Play While Paused: Time and Space in Videogame Pause Menus

Matthew D. Schmalzer

Abstract

Alexander Galloway (2006) argued that "to live today is to know how to use menus" (p. 17). This is especially true in videogames: To play videogames is to use menus. While menus proliferate in videogames, they are practically synonymous with the act of pausing, appearing whenever a game is suspended. This essay explores how both the act of pausing and navigating pause menus offer perceptions of temporality and space that situate videogame play as not simply explorations of game spaces; they also reveal the computational modes of time and space that are always present in all aspects of videogame play.

Menus proliferate in videogames. They are designed with countless aesthetics, employ many navigational methods, and are used for a myriad of functions. One of the first things a player sees when booting up a videogame is almost always a menu with options such as "new game" and "load game." There are menus for settings that allow players to tweak the technical parameters of their game. Menus are used for dialogue and combat and managing inventories. Shopping, crafting equipment, and completing quests are all often relegated to menus. Because of the wide variety of menus, it is impossible to talk about all instantiations generally. In this paper I narrow my discussion of menus, if only slightly, to the pause menu. Pause menus include menus that, when accessed, suspend most of the machine's actions that directly affect the game world.

Pausing has not always been a standard feature of videogames. Arcade cabinets of the 80s were designed to take as many quarters from players as possible, so a pause act would only diminish any cabinet's return on investment by potentially extending players' time on machine without inserting more coins. However, as videogames found their way into players' living rooms, play was no longer contingent on paying for playtime. A household's demands on a player's attention was not always conducive to long unbroken play sessions, either, so

Author Biography

Matt Schmalzer studies the aesthetics of digital media at North Carolina State University.

pausing became integral to play in this new environment. Additionally, other technologies like the VCR and cassette player conditioned users to expect the ability to pause their media. Accordingly, the Atari 5200 released in 1982 was unique in that it featured a controller with a button specifically dedicated to pausing. Most official games released on the Atari 5200 were simply ports of Atari 2600 or arcade games, and as such there was no special functionality when pausing. It simply froze the game state, allowing the player to leave the game and return to it later (or possibly consider their next move in the virtual world). As videogame systems became more powerful, the games on those systems became larger and more complex. Some games began leveraging the pause state to add gameplay features. *The Legend of Zelda* (Nintendo 1986) on the Nintendo Entertainment System, for example, featured a menu that completely eclipses the world of the game when it is accessed, and in its place presents a map, selectable inventory, progress indicators, and the save function (see Fig. 1). These kinds of menus have become extremely common in modern gaming, particularly in single-player adventure games with large diegetic item inventories like *Mass Effect* (BioWare, 2007), *The Elder Scrolls V: Skyrim* (Bethesda Game Studios, 2011), and *The Witcher 3* (CD Projekt Red, 2015). But even if there is little functionality in some pause menus, single player games of all sorts commonly feature a pause menu. It should be noted that just because they have become common, does not mean all games feature pause menus. Some games do not include pause menus for artistic or aesthetic reasons like Penn and Teller's (Imagineering, 1995) *Desert Bus*, which presents players with a marathon eight hour bus drive from Tucson, Arizona, to Las Vegas, Nevada. Synchronous multiplayer games like *Counter-Strike* (Valve, 2000) or *League of Legends* (Riot Games, 2009) and massively multiplayer online games like *World of Warcraft* (Blizzard Entertainment, 2004) are not able to be paused, either, because multiple players experience the same virtual space, and one player pausing would interrupt other players' actions in that space.

The kinds of actions that typically draw the most attention from players and scholars alike happen in what Jesper Juul (2005) dubbed the "game space" (p. 165), or the diegetic space players are able to navigate within. These kinds of actions are what Alexander Galloway (2007) called "diegetic operator acts," which are actions taken by the player that directly affect the game's fictional environments (p. 22). For example, diegetic operator acts occur when players direct player-characters to run, jump, fly, or drive through environments. "Nondiegetic operator acts" (p. 2), on the other hand, referred to actions taken by the player that do not directly affect the diegetic game world but still affect the game's state in some way, such as configuring the game's settings or equipping armor to a character through an abstracted menu interface. Galloway admits that the line between diegetic and non-diegetic acts is blurry (equipping armor in a menu affects the character within the diegetic game world after all, does it not?), and I will explore this precarious separation, but what is important here is the interface that is used to perform these actions. When players execute nondiegetic operator acts, they do not navigate a game world; instead, they navigate menus. Galloway argued that "to live today is to know how to use menus" (p. 17), and this is especially true in videogames. Despite the lion's share of critical focus being given to game worlds,

Figure 1: *The Legend of Zelda* Pause Menu

Screenshot of *The Legend of Zelda* pause menu

to play videogames is to use menus.

Pausing games is often discussed as a way to freeze the game world to one specific game state. Christopher Hanson (2018), for instance, maintained that "pausing a game allows players to temporarily suspend play, effectively interrupting the game's flow of time" (p. 58). This is not entirely true all of the time. Pausing may suspend some of the computer's actions that are associated with the game world, but it does not necessarily suspend play. Pause menus allow the player to take any number of actions from saving their game to customizing their character. The navigation of these menus and the configuration that they allow is central to play. Galloway (2006) went as far as claiming that the actions taken within menus "are often the very essence of the operator's experience of gameplay" (p. 14). At the same time, while the actions taken within the menu are important aspects of videogame play, it is

undeniable that accessing and navigating pause menus produces a drastically different experience of play than navigating the spaces of game worlds.

In this essay I explore how both the act of pausing and navigating pause menus offer perceptions of temporality and space that situate videogame play as not simply explorations of game spaces; they reveal computational modes of time and space that are always present in all aspects of videogame play. To uncover these temporalities I will begin with a review of literature on time in videogames that establishes some of videogame's and videogame player's temporal frames: play time, diegetic time, and computational time. I then move to a close reading of two pause menus that offer different understandings of temporality based on players' perceptions of the centrality to play of the specific pause menus. I end with discussions of ways in which speedrunners use pausing and pause menus to subvert standard notions of play, and in so doing completely reorient assumptions surrounding our attunement to computational temporalities and our perceptions about the construction of (and our interactions with) game worlds as diegetic spaces of play. Introducing the practice of speedrunning demonstrates the ways that typical conceptions of temporality and space in videogames are incomplete, and fail to grasp all of the technical and experiential dimensions of gameplay by subverting designed notions of videogame play.

