CHAPTER T-1

Twine as Platform

We have introduced Twine, and indeed, Twine has more widely been studied, primarily through the works it enables. This raises a preliminary question: What is Twine for, and what do we call the things it makes? This simple question holds the shadow of a much larger history of definitional tension surrounding games and what "counts." More importantly, who gets to decide what—and, by extension, who—counts as part of the discourse of game design? The shadow of a history of misogyny, exclusion, racism, labor abuses, and general awfulness in video game culture looms large over this question. We will wrestle with Twine's place (and our own) in this history throughout our study of Twine, and if you engage in the making of Twine works as our practical chapters invite you to do, you, too, might find yourself facing questions of where your work fits—and what it should be called.

In academic circles, the desire for definitional clarity might be understood through the discourse of formalism or (broadly) the placement and understanding of a work according to its structure. And formally speaking, Twine works are near immediately recognizable unless their creators go through significant work modifying the final interface: while each story format (a set of rules that overlay Twine's central logic) has its signature interface design, the general structure of

passage-driven, hyperlinked narrative holds. The two generations of the Twine editor are fundamentally similar, as the dominant metaphors remain consistent, but they differ in the details. Rather like a branching Twine narrative, the history of Twine and its significance as a platform is a threaded, nonlinear tale, and how we tell it depends on where we begin in defining Twine works. This is no small decision, so before we make it, let me move toward a definition that will resist completion and finality as we move through this work. The front page of www.twinery.org offers a straightforward summary of the platform that notably makes no mention of games, a deliberate omission we will return to shortly:

> You don't need to write any code to create a simple story with Twine, but you can extend your stories with variables, conditional logic, images, CSS, and JavaScript when you're ready.
>
> Twine publishes directly to HTML, so you can post your work nearly anywhere. Anything you create with it is completely free to use any way you like, including for commercial purposes.
>
> Twine was originally created by Chris Klimas in 2009 and is now maintained by a whole bunch of people at several different repositories. (Klimas, "Twine")

This initial definition suggests that Twine works are, most fundamentally, stories. The reality is more complicated. Twine has been used recursively as a tool to build tutorials; rhetorically as a tool for arguments and essays; abstractly for poetry and generative art; and educationally for making materials across disciplines, to name only a few instances.

More recently, the Twine Cookbook, maintained by Dan Cox, breaks down these features into usable demos across the many versions, or formats, of Twine. The Cookbook notes that the terms used in Twine are intended to be not limitations but opportunities: "Anything made using Twine can be called by any name. They are no rules on naming conventions and everything from experimental games to more traditional novels can be created in Twine. Everything is welcome. In

general, the Twine editor calls individual projects Stories" (Cox, "iftechfoundation / twine-cookbook").

When we call Twine a software platform for the development of games and interactive stories, we risk being simultaneously reductive and overgenerous with our description: not all things made with Twine fall into these easy categories, and as a software platform, Twine can make pretty much any genre of interactive text the user envisions. The term *software platform* evokes Lev Manovich's discussions of the power of "cultural software" in shaping (and allowing users to shape) culture, from Adobe Photoshop and Flash to Microsoft's Visual Studio (Manovich 3). Within this space, Manovich argues for the need for software studies focusing on a range of categories of application: within his hierarchy, Twine falls perhaps most easily into the category of "media software" or content creation software broadly (Manovich 24). It is in its resemblance to the tools of this category (graphical interface–driven metaphors of making) rather than the programming-driven category that Manovich describes as falling outside of this mainstream thanks to the dividing line of code: "Today, a typical professional graphic designer, film editor, product designer, architect, music artist—and certainly a typical person uploading videos to YouTube or adding photos and video on her/his blog—can neither write nor read software code. (Being able to read and modify HTML markup, or copy already prepackaged lines of JavaScript code is very different from programming)" (Manovich 31).

The dismissal of basic web development as "different" from programming in Manovich's parenthetical is notable, particularly as the book *Software Takes Command* was published in 2013—the same year public media attention was drawn to Anna Anthropy's "Twine revolution," where she offered an explanation of Twine's appeal that similarly put it in a category separate from programming: "This last year . . . people have really adopted Twine, which is a free tool for making text games. And aside from being free, it's really not programming at all—if you can write a story, you can make a Twine game" (Ellison).

Both Manovich and Anthropy draw a line around programming, a term that carries with it heavy baggage of gatekeeping and a recent

history of exclusion (again, the shadow of who counts—and who owns—the culture of Silicon Valley and its global influence looms large). They focus their gazes on something else: the type of cultural software that allows anyone to make, presuming some foundational digital literacy. This association of Twine with the absence of programming is, of course, illusory: as you will experience in the practical chapters, Twine is entangled with code, and the code is at some level inescapable. "Code" itself has many layers of meaning and nuance: markup languages such as HTML primarily annotate and structure content, while scripting languages such as JavaScript center on interactivity. Twine adds its own layers over both, but in simplifying, it also imposes its own new structures and abstractions. While the graphical user interface (GUI) significantly draws the user into hyperlinked visual making, the passage boxes awaiting content must ultimately be programmed in that strict rules must be followed to ensure readability following the procedures of Twine's underlying machine. This contradiction is at Twine's heart: it is a piece of cultural software that allows a user to build complex interactivity toward many ends, and it invites the user into a rabbit hole of complexity where the entryway is paved with language, not code. As the user moves forward, Manovich's dismissal might even be reassuring: this disguised HTML, and precorralled JavaScript, is not programming at all.

This allure of Twine is, of course, not true at a fundamental level: Twine is code, and Twine-making is programming, but its structures are designed with user experience at the forefront. The tension between what Twine makes easy and what Twine makes possible is immense, and as is common to communally supported software projects, the complexity of entry to Twine has risen with its increased versatility even as the variety of entry points and tutorials available has also grown, complete with whole texts dedicated to learning Twine: Melissa Ford's *Writing Interactive Fiction* and Anna Anthropy's *Making Games with Twine*. Both of these books are notably aimed at the game development community of Twine.

Twine and Games

Though the definition of Twine provided on its own website omits the term *games*, a history of the platform that begins with usage (or starts in the tutorials, textbooks, and examples that have gained notoriety) cannot escape the term *games*. This term is hotly contested, all the more so as of 2019, as I type these words five years after the anniversary of Gamergate (whose specter haunts Twine and this work). In their provocatively titled book *Real Games*, Mia Consalvo and Christopher Paul examine the contested definitions of *game* and its impact on what gets studied, critiqued, and ultimately preserved: "Game studies academics are themselves variably interested in what constitutes a real game as a way to legitimate the field and define an area of study. What gets left out of structuralist arguments is the value judgment going into labels such as game or not game. If something is not a game, then it is decidedly less important from the field's perspective" (Consalvo and Paul xxv–xxvi).

