Playable Quotes for Game Boy Games

Joël Franusic
Independent
Alameda, CA, USA
joel@franusic.com

Kathleen Tuite
Independent
Santa Cruz, CA, USA
kathleen.tuite@gmail.com

Adam M. Smith
University of California, Santa Cruz
Santa Cruz, CA, USA
amsmith@ucsc.edu

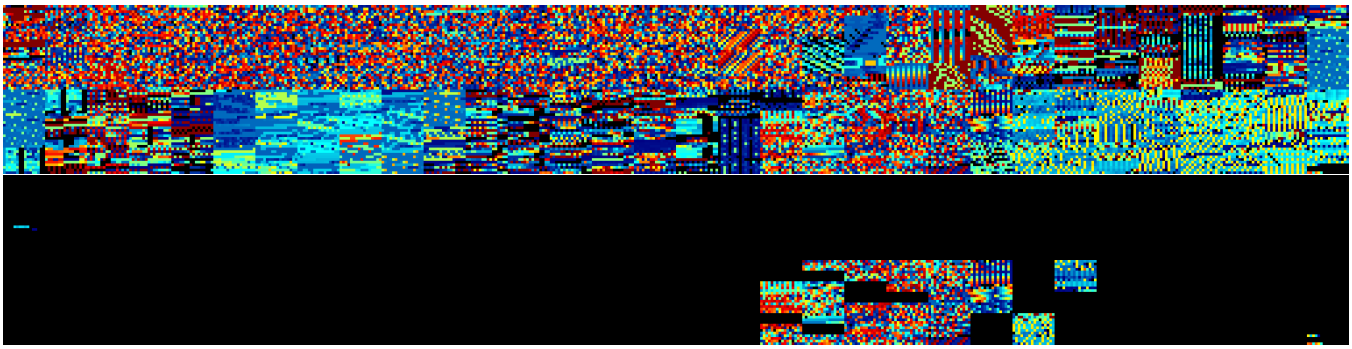

Figure 1: Read-only memory (ROM) image for *Tetris*. Top: Original game data describing game modes, rules, music, art, etc. Bottom: Masked ROM preserving only those blocks (about 6%) accessed during a short recorded gameplay segment. When combined with a savestate and player input sequence, the masked ROM is sufficient to replay the segment and even respond to novel inputs. However, it does not support gameplay involving game modes or art that was not seen during the quoted segment.

ABSTRACT

When discussing a work of literature, text quotes can provide access to specific pieces of content and allow them to be placed in a larger context. However, we have no obvious analog for this notion of quotes in the medium of videogames. In this paper, we introduce the concept of playable quotes. To support the needs of developers, educators, streamers, and other stakeholders, playable quotes should be playable (directly interactive), partitioned (small slices of larger works), permanent (able to outlive the original work), and performative (able to demonstrate specific styles of play). Focusing on the Game Boy hand-held game console, we describe Tenmile, a deployed prototype for creating and sharing self-contained playable quotes of Game Boy games that satisfy our key requirements. Using Tenmile's technical design as a template, we lay out a strategy for implementing similar notions of playable quotes across other computing platforms and raise issues that should be considered before continuing beyond Game Boy.

CCS CONCEPTS

• **Applied computing** → **Computer games**; • **Information systems** → *Multimedia databases*; *Multimedia and multimodal retrieval*.

FDG 2023, April 12–14, 2023, Lisbon, Portugal

ACM ISBN 978-1-4503-9855-8/23/04…$15.00
https://doi.org/10.1145/3582437.3582479

KEYWORDS

Game Boy, emulation, speedrun, program slicing

ACM Reference Format:
Joël Franusic, Kathleen Tuite, and Adam M. Smith. 2023. Playable Quotes for Game Boy Games. In *Foundations of Digital Games 2023 (FDG 2023), April 12–14, 2023, Lisbon, Portugal.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3582437.3582479

1 INTRODUCTION

How would you quote a videogame? You might quote some review text or character dialog, use a clip from a trailer video, capture a screenshot, or record a short video of play. But none of these are playable in the way that a quote of text is readable. This paper offers an alternative: *playable quotes*.

Beyond quoting text, it is straightforward to crop images and clip videos to share their most relevant details. As the dimensionality of media increases, starting from text, moving to images, and even adding a time dimension for video, the basic mechanisms of quoting continue to work until we add the dimension of interactivity.

For quoting games in particular, we need a way to consider locations and boundaries in the space of play. The breakthrough work of Kaltman et al. [23] provides us with a notion of *location* within the space of play comparable to a bookmark for a specific moment in a game, allowing that moment to be cited, discussed, and compared with other moments in games.

We take this a step further by demonstrating that it is possible to specify *bounds* within the space of play. If we think about a videogame as a collection of data (e.g. compiled machine code and art assets), we show that it is possible to form playable quotes by masking out large fractions of this data while keeping just enough so that the result is still able to be played. Fig. 1 shows a masked

version of *Tetris* data that preserves the ability to reproduce a few specific seconds of gameplay.

In this paper, we argue that playable quotes address unmet needs of game developers, educators, streamers, and other stakeholders. To meet these needs, playable quotes need to satisfy four key properties: (1) they need to be playable; (2) they need to be small, delimited slices of the original work; (3) they need to have a kind of durability that might allow them to outlast the work from which they are quoted; and (4) they need to directly demonstrate how to perform (play) during the quoted moment. These use cases and properties are elaborated in the next section.

Illustrating the human experience of working with playable quotes, this paper describes Tenmile, a publicly deployed prototype in which users can create, share, and experience playable Game Boy quotes in a web browser. Using Tenmile as a template, we sketch a method for extending the notion of playable quotes for platforms beyond Game Boy, such as contemporary desktop software.

2 PLAYABLE QUOTES

This section reviews a documented need for playable quotes and provides plausible use cases to derive our set of key properties.

2.1 Inspiring Use Cases

Anderson et al. [1] conducted an interview study to understand the information needs that might be satisfied by future videogame moment search engines. Although their research questions focused on issues such as how users might form search engine queries (e.g. by using descriptive text, screenshot images, or short video clips of gameplay), their documentation of the users' ultimate goal points at the need for something like playable quotes. Where web search engines return snippets of web content (such as a block of text or a thumbnail image), game moment search engines would ideally offer snippets of gameplay that could be quickly browsed before going deeper into the target interactive experience.

The Game and Interactive Software Scholarship Toolkit (GISST) [23] makes it possible to cite specific moments in games akin to how we can reference specific page numbers in specific editions of books. Further, it intends to capture specific performance recordings that demonstrate styles of play or record how a game responds to player choices. Citations created with GISST can be embedded into webpages where they offer readers the experience of directly playing through the moments of interest. Kaltman et al. argue that GISST brings us closer to "a new class of scholarly support tool for arguments about games and other interactive software that would benefit from the inclusion of audiovisual and executable references."

Below, we elaborate on Anderson's documented search engine user profiles to illustrate how playable quotes might offer valuable new experiences for a variety of videogame stakeholders.

Speedrunners are people who compete to play games (or specific segments of them) in the fastest humanly[1] possible way. Speedrunners will seek out recordings of individual record-breaking speedruns for examples of glitches and tricks. Often, their ultimate goal is to learn from these specific examples and set up an environment where they can practice new techniques to improve the timing of their own runs. Rather than sharing a newly developed speedrunning technique via descriptive text or non-interactive video, playable quotes might allow specific performances to be shared and have their frame-by-frame button inputs reviewed and revised by others. While it is common for speedrunners to share input sequence recordings (referred to as *movies*), these recordings can only be played back by those who have access to the original game.