Parallel Temporal Frames

Time in videogames is never straightforward. Foundational game theorists such as Johan Huizinga (1950) and Roger Caillois (1961) have noted that games allow players to experience time differently than they do in their daily lives. Juul (2005) picked up on this thread when he delineates "play time," the time it takes to play the game, from "fictional time," the time that flows within the diegesis of the game (p. 141). In some games, there is a one to one correspondence between these temporal spaces. Juul (2005) argued that action games tend to match temporalities, as when the player inputs a command, say to fire a gun, the action almost immediately happens within the game world (p. 142–143). However, this is not always the case. He noted that a year in the fictional time of *Sim City 4* (Maxis, 2003), for example, only takes two minutes in play time (Juul, 2005, p. 143). Fictional time flows much more quickly than play time in this case.

Juul's (2005) categorizations of time in videogames is far from the only one, but it has shown to be wildly influential, forming a basis for many other further analyses (ex. Hitchens, 2006; Nitsche 2007; Wei, Bizzocchi, & Calvert, 2010). These studies follow in the tradition of Gérard Genette's (1979) classic narratology that splits a narrative into two levels: story and discourse. Story refers to the telling of events in the order they are described, which is analogous to play time, and discourse refers to the diegetic ordering of events, which is analogous to fictional time. Other scholars do not draw directly from this tradition, but still root their analysis of time primarily in the temporal frames afforded by a game's diegesis (ex. Nikolchina, 2017; Jayemanne, 2019). While these temporalities are clearly at play in videog-

ames, they are far from the only temporalities present, and privileging diegetic and experiential temporalities as the basis of understanding videogame time ignores important ways that videogames fundamentally operate and how players interact with the videogame's structure.

It is difficult, perhaps impossible, to develop an all encompassing theory of videogame temporality that accounts for all temporalities present in every type of videogame. To this end, José Zagal and Michael Mateas (2010) helpfully introduce the concept of "temporal frames." Zagal and Matteas (2010) envisioned temporal frames as a generalizable concept that is malleable enough to fit any type of temporality videogames may present. They described temporal frames as supporting "the ability to define many possible frames as needed to perform an analysis of the temporal phenomena in a given game and also allows one to see the structure that is shared across all temporal frames (i.e., why the domain of game-world events and the domain of real-world events each constitute a flow of time)" (Zagal & Mateas, 2010, p. 845). As concrete examples, Zagal and Mateas (2010) introduced four frames: "real-world time, game-world time, coordination time, and fictive time" (p. 848). While they introduced these as only a few frames among many, they made a point of excluding certain temporalities from consideration. "Videogames," they noted, "execute on a computational infrastructure, in which the fundamental state changes are the billions of computational state changes happening per second." (Zagal & Mateas, 2010, p. 846). Because these state changes happen outside of the direct perception of players, they concluded that an analysis of these temporalities "would fail to provide a description of game time relevant to players and designers" (Zagal & Mateas, 2010, p. 847).

This, as Mark Hansen (2015) pointed out, is the defining feature of new computational media: they "operate predominantly, if not entirely, outside the scope of human modes of awareness (consciousness, attention, sense perception, etc.)" (p. 5). Shane Denson and Andreas Jahn-Sudmann (2013) described this as a "blindness to computational temporality" (p. 15). We do not directly see the temporalities at play. This does not mean that we cease to engage with computers in sensory ways, but we need to remember that any on-screen or auditory output is a result of extremely rapid micro-processes that are well below any human's perceptual thresholds. Even if players do not directly perceive these minute temporalities, the computer's temporal frames are vitally important to gameplay.

But what exactly are the computer's temporal frames? Perhaps this is best exemplified with an excerpt from Patrick LeMeiux's (2014) account of a Nintendo Entertainment System, specifically an NES-4021, receiving a lone signal from a controller. He began:

> Press "START." Sixty times a second an electrical impulse is sent from the Nintendo Entertainment System (NES) 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. After receiving a high pulse for 12 microseconds 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 (i.e., 0V), is interpreted as a 0 by the NES's central processing unit (CPU)—a modified version of MOS Technology's popular 6502 processor called the Ricoh 2A03. A "high" current pulse (i.e., +5V) is registered as a 1.

The events that this lengthy account describes happen in less than a sixtieth of a second. Whenever videogames operate these kinds of imperceptible micro-actions occur countless times. These temporalities are what scholars interested in areas like code studies (ex. Hayles, 2004; Sample, 2006; Kirschenbaum, 2008) and platform studies (ex. Montfort & Bogost, 2009; Altice, 2015; Arsenault, 2017) tend to be most concerned with. The flows of electrical pulses coursing through gaming systems come to represent individual bits, 1s and 0s, that form the abstract code governing videogames. Changing bits lead to computational state changes happening on the order of billions of times per second. These discrete, rapid state changes characterize the computer's temporal frames.

Before delving into the ways that these unperceived temporalities come to matter to players through the pause menu, in the next section I will first explore some of the diegetic and experiential temporal frames typically associated with pause menus. These temporal frames, I attempt to show, have largely been ignored or undertheorized because of the ways pause menus have been read as paratextual aspects of videogame play.

Temporalities of the Pause Menu

I have described pause menus as menus that, when accessed, suspend most of the machine's actions that directly affect changes in the game world. When the game world's temporalities are paused we tend to think that the entirety of the game stops. For example, Christopher Hanson (2018) argued that play time, as well as fictional time, ceases when the game world is paused. He said: "Pausing virtually any game disrupts and undermines a flow state by interrupting play and pulling the player out of the game" (2018, p. 72). This implies that the player is taken completely out of the framing of the game when the game world is paused so that they are no longer playing. The idea that the game is completely put in suspension when the pause menu is accessed inherently privileges the videogame as, first and foremost, a game world. The term "pause menu," however, can be a misnomer as many temporalities do not necessarily stop when the "game" is paused.

Hanson's (2018) claim that pause menus cease play requires some clarification as to what constitutes "the game" for players and scholars alike. Mia Consalvo (2007) borrows and applies to videogames another concept from Gennette: paratext (p. 8). Drawing on Peter Lunenfeld (1999), Conslavo (2007) described paratexts as any text that surrounds the central text of a videogame focusing on aspects like advertising, guides, and reviews as examples of paratexts, but she also located cheat codes as another major instance. Cheat codes are an interesting example because they are a part of the game's code, but players often relegate them

to being outside of the normal experience of play—as Consalvo (2007) put it, cheat codes are not the core of the experience of play. They do not constitute the central, normal, or standard form of play, instead becoming an "addendum" to gameplay (p. 30). Peter Lunenfeld (1999) argues that the distinction between text and paratext can often become difficult, and perhaps impossible, to determine in regards to digital media (p. 19), and I agree. The pause menu, like cheat codes, tends to be an element of the game's code that is treated as a paratext by some players and scholars alike.