In opening this volume, it is tempting to position Twine as a games platform and to categorize Twine works as games. Developing such a common framework would give us the language of games studies for addressing Twine's value—and critiquing its structures—but more urgently, it would also give us an easy case for Twine's significance. Such a claim would likely not go uncontested for long: in an entire volume dedicated to reclaiming and examining cases of "not games" ranging from Facebook games to walking simulators, Consalvo and Paul do not mention Twine or even interactive fiction. The "not games" they identify as edge cases are in many ways closer to gamelike expectations than Twine works. This is not to say there has been no intersection of this discourse: indeed, there is a fundamental awareness of formalism embedded in Twine. Twine creators have wrestled with the question of the platform's game-ness and, in doing so, give us an entry point into positioning Twine as a form.

The extremism with which the word *game* is regulated inspired the Twine metawork *Is This a Game?* released by the Game Police in 2013. The work asks users humorously to consider the degradation of language that might result if the player calls the work in question a game.

This "linguistic singularity" path, if pursued, results in the player faced only with the word *game* repeating meaninglessly. Commenting on the game's message, critic Steve Haske observed, "Meanwhile, you can choose to change your mind, rescinding your decision to call this thing a game. It creates an interesting food-for-thought Catch-22: if you opt out, then you haven't just played a game. If you don't, you may not have the 'game' experience you thought you would (though you can confusingly find an inherent design)" (Haske).

Figure 2 captures the rhetorical style of the game (or not game), which deliberately uses a mostly unmodified version of the Twine Sugarcane style sheet, capturing the aesthetic associated with Twine most widely at the time. The game later takes this a step further, demonstrating how its meaning eventually collapses under the weight of the word *game* by literally replacing all other previously displayed text and offering only the same word as a choice. While perhaps unsubtle, this work is very much a product of its time, offering a playable entry point into the controversy over whose work counts in the game world.

This controversy was not one with merely academic stakes. In the same year, the literal "game police" of Steam Greenlight were deciding

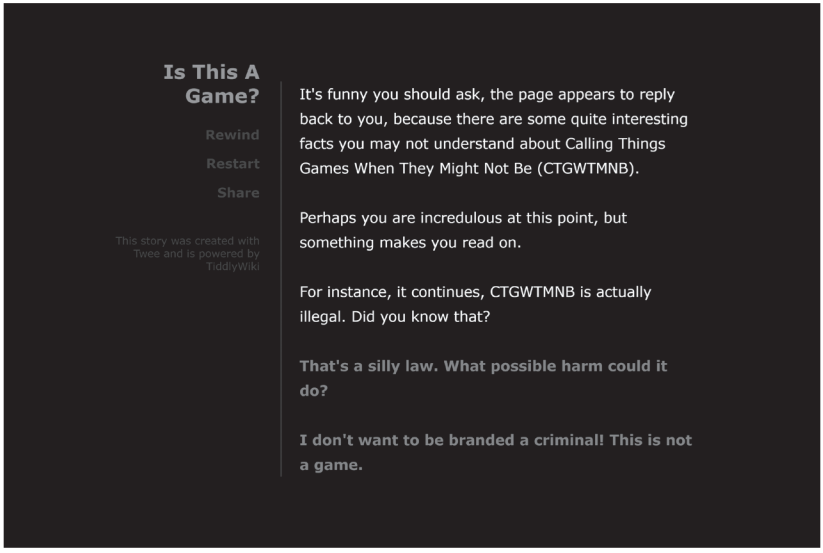

Figure 2: *Is This a Game?* escalates questions of formalism

whether a Twine game, *Depression Quest*, could be included on the game storefront. Being present on Steam opens up a market of opportunities for a designer, and Zoë Quinn's work, tagged by them as "interactive (non)fiction," would be the subject of hostility and debate. While the game was released in 2013 and had already been recognized as a game within independent spaces (including winning Best Narrative Game at Boston FIG and Official Selection at Indiecade 2013), one of the most popular discussion threads on the Steam Greenlight page asked, "Can this be counted as a game?" (Quinn).

Also in 2013, as questions of game-ness were rising around the independent-developer scene, critic Leigh Alexander offered a provocation on Twitter in defense of the type of experiences represented by *Depression Quest* and other works: "When people say games need objectives in order to be 'games,' i wonder why 'better understanding another human' isn't a valid 'objective.' . . . Games need 'challenges' and 'rules,' isn't 'empathy' a challenge, aren't preconceptions of normativity a 'rule'[?]" (Alexander). Alexander's tweet was not well received and escalated the debate as others joined in defending the formalist approach as essential to drawing lines to define the object of study.

Designer and critic Raph Koster responded to this provocation with a blog post entitled "A Letter to Leigh" that, among other critiques of noninteractivity, asked why the games don't in turn show more empathy for him as a player: "But I also find myself looking to the future, where I hope the games have empathy for the player, rather than the other way around, because it is a far harder artistic, and empathic, challenge to understand an opposing point of view than it is to present one's own. I'll be entertained by a rant I agree with, and angered by a rant I don't, but a debate is far more likely to change my mind" (Koster).

Darius Kazemi's "On Formalism" offers a playable response to that letter, taking a quote from Koster and centering it on the screen while offering a critique of Twine as a platform through code. It opens unassumingly, presenting as a classic Twine 1.X[1] work with the hallmarks of

1 For readers not familiar with this software naming convention, 1.X indicates any of several serialized releases in the first series of an application (1.1, 1.2.3, 1.999, etc.), 2.X indicates any release in the second series, and so on.

SugarCube (a story format for Twine with a side bar and restart button). Press "click to continue," however, and the screen breaks free. The passage starts to move, and the player's clicks turn into a weapon gradually reducing the words to nothing (as shown in figure 3). When the game was posted by Porpentine to the *Free Indie Games* blog in 2013, it inspired a debate about the definitions of interactivity and dialogue (Porpentine, "On Formalism").