Streamers and YouTubers are video content creators who often aim to discuss design details and play style possibilities for videogames. Often, they would like to directly demonstrate gameplay to their audiences to make a specific argument. Aiming to put on a specific show, they would benefit from the ability to directly jump to points of interest in larger games. A prepared set of quotes might allow streamers to increase the production value of their shows by presenting a more focused and responsive performance.

Educators who teach the design of games often want to demonstrate the design choices and varieties of game mechanics from a wide range of games. Where the prepared collection of playable quotes helped the streamer fluently navigate a single game, such a collection might help the educator demonstrate key ideas across several games spread across platforms and history. This need is currently only partially met by the use of gameplay video clips. The use of playable quotes might offer educational materials for teaching game feel [39] or how a game responds differently to different player choices in a specific moment without needing to tediously replay to that moment from the most recent savepoint supported by the game itself.

Game designers and developers, who rapidly churn through many variations of design ideas and game assets [45] during their creative process, often want to document which moments encountered during their playtests were particularly fun, painful, or embarrassingly buggy. A suitably compact representation for playable quotes might allow direct recordings of these moments to be stored and shared with the larger design team for review even while the full details of the temporary game build seen by the playtesters is discarded. Beyond a video recording of gameplay, developers might be able to make specific use of game input recordings to diagnose and reproduce bugs demonstrated in a recording.

Anderson's final category of **scholars** includes people who might want to discuss rare or unique moments, examples of superior or aesthetic play, or even historically significant moments. Anderson described how game historian Henry Lowood offered *Move 37* [28] in a pivotal 2016 match of the traditional board game Go between human expert Lee Sedol and the machine player AlphaGo. A playable quote of *Move 37* and its immediate gameplay context (perhaps reproduced inside of a videogame adaptation of Go) might allow critics and other audiences to make much more concrete arguments about what was at stake at that point in the game and what futures could have followed from different moves. Kaltman's GISST [23] demonstrates several other areas of potential scholarly impact for playable quotes, and this team's recent *Digital Humanities Quarterly* article [22] is the first example of a scholarly publication that embeds interactive moments.

[1] In the related area of tool-assisted speedruns (TAS), *super-human* gameplay sequences are generated with the help of computer programs and game memory visualizations. The authors of this paper are grateful for the informal literature of the TAS community that has offered deep insights into the low-level execution details for classic videogames. The curious reader is encouraged to start their journey at https://tasvideos.org/

2.2 Key Properties

In order to support the use cases sketched above, playable quotes need to have certain properties. We use the properties listed below to guide the technical implementation and user experience design of our deployed prototype system (see Section 4).

Playable: Textual quotes must be readable, and video clips must be watchable. Analogously, playable quotes need to be playable. Games can create meaning by the way they do (or even don't) respond to audience choices, and for a quote to convey that meaning compactly, the quote should offer similar responsiveness. Beyond just letting you watch a recording of play, playable quotes should let you grab the controls and test out your own style of play during the quoted moment. The playability of a playable quote supports the speedrunners desire to train specific techniques in specific context and supports the educator's desire to demonstrate how the game responds to different choices.

Partitioned: To function as a compact quote that conveys specific ideas (while being easy to archive, distribute, and embed), the small bit of content of a playable quote should be intentionally partitioned from the content of the larger work being quoted. The bounds set on a quote can shape and focus the discussion of the original work while masking out irrelevant or distracting details. A skilled creator of playable quotes will delimit their quotes in a way that allows replay with the styles and effects of play they want to communicate and little more. The compactness that results from partitioning will support the scholar's intent to archive important moments and the designer's intent to keep only the important parts of ephemeral versions of their game. Meanwhile, the specificity that results from partitioning could help the streamer navigate to the precise moment of play needed for their show or even allow their live audience to submit alternate performance takes on a given scenario without needing to redistribute the entire original work.

Permanent: Unlike a page number for a book or a uniform resource locator (URL) on the web that can become a *dead reference* when the associated book or webpage is lost, we require that playable quotes be permanent in the sense of potentially being able to outlast the work they quote from. This durability of quotes will ensure they can still communicate key ideas even when they are sliced from larger works that are considered ephemeral (e.g. the latest debug build of a game, a live game that responds to the shifting discourse on social media, or a game built by a student for a university course). Permanence may also aid in archiving, bypassing the need to set up and play through historical games from scratch. Beyond the citations offered by GISST, permanent playable quotes might someday allow embedding key moments of a game into scholarly documents as easily as we embed figures. Publishing a paper that uses a GISST-style reference currently requires authors to convince publishers that it is worthwhile to archive and host a copy of the *complete* original work associated with each cited interactive moment.

Performative: Playable quotes that are playable, partitioned, and permanent would be already quite useful for a number of stakeholders. However, there is a difference between a slice of a game's data and a slice of the game's space of play. Analogous to the difference between a written screenplay and the final film (with many details of timing, posture, and pacing added through the creativity of actors), we require that playable quotes record specific styles of gameplay to be performed during the quoted moment. These demonstrations of play will layer on additional channels of communication to what is otherwise a bookmark in the space of play without guidance on how to perform in that moment. For the speedrunning use case described above, these performances are precisely the objects of interest (as most players already have full access to the games they are interested in speedrunning). For the community of scholars, because it may be the case that the historical game of interest no longer has any high-skill players available, playable quotes might function as a valuable primary source for study of historical styles of play.

3 GAME BOY FEASIBILITY STUDY

To illustrate our notion of playable quotes, we first turned our attention to the Game Boy handheld game console. (This section describes our technical proof of concept that inspired the Tenmile system described in Section 4.) Since its original commercial release by Nintendo in 1989, the Game Boy platform has been a historic and ongoing site of economic and cultural activity. For example, speedrunners continue to improve techniques and completion times for games like *Metroid II: Return of Samus* [38]. Platform studies scholars examine the Game Boy to develop arguments about how material constraints inform the development of media platforms [11, 35]. Enormous collections of licensed and unlicensed Game Boy games are now available for download and play through the piracy-adjacent ecosystems of abandonware [6] and ROM-hacking communities [29]. Access to historical games inspires designers in the present day to develop fresh collections of homebrew Game Boy games [43] and it motivates the creation and maintenance of authoring tools like GB Studio [27] that intend to make the Game Boy platform welcoming even to first-time game designers. Absent a usable notion of playable quotes, much of the content of this deep and growing catalog of Game Boy games is unavailable for direct inclusion in public discourse.

Originally, Game Boy games were distributed in the form of plastic cartridges containing circuit boards which were primarily composed of memory chips. In the present day, homebrew Game Boy games are often distributed as digital files (read-only-memory images, or simply *ROMs*) that describe the data to be loaded onto the rewritable memory of flash carts or into software emulators [5]. For the sake of illustrating the concept of playable quotes, we assume that all of a game's relevant details are contained in the ROM data (setting aside beyond-ROM hardware such as the Game Boy Camera) and demonstrate all gameplay using emulators.