Take for example Mark Wolf (2001) who described time spent in pause menus as "interludes," "breathers," and "moments in which the game's interactive potential is briefly suspended" (p. 83). Samuel Tobin (2013) picked up on this line of thinking by placing pause menus into a broader category of "non-diegetic moments" that also includes load screens and item inventories. These moments, according to Tobin (2013), "allow the player to sit back, stretch, and otherwise withdraw from the game posture and from the demands of time and speed" (p. 136). Here we see pause menus placed outside of not only the diegesis of the game world, but also outside of the temporalities of gameplay completely. I do not deny that time spent in pause menus *can* facilitate these breaks, but it is not the only mode of temporal interaction players have with pause menus. What I hope is clear with my use of the term paratext here is *not* that the pause menu is placed by scholars outside of any meaningful interactions with a game completely, rather that the pause menu is often situated paratextualy to the central experience of gameplay, which is typically located in the space of game worlds.

It could be helpful to compare an example of pause menus that many players treat in ways similar to those Wolf and Tobin describe to one that facilitates play more directly. Action game pause menus tend to offer few options that players would consider ludic or central to gameplay at all. Much of the pleasure of playing these games comes from maneuvering through environments and the challenges that lie within them, but pausing takes the player to an alternate, flat space that forces them to navigate hierarchical lists instead of diegetic worlds. Instead of maneuvering a character through territories filled with puzzles or enemies, they find themselves maneuvering a cursor while the game world is suspended. The menus, of course, serve many purposes, but those functions are not central to the core elements of gameplay that involve actions taken within environments, so pausing can be viewed as paratextual.

One representative example of these kinds of action games is *Call of Duty: Modern Warfare 3*'s (Infinity Ward & Sledgehammer Games, 2011) campaign mode (Fig. 2.). The *Call of Duty* series of game's campaign modes are first-person shooters that put the player in the position of a soldier in a variety of realistically rendered settings tasked with any number of missions that can involve stealth, chase sequences, and, of course, fire fights. When paused, the entirety of *Modern Warfare 3*'s game world is suspended, including any computational processes associated with those spaces. Very few of the operations or information presented within the menu affect the game world directly apart from some information about the play-

Figure 2: *Call of Duty: Modern Warfare 3*'s Campaign Mode Pause Menu

Screenshot of *Call of Duty: Modern Warfare 3's* Campaign Mode Pause Menu

er's current objectives and location within the diegesis of the game world. The player is able to change technical parameters such as controller sensitivity and screen resolution within the "options" submenu, which falls outside of any diegetic or ludic frames, but most of the functionality within this pause menu is related to manipulating fictional time. The player can resume from where they paused the game state (resume game), revert the game state to one of two previous states (last checkpoint and restart mission), or suspend the game at its current state indefinitely (save and quit). Instead of a fluid flow of time, the pause menu encourages the player to take control of fictional time revealing time in videogames to be halting, circular, and recursive. These kinds of alternative temporalities allow the player to manipulate time in ways that undermine "chrononormativity," a term Matt Knutson (2018) borrows from Elizabeth Freeman (2010) which describes socially constructed, standardized ways of being in time that Knutson argues videogames often undermine. These alternative temporalities give the player freedom from rigid temporalities through the control they give over time, which may be a productive disruption of Mihaly Csikszentmihalyi's (1990) classic conceptions of flow that Hanson (2018) invoked. But, even if these manipulations of time are geared at controlling the game state within the game world, they are not framed as being diegetic nor ludic. The language of the menu itself emphasizes the menu's position as outside of gameplay by labeling the option to continue as "resume game." The game world is always privileged as the cite that gameplay happens, while the pause menu is a paratextual unsettling of the game's fictional temporalities.

However, this is not always the case. In many games the pause menu is integral to play. It does not remove the player from the game; instead, it facilitates different types of play integrating itself within the text. This is especially prevalent in genres of role playing games (RPGs). A representative example of these menus comes from *The Legend of Zelda: Breath of the Wild* (Nintendo EPD, 2017). Early in the game the protagonist, Link, emerges from a cave onto an overlook. A cutscene plays wherein Link runs to the edge of a cliff and looks out onto the vast land of Hyrule. *Breath of the Wild* features an enormous world to explore, and the vista that Link, and the player, are met with introduces them to just how large the spatial dimensions of the game world are, preparing the player for a particular kind of spatial and temporal engagement with the game. In the distance the sun rises, and, as the game reverts to player control, a clock appears at the bottom right of the screen. This sequence tells the player that they will travel across the sprawling land of Hyrule, day and night, to uncover the secrets hidden in the distant peaks and castles that seem so far out of their reach at that moment. However, as the player guides Link down from the perch, they run across tree branches lying on the ground. Link can pick up the tree branches by pressing "A" on the controller, and the first one will automatically be equipped as a weapon. As he continues descending Link comes across more tree branches, a torch, and an axe, all of which can be equipped as weapons, but only one item may be equipped at a time. To swap weapons, the player must enter a pause menu (Fig. 3). In the first minutes of a game seemingly dedicated to exploring an immense game world over days and weeks of diegetic time, the player is prompted to stop the game's diegetic time to access a pause menu.

However, unlike the *Call of Duty* title, it is difficult to say that the player is taken out of gameplay when they access these menus. When any item, including weapons, are obtained within the game world they can be found in an inventory slot under one of seven tabs within the pause menu. There are limited slots for every category of equipment (weapon, bow, and shield), so the management of those items within the menu is integral to gameplay strategies meaning play time still flows while the game world is paused. Also, actions taken within the pause menu directly affect the game's diegesis. For instance, swapping weapons and armor or eating food affects the diegetic character. This is emphasized by the representation of Link that appears in the menu which is animated based on the actions the player selects within the menu. Armor and equipment will appear on his body or he will scarf down apples, for example, and when the pause menu is exited Link will accordingly be wearing different armor or have more health. This is not simply a break from gameplay; strategic and diegetic actions are taken in this space even if diegetic time has ceased.

However, there is more to the temporalities of a pause state. These accounts so far solely privilege diegetic and experiential temporal frames. In the next section I will complicate these by introducing how computational temporal frames interact with the other layers of temporality through an analysis of Alexander Galloway's (2006) descriptions of time in *Shenmue* (Sega AM2, 1999) that casts the temporalities of pause menus like the one found in *Breath of the Wild* in a slightly different light.