In these critiques, formalism is a stand-in for the larger debate of where games begin and end—a debate that is used primarily to exclude and gatekeep—that also asks us to question our relationship with games and the assumptions that the very term makes us bring to our interactions with a work. Game designer Robert Yang's own blog post in response to Koster's (entitled, recursively, "A Letter to a Letter") further highlights the problematic aspects of Koster's claim, which Kazemi's work makes playable (while resisting "dialogue"): "I do think that you imply that this inability to separate content from form is an inherent (formal) weakness of personal games and the ways they mean things. That, because these games can't fit into a formalist frame,

Figure 3: The weaponization and transformation of "On Formalism" after clicking

they are thus less gamelike. Instead, I'd argue that this is a weakness of a traditional formalist approach: mechanics are often boring / limit what authors can do with games. . . . 'Dialogue,' on an oppressor's terms, rarely results in empathy" (Yang).

In labeling something as a game, we might limit it; similarly, in labeling something not a game, we potentially exclude it, and its creators, from the discourse of games and what games might be. These challenges are worth pausing on here, as we operate with an awareness and interest in Twine's place as a hypertextual platform but also see the undeniable significance of claiming this platform in the name of games: not just for what it does for Twine but for what it does for games. It is no coincidence that these definitional debates accompany the cultural challenges to Twine, which we will examine in more detail throughout this book. Given these tensions, I want to stress that in examining Twine as a platform, we are not taking a formalist lens; instead, we want to consider how Twine's affordances have played a role in making certain types of experiments easier. To start with a simple claim, many of the games and experiences that have been made in Twine center on the personal, and the platform's affordances seem to map well to expressions that put the mechanics of choice—and denial of choice—at the forefront.

In her examination of the framing of Twine as a game platform alongside these tensions, Alison Harvey notes that the community's fundamental ethos plays a major role in framing the type of work produced: while many game tools offer tutorials based on shooting and conflict, Twine collections and tutorials place their emphasis on "a different set of preferred affordances" (Harvey 98). Look at the tutorials for Stencyl, GameMaker, Unity, and so forth: mechanics of movement, violence, and acquisition dominate the expressive palette, pushing early designers to imitate that which is already established (as broadening the vocabulary of a graphical game is a different challenge than empowering verbs in Twine). Similarly, game designer and Twine luminary Mattie Brice expressed her thoughts on games as objects in response to these debates:

> There is much to be said in the way of a game's form. How is it structured, and how does that structure make a difference? Let's say someone

> submits something that doesn't look like a poem to a poetry contest. The judges don't necessarily go "This isn't a poem, therefore, it is not worth considering." Rather, the form itself critiques the established genre, it says "I'm a poem, and what are you going to do about it?" The formal genres in writing are for convenience only—ultimately, the kind of criticism needed for flash fiction, prose poems, short stories, novellas, and novels, is ultimately one in [*sic*] the same. Maybe everything is really just poetry. Boundaries, bones of old men before us, are only there to be transgressed. (Brice)

As a platform, Twine inherits this contradiction. It is structurally familiar and formally suggests so many antecedents that it does not at first glance appear transgressive—and yet it transgresses and transforms.

Transforming Hypertext

If we limit the lens of Twine as a platform to games, we ignore the other spaces that Twine has transformed, including hypertext itself and interactive fiction more broadly. Drawing on interviews with developers and community members as well as the embodied history of Twine within forums, mailing lists, and the code database, we will position Twine as a communal, open-source project. Positioning Twine alongside other platforms of the web (including precursors HyperCard, Storyspace, and Flash) offers insight into Twine's significance, which is not only a matter of interface and affordances. We will consider Twine's positioning within communities such as Glorious Trainwrecks, Tumblr, itch.io, and Philome.la and how the circulation and discourse within these spaces have shaped Twine's life-span and influence. Astrid Ensslin and Lyle Skains observe that Twine's rise is a rejection of exclusivity and platform control enabling a "writerly reader," or "(w)reader": this "(w)readerly empowerment through co-creation of narrative meaning cannot be imposed through forms, texts, and theories that imply exclusivity of access and assume that deconstructivist thought can be implemented through manifest literary materiality. Instead, movements like the Twine community and participatory social media writing have shown that genuine

wreadership has to come from users themselves, driven by the aesthetic and social needs of their own communities . . . and the desire to get published as an experimental creative writer" (Ensslin and Skains). While Twine is "owned" in a sense—and, with the increased control of the Interactive Fiction Technology Foundation (IFTF) over its future, communally owned—it is not a closed or corporate platform, and its output is entirely open to reverse engineering, making it a purer form of hypertext in that it fundamentally compiles into open web standards.

Twine is responsive to its moment and the platforms that precede it: most of the earlier platforms are united by their reliance on proprietary, corporate-owned technologies. The web is littered with the unplayable or occasionally emulated remains of works built on these platforms: Apple's HyperCard, perhaps the first popular hypertext platform, vanished as the company shifted direction; Eastgate's Storyspace supported hypertext works sold on removable media that are now almost entirely unplayable; and so forth. The proprietary ecosystems and walled gardens of currently popular ecosystems for games and electronic literature, such as iOS and Android, are similarly fraught with demands for continual updates that, if unmet, render work unplayable. By contrast, the web's standards have been relatively reliable. It would be an exaggeration to say that HTML is still what it once was—open a browser source of the original HTML and HTML 5 and compare, and the tags are certainly similar, but the act of translation required is daunting. Fundamentally, we rely on this backward compatibility: we assume that all other platforms might fail us, but the web lives on. The dominant force in emulation is the Internet Archive: Jason Scott and his team have made it possible to reexperience many of the works created on platforms that have fallen by the wayside.

There are many visual entry points into hypertext, but most of them bring with them expectations of a corporate purpose or information architecture–driven organization system. Adobe Dreamweaver, with its drag-and-drop interface elements and GUI-driven editor, is in stark contrast to the playfulness of Flash. The WordPress interface (and similar content management systems) emphasizes a separation between form and content, offering modifiable themes and blocks of content

that do not easily lend themselves to narrative. Meanwhile, opening up an .html file and starting from scratch can quickly become a logistical nightmare when it comes to tracking: nonlinear work requires fragmentation, and those fragmentations require significant marking with IDs (and tracking of past links) to navigate. In an early reflection on hypertext literacy, "Nonce upon Some Times," Michael Joyce notes that the "paradox" of hypertext relies on rereading, and that same rereading makes development difficult without a dedicated tool for visualizing the work:

> Hypertext fiction in some fundamental sense depends upon rereading (or the impossibility of ever truly doing so) for its effects. Yet in a sufficiently complex and richly contingent hypertext it is impossible to reread even a substantial portion of the possible sequences. Indeed for any but a reader who has consciously blazed his way through the thicket (breadcrumbs, in fact, have become a technical term for computer tools designed to keep track of the reading of hypertexts) it is unlikely that successive readings by a single reader will be in any significant way alike. Even in less vigorous hypertext systems such as current instantiations of the World Wide Web, bereft of the systematic memory that shapes possible readings, the linked surfaces of possibility themselves compound. (Joyce)

Joyce writes in the earliest stages of hypertext, before the link became utilitarian and familiar, so transparent as to become unremarkable. However, he draws our attention to the ways hypertextual linking can be playful, creative, and confounding, defying the utilitarian future of the web.

