 and Mario Kart 64. The night was also punctuated by a film screening of “Streaming: All Becoming Real,” a collaborative Twitch.tv performance by Los Angeles artist Meghan Gordon and Cyghfer.

5.12. Afterword

In the aftermarket of the Nintendo Entertainment System, long after games like Super Mario Bros. have been on store shelves, play becomes a form of game design. And
although this play is infinitely plastic, it may not hold the attention of a given community. There is an endgame in the aftermarket and, through their own practices, discourses, and communities, players like Happy Lee, Andrew Gardikis, and Cory Arcangel each attempt to exhaust Super Mario Bros. For Happy Lee Super Mario Bros. exists as a single frame, for Andrew Gardikis, a single second. And for artists like Cory Arcangel, the endgame is equally unplayable. Or is it? The material plasticity of the hardware and recursive play of the communities offers a potential line of flight--a new horizon of play after the aftermarket.

Although the tool-assisted speedrunning, real time attacks, and ROM hacks discussed in Chapters 2, 3, and 4 can be executed accurately in software using console emulators, original hardware has become an important standard for all three communities. In this sense, the Nintendo Entertainment System as a platform becomes the single measure, connecting the diverse set of metagaming practices occurring in the aftermarket. In the case of speedrunning, executing human play on the original electronic consoles and cartridges functions as a “ground truth” for comparing times across different regions and communities. The tool-assisted speedrunning community has also begun the arduous process of “hardware verification” in order to preserve the integrity of their most historically important runs. Brothers was inspired by the complex input devices like Peter Greenwood’s “TASbot” or True’s “NES / SNES Replay Device” and “MultiReplay”--input devices engineered to sync linear inputs to a console in order
to playback tool-assisted speedruns in realtime. ROM hackers likewise have discovered methods to expand the functionality of vintage hardware to play new games produced on old platforms. By outfitting cartridge mappers with new, widely accessible RAM and ROM, software experiments can be retrofitted to actual hardware. Platform Games deployed the hardware, software, and code of Nintendo’s Entertainment System and Super Mario Bros. as not only as a material constraint on the work being shown at Babycastles but as a material collaborator which pushed back and moved play outside of artistic or authorial intention.

Beyond attempting to articulate the histories, practices, and attitudes of tool-assisted speedrunners, real time attackers, ROM hackers, and hardware reproducers, another element of Platform Games was the attempt to play with the work of a another player, designer, or artist—to extend artistic endgames into a space after the aftermarket. Fucked Marios follows Ryerson’s online presence, attempting to synthesize her blog with her daily tumbler. Brothers takes its inspiration from Peter Greenwood’s practice engineering bots to autonomously play tool-assisted speedruns. Coin Heaven is an homage and challenge to Cory Arcangel’s Super Mario Clouds, failing to rebuild the famous artwork through the methods of ROM hackers. RSG-SMB-TAB-SSS is after Alex Galloway’s work at Eyebeam in 2003—a platform designed to play an impossible composition. Single Switch was designed according to a speculative thread on AbleGamers.com in which players with low manual dexterity or even no hands
imagined alternative controllers that would make classic platforms accessible to different kinds of bodies while simultaneously inspiring different kinds of play. Exercises in Play is an explicit citation of Raymond Queneau’s Exercises in Style but takes Super Mario Bros. as its subject. These projects are not just examples of aftermarket game design philosophy but also after-aftermarket game design philosophy.

Furthermore, pieces like Conflict Free, Mario Paint, and, most explicitly, Magic Circle, are critiques of the exhibit itself, the institution of game art, and the enclosing effects of videogames in general. These works are an attempt to function as an afterword to Platform Games--another kind of aftergaming also present in this dissertation.

Aftermarket documents the community histories and engages the material practices of tool-assisted speedrunning, real time attacking, and ROM hacking in the aftermarket of the Nintendo Entertainment System between 2004 and 2014--a ten-year period set roughly twenty years after the release of Super Mario Bros. in 1985. A close analysis of Morimoto’s moSMB3.mov (2003), Andrew Gardikis’ attempt to achieve a 4:57 (2004-2014), and Cory Arcangel’s Super Mario Clouds (2003, 2009) not only reveals the history of play, but also shows how communities of players have been playing, making, and studying videogames over the last ten years. In doing so, tool-assisted speedrunners, real time attackers, and ROM hackers deploy the methods of academic disciplines such as platform studies, software studies, and critical code studies (and often well before these academic fields have emerged in any meaningful way). This kind
of rigorous play occurring in the aftermarket cannot be reduced to conspicuous consumption or ahistorical escapism, but is a practical and productive form of critical making—a **game design philosophy** that combines media archaeology, media theory, and media art. Beyond documenting examples of the history of play from these three communities, *Aftermarket* produces a comparative study based on players’ engagement with a single game: *Super Mario Bros*. More recognizable than Mickey Mouse, Mario represents the rebranding of videogames from “computer systems” into “entertainment systems” and following this collapse, enclosure, and privatization of play in 1985, the practices of tool-assisted speedrunners, real time attackers, and ROM hackers transform the plumber into a platform for making new games in the aftermarket of the videogame industry.
Works Cited


164


https://www.youtube.com/watch?v=xuXYuUMcxyE.


MN: University of Minnesota Press.


Minneapolis, MN: University of Minnesota Press.


http://www.speedrunslive.com/faq/.


http://tasvideos.org/WelcomeToTASVideos.html.

http://tasvideos.org/GameResources/NES/SuperMarioBros.html#The21FrameRuleSmb1Smb2j.


http://www.dslreports.com/forum/r8636929-.


http://tasvideos.org/SiteHistory/MorimotoAboutSMBMovie.html.

Biography

Patrick LeMieux is a media artist, game designer, and Ph.D. candidate in the Department of Art, Art History, and Visual Studies at Duke University. For more information please visit http://patrick-lemieux.com/.