Figure 3: *The Legend of Zelda: Breath of the Wild*'s pause menu

Screenshot of *The Legend of Zelda: Breath of the Wild*'s pause menu.

Hidden Temporalities

To demonstrate why the computational level, or what Craig Lindley (2005) called the "generative substrate," is important to take into consideration when discussing temporality I will briefly investigate Galloway's (2006) description of placing the game *Shenmue* into what he dubbed an "ambience state" (p. 10). An ambience state occurs when the player refrains from entering any inputs causing the game to idle. Galloway illustrated *Shenmue*'s ambience state through the following: "No stopwatch runs down. No scores are lost…It rains. The sun goes down, then it comes up. Trees stir" (p. 10). He focuses on two things here: the audio-visuals of the game world (rain, sun, and trees) and markers of progress within the ludic structure of the game (clocks and scores). Both of these elements are predicated on outputs that the player can perceive. Players see the representation of trees moving and, conversely, do not see any timers ticking down. Based on these observations, Galloway concluded that "things continue to change in an ambience act, but nothing changes that is of any importance" (p. 10), and this would seem to be true if we ignore the computer's processes in favor of what we perceive of the game world.

The claim that nothing happens of any importance is a dubious one for Galloway's particular example. First, fictional time still passes while in an ambience state, and events within the

game world of *Shenmue* are tied to particular fictional times. Waiting can lead to missed opportunities, or it can be used productively to trigger certain game states. Additionally, while the game never indicates it directly to the player, there is actually a time limit to complete the game. If the player takes too many in-game days to complete the game's storyline a cutscene triggers where the protagonist is attacked, signaling that the player is out of time and cannot continue. This is effectively a game-over. Players are given 137 diegetic in-game days, which equals 1 day, 10 hours, and 15 minutes in play time. So, while there is no stopwatch ticking down that the player directly perceives, the computer still keeps track of time and triggers particular game-states in a supposed ambience state. Of note, this time continues to flow even when the game is supposedly paused. The main character Ryo's Timex can always be checked, on the street or in the menu, and it is always ticking so long as the game system is powered on and the game world is loaded.

But there are processes that are even more opaque to players. For example, the gambling mini-game "Big or Small'' can be played at multiple places within the game world. The player-character is prompted to place a bet with fictional currency on the result of a dice roll. The roll seems to be completely random, but computer programs are deterministic and incapable of producing truly random outputs. The output is governed by what is colloquially called a random number generator (RNG), but is more accurately a pseudo-random number generator (PRNG). PRNGs generate a sequence of numbers through an algorithm that uses an arbitrary starting state, called a seed. There are multiple methods for setting a seed, such as assigning a number based on the system's real-time clock or on how long it takes the player to select the option to load a game file. When the player initiates the Big or Small mini-game the PRNG is queried and the number in the PRNG's sequence at that particular moment determines the result of the dice roll. If the same seed is used, then the same sequence of numbers is generated, meaning that a player with an understanding of the minutiae of the PRNG's mechanics can manipulate the seemingly random occurrences. Waiting, then, can be an important strategy to advance the PRNG's number to receive a desired result. It is important to note that the PRNG cycles through its sequence extremely rapidly and without any direct audio-visual indication that it is happening at all. This makes it nearly impossible for players to manipulate the PRNG, but it does mean important actions that directly impact the game world happen while the player-character stands still. The dice are not nearly the only facet of the game governed by the hidden PRNG either. Certain characters' dialogue choices, character's walking paths, and the results of other mini-games are, in part, influenced by the PRNG. It is an essential component of the world of *Shenmue*, even if it is never represented on the screen.

This is not to say that there is no purchase to the concept of ambience states, nor that they do not exist within videogames. What I am drawing attention to is the need to understand the computational temporalities of videogames in tandem with representational and experienced temporalities. A model of time in games that exclusively privileges the differences between play time and fictional time while conceiving of the videogame solely as a fictional

game world ignores the important aspect of the computer's time, which is essential to the operation of videogames.

Let's return to the example of *Breath of the Wild*'s pause menu to see how our understanding of temporalities is altered when we consider the computer's temporalities alongside other temporal frames. From the perspective of the game world no time passes between when the player accesses and exits the pause menu. From players' subjective temporal frames they may spend seconds, minutes, or even hours in the pause menu, but from the vantage point of the frozen diegetic temporal frames, players match the rapidity of computational temporal frames while in the pause menu. Because of this freezing of diegetic time, the player is able to input as many commands as they would like in a micro-temporal gap in the fictional flow of time. The pause menu, then, allows players to attune themselves to the machine's temporalities by entering commands while fictional time is paused.

Of course, actions taken within the menu are vastly different in nature from the actions that can be taken within the game world. *Breath of the Wild's* pause menu is prominently used for configurations of inventories and using items that provide beneficial effects to the hero, while the game world allows for actions like climbing, running, and combat which cannot normally be inputted with the rapidity of the computer's temporalities, but, nonetheless, the player is effectively able to perform gamic actions at speeds that approximate a computer's temporalities through the use of menuing.

This example assumes a player interacting with games in more or less conventional ways, which is not always the case. Players do not solely play in prescribed ways intended by developers, which Stephanie Boluk and Patrick LeMieux (2018) called a "standard metagame" (p. 36). Boluk and LeMieux (2018) defined metagame "as a truly broad label for the contextual, site-specific, and historical attributes of human (and nonhuman) play (p. 17). In the next sections I will be focusing on player practices that subvert standard metagames: speedrunning. An analysis of the "contextual" and "site-specific" ways speedrunners interact with pause menus presents an alternative picture of the ways players become attuned to computational temporalities.

Speedrunning the Paused Game

Despite our inability to directly perceive some temporalities, we still become attuned to them in various ways. I use the term "attunement" in the same sense as James Ash (2013). Ash (2013) described attunements as an internalized connection to a system or situation that has "a culpable effect on the body of users, inscribing and generating new associations in thought and action." (p. 35). He analysed the ways players become attuned to elements of *Call of Duty 4*'s (Infinity Ward, 2007) multiplayer game modes (Ash, 2013). Various aspects of play become unconsciously inscribed in player's actions such as thumbstick sensitivity, weapon reload times, and map sight-lines. Player's become connected to the game's minute

affordances through their repeated interactions with it. I argue that players become attuned to the computer's temporalities in much the same way, which is revealed through player's use of the pause menu.