To understand how we could build playable quotes for Game Boy games, consider how data is stored and computed during gameplay. The static data of a game is stored in ROM: it is accessed but not modified during play. This data (which includes executable machine code) defines the various modes of a game, the rules at work in each mode, as well as the music, visual art, and other design details. The dynamic data of a game is stored in a number of different random-access memories (RAMs). Most important game state data that varies from moment to moment is stored in working memory (WRAM) while video memory (VRAM) is used to drive the console's

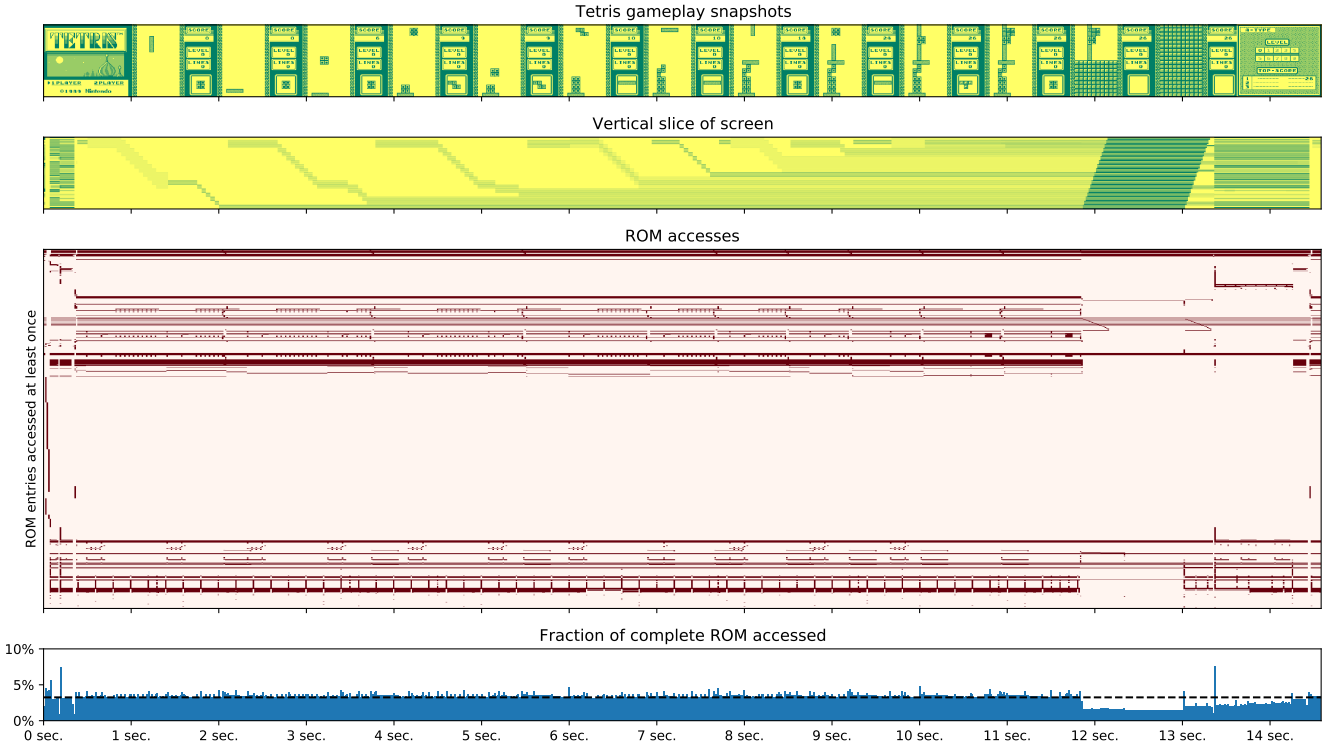

Figure 2: ROM access patterns for *Tetris* gameplay (during which the player quickly traverses game menus and then cycles between pressing A, LEFT, and DOWN to mash their pieces into a stack before losing the game and starting a new gameplay session). On a typical frame of core gameplay for Tetris, only about 3% of the ROM bytes are accessed.

display. While some games embed additional cartridge memory chips (CRAM), most do not.

Recall that we want to produce *permanent* and *playable* quotes of specific moments in a game: they should allow the audience to continue some amount of play starting at the quoted moment without needing access to any other reference materials. To set up an emulator for continuation of play (or to make a GISST-style citation for that moment), we typically need a ROM file and a savestate file. The savestate file describes the various RAMs mentioned above as well as a few extra details for the state of the main processor, memory controllers, and peripheral devices such as the sound chip. For Game Boy, ROM files range in size from 32 KiB to 8 MiB whereas savestate files are usually orders of magnitude smaller. The largest components of a savestate file are usually the 8 KiB WRAM and 8 KiB VRAM states. To *partition* the game, we can try to carve away irrelevant details from this pile of data while keeping just enough to reproduce (or *perform*) the kinds of play we want to communicate.

Fig. 2 visualizes the patterns of access to ROM data over time during a few seconds of gameplay for *Tetris* [32]. During the recorded gameplay performance, the player taps through some of the game's menus to enter the core gameplay mode, then carelessly stacks their puzzle pieces in a rush to reach the game-over animation. After this animation, the player taps through a high-score entry screen and then traverses menus to start a new session of core gameplay. In this visualization, note how little ROM data is accessed during each moment of interaction (as a fraction of all available ROM) and that access patterns are largely stable within a given gameplay mode (e.g. core gameplay versus menus), shifting significantly only when the player transitions to a new mode.

We analyzed memory access patterns by modifying a popular Game Boy emulator. In particular, we patched the implementation of memory models inside of PyBoy [18] and created analysis plugins that recorded the precise addresses of memory and their order of access between each frame of interaction for all of the large memories (ROM, WRAM, VRAM, and CRAM) [34]. Going beyond this instrumentation, we created additional plugins that enforced access to a *masked* view of each memory: gameplay would be interrupted if play continued in a way that attempted to read from a memory address not included on a strict allowlist.

With this instrumentation enabled, we played a number of different Game Boy games and considered a number of different strategies for building the allowlists. In the Tetris example above, a quote that captures only the core gameplay mode only involves about 6% of the ROM, covering roughly 2 KiB data. This is comparable to the amount of data needed to encode two uncompressed copies of the plain-text version of this paper's abstract (not bad for a precise executable statement of the look and feel of *Tetris*). While patterns of WRAM access qualitatively resembled those of ROM, the smaller overall size of WRAM and the relatively larger fraction of it needed to resume play made it less interesting for partitioning. Meanwhile,

almost the entire contents of VRAM was needed to reproduce even a single graphical frame of continued gameplay.

These observations convinced us that a feasible representation of playable quotes for the Game Boy would consist of a masked view of the ROM combined with a savestate file containing unmasked views of RAMs. To satisfy the *performative* requirement of playable quotes, we proposed to augment this data with a precise recording of the player's button-inputs during the quoted gameplay segment.

To expand the world of playable quotes beyond our feasibility study with PyBoy, we needed to figure out how to provide public audiences with access to a memory-tracing emulator, offer them a means of recording input events with sufficient precision to re-perform gameplay demonstrations, and define a convenient data format for sharing bundles of masked ROMs, savestates, and input event movies. These details are described in the following section.

4 TENMILE PROTOTYPE

To feel out the capabilities of playable quotes for Game Boy, we needed to create, share, and discuss quotes from many different games. In this section, we describe Tenmile,[2] a publicly deployed prototype system intended to sketch a concrete workflow for using playable quotes in public discourse. The following subsections describe the user experience of Tenmile (including the novel activity of *riffing* on a playable quote) and how its technical design enables sharing and embedding playable quotes analogously to how we share and embed image files.

4.1 User Experience Design

At the center of Tenmile's user experience is interaction with our quote player widget (seen in Fig. 3). Usable on both desktop and mobile web browsers, the widget supports the performance of existing playable quotes for Game Boy *and the creation of new quotes.*

The widget is styled after the look of the original Game Boy hardware, but it includes buttons to load new quote files (described later) or to switch interaction states. The experience of watching and then riffing on a playable quote is intended to resemble the experience of watching a friend play a Game Boy game and then getting a chance to temporarily continue play as they hand you the console at a critical moment.

4.1.1 Interaction States. Following the *progressive disclosure* design principle [37], our quote player widget is intended to incrementally reveal the unfamiliar nature of playable quotes. Initially, quotes appear to be plain image files. Next, they are revealed to behave like video clips. Upon touching any of the on-screen or keyboard controls, the user realizes that the quoted performances are interactively replayable almost like they were copies of the original game. Users with access to complete ROM files may eventually learn that the widget supports play beyond quoted moments as well as the ability to record and compile new quotes from inside of the widget. Fig. 4 summarizes the widget's interaction states.