Reconsidering Joyce's concept of hypertext, and particularly his emphasis on defining the link, Emily Short notes, "From the perspective of more than twenty years later, many of Joyce's observations feel like first pen-and-paper cartographical attempts on a territory that has now been explored very extensively on foot" (Short). The type of nuanced links that Joyce and Short describe were not built into the initial Twine but evolved thanks to user-developers pushing Twine's utility forward.

Among those, one of the most significant developments is the cycling link, a structure that allows the user to click and replace a piece of text repeatedly from a set of options prewritten by the designer. (We retrace this evolution in the next practical chapter, concentrating on textual variation.) Porpentine documented the impact of Leon Arnott's cycling link macro in a blog post examining *Candy Ant Princess*, a game by Whisperbat that makes extensive use of the system to allow the player to make aesthetic choices that occasionally impact play. As Porpentine summarizes, this makes the difference between creating passages for every link and treating links as choices directly, as shown in figure 4 (Porpentine, "Live Free, Play Hard").

Though this will seem an odd observation in a book about Twine, in many ways, Twine appears unnecessary. It is an interface built on top of hypertext: everything that can be accomplished in Twine can be accomplished with HTML and JavaScript, albeit with more difficulty. Thus considered as a platform, Twine is not about the resulting work; it is entirely about the means of production. At the same time, Twine's particular mechanisms (the things that each generation, and each story format, makes easy) transform the resulting work, with cycling links as

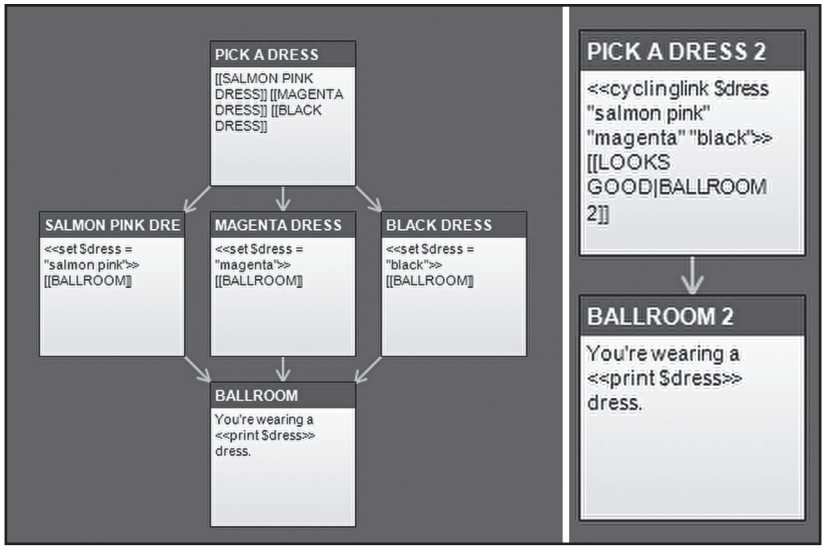

Figure 4: Choosing clothing in Twine 1.X with and without cycling links

just one example of the expanding vocabulary Twine offers with each iteration. Twine's tools allow creators to create new poetics of the link and particularly allow someone who might use Twine next to pick up and expand those poetics with relative ease. The consistency of the interface shapes the experience and the player's expectations for what a Twine work can be.

Twine's construction as a tool for facilitating a type of web production is far from unique. WYSIWYG (What You See Is What You Get) web editors have been around for almost as long as websites themselves, from the built-in tools of community hubs such as GeoCities and Angelfire to the unwieldy, mangled-code-generating FrontPage and Dreamweaver, but without the writerly metaphor that Twine's story assumption foregrounds. All this is a roundabout way of saying that what Twine makes easy is superficially simple but difficult in practice. The shortcuts are not just pragmatic code solutions but also visual navigation, and thus the most crucial element of Twine for many users is the GUI itself. To recall Steven Johnson's *Interface Culture*, the link is the center of hypertext, the first marker of meaning-making in digital navigation whose current ubiquity makes us forget its initial impact: "Ask any Web user to recall what first lured him into cyberspace; you're not likely to hear rhapsodic descriptions of a twirling animated graphic or a thin, distorted sound clip. No, the eureka moment for most of us came when we first clicked on a link, and found ourselves jettisoned across the planet. The freedom and immediacy of that movement—shuttling from site to site across the infosphere, following trails of thought wherever they led us—was genuinely unlike anything before it" (110).

It is this origin point of the link where we find Twine: a realization of the "freedom and immediacy" of Johnson's web, built on top of not a game interface but a hypertextual one that would shape its affordances and the expectations of its users going forward.

Contextualizing Twee

A short history of Twine starts not with Twine itself but with a scripting language called Twee, based on an earlier writing system called

TiddlyWiki, a tool for creating user-modifiable hypertexts, or wikis. Twee lacks the visual interface of Twine but encodes the fundamental underlying mechanics: it is the scripting language that precedes the graphical interface. In the interview he gave for this book (appendix I), Klimas described TiddlyWiki's self-modifying codex as too limited for the types of hypertexts he wanted to create—thus he set out to visualize a better tool, which would become Twee:

> I ran across this technology called TiddlyWiki, and it was this really clever thing where it was this self-modifying Web page. . . . You download it to your computer, you can edit it, it's like a wiki but there's no server component to it at all, and so it's like a very simple . . . DIY hypertext. And so I started editing and playing out stuff in there and experimenting with that medium . . . and it just got very disorienting, actually, to try to edit it from inside . . . where I'd click links, and follow them, and it's like—where am I? And so I'd get lost in my own stuff, and that was sort of the genesis: I want to build a tool that will help me do this better. (Klimas, appendix I)

In his oral history of Twee, Dan Cox notes that several derivative works perform a similar function, using output to bridge to other formats: Cradle, or UnityTwine (2015); Yarn (2015); Tweego (2013); Twee2 (2015); Entweedle (2015); and Entwee (2016); among others (Cox, "An Oral History of Twee"). The range of these should not necessarily be confused with influence—many of these projects are driven by the needs of their developers.