Speedrunners employ a technique called "pause buffering" to directly attune themselves to the computer's temporalities. Speedrunners are players that achieve arbitrary goals within games as quickly as they can, which usually revolves around completing games as fast as possible. Time flows differently for speedrunners than it does for most players. Fictional time is largely unimportant, while play time is paramount and constantly flows no matter what. Ultimately, speedrunners attempt to minimize play time as much as possible, but to do so most effectively they must often exploit glitches and bugs which require extremely precise inputs in order to match the computer's temporalities.

Speedrunners are uniquely aware of the computer's temporalities. LeMeiux (2014) uses his above account of the Nintendo Entertainment System's receiving of inputs to illustrate some of the ways a specific subset of speedrunners, tool-assisted speedrunners, are aware of and manipulate the minute processes of the game's hardware and software. The extremely fast exchanges of electricity through the NES's wires affect players that push a game to its limits when it comes to speed. Even the laws of physics in the form of travel speeds of signals in a local system are important bottlenecks that cap players' possible inputs. James Newman (2019), in his analysis of a speedrun of *The Legend of Zelda: Ocarina of Time*, similarly noted that techniques used by speedrunners "are contingent on techniques and opportunities made available through the creative exploitation of the materiality of the code and whose trajectories emerge from errors, inconsistencies, and quirks in the ways data are stored and flow through the system" (p. 10). The minute idiosyncrasies in the ways the system processes information is vitally important to speedrunners, and both the virtuosic inputs and encyclopedic knowledge of the games make speedrunning what Rainforest Scully-Blaker (2014) called a "Practiced Practice."

A part of this practice is a technique that allows players to slow the flow of the computer's time so they can enter commands at specific points during the computer's operations called pause buffering. This is common in speedruns for many games, including *Breath of the Wild*, and a myriad of other titles such as *Super Mario 64* (Nintendo EAD, 1996), *Grand Theft Auto: San Andreas* (Rockstar North, 2004), *Super Monkey Ball* (Amusement Vision, 2001), *Mario Kart Wii* (Nintendo EAD, 2008), and *Resident Evil 5* (Capcom, 2009), among many others. Pause buffering works slightly differently in every game, but the idea behind the technique is consistent. The pause menu is rapidly accessed and exited allowing diegetic temporalities to lurch forward incrementally. This means that the computational temporalities advance in minute increments as well, which players are able to take advantage of. Inputs are entered into those small gaps while the pause menu is not active, granting the player a high level of precision of inputs in relation to the computer's temporalities.

The technique is perhaps most famous for the exploits it makes possible in Newman's (2019) object of study: *The Legend of Zelda: Ocarina of Time* (Nintendo EAD, 1998) for the Nintendo 64. When *Ocarina of Time*'s pause menu is exited to return to the game world there are a few "lag frames." A frame is the individual generation of the visuals on the screen by the computer, and lag refers to a delay between the player's inputs and the computer's ability to register them. So, a lag frame is a very short window where inputs cannot be registered by the computer. The North American version of *Ocarina of Time* runs at an average of about twenty frames per second meaning every frame lasts five-tenths of a second. The amount of lag frames when exiting the pause menu varies depending on a variety of factors, but even a couple of lag frames is long enough for players to enter a command during the lag frames. The computer does not register commands until diegetic time continues to flow, so if the player holds the input through the lag frames then that input is initialized at the first possible frame after the lag frames. Through this method speedrunners can rapidly enter and exit the pause menu to input commands at specific frames, which makes difficult exploits that require precise and rapid inputs much easier to execute. Pause buffering can be used to make many highly difficult exploits possible, which effectively allows the main character, Link, to fly through the air (bomb hovering), slide clear across the map at extreme speed (superslide), or even walk right through solid walls (door of time skip).

The "megaflip," in particular, is a useful exploit that is relatively simple to explain, and, while it does not require nearly as much temporal precision as other tricks in the game, it illustrates the power of pause buffering nicely.[1] A megaflip is used to make Link jump farther than he usually is able to, allowing large gaps to be crossed, and giving access to normally inaccessible areas. To perform this trick the player must first use a bomb, roll into the ensuing explosion while holding a shield, and then backflip out of the blast, receiving a boost in momentum. However, bombs have a fuse, so the timing of when to initiate the roll into the bomb to match its explosion is fairly precise. There are exactly five frames, or a quarter of a second window, that the roll can be initiated that will result in a successful megaflip instead of simply receiving damage. To attune themselves to this small temporal gap, players pause buffer the bomb's fuse. Once the fuse gets close to its explosion, players can begin to cycle in and out of the menu, attentive to the ways diegetic time minutely increments every cycle. For this trick in particular, players are cued in to the appearance of the bomb itself. An online guide to performing the trick describes the visual cues:

> "The five frames in order are as follows:
> 1. Small blue bomb #1
> 2. Small blue bomb #2
> 3. Small red bomb
> 4. Big blue bomb
> 5. Big red bomb (then it explodes on the next frame)" ("Megaflip")

Rolling during any of these rapid, barely perceptible, frames will result in a successful megaflip.

The pause menu allows players to play the game frame to frame. Speedrunners, in effect, use the pause menu to slow machine time enough so that play time can be attuned to the computer's micro-temporalities. Pause menus, then, can extend the micro-temporal moment to allow players to insert commands at the computer's temporalities in both the game world and pause menu. Of course, the pause menu is not essential to performing these tricks. However, once players are able to attune themselves to the computer's temporalities through tactics that slow the computer's temporalities, they are then able to become more aware of the ways the computer's temporalities work, and eventually speed their own actions up to be in line with the computer's, even without the buffer. And this is exactly what happens in speedrunning. Players initially rely on pause buffering, but pause buffering extends real time, which is not conducive to the fastest speedrun. Pause buffering even extremely precise tricks is used less and less by speedrunners as they become more experienced with the game, demonstrating the ways in which they become attuned to the computer's temporal frames through the use of a pause menu.

Interestingly, when played frame by frame, the game's temporalities become a kind of menu. Temporality, itself, is treated as discrete options to be selected from. When megaflipping in *Ocarina of Time*, during every frame there are a few options that can be selected: Either wait and advance to the next frame, or initiate roll, which will lead to a successful or unsuccessful megaflip depending on the particular frame chosen. Pause buffering brings these kinds of branching selections of actions at certain diegetic and computational moments to the fore because the temporalities become segmented into discrete units, but this is how we interact with videogames at all times: actions are selected at specific moments and result in an output based on the temporal state of the game. The more expert a player becomes in a game the clearer they can see the various branching menu hierarchies behind the temporalities. It's not that pause buffering causes this segmentation, it just reveals it more clearly.