Upon entering the **watching** state, the player is greeted with an animation of pre-recorded gameplay. Rather than offering a time seek bar and explicit duration suggestive of potentially long-duration video files, we offer the feeling of an animated GIF file

[2]This project codename references a campground in California's Sierra Nevada where the authors developed the project concept.

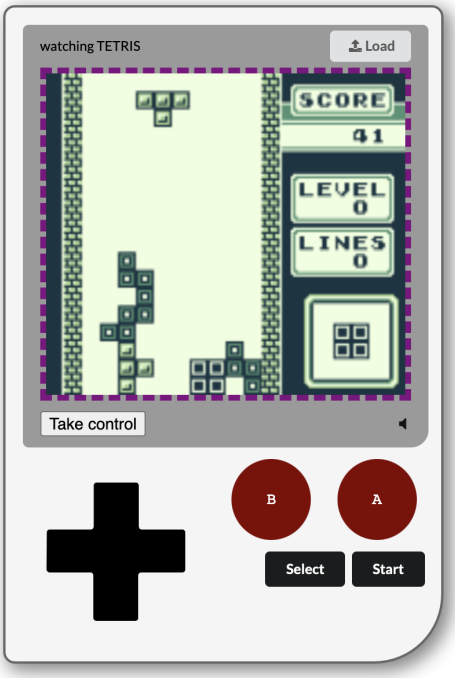

Figure 3: Screenshot of the quote player widget, showing playback of a *Tetris* quote in the *watching* state. The user can use their keyboard or on-screen controls to take control of play and continue in the *riffing* state.

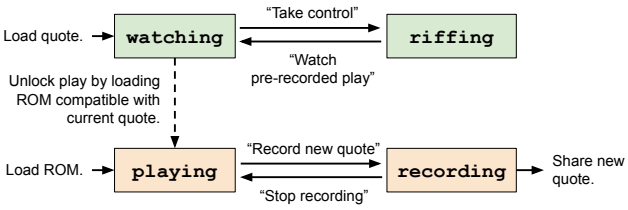

Figure 4: Interaction states in our quote player widget. Most users will begin by loading a pre-existing quote file while advanced users might use a collection of quote files to help navigate the space of unbounded play for a complete game they possess.

or arcade game attract mode: the (almost always short) gameplay segment automatically plays in a loop without user intervention. While the widget allows users to explicitly load their own quotes, most users will first see the widget with a specific quote pre-loaded. We allow users to construct links to our quote player webpage with the URL of existing files on the web. Using inline frames [9], web content creators can embed playable quotes in their pages similar to how they embed image and video content.

While watching a quote, the user can "Take control" to enter the **riffing** state. Riffing leans on the combination of the *playable* and *performative* properties of playable quotes. Named in reference

to the musical[3] term, riffing allows the user to improvise their own performance based on the example offered in the pre-recorded input data. In their improvised performance, the user may decide to have their character walk left instead of right, jump earlier or later than in the original recording, or simply sit still and listen while the background music continues to play out. As this unanticipated behavior might cause our prepared Game Boy emulator to attempt to access out-of-bounds ROM data, we use continuous memory tracing while riffing to detect when the user is leaving the effective bounds of the quote. Rather than seeing an explicit error message when this happens, we quietly return the state of the emulator to the one at the beginning of the quote (as if they had reached the natural loop point of the GIF), leaving them in the riffing state in full control of the game. A "Watch pre-recorded play" button allows the user to return to the watching state. So long as memory accesses remain within those supported by the quote, there is no limit on how long a player can remain in the riffing state without the need to reset to the beginning of the quote.

If the user has access to a complete ROM file, they can load it instead of a quote file to reach the unrestricted **playing** state. In this state, our widget acts much like many other web-based Game Boy emulators. If the user was just watching a short quote and they have access to the full game associated with that quote, they can load that ROM to continue unrestricted play from the moment embedded in the quote. This advanced interaction pattern is intended to support using quote files as shareable bookmarks in the space of play with the added benefit of being permanent previews of the marked moments for users who lack access to the full version of the quoted work.

Importantly, users transition from the playing state to the **recording** state with an obvious "Record new quote" button. While recording, our widget continuously displays a message with an estimate of how much of the original work (as a fraction of total ROM bytes) will be embedded into their quote file. This small bit of feedback is intended to help users develop a feel for what makes one quote larger than another and how their recent gameplay actions have impacted memory access patterns (which has implications on how much freedom for riffing their quote will offer to others). The result of a completed recording is a compiled quote file that can be downloaded, hotlinked, and otherwise shared using any of the familiar tools for sharing image files. Upon finishing a recording, the user is returned to the playing state at the moment the recording began (setting them up to record another take, possibly playing a different way to create a differently shaped space of riffable play or offering a better performance of the gameplay technique they are demonstrating).

As with the watching state, web content creators can link or embed our widget in the playing state to offer immediate interaction without the need for users to explicitly load a ROM file. This mechanism allows game developers to share their Game Boy games in an easy-to-quote form that might help other users to respond with specific design commentary or bug reports. Because ROM files can be loaded from other websites (via their public URLs), this mechanism also opens up public archives of ROM files to be quoted without the need to deploy any new software to those sites or for quote-creators to explicitly download any data or tools.

[3]To riff on a piece of music is to improvise a variation on a short and often repeated segment of an earlier work.

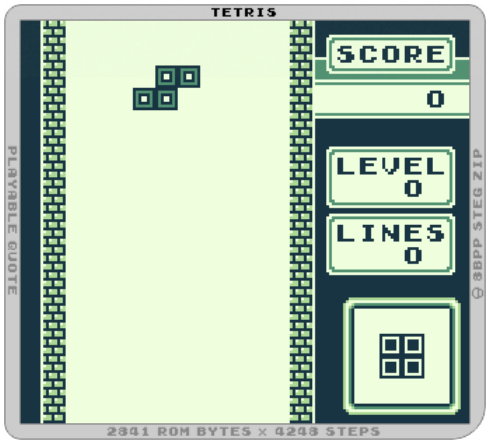

Figure 5: Example quote file for *Tetris* that is scoped to demonstrate some but not all of the game's core mechanics. Small variations in the brightness of each pixel encode a *masked* copy of the game's ROM and other data required to offer an interactive reproduction of a specific moment of gameplay. Clues in the image frame suggest deeper meaning.

4.1.2 Quote Files. In the spirit of progressive disclosure, quote files are intended to first appear as images, but they need to have clues that lead audiences to their more advanced affordances. Fig. 5 contains a Tenmile playable quote of *Tetris* in which the player can watch a sample of typical gameplay before trying it themselves. The quote was constructed so that the player can clear a few lines of blocks at a time but never exercise a tetris (clearing four lines at a time) or access any of the game's other modes.

Our quote files are intended to visually parse as game screenshots embedded in a gray frame evocative of the plastic cartridges used to distribute the original Game Boy games. Reading the text in the frame counter clockwise, the viewer sees the clues "Tetris," "playable quote," "2841 ROM bytes x 4248 steps," and "8bpp steg zip." This image-as-playable-artifact encoding references the way in which games for the PICO-8 [20] fantasy console are distributed as cartridge-shaped image files where the data needed to execute the game is steganographically encoded (8 bits per pixel) into the low-order bits of image pixel brightness.

By saving our quote files in the PNG image file format and including at least one transparent pixel, we found that users could share quote files on various social media platforms (such as Imgur and Twitter) without the encoded data being lost.