TiddlyWiki creator Jeremy Ruston remembers the allure of early web wikis that inspired his design of the system and particularly the idea of breaking out a single document model of development, which in turn would inspire Twee and Twine:

> The allure of the wiki for me was the feeling that it could eventually disrupt the prevailing paradigm of print-oriented documents and emails.
>
> After watching people use wikis for a few years, I noticed that power users made extensive use of the ability to open multiple wiki pages at

> once in several browser tabs, making it easier for them to compare and review pages, to copy text between them and to act as a sort of queue of pages yet to be read.
>
> I felt that this ability to manipulate multiple pages at once was central to the ability to refactor a wiki, and it is generally accepted that a wiki that is lovingly refactored tends to be more useful. And yet, standard wiki user interfaces have always been designed exclusively for the presentation and manipulation of single pages at once. (Ruston)

The newest iteration of TiddlyWiki maintains that concept of nonlinear organization, inviting users with the question, "Have you ever had the feeling that your head is not quite big enough to hold everything you need to remember?" (TiddlyWiki). This evokes similar nonlinear tools built around organization, such as Evernote and Microsoft OneNote. The original Twee website maintained by Klimas in 2005 explains Twee's origin as emerging from his desire to have a text-driven interface for working with TiddlyWiki:

> Twee is a supersimple markup language for TiddlyWikis. It was invented when Chris spilled water on his laptop's trackpad, which knocked it out of commission temporarily, and he still wanted to work on his TiddlyWiki.
>
> In short, Twee lets you turn plain text files that look like this:
>
> ```
> :: Twee [systemConfig]
> Twee is a supersimple [[markup language]] for
> ~TiddlyWikis.
> ```
>
> . . . into living, breathing TiddlyWikis. Right now, it allows you to target the latest version of TiddlyWiki, TiddlyWiki 1.2.39, Twinkie, and iPods. It also includes untwee, a tool that converts existing TiddlyWikis to Twee source code. (Klimas, "Code and Other Oddments")

The legacy of the text version comes through in the earliest graphical iterations of Twine. In Twine 1.X, users were not by default introduced to the concept of incorporating Cascading Style Sheets (CSS)

and JavaScript into their work. Jane Friedhoff describes Twine's original user interface as using the "corkboard paradigm," which essentially means that Twine offers the same visual space and freedom as rearranging materials on a corkboard, including ease of movement and the ability to get a big-picture perspective. She notes, "This kind of visual, spatial practice is relatively rare in the coding world (outside of patching languages, such as MaxMSP), but it is very similar to the way many writers plan and organize their stories" (Friedhoff).

The passage interface did not distinguish between, or provide a specific space for, adding such code, leaving users to follow tutorials to incorporate "tagging" to mark special passages for this purpose. Twine 2.X is more elaborate in its initial assumptions and includes by default a separate style sheet and scripting area, with boilerplate guidance for incorporating CSS to properly link to the tags and structures of Twine. Twine 2's most dramatic innovation is the browser-based editor, which offers the next level of accessibility for users unable to install software. The 2.X editor is still not ideal for use outside of a desktop or laptop computer (similarly, Twine works are mobile-passable but only mobile-friendly when intentionally modified by the designer with the touchscreen user in mind). This shift reflects a shift in assumptions about the use cases for Twine, which have over time fallen more into classroom usage as well as general interest as an introductory development tool.

The story formats—or rulesets and paradigms that provide different ways of making in Twine (discussed in more detail in chapter P-1)—included in Twine 1.X are Jonah, Sugarcane, and Responsive. Later, SugarCube and Snowman would appear. By far the dominant story format for designers was Sugarcane (and by extension, SugarCube): the other two pushed more specific aesthetics onto users. The format called Jonah emphasizes the single page, requiring text that is designed to stretch and accrete rather than replacing one passage with another. Twine 2 removes the awkwardness of Twine's original distinction between a .tws source file and the .html output, removing the need for tracking and preservation of both the Twine editor's code and the browser's readable output. From a preservation standpoint, this strengthens Twine's longevity, as the final .html is its own complete

archive. Notably, a tool for reverse engineering .html works in Twine 1.X now exists, which eliminates the need to have the source file in order to investigate the complete work. As a platform, Twine is thus continually expanding outward in the hands of its users. The Twine 2 Monogatari story format, for instance, allows users to build from the Twine syntax to create web-friendly visual novels (Pinheiro).

Twine's Spread

Traditionally, platform studies approaches have examined corporate-controlled ecosystems, usually regulated by the producer of the hardware or software in question (Bogost and Montfort). Open-source platforms raise different questions and simultaneously offer clearer attribution, thanks to the documentation of contributions on platforms such as GitHub and murkier ownership and control. Friedhoff notes that the lack of a regulated distribution model is essential to the success of Twine's more queer, erotic, political, and otherwise experimental titles that would likely not make it past the review standards of most other platforms (Friedhoff). Simply put, most Twine works would not—could not—exist in the wild without Twine's self-distributed, easily spreadable modality.

The Interactive Fiction Database (IFDB), an archive of digital writing now sponsored by IFTF, primarily chronicles work by designers who came in contact with or embraced the term *interactive fiction* for their work, while the curation blog *Free Indie Games*—founded by Terry Cavanagh and featuring several developers, including Porpentine—offers a counterhistory heavily emphasizing Twine from 2012 to 2014 (*Free Indie Games*).

Probing the history of Twine through the IFDB returns a few false starts: humorously, Anna Anthropy's Twine iteration of Nintendo Power's 1990 feature *Dragon Warrior Text Adventure* is listed under this date despite having been published in 2013. The game is primarily notable for its nostalgia (Nintendo Power and Anthropy). While this is one of the more extreme examples of Twine as a preservation/emulation tool, this trend continues in the second earliest publication noted: the 2006

entry for *Escape from the Crazy Place* refers to the 2017 version of what the authors call "a preposterous blob of literary jelly" that has previous lives in physical text, classic HTML, and the 2006 version in the Text Adventure Development System (TADS; Guest and Etheridge). The work is also an instance of collaborative Twine-writing, as its lead author describes in the Glorious Trainwrecks post announcing the Twine version:

> Written over 33 years, Escape from the Crazy Place is a sprawling TWINE game with over 90,000 words of text. It is also an example of exquisite corpse writing, combining the talents of around twenty different authors. Some wrote just a passage or two, others wrote dozens.
>
> This new TWINE version was originally intended to be a trimmed-down, more polished version of the 2006 TADS 2 version, but myself and my friends Loz Etheridge and Mark Bailey got a bit carried away, and somehow or other the 2017 version ended up being two-and-a-half times the size of the original. The game will continue to expand as I intend never to stop adding to it. (Guest)