Spaces and Databases

Another dimension to the pause menu that affects temporalities has been ignored thus far in this analysis. There is spatiality to pause menus; we travel through them. Lev Manovich (2001) noticed that we use the term "navigate" when we interface with menu heavy platforms like the internet or IRC channels (p. 272-273), and the same could be said for videogames. Navigating these spatial dimensions, just like navigating a player-character through a game world, takes time to traverse. This navigation can be central to play, especially for speedrunners. However, menus typically are not spatial in the same ways that game worlds are spatial. There is often no depth to menus, nor are there characters that interact with diegetic worlds like the immense lands we see in *Breath of the Wild* where Link forges his way across the map battling enemies and scaling mountains. Instead, in menus cursors navigate flat space and select options from lists, but we will see that on a foundational level there is not much difference between the ways menus and game worlds operate because of the computer's use of databases.

The lists and hierarchies inherent to menus can be thought of as a database. In the context of digital media, Manovich (2001) defined databases as "a collection of items on which the user can perform various operations—view, navigate, search" (p. 219). This is a particularly apt description of the menus we have seen already, especially *Breath of the Wild's* collections of equipment that can be equipped, de-equipped, tossed, and sorted. Again, we see spatial metaphors here in regard to the ways players interact with databases (navigate, search), just as we did with menus, but Manovich (2001) argues that videogames are not primarily experienced as databases. Instead they are the territory of algorithms, which, in turn, implies action. We don't always notice this as, according to Manovich (2001), it is the "narrative shell of a game" that "masks" the game's "simple algorithms" (p. 222). This is true to an extent, but the database logic of the pause menu reveals videogames as not only action oriented algorithms, but also as navigable databases of information.

On a fundamental level, videogames are made of databases. To illustrate this let's look at how Nintendo's Gameboy stores information.[2] The Gameboy has a 16 bit address space, which means that its CPU can access up to 65,536 unique bytes of data, each of which is mapped in memory to its own hexadecimal value. Every range of hexadecimal values is used for a different function. For example, addresses $8000 through $9FFF are mapped to video RAM, which contains memory pertinent to the visuals displayed on the screen, while addresses $A000 through $BFFF are dedicated to external RAM, or memory located on the game cartridge. Fig. 4 demonstrates how the memory map looks when laid out visually. There is a spatiality to the map as the different addresses and areas of memory are stacked on top of one another through the system's abstracted code.

This spatiality affects the ways that all games access and present information, but we can see it more clearly through certain gameplay practices. An interesting exploit, again found and used by speedrunners, exists in *Super Mario Land 2: 6 Golden Coins* (Nintendo R&D1, 1992) that demonstrates the database logic of the Gameboy's memory. Half the external RAM (ExRAM) and the entirety of the working RAM (WRAM and WRAM Echo) are used to store twelve-kilobytes of level data, or the space that Mario navigates. Using an exploit called the "Pipe Glitch" Mario can exit the intended area of the level data to fall out of the working RAM and into the other areas of memory. When he lands in these other areas, it looks like a glitchy mess of tiles, but is actually a representation of the game's memory data (Fig. 5). Every byte of data within the areas that Mario is intended to navigate corresponds to a different visual tile within the level, but because types of tiles correspond to particular values, every byte of memory, whether it was intended to be viewed on screen or not, can be interpreted as level data. Interestingly, every tile retains some properties from the level data. There are pipes, bricks, coins, etc., although the visuals do not always correspond to their correct behavior. Nevertheless, Mario can still interact with the game's memory through these tiles. Picking up a tile that acts as a coin or breaking a tile that acts as a brick, for example, changes the value of that byte of information to a different value because the tile's state has changed. Doing so effectively rewrites the code of the game. Of note to speedrunners, nav-

igating Mario to hex address $A2D5 and breaking the block that sits there will immediately trigger the game's end credits, potentially ending the game in only minutes. The memory map essentially becomes a navigable, if cumbersome, menu that allows the player to select specific options from a large list of information. It just so happens that the list in this case is

Figure 4: Diagram of Gameboy's memory map

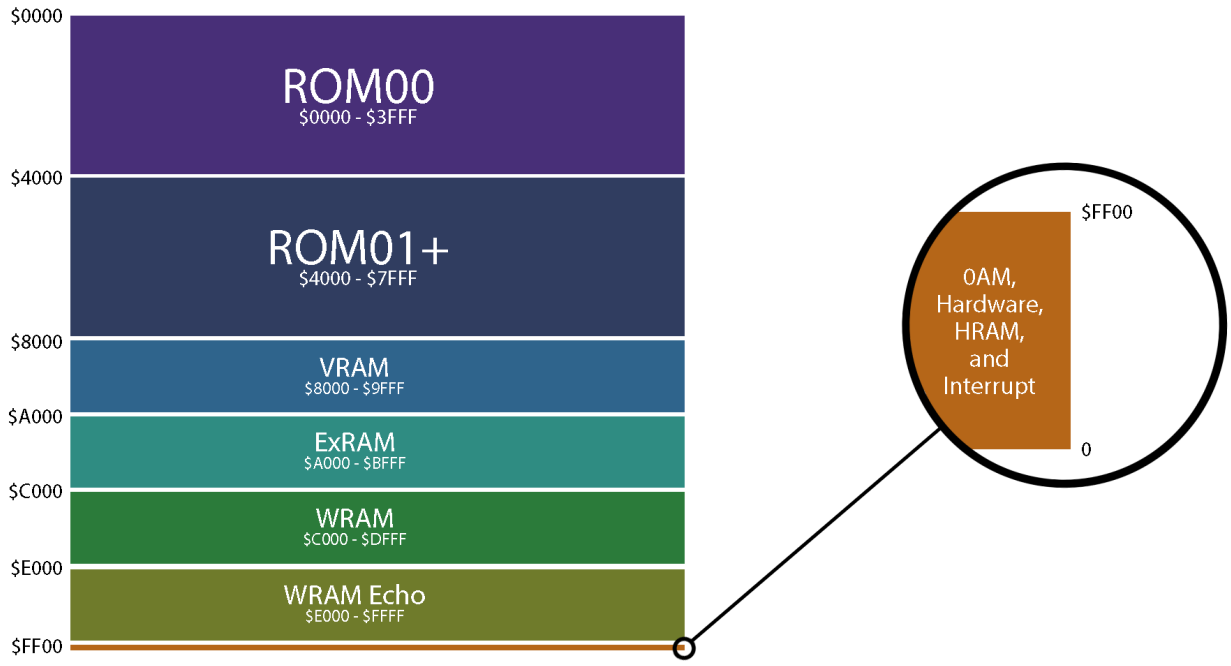


Diagram of Gameboy's memory map

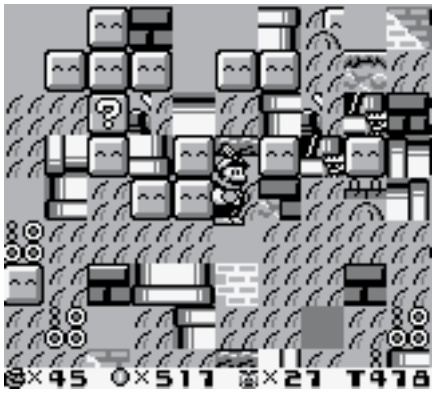
constituted by the database of the game's memory map interpreted through the logics of the game's level data.