In the unlikely event that a far future media archaeologist is attempting to decode one of our quote files without access to this paper, the "8bpp steg zip" note is intended to be enough of a clue to allow the archaeologist to recover a zip file containing plaintext documentation that might allow successful decoding of the remaining details of the quote assuming they have access to the GitHub Arctic Code Vault [21]. This design detail is a nod to how the concerns of having permanent quotes requires having permanent access to quote-reading tools as well.

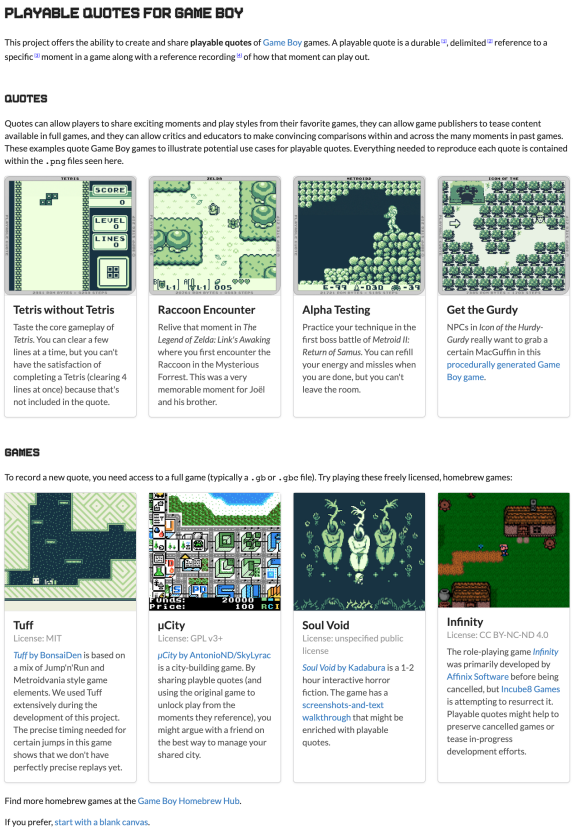

Figure 6: Landing page for the Tenmile project offering quotes to view and full games from which new quotes can be created.

4.1.3 Project Landing Page. To add context to the widget described above and to bootstrap creation and sharing of quote files, we created a public landing page for the Tenmile project on the web: **https://tenmile.quote.games/**

Our landing page offers a number of example quote files, each visually represented by embedding the raw quote file into the webpage as a normal image element. A quote from Tetris emphasizes control over a quote's bounds by explaining what is and is not available to experience while riffing. A quote from *The Legend of Zelda: Link's Awakening* is suggestive of how someone might share personally meaningful moments of a game with people who have never played the game before. The quote from *Metroid II: Return of Samus* shows how a quote can be used as both a discourse object and a training environment in the speedrunning community (even if our recorded performance is not particularly impressive). The final quote coming from the procedurally generated [15] Game Boy Game *Icon of The Hurdy Gurdy* represents a role for permanence in storing a playable slice of a larger game that is soon to be discarded.

Below the example quotes, we offer links to our quote player with the full ROM for a few non-commercial games pre-loaded. The first game, *Tuff* [44], was used extensively during our feasibility study with PyBoy and in later development of Tenmile. The game *µCity* [14], in addition to showing support for Game Boy Color games, entices the user to share quotes of their unique creations (simulated city designs that represent data not found anywhere in the original game ROM but instead created by players). *Soul Void* [16] is a lengthy interactive horror fiction for which playable quotes might offer an interesting alternative to screenshot-and-text walkthroughs. Finally, *Infinity*[4] (an in-development continuation of a canceled Game Boy game from 2001) is offered as an example of how quotes might be used to record the incremental development of games that have yet to see an official release.

Our landing page with curated widget links is intended to set up Tenmile as a *casual creator* tool. Compton [8] describes casual creators as a genre of software (not a class of users) existing "halfway between productivity software and entertainment software: like productivity software they are designed to create artifacts (virtual or otherwise) during their use, but like entertainment software, the purpose of the experience is primarily diversion or pleasure, and the production of artifacts is only secondary to that purpose."

Compton characterizes the features common to many existing casual creator tools by describing design patterns. Intentionally following some of these patterns, we instantiate the "no blank slate" pattern by offering views onto our widget with quotes or whole games pre-loaded. The "self-promotion" pattern is expressed in how we allow the quote-creator to compile quotes down to shareable URLs and curious image files compatible with current social media platforms. The "make it real" pattern is manifested in how the quote-creator is creating new games (not just videos of them) through their recorded performance in a way that offers others a view into games they might have never touched otherwise. Finally, the "transformation" pattern maps to how users can create exciting new artifacts as a result of seemingly everyday actions: simply playing a game and not touching pixel art or machine code.

Unexpectedly, our implementation of these design patterns at the level of the Game Boy platform offers new casual-creator style interactions around existing productivity software. For example, consider the professional-grade chiptune music sequencer *Little Sound Dj* [25] (also known as LSDJ). It might now be possible to share whole albums, individual songs, or individual phrases within them as playable quotes of the LSDJ cartridge, with artist-specified levels of audience freedom to riff and remix the work released. Fans could self-promote their own remix by recording their own riff on the artist-provided quote after they unlock play by loading the full (freely available) LSDJ ROM into the widget.

4.2 Technical Design

This section reviews low-level technical details of implementing playable quotes for Game Boy inside of modern web browsers.

4.2.1 Interactive Widget. Recall our desire to reproduce the capabilities demonstrated in our PyBoy feasibility study in a format compatible with desktop and mobile web browsers. In search of a browser-native Game Boy emulator, we encountered a variety of JavaScript-based emulators with varying levels of documentation, game support, and ease of modification. Critically, we required an emulator with support for creating and loading savestates and

[4] https://www.infinitygbc.com/

sufficiently deterministic playback of recorded input sequences to illustrate our idea of *performative* quotes.

We selected the aged and unmaintained emulator GameBoy-Online (GBO) [19] for its surprising ease of integration. The adoption of GBO by the developers of the GB Studio game authoring tool was an encouraging endorsement as well. Where we needed to make multiple edits to core elements of PyBoy to implement memory tracing, we were surprisingly able to use GBO without any quote-related source modifications. Instead, we used the Proxy [10] feature of JavaScript to install instrumentation objects at the sites of key methods and arrays of GBO at runtime. In particular, we intercepted property access (array indexing) operations on the `gameboy.ROM` array to snoop ROM accesses. We also intercepted calls to the `gameboy.JoyPadEvent` method to record input events precisely (regardless of whether they came from a hardware keyboard on an on-screen button) and to the `gameboy.run` method to precisely count emulator timesteps (regardless of the passage of wall-clock time). While our implementation strategy could be interpreted as a kind of ad-hoc aspect oriented programming [24], it is better understood as an instance of reverse engineering practices already in widespread use in game hacking communities.[5]

Achieving sufficiently deterministic replay with GBO revealed certain curiosities about the Game Boy platform. For example, we learned that many Game Boy games introduce the feeling of randomness by noting the display scanline when a button state change occurs (checking the status of the `LY` register associated with the liquid crystal display driver hardware). We found the *Telling LYs* [46] test ROM invaluable when testing our ability to replay input events with the necessary sub-frame precision. Despite our efforts, we were only able to achieve deterministic replay for most games most of the time. Additional exploration and bug fixes within GBO's 10,000 line JavaScript implementation seem to be required to improve the performative aspect of Tenmile's quotes further.

The entire Tenmile project is implemented using only about 1,000 lines of new JavaScript code, most of which is dedicated to managing the user interface state transition system and quote file encoding rather than low-level Game Boy emulator concerns. In a version of Tenmile adapted to quote from Super Nintendo games, only a fraction of this code would need to be changed.