TADS, originally released in 1988 and last updated in 2006, stands in contrast to Twine in its code-focused approach that resembles the programming language C. It also offers a different, code-grounded vision than another system often mentioned in the same breath as Twine: Inform 7. Created in 1993 by Graham Nelson, Inform 7 is Twine's most popular cousin in the interactive fiction arena. Inform 7 boasts the appeal of "natural language processing correlation between system and output" that has been noted in early interactive fiction platform studies: Alex Mitchell and Montfort note that both TADS and Inform, the dominant interactive fiction creation platforms as they were writing in 2009, are driven by software objects and that the model of object-oriented, category-driven programming, in turn, suggests a "simulationist" approach to design (Mitchell and Montfort). Mitchell and Montfort draw the term *simulationism* from the interactive fiction community, defined as "the tendency towards deeper and less abstract simulation of physical (and possibly emotional) properties of the game

world, not for limited domains that the author has chosen, but as a general framework," with corresponding challenges for development: "Interactive fiction systems already face the problem of generating human-like text to describe situations arising in games. The list of objects in a drawer is generated from the underlying world model. The problem with simulationist IF is that this becomes a magnitude more complicated" (Mitchelhill).

Twine is the antithesis of this model. Originating in JavaScript (a non-object-oriented programming language that arguably has become more object-oriented as a result of increased pressures on web interactivity), Twine lacks the strict structures and classes of its C-esque counterparts. While it is possible to build a world model within it that might be termed *simulationist*—see our discussion of Porpentine's *With Those We Love Alive* in chapter T-3—such development is not built into the system in the way that Inform 7 has responded to the needs of designers building complex world models.

Other platforms for the creation of interactive fiction are more pragmatic in their approach to potential users, recognizing that a knowledge of programming is required to progress in developing with their tools. To return to Inform 7, its system visuals are secondary to text, and the authoring of natural language follows the most orderly rules of coding: while the blank page of an open Inform 7 game might look like a Microsoft Word document at first glance, freedom of writing style exists only inside the quotation marks that delineate strings. The rigor of the language is necessary for Inform 7's primary metaphors—the designer must first create the world and then define the rules by which the player might interact with that creation. Thus the structures of basic Inform 7 look like sentences but follow predetermined rules, as in this example:

```
The Office is a room. The description of Office is "Despite all your best
intentions of cleaning, the office is covered in papers, none of them
useful." The desk is a supporter in the Office. The laptop is on the desk.
    Instead of booting the laptop:
    say "The last thing you want to do is see the
    state of your emails."
```

Note some of the conventions: quotation marks indicate a string, or a sequence of characters that the language will not attempt to parse and understand. All the other sentences must be readable to the parser: words such as *description* and *supporter* are defined in Inform and create certain properties. The "instead" rule allows the system to intercept certain verbs and respond—so if the player tries to type "boot the laptop," the phrase after it will appear to discourage them from continuing down that path of action. Once broken down, the structures and demands of the language on the writer become apparent immediately (even before the would-be creator descends into the more clearly programmatic metaphors of data structures, logic, and event-driven "scenes" that enable a complex state of play). Mitchell and Montfort end their analysis of Inform and TADS with a reminder that "it is useful to consider the less-than-obvious ways in which these systems might influence the shaping of stories and worlds" (Mitchell and Montfort). To extend this argument, I noted that it is necessary to consider the less-than-obvious ways in which these platforms are reconstructing game culture.

The original Twine macros reveal the code intensity behind the extension of links in the early formatting and syntax of Twine's vocabulary. To return to the poetics of the cycling link, the macro was described in its creator's introductory post on Glorious Trainwrecks as a simple enhancement: "This simply produces a link whose text cycles between a number of values whenever you click on it. It otherwise leads nowhere. You can use it as a silly clicky trinket, a cheap alternative to the <<replace>> macro, or (as detailed below) as an input interface element" (Twine).

In 2012 (right before Twine's rise on the scene), Montfort and Short noted in their examination of the state of interactive fiction that the move to the browser was driving pushes for change in platforms that typically had ignored and standardized the aesthetics of the user interface:

> Presenting IF in a browser window generates its own new set of player and author expectations. Typography and text styling has for a long

> time been at best a secondary concern: interpreters on different operating systems present text in different ways, in different fonts, colors, and marginal arrangements. Traditionally, the tools used by the IF community have offered the author only limited control over this presentation. Portability across a large number of platforms (including small-screen mobile devices and computers being run with a screen reader by blind players) was often considered more important than the ability to craft a specific visual experience, and providing an attractive textual surface was often seen as the job of the interpreter creator rather than the author of a specific game. (Montfort and Short)

We return to this question of the interface in more detail later in this work, with chapter T-5's examination of Twine's entanglement with camp aesthetics. However, it is important to note that Twine's rise as a competitor to other interactive fiction platforms comes from both the ease of making and the ease of spreading work.

Open-Sourcing Twine

Placing Twine's history alongside this other most dominant platform for writing interactive fiction, Inform 7, illuminates their important differences as well as their fundamental similarities as platforms driven by their user communities. At NarraScope 2019, the first conference hosted by the IFTF, both Klimas and Inform 7 creator Graham Nelson offered "state of the platform" talks to audiences of players, developers, and scholars.

Klimas addressed the past and future of Twine for his audience. This moment was part of a shift in Twine's history, as Klimas documented some of the challenges of Twine as well as his hopes for the platform's future in the hands of the organization. At the core of his aspiration is Twine's commitment to open-source and open-access. The open-source nature of Twine has not been without its consequences. In 2018, Netflix released "Bandersnatch," a groundbreaking episode of its *Black Mirror* series in which suitably equipped viewers could select links to determine the unfolding of the narrative—an embrace of interactive

video, or hypertext, or choose-your-own-adventure gaming, depending on perspective. Significantly, "Bandersnatch" also involved at least a glancing encounter with Twine. The "Bandersnatch" creative team has acknowledged using Twine, among other applications, in preparing the treatment (roughly speaking, the prototype) for the project (Rubin). Creator Charlie Brooker described Twine as the tool that assisted in his big-picture thinking for the episode: "Every time I had an idea I put it in a box, and you can move them around. It's a bit like making a giant patchwork quilt" (Rubin). In his NarraScope talk, Klimas reports reaching out to the "Bandersnatch" team but receiving only resounding silence. This reaction was probably predictable, given problematic claims of influence, authorship, and credit that crop up regularly in show business. There was, for instance, an ongoing lawsuit from the publisher Chooseco over the use of the choose-your-own-adventure concept, to which the company claims proprietary rights (Kaminsky).