The diegetic space of videogames, then, really is a mask, as Manovich (2001) claimed, but it does not only mask the game's algorithms. Videogames are not three dimensional game spaces, at least not literally. They are databases that are accessed through complex menus. Another salient example comes from speedrunners of *Pokémon: Yellow* (Game Freak, 1998) who exploit various glitches to change where the player-character travels when they exit the house the game starts in.[3] Instead of stepping outside into Pallet Town ready to begin their Pokémon adventure, the player-character finds themself walking into the Hall of Fame, the final room of the game where the credits are initialized. Pallet Town is not, in a literal sense, spatially outside of the player-character's house. Every possible location in the game simply has an address assigned to it and every door is programmed to send the player-character to a particular address. Changing that address through exploits breaks the spatial continuity of the game world and reveals the game world to be constructed from a database of information, while also revealing the game world to be a menu that allows the player to navigate that

database.

These examples come from games on the relatively technologically simple Gameboy with its somewhat limited storage capabilities. This makes it more manageable to demonstrate how the games draw information from the database and represents it on the screen. Also, Gameboy games were built in low level assembly languages, so there is a fairly direct connection between the code and the materiality of cartridges and consoles allowing us to look directly to where every piece of coded information is located in a nice, neat spreadsheet. More recent games are often built on large, powerful game engines like Unreal Engine or Unity, but even if they do not use popular engines they still can become elaborate and massive entan-

Figure 5: Exploring the Memory of *Super Mario Land 2*



Exploring the memory of *Super Mario Land 2*

glements of software and hardware. It can be more difficult to understand the complicated processes happening behind the screen, although some players (particularly speedrunners) are attuned to these processes despite their complexity. However, even if we do not understand the specific ways games access and store information in databases we do understand that our actions produce specific on screen results that are drawn from somewhere. When we, say, notch an arrow in *Breath of the Wild*, we have selected particular computational functions from the storage databases that make up the game. In fact, the controller, itself, could be thought of as a menu of buttons that presents context specific options within the game that correlate to certain values in memory at specific times. Every action we make, whether in a menu proper or game world, fundamentally becomes a selection from the game's memory in some capacity.

Pause menus have their own spatial dimensions and logics regarding navigation and functionality compared to what I have been calling game worlds, but, at their core, both game worlds and menus are interfaces that allow the player to manipulate the database of the game's memory. The space of the pause menu, and the many other menus that are so common in video games, forefront the underlying processes of the computer, whereas the

diegetic game world tends to mask it. This is not always the case. For instance, Kristine Jørgensen (2013) argues in *Gameworld Interfaces* that the game world itself can become an information rich environment that gives players access to the underlying functions of the game's systems; however, in general, diegetic game worlds tend to emulate real-world logics such as three dimensional space and realistic physics that obscure the computational nature of games. Although it is not completely disconnected from the temporal and spatial dimensions of the game world, the pause menu's lists of information becomes a space of play that demonstrates the ways the computer conceives of space as a database.

Conclusion

The pause menu both attunes players to the temporalities of the machine while also privileging the ways that computers spatially store information in databases. Organizing equipment in *Breath of the Wild's* pause menu, for example, may use a different interface than battling an enemy in the game world and the temporal-spatial dimensions may appear wildly different, but, from the computer's perspective, both actions are a process of accessing particular pieces of memory stored in databases based on the player's inputs coupled with various computational processes. The pause menu, with its tables, statistics, and hierarchies, demonstrates this clearly to the player, but it is always present in every facet of play making the pause menu a kind of window into the computer's temporality and spatiality. When we look into the micro-temporal operations that videogames undergo to create game worlds, which are revealed through alternate metagames like the ones speedrunners engage in, we find that videogames need not simply produce experiences for players through their diegetic levels, which they undoubtedly do. Other spaces of play, like the pause menu, reveal a fascinating account of time and space as well.Last section of the body text goes here. .5 in spacing again.

Endnotes

1. For an example of pause buffering see Youtube user 117rinari's video titled "Zelda Ocarina of Time - Pause Buffered Megaflip": https://www.youtube.com/watch?v=dm6fCQgi-KE

2. Valuable information regarding the Gameboy's memory and *Super Mario Land 2: 6 Golden Coins* exploits is taken from Youtube user Retro Game Mechanics Explained's Youtube video "Super Mario Land 2 - Memory Exploration": https://www.youtube.com/watch?v=FPzuYWb-nln4

3. For an example of *Pokémon: Yellow* (Game Freak, 1998) exploits see Youtube user Nerfmaster119's video titled "[TAS] Pokemon Yellow Credits Warp Any% In 2:14.21 By Nerfmaster119": https://www.youtube.com/watch?v=fSy3ryTSQIQ

References

Altice, N. (2015). *I am error: The Nintendo Family Computer/ Entertainment System platform*. Cambridge: The MIT Press.

Amusement Vision. (2001). *Super monkey ball*. Gamecube: Sega.

Arsenault, D. (2017). *Super power, spoony bards, and silverware: The Super Nintendo Entertainment System*. Cambridge: The MIT Press.

Ash, J. (2013). Technologies of captivation: Videogames and the attunement of affect. *Body and Society, 19*(1).

Bethesda Game Studios. (2011). *The elder scrolls V: Skyrim*. PC, PlayStation 3, and Xbox 360: Bethesda Softworks.

BioWare. (2007). *Mass effect*. PC, Playstation 3, and Xbox 360: Microsoft Game Studios

Blizzard Entertainment. (2004). *World of warcraft*. PC: Blizzard Entertainment.