4.2.2 Adjusting Quote Bounds. During a recording session in Tenmile, we build up a byte-precise list of ROM entries accessed. When we create masked ROMs that include exactly these bytes and no others, the resulting quotes are very brittle: they tolerate very little variation in alternate gameplay behavior. In other words, they leave little room for riffing even when very long segments of recorded gameplay are used to produce them. Worse, because of imperfect determinism in replays, it is sometimes the case that the originally recorded sequence of input events cannot be fully replayed without encountering an out-of-bounds ROM access.

As early as the PyBoy experiments, we had found it useful to adjust the bounds of a quote by expanding our allowlists (masks) to include extra entries that were close to those that were confirmed to be read during the original recording. In particular, we began to interpret memories as being grouped into larger memory pages even though the original hardware did not work this way. If a single byte in a page was accessed during a recording session, we would adjust our mask to allow accessing any other byte in the same page.

In the deployed version of Tenmile, we found that using a page size of 256 bytes (comparable to the character length of this sentence) offered plenty of room for in-bounds riffing without accidentally including support for unintended play styles.

To prevent accidentally including entire games in Tenmile quotes (which might occur if the user manages to cause a virtual crash followed by recording the boot sequence for the game), we forcibly exclude a certain segment of the ROM that we know is only used when the Game Boy hardware is first starting up. Conversely, we forcibly include another piece of the ROM encoding the game's title even when those bytes are not read during the quoted performance. The game's title is needed for display in the widget and the generation of certain quote metadata.

4.2.3 Quote File Format. Recall that we package playable quotes in the form of game screenshot images with additional data steganographically encoded into the pixel values. Because of the fixed size of the Game Boy screen, screenshots offer a fixed budget for encoding the extra data needed to bring a playable quote to life in our widget. This budget accidentally implements another important safety feature that prevents creation and distribution of quotes that propagate unreasonably large slices of their original works.

As suggested by our "8bpp steg zip" clue, we encode the data of a zip file into the least-significant bits of the color values of each pixel. Because the Game Boy display uses a low bit depth color representation, screenshot data and zip file data do not collide. The encoded zip file contains several other files described below.

`ROM.bin` is a ROM image compatible with existing Game Boy emulators (assuming it can be paired with a valid savestate). In this file, the masked entries of the original ROM are replaced with the value `0`, allowing the file to be compressed significantly.

`ROM.mask` is another byte array shaped identically to the one in `ROM.bin`. However, values in this array are simply either `0` or `1`, describing whether reads to the corresponding entry in the ROM array should be allowed.

The data in **`initialState.msgpack`** and **`actions.msgpack`** describe serialized JavaScript objects for the GBO emulator, which are needed to reproduce the savestate associated with the start of a recorded performance and the per-step input events needed to reproduce the performance.

The final file, **`README.md`**, includes quote metadata such as the SHA-256 digest of the ROM image used as the source for the quote. It also contains some narrative clues to help others decode and successfully interpret the other files in the zip archive in case the rest of the Tenmile project is lost.

The Tenmile project page includes a web-based tool that can unpack a quote file into a visual and textual summary of its contents. The ROM byte visualizations in Fig. 1 were generated by applying this tool to the quote in Fig. 5.

While the steganographic encoding strategy ended up as a cute reference to the PICO-8 game distribution format, it comes after discarding two alternatives. One way to combine a PNG image and a zip file is to simply concatenate them. This results in a file that can be directly viewed in web browsers and easily unzipped after

[5] Outside of the browser environment, we instead might reach for the Frida dynamic instrumentation toolkit to express runtime interception logic: https://frida.re/

renaming the file to have the right extension (`.zip`). Another way is to pack the zip data into a *data chunk* (a peculiar feature of the PNG file format). We found that current social media platforms understandably strip this kind of non-visual data from image files hosted on their platforms. Converting our losslessly-compressed PNG files into lossily-compressed JPEG files ends up destroying the fine details needed to play them. However, including at least one transparent pixel in our PNG files seemed to be sufficient to discourage image hosting services from attempting to quietly convert them to smaller JPEG files.

5 PLAYABLE QUOTES BEYOND GAME BOY

Game Boy games target a decades-old hardware platform using minuscule amounts of code and data compared to contemporary software like desktop productivity suites or mobile videogames. As such, certain aspects of Tenmile should not be taken to represent the more general notion of playable quotes. In this section, we walk through a hypothetical implementation of playable quotes that would be compatible with making quotes of modern desktop software and distributing those quotes for riffing-style interaction on other modern desktop computing platforms.

Suppose you want to develop an argument about the historical evolution of user interface designs for spreadsheet tools. You might like to include a playable quote of VisiCalc, the first modern spreadsheet application, running on an Apple II. Next, you would like to contrast this with a quoted moment of Microsoft Excel running inside of Windows XP. Finally, you want to show how the very same calculation expressed in the other scenarios is entered into cloud-based Google Sheets as viewed in the Chrome browser on a recent laptop with Chrome OS. While there are lightweight emulators for the Apple II comparable to GBO and even more heavyweight JavaScript-based emulators for XP-compatible machines that might still support a Tenmile-like experience, it seems unavoidable that a more general implementation of playable quotes can only be built on a much more general emulation platform.

First, we should seek out an emulation platform that can simulate a fairly wide variety of other hardware platforms. The QEMU [4] project fits the bill here. In particular, QEMU supports saving and loading whole-system state snapshots, and QEMU itself runs on several different platforms (regardless of the platform it is intending to emulate). In the scenario where we run Chrome OS in a QEMU-based virtual machine (VM), the VM's hard disk image plays the role of the Game Boy ROM file, now potentially six orders of magnitude larger.[6] The contents of the savestate file again encode the state of working RAM and video RAM along with the state of the CPU and peripheral devices, all correspondingly larger.

As QEMU is a mature and often-extended emulation platform, it already offers robust support for injecting instruction-level instrumentation and even tracing accesses to simulated physical memories. QEMU even offers record/replay functionality for implementing (nearly) deterministic execution of code that is sensitive to the timing of events external to the VM, such as how quickly virtual disk accesses are resolved or when packets arrive on virtual network interfaces. These features may eliminate much of the reverse engineering effort that was required in Tenmile's use of GBO. Alternatively, it may be possible to directly apply Mozilla's *RR* [31] record-replay system directly to the system-level emulator.

As with Game Boy, we should do a feasibility study to understand the access patterns for this modern system's major memories. Despite the startling increase in overall scale from Game Boy, we expect to see access patterns qualitatively resembling those for Game Boy. However, now it would probably be worthwhile to apply tracing and masking to both working and video RAM. As a result, we expect that the size of typical quote files would not grow in strict proportion to the quoted system's total data storage capacity. Perhaps the Google Sheets quote would include just 38 MiB of quoted data out of the VM's 64 GiB image size (0.06% rather than the 6% seen for *Tetris* earlier).[7]

Tempering the optimism above, consider that for cloud-based software the rules defining what happens when a given button is pressed are no longer guaranteed to be found in the equivalents of Game Boy ROM and RAM. Critical parts of software might only be downloaded from remote servers. Certain interactions with software would seem unquotable: either network traffic to remote hosts is recorded and replayed exactly (precluding riffing) or the network traffic is released out into the real internet where different responses might come back (precluding permanence). Sarma addresses a variation on this concern in an attempt to abstract a wide variety of artificial intelligence evaluations by way of whole-desktop system emulation [36]. Sarma showed that gameplay for cloud-based games could be replayed from a state snapshot by simulating a persistent network dropout beginning at the start of the recorded gameplay segment. Once some parts of a web-based game have been loaded into the local machine's browser, many interactions can proceed without updates from a remote server. Attempts to communicate with the server consistently time out (masking whether the remote server still exists or would return the same response even if it did). So long as all network communication happens *outside* of the quoted segment, we should still be able to produce quotes that are playable, partitioned, permanent, and performative.