Setting the "Bandersnatch" story aside, Klimas pointed to similar uses of Twine as a prototyping tool for professional, profitable endeavors ranging from the choose-your-own-adventure graphic novel *Romeo and Juliet* by Ryan North to the opening sequence of the game *Firewatch* but also pulled up a more stark testament to Twine's lack of financial support even as he showed this economic potential. (At the time, the Patreon to support Twine was at less than $800 a month.) This echoed discussions of financial realities in narrative games that are unavoidable: the conference opening keynote included shots of a game in progress, abandoned for being too expensive to viably complete. Nelson, speaking during the Q&A, noted, "I'm not doing anything to help that," reflecting on the type of ambitious game that the developer works on and the realities of the limitations of open-source tools: "We're making a really good box," but "every step you take along that road makes it harder to get access to what's outside" (Nelson).

Chris Klimas noted in his discussion at Narrascope that he is aware of the challenges that arise when the user drawn to Twine by the promise of no programming seeks more control over the logic of their play and hits the wall of code and assumptions that go with it. Reflecting on the question, "How do you assist people in getting over that wall?" Klimas

pointed to his then current work on the story format Chapbook, introduced at the conference. In his initial guide, Klimas planned a section labeled "Advanced" but expected that it would need a disclaimer: "You'll need to understand JavaScript." (We discuss JavaScript in passing in upcoming practical chapters and in some depth in chapter P-3, where an example explores the integration of JavaScript code within Chapbook.)

This decision reflects a constant tension at the heart of design work on the platform: the balance of ease of use and capabilities. To again evoke the spirit of the dearly departed Flash platform, the breakdown of this balance can cause users to flee to new platforms or encourage them to never upgrade—many users stuck with old versions of Flash not just because of financial investment but because of the learning curve that went with each iteration's significant extension of the base feature set into a more and more algorithmic world. This resistance is also a reminder of the incredible frustration that can await the artist and writer in a world of ever-changing and proprietary tools; by contrast, the open-source tool offers the hope of consistency or at least the promise of continued availability.

With that said, our venture into Twine as a platform will not be without its challenges, and the interface underlies a greater complexity than you might expect. Chris Klimas noted that even some of the most fundamental functionality of Twine is more complex for the user than it might at first appear: "Plugging images into Twine, which is a really basic idea, is hard. You have to understand how URLs work. There's the comfort and the size of the box" (Klimas, "Twine: Past, Present, Future"). Twine's "box" of utility is continually growing, from the 2006 Twee with its off-putting lack of graphical interface, to the first Twine GUI in 2009, to the 2014 Twine 2, which looked to the web as a workaround for the frustrations presented by the walled garden of the Apple Store and Android Market.

The reach of Twine is also increasing: Twine 2.3.1's downloadable version (functional on Windows, Linux, and Mac) has reportedly exceeded twenty-five thousand downloads. The Twine community as of Klimas's 2019 report included 3,000 members on a Discord chat and 2,300 on the unofficial subreddit. In his own assessment of Twine's

reach, Klimas observed three main groups using Twine, all with different needs. Creative professionals (mostly game designers) using Twine as a prototyping tool, from the Netflix "Bandersnatch" team to the writers of the indie game *Firewatch*, rarely release those early iterations but may acknowledge Twine in postmortems on their work. Educators such as the authors of this book are the second primary user group, with Twine's reach extending to classrooms in India and well outside of game design programs (frequently as an alternative to the traditional paper-writing research assignments of many disciplines). And of course, finally, Klimas noted the indie creators and the recognition their work has brought Twine in a range of communities. These voices range from those distributing work on itch.io and Steam to artists exhibiting at the Whitney Biennial. Klimas observed that reaching (and keeping up with) this audience presents its own challenges, offering, self-deprecatingly, "I'm really not cool, and these people tend to be really cool."

The Twine platform was "adopted" as a recognized platform by the IFTF, a decision driven by the need for maintenance and institutional support. The Twine committee of the IFTF consists primarily of the developers responsible for building the story formats and tutorials that power the Twine community: Leon Arnott, Thomas Michael Edwards, Dan Cox, M. C. DeMarco, David "Greyelf" Tarrant, Colin Marc (stepped down 2019), and Klimas. This team is notably less diverse than the set of indie artists we highlight throughout this work, and tensions between the community and the guiding developers can be high—as Klimas observed, people frequently blame an imagined "they" for changes in Twine rather than seeing open-source projects as authored by dedicated creators donating their time to the project. Code authorship is visible when discussing an open-source project like Twine, but in some ways, that leads to less of a sense of creative control. The economics of this model are perhaps unsustainable: challenges include basic finances, such as paying to become a registered or "signed" application to enable users to more easily install the Twine platform on their computers. Other ambitious goals, such as modernizing the development workflow to add a Twine package manager and collaborative tools, are likely out of reach and also raise their own questions:

If we cannot even easily define what Twine makes, who should—and will—decide where Twine goes?

In an inherently decentralized community, there are whole groups who use Twine but aren't part of the conversations about the future. The same challenges we face defining Twine's scope also make it difficult to plan its development road map, which, as Klimas noted, requires balancing the needs and requests of the experts versus the teachers and students working with Twine in classroom settings without code experience, who are thus perhaps less likely to post concerns and issues in GitHub. Even the decision about where to place resources changes the platform's reach. Klimas and collaborators abandoned a plan to move the Twine support forum to Stack Overflow, a popular website for coding support, given the emphasis on programming and the conventions that might be particularly daunting to newcomers. Previous community hubs, such as Google Groups and the Twine forums, have run into problems of spam and moderation, while gamer-favored platforms such as Discord and Reddit bring in whole new potentials for toxicity.

As we will argue throughout this work, Twine's in-betweenness is its strength: it is the source of the platform's influence and what makes Twine relevant in conversations ranging from the future of education to the unrealized potential of electronic literature to the need to transgress existing boundaries in games. As cultural software, it is itself hypertextual, linked into communities that may never themselves intersect. Its survival and evolution to this point, refusing to diverge toward a commercial approach, is both admirable and unusual and ultimately the source of Twine's revolution.

Works Cited

Alexander, Leigh (@leighalexander). "When people say games need objectives in order to be 'games,' i wonder why 'better understanding another human' isn't a valid 'objective.'" Twitter, April 8, 2013. https://twitter.com/leighalexander/status/321152113021448193.