Boluk, S. & LeMieux, P. (2018). *Metagaming: Playing, competing, spectating, cheating, trading, making, and breaking videogames*. Minneapolis: University of Minnesota Press.

Caillois, R. (1961). *Man, play, and games*. Chicago: University of Illinois Press.

Capcom. (2009). *Resident evil 5*. PlayStation 3 and Xbox 360: Capcom

CD Projekt Red. (2015). *The witcher 3: Wild hunt*. PC, PlayStation 4, and Xbox One: CD Projekt.

Consalvo, M. (2007). *Cheating: Gaining advantage in videogames*. Cambridge: The MIT Press.

Csikszentmihalyi, M. (1990). *Flow: The psychology of optimal experience*. New York: Harper & Row.

Denson, S. & Jahn-Sudmann, A. (2013). Digital seriality: On the serial aesthetics and practice of digital games. *Eludamos, 7*(1). Retrieved from http://www.eludamos.org/index.php/eludamos/article/view/vol7no1-1

Freeman, E. (2010). *Time binds: Queer temporality, queer histories*. Durham: Duke University Press.

Galloway, A. (2006). *Gaming: Essays on algorithmic culture*. Minneapolis: U of Minnesota Press.

Game Freak. (1998). *Pokemon: Yellow*. Gameboy Color: Nintendo.

Genette, G. (1979). *Narrative discourse*. New York: Cornell University Press.

Hansen, C. (2018). *Game time: Understanding temporality in video games*. Bloomington: Indiana University Press.

Hansen, M. (2019). *Feed forward: On the future of twenty-first-century media*. Chicago: University of Chicago Press.

Hayles, N. K. (2004). Print is flat, code is deep: The importance of media-specific analysis. *Poetics Today, 25*(1).

Hitchens, M. (2002). Time and computer games, or "no, that's not what happened." In K. K. W. Wong, L. C. C. Rung, & P. Cole (Eds.), *Proceedings of the Third Australian Conference on Interactive Entertainment* (pp. 44–51). Perth: Murdoch University Press.

Huizinga, J. (1950). *Homo ludens: A study of the play-element in culture*. London: Routledge.

Imagineering. (1995). *Desert bus*. In *Penn & Teller's smoke and mirrors*. Sega CD: Absolute Entertainment.

Infinity Ward. (2007). *Call of duty 4: Modern warfare*. Playstation 3 and Xbox 360: Activision.

Infinity Ward & Sledgehammer Games. (2011). *Call of duty: Modern warfare 3*. Playstation 3: Activision.

Jayemanne, D. (2019). Chronotypology: A comparative method for analyzing game time. *Games and Culture*.

Jørgensen, K. (2013). *Gameworld interfaces*. Cambridge: The MIT Press.

Juul, J. (2005). *Half-real: Video games between real rules and fictional worlds*. Cambridge: The MIT Press.

Kirschenbaum, M. (2008). *Mechanisms: New media and the forensic imagination*. Cambridge: The MIT Press.

Knutson, M. (2018). Backtrack, pause, rewind, reset: Queering chrononormativity in gam-

ing. *Game Studies, 18*(3). Retrieved from http://gamestudies.org/1803/articles/knutson

LeMieux, P. (2014). From NES-4021 to moSMB3.wmv: Speedrunning the serial interface. *Eludamos, 8*(1). Retrieved from https://www.eludamos.org/index.php/eludamos/article/view/vol8no1-2/8-1-2-html

Lindley, C. (2005). The semiotics of time structure in ludic space as a foundation for analysis and design. *Game Studies, 5*(1). Retrieved from http://www.gamestudies.org/0501/lindley/

Lunenfeld, P. (1999). *The digital dialectic: New essays on new media*. Cambridge: The MIT Press.

Manovich, L. (2001). *The language of new media*. Cambridge: The MIT Press.

Maxis. (2003). *Sim City 4*. Windows PC.

"Megaflip." *OoT Speedruns Wikia*. Retrieved from https://oot-speedruns.fandom.com/wiki/Megaflip

Montfort, N. & Bogost, I. (2009). *Racing the beam: The Atari Video Computer System*. Cambridge: The MIT Press.

Newman, J. (2019). Wrong warping, sequence breaking, and running through code: Systemic contiguity and narrative architecture in *The Legend of Zelda: Ocarina of Time* any% speedrun. *Journal of the Japanese Association for Digital Humanities, 4*(1).

Nikolchina, M. (2007). Time in video games: Repetitions of the new. *Differences, 28*(3).

Nintendo EAD. (1986). *The legend of Zelda*. Nintendo Entertainment System: Nintendo.

Nintendo EAD. (1996). *Super mario 64*. Nintendo 64: Nintendo.

Nintendo EAD. (1998). *The legend of Zelda: Ocarina of time*. Nintendo 64: Nintendo.

Nintendo EAD. (2008). *Mario kart Wii*. Wii: Nintendo.

Nintendo EPD. (2017). *The legend of Zelda: Breath of the wild*. Nintendo Switch: Nintendo.

Nintendo R&D1. (1992). *Super mario land 2: 6 golden coins*. Gameboy: Nintendo.

Nitsche, M. (2007). Mapping time in video games. In A. Baba (Ed.), *Situated Play: Proceedings of the Third International Conference of the Digital Games Research Association* (pp. 145–151). Tokyo: University of Tokyo.

Riot Games. (2009). *League of legends*. PC: Riot Games.

Rockstar North. (2004). *Grand theft auto: San Andreas*. Playstation 2: Rockstar Games.

Sample, M. (2016). Code. In H. Lowood & R. Guins (Eds.), *Debugging Game History: A Critical Lexicon*. Cambridge: The MIT Press.

Scully-Blaker, R. (2014). A practiced practice: Speedrunning through space with de Certeau and Virilio. *Game Studies, 14*(1).

Sega AM2. (1999). *Shenmue*. Playstation: Sega.

Tobin, S. (2012). Time and space in play: Saving and pausing with the Nintendo DS. *Games and Culture, 7*(2).

Valve. (2000). *Counter-strike*. PC: Valve.

Wei, H., Bizzocchi, J., & Calvert, T. (2010). Time and space in digital game storytelling. *International Journal of Computer Games Technology*. Retrieved from https://www.hindawi.com/journals/ijcgt/2010/897217/

Wolf, M. J. P. (2001). Time in the video game. In M. J. P. Wolf (Ed.), *The Medium of the Video Game* (pp. 77–93). Austin: University of Texas Press.

Zagal, J. & Mateas, M. (2010). Time in video games: A survey and analysis. *Simulation and Gaming, 41*(6).