The way that Tenmile's progressive disclosure process starts with the metaphor of playable quotes as image files needs to be rethought in the case of quotes for platforms that are significantly more heavyweight. Heavier quotes might be packaged in ways that evoke the affordances of video files. Though users will excuse the larger file size of playable quotes that present themselves as high-resolution videos, they will also expect to not need to pay the full download cost before they can start watching or riffing on a quote: surely, the video should support streaming! Where Tenmile used a fixed memory mask for the entire quote, it might be useful to track memory accesses over time in a way that leads to a kind of time-varying mask. Instead of recording whether a given byte of storage should be allowed to be accessed, we might store a time or

[6] Compare a 1 TiB hard disk with a 1 MiB cartridge ROM.

[7] While we conjecture that quotes of modern desktop software are likely to be small, we want to offer system builders a few specific warnings. First, desktop system are capable of accessing billions of distinct memory and disk bytes per second. Second, via features like direct memory access (DMA), memory transactions can be initiated by a variety of non-CPU devices on the system bus. Third, even a small amount of instrumentation overhead can negatively impact the feeling of responsiveness expected in certain interactive experiences. Oh, and be prepared for the fact that commercial games often employ self-modifying code and other anti-debugging features in their implementation of copy and cheat protection mechanisms [47].

streaming-byte-count after which the byte should be allowed to be accessed. Where users are familiar with the way that video playback pauses in the face of incomplete buffering, streaming features of heavyweight quotes might allow incrementally expanding spaces for riffing as additional savestate details continue to download. The increasing sophistication of streaming video quotes might justify the development of similarly sophisticated quote cropping and editing tools, potentially patterned after user interface metaphors already used in video editing software.

An ability to casually quote from desktop software as easily as one can take a screenshot of their desktop opens up certain non-obvious security and privacy considerations. It is one thing to accidentally include a partial view of your personal email in the background of a larger screenshot. However, with playable quotes, interactive viewers of the quote may now be able to scroll the rest of your email into view and potentially even extract your personal credentials from the larger blocks of memory that get included in your quote. A social hub for sharing desktop quotes (something halfway between YouTube and Tenmile's landing page) could easily be the site of an information security disaster. Desktop quoting tools should include multiple layers of safety and privacy features that do not have an equivalent in the Game Boy scenario.

6 RELATED WORK

Although we have positioned playable quotes as an extension to game moment citations in GISST [23], we acknowledge a deeper history to playable quotes below.

Beginning in the 1990s, "Action Replay" was the brand name of a line of game cheating devices that were marketed to enhance players' experience with PC, set-top, and hand-held game consoles. Beyond supporting cheats equivalent to making small ROM edits (such allowing the player to begin the game with 99 lives or setting the damage incurred by enemy attacks to zero), many Action Replay devices supported a feature in which the equivalent of whole-system save and restore was implemented at the level of hardware [12]. Frozen states could be shared with others who, on some platforms, could use them to continue play from the saved moment even without access to the original game.

In the literature of programming languages, our tracing and masking process might be recognizable as a form of *program slicing* [41]. While programs are often sliced by applying static analysis to code in high-level programming languages, this literature extends into the realm of dynamic analysis of compiled machine code in the form of *binary rewriting* [42] (often building on the same dynamic instrumentation and hardware emulation strategies that we use).

In the computer systems literature, *eidetic systems* [13] are those capable of being perfectly restored to any previous state they inhabited. Unwittingly, Tenmile implements a number of techniques from this area to produce compact replay logs. Where this literature is often focused on engineering new systems (for which software can be freshly compiled to utilize new features), expanding the idea of playable quotes will require reverse engineering a number of older systems and support for pre-compiled binaries from the wild.

Meanwhile, communities of videogame players have long been creating and sharing moments in video games. For platforms like Game Boy, these moments are either shared as videos [17] or as button press traces that are recorded on specially modified video game emulators [7]. Despite the fact the data in these archives is not simultaneously playable, partitioned, permanent, and performative, it has already been used to advance artificial intelligence [2] and information retrieval [48] research. Much of the potential analysis and discourse that motivates the development of playable quotes is already partially enabled by these archives.

In game design communities, a number of projects seem to implicitly represent many of the features of playable quotes while following the metaphor of being *museums* of play. These museums are hand-crafted, original works of interactive media that intend to preserve and interactively present specific details from earlier games. For example *The Museum of Mechanics: Lockpicking* [30] is a "browsable, playable collection of lockpicking mechanics from various games." Barr's *v r 3* [3] is an "exhibition of digital water in two galleries" offered to the player as a navigable three dimensional space. Other experiences such as *noclip.website* [33] and VGMaps.com [26] offer reduced senses of playability but increased senses of how authentic the reproduced content is to the source from which it was drawn. Beyond these, the historical phenomenon of shareware [40] and the ongoing practice of releasing demo version of software titles is also similar. While none of these examples is simultaneously playable, partitioned, permanent, and performative, they nevertheless appear to service a set of needs that overlaps with our experience goals for playable quotes.

7 CONCLUSION

In this paper, we set out to illustrate a notion of playable quotes that blends the expected affordances of text quotes and videoclips with the hands-on responsiveness of videogame interaction. We found in the literature a record of latent desire for playable quotes and used this to derive a set of key properties for satisfying implementations. We also found a starting point for a technical implementation: GISST citations composed of a classic game ROM plus a savestate file and optional player input recording. Turning to the Game Boy handheld game console, we found that the memory accesses to a game's ROM (the largest data component of a GISST-style citation) were surprisingly sparse and consistent over time. With this observation, we were able to dynamically instrument the execution of an existing browser-based Game Boy emulator so that it could record and enforce a set of memory access bounds during replay of a recorded performance or the riffing interactions with a live human player. Following the paradigm of progressive disclosure, we packaged our quotes into a form that felt like a familiar image file with certain additional special capabilities revealed only when relevant. Using our deployed prototype system, Tenmile, users can learn to create and share playable quotes of Game Boy games and use familiar web content authoring techniques to embed them into larger webpages or discourse on social media. Taking the technical design of Tenmile as a template, we stepped through a hypothetical development of playable quotes for desktop software, uncovering a number of important areas of future research.

The tools to create and share playable quotes now exist (at least for Game Boy), so we invite the research community to help us explore what they can be used for and how to make those uses more accessible and impactful.

REFERENCES

[1] Barrett R. Anderson and Adam M. Smith. 2019. Understanding User Needs in Videogame Moment Retrieval. In *Proceedings of the 14th International Conference on the Foundations of Digital Games* (San Luis Obispo, California, USA) *(FDG '19)*. Association for Computing Machinery, New York, NY, USA, Article 20, 10 pages. https://doi.org/10.1145/3337722.3337728

[2] Yusuf Aytar, Tobias Pfaff, David Budden, Tom Le Paine, Ziyu Wang, and Nando de Freitas. 2018. Playing Hard Exploration Games by Watching YouTube. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems* (Montréal, Canada) *(NIPS'18)*. Curran Associates Inc., Red Hook, NY, USA, 2935–2945.

[3] Pippin Barr. 2017. "v r 3". https://pippinbarr.com/v-r-3/info/

[4] Fabrice Bellard. 2005. QEMU, a Fast and Portable Dynamic Translator. In *Proceedings of the Annual Conference on USENIX Annual Technical Conference* (Anaheim, CA) *(ATEC '05)*. USENIX Association, USA, 41.

[5] Brett Bennett Camper. 2005. *Homebrew and the social construction of gaming: community, creativity, and legal context of amateur Game Boy Advance development.* Ph. D. Dissertation. Massachusetts Institute of Technology.