Bogost, Ian, and Nick Montfort. "Platform Studies: Frequently Questioned Answers." *Digital Arts and Culture*, 2009. http://bogost.com/writing/platform_studies_frequently_qu_1/.

Brice, Mattie. "Triptychs." Mattie Brice's website, April 13, 2013. http://www.mattiebrice.com/triptychs/.

Consalvo, Mia, and Christopher A. Paul. *Real Games: What's Legitimate and What's Not in Contemporary Videogames*. MIT Press, 2019.

Cox, Dan, ed. "iftechfoundation / twine-cookbook." 2017. GitHub, 2019. https://github.com/iftechfoundation/twine-cookbook.

———. "An Oral History of Twee." Digital Ephemera, June 8, 2019. https://videlais.com/2019/06/08/an-oral-history-of-twee/.

Ellison, Cara. "Anna Anthropy and the Twine Revolution." *Guardian*, April 10, 2013. https://www.theguardian.com/technology/gamesblog/2013/apr/10/anna-anthropy-twine-revolution.

Ensslin, Astrid, and Lyle Skains. "Hypertext: Storyspace to Twine." In *The Bloomsbury Handbook of Electronic Literature*, edited by Joseph Tabbi, 295–310. Bloomsbury, 2017.

Free Indie Games (blog). "About." Accessed July 29, 2019. http://www.freeindiegam.es/about/.

Friedhoff, Jane. "Untangling Twine: A Platform Study." Proceedings of the 2013 DiGRA International Conference, 2013, 10.

Guest, J. J. "Escape from the Crazy Place." Glorious Trainwrecks, February 21, 2017. https://www.glorioustrainwrecks.com/node/6547.

Guest, J. J., and Loz Etheridge. "Escape from the Crazy Place." Interactive Fiction Database, 2006. https://ifdb.tads.org/viewgame?id=ny5d87fqbeh3pnuz.

Harvey, Alison. "Twine's Revolution: Democratization, Depoliticization, and the Queering of Game Design." *GAME* 1, no. 3 (2014). https://www.gamejournal.it/3_harvey/.

Haske, Steve. "'Is This a Game?' Forces You to Contemplate the Philosophical Definition of Games." Complex, June 9, 2013. https://www.complex.com/pop-culture/2013/06/is-this-a-game-forces-you-to-contemplate-the-philosophical-definition-of-games.

Johnson, Steven. *Interface Culture: How New Technology Transforms the Way We Create and Communicate*. San Francisco, CA: Harper, 1997.

Joyce, Michael. "Nonce upon Some Times: Rereading Hypertext Fiction." *Modern Fiction Studies* 43, no. 3 (1997): 579–97.

Kaminsky, Michelle. "Chooseco, 'Choose Your Own Adventure' Trademark Owner, Sues Netflix over 'Bandersnatch.'" *Forbes*, January 14, 2019. https://www.forbes.com/sites/michellefabio/2019/01/14/chooseco-choose-your-own-adventure-trademark-owner-sues-netflix-over-bandersnatch/.

Klimas, Chris. "Code and Other Oddments." Gimcrack'd, March 28, 2006. https://web.archive.org/web/20060328165735/http://gimcrackd.com/etc/src/.

———. "Twine: Past, Present, Future." *IFTF Narrascope Conference*, June 15, 2019. https://2019.narrascope.org/pages/schedule.html.

———. "Twine Is an Open-Source Tool for Telling Interactive, Nonlinear Stories." Twinery.org, 2019. https://twinery.org/.

Koster, Raph. "A Letter to Leigh." Raph Koster's website, April 9, 2013. https://www.raphkoster.com/2013/04/09/a-letter-to-leigh/.

Manovich, Lev. *Software Takes Command*. Bloomsbury, 2013.

Mitchelhill, James. "Simulationism and IF (Long)." Google Groups, October 1, 2005. https://groups.google.com/forum/#!msg/rec.arts.int-fiction/o-Y2qK8_KLE/Qrwmdv0L5k4J.

Mitchell, Alex, and Nick Montfort. "Shaping Stories and Building Worlds on Interactive Fiction Platforms." eScholarship, December 2009. https://escholarship.org/uc/item/6pk7s4n6.

Montfort, Nick, and Emily Short. "Interactive Fiction Communities: From Preservation through Promotion and Beyond." *Dichtung Digital* 41 (September 2012). http://www.dichtung-digital.org/2012/41/montfort-short/montfort-short.html#10.

Nelson, Graham. "Opening Inform." *IFTF Narrascope Conference*, June 15, 2019. http://emshort.com/narrascope/talk.html.

Nintendo Power, and Anna Anthropy. "Dragon Warrior Text Adventure—Details." Interactive Fiction Database, August 2, 2013. https://ifdb.tads.org/viewgame?id=vbs1pvv73c2p18i2.

Pinheiro, Haroldo de Oliveira. "haroldo-ok / twine-monogatari." GitHub, 2019. https://github.com/haroldo-ok/twine-monogatari.

Porpentine. "Live Free, Play Hard: The Week's Finest Free Indie Games." Rock Paper Shotgun, April 28, 2013. https://www.rockpapershotgun.com/2013/04/28/live-free-play-hard-the-weeks-finest-free-indie-games-26/.

———. "On Formalism (Darius Kazemi)." *Free Indie Games* (blog), April 25, 2013. http://www.freeindiegam.es/2013/04/on-formalism-darius-kazemi/.

Quinn, Zoë. "Steam Greenlight: Depression Quest." Steam, December 4, 2013. https://steamcommunity.com/sharedfiles/filedetails/?id=200770535.

Rubin, Peter. "How the Surprise New Interactive *Black Mirror* Came Together." *Wired*, December 28, 2018. https://www.wired.com/story/black-mirror-bandersnatch-interactive-episode/.

Ruston, Jeremy. "History of TiddlyWiki." TiddlyWiki, September 23, 2014. https://tiddlywiki.com/static/History%2520of%2520TiddlyWiki.html.

Short, Emily. "Links and Structures from Michael Joyce to Twine." *Emily Short's Interactive Storytelling* (blog), July 27, 2019. https://emshort.blog/2019/07/27/michael-joyce-on-hypertext-links/.

TiddlyWiki. Accessed December 20, 2018. https://tiddlywiki.com/.

Twine, Leon. "Twine Macro: << Cyclinglink >>." Glorious Trainwrecks, January 28, 2013. https://www.glorioustrainwrecks.com/node/5020.

Yang, Robert. "A Letter to a Letter." *Radiator* (blog), April 10, 2013. https://www.blog.radiator.debacle.us/2013/04/a-letter-to-letter.html.