[6] Sarah Coleman and Nick Dyer-Witheford. 2007. Playing on the digital commons: collectivities, capital and contestation in videogame culture. *Media, culture & society* 29, 6 (2007), 934–953.

[7] The TASVideos community. 2022. TASVideos: Tool-assisted game movies. https://tasvideos.org/

[8] Katherine E Compton. 2019. *Casual creators: Defining a genre of autotelic creativity support systems.* Ph. D. Dissertation. UC Santa Cruz.

[9] MDN Web Docs Contributors. 2022. <iframe>: The Inline Frame element. https://developer.mozilla.org/en-US/docs/Web/HTML/Element/iframe

[10] MDN Web Docs Contributors. 2022. Proxy - JavaScript. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Proxy

[11] Alex Custodio. 2020. *Who Are You? Nintendo's Game Boy Advance Platform.* MIT Press, Cambridge, MA.

[12] Datel 1991. *Action Replay III Instruction Manual Datel A500-A2000.* Datel.

[13] David Devecsery, Michael Chow, Xianzheng Dou, Jason Flinn, and Peter M. Chen. 2014. Eidetic Systems. In *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*. USENIX Association, Broomfield, CO, 525–540. https://www.usenix.org/conference/osdi14/technical-sessions/presentation/devecsery

[14] Antonio Niño Díaz. 2018. µCity 1.2. https://github.com/AntonioND/ucity SHA256:39a254.

[15] Tamara Duplantis, Isaac Karth, Max Kreminski, Adam M Smith, and Michael Mateas. 2021. A Genre-Specific Game Description Language for Game Boy RPGs. In *2021 IEEE Conference on Games (CoG)* (Copenhagen, Denmark). IEEE, IEEE Press, Piscataway, NJ, 1–8.

[16] Brooke Edenfield. 2019. Soul Void. https://kadabura.itch.io/soul-void SHA256:2e64a6.

[17] emumovies.com. 2022. EmuMovies. https://emumovies.com/

[18] Mads Ynddal et al. 2022. PyBoy. https://github.com/Baekalfen/PyBoy

[19] Grant Galitz. 2016. GameBoy Online. https://github.com/taisel/GameBoy-Online

[20] Lexaloffle Games. 2022. PICO-8. https://www.lexaloffle.com/pico-8.php

[21] GitHub, Inc. 2020. GitHub Arctic Code Vault. https://archiveprogram.github.com/arctic-vault/

[22] Eric Kaltman, Joseph Osborn, and Noah Wardrip-Fruin. 2021. From the Presupposition of Doom to the Manifestation of Code: Using Emulated Citation in the Study of Games and Cultural Software. *DHQ: Digital Humanities Quarterly* 15, 1 (2021).

[23] Eric Kaltman, Joseph Osborn, Noah Wardrip-Fruin, and Michael Mateas. 2017. Getting the GISST: A Toolkit for the Creation, Analysis and Reference of Game Studies Resources. In *Proceedings of the 12th International Conference on the Foundations of Digital Games* (Hyannis, Massachusetts) *(FDG '17)*. Association for Computing Machinery, New York, NY, USA, Article 16, 10 pages. https://doi.org/10.1145/3102071.3102092

[24] Gregor Kiczales, John Lamping, Anurag Mendhekar, Chris Maeda, Cristina Lopes, Jean-Marc Loingtier, and John Irwin. 1997. Aspect-oriented programming. In *ECOOP'97 — Object-Oriented Programming*, Mehmet Akşit and Satoshi Matsuoka (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 220–242.

[25] Johan Kotlinski. 2021. Little Sound Dj. https://www.littlesounddj.com

[26] Jonathan Leung. 2022. VGMaps.com: The Video Game Atlas. https://vgmaps.com/

[27] Chris Maltby. 2022. GB Studio. https://www.gbstudio.dev/

[28] Cade Metz. 2016. In Two Moves, AlphaGo and Lee Sedol Redefined the Future. https://www.wired.com/2016/03/two-moves-alphago-lee-sedol-redefined-future/

[29] "Nightcrawler". 2022. ROMhacking.net. https://www.romhacking.net/

[30] Johnnemann Nordhagen. 2022. Museum of Mechanics: Lockpicking. https://dimbulbgames.itch.io/museum-of-mechanics-lockpicking

[31] Robert O'Callahan, Chris Jones, Nathan Froyd, Kyle Huey, Albert Noll, and Nimrod Partush. 2017. Engineering Record and Replay for Deployability. In *2017 USENIX Annual Technical Conference (USENIX ATC 17)*. USENIX Association, Santa Clara, CA, 377–389. https://www.usenix.org/conference/atc17/technical-sessions/presentation/ocallahan

[32] Alexey Pajitnov. 1989. Tetris. [Game Boy]. SHA256:0d6535.

[33] Jasper St. Pierre. 2022. noclip: A digital museum of video game levels. https://noclip.website/

[34] Redacted For Peer Review. XXXX. Redacted. https://example.com/redacted

[35] Daniel Reynolds. 2016. The Vitruvian Thumb: Embodied Branding and Lateral Thinking with the Nintendo Game Boy. *Game Studies* 16, 1 (2016).

[36] Arindam Sarma. 2020. *Abstracting AI Evaluation Environments with Virtual Machines.* Master's thesis. University of California, Santa Cruz.

[37] Phillip B. Shoemaker. 1999. Designing Interfaces for Handheld Computers. In *CHI '99 Extended Abstracts on Human Factors in Computing Systems* (Pittsburgh, Pennsylvania) *(CHI EA '99)*. Association for Computing Machinery, New York, NY, USA, 126–127. https://doi.org/10.1145/632716.632794

[38] speedrun.com. 2022. Metroid II: Return of Samus - speedrun.com. https://www.speedrun.com/m2

[39] Steve Swink. 2008. *Game feel: a game designer's guide to virtual sensation.* CRC press, Boca Raton, FL.

[40] Lisa Takeyama. 1994. The shareware industry: some stylized facts and estimates of rates of return. *Economics of innovation and new technology* 3, 2 (1994), 161–174.

[41] Mark Weiser. 1984. Program slicing. *IEEE Transactions on software engineering* SE-10, Issue 4 (1984), 352–357.

[42] Matthias Wenzl, Georg Merzdovnik, Johanna Ullrich, and Edgar Weippl. 2019. From hack to elaborate technique—a survey on binary rewriting. *ACM Computing Surveys (CSUR)* 52, 3 (2019), 1–37.

[43] Nick Westwood. 2022. Game Boy Homebrew Games: a collection by RetroBreak. https://itch.io/c/577395/game-boy-homebrew-games

[44] Ivo Wetzel. 2014. Tuff. https://bonsaiden.github.io/Tuff.gb/ SHA265:ca2822.

[45] Megan A. Winget and Caitlin Murray. 2008. Collecting and Preserving Videogames and Their Related Materials: A Review of Current Practice, Game-Related Archives and Research Projects. https://doi.org/10.48550/ARXIV.0811.3137

[46] Damian Yerrick. 2018. Telling LYs? https://forums.nesdev.org/viewtopic.php?f=20&t=18026 SHA256:968e1d.

[47] Nezer Jacob Zaidenberg. 2020. *Game console protection and breaking it.* IGI Global, Hershey, Pennsylvania. 449–461 pages.

[48] Xiaoxuan Zhang, Zeping Zhan, Misha Holtz, and Adam M. Smith. 2018. Crawling, Indexing, and Retrieving Moments in Videogames. In *Proceedings of the 13th International Conference on the Foundations of Digital Games* (Malmö, Sweden) *(FDG '18)*. Association for Computing Machinery, New York, NY, USA, Article 16, 10 pages. https://doi.org/10.1145/3235765.3235786