## CHAPTER P-1

# From Links to Stories

> ◊ *Twining* is not simply a how-to book, so the step-by-step examples in this chapter are accompanied by comments designed to put practical learning in context. If for some reason you're more interested in the instructions alone, you'll find each action item set like this paragraph, boxed and marked with a special character. We'll use this convention for all the practical chapters.

Supporting materials for this chapter can be found online at https://github.com/AMSUCF/Twining. For most of our examples, you'll find two documents: a web page (.html), which is the finished version of the project, plus a plain text file (.txt) containing all the code we discuss. In projects with multiple pieces (or "passages," as you'll shortly learn to call them), we've indicated the passage to which the code belongs.

We're providing these resources as an invitation to tinker, play, and remix. There are two ways to make our code your own. The Twine 2 application allows you to import any published Twine file you find as a web page. The procedure is discussed later in this chapter. You can import any

of our examples and see all the structure and code. On the other hand, if you want to work through our examples step-by-step, you may want to copy and paste from the text files to save yourself a lot of tedious typing.

## Getting Ready to Write

You can access Twine via the web at www.twinery.org. There are two options: download a local copy or use the program online. We recommend downloading and installing if you can. The online version is fine for beginners, but it limits more advanced work (involving external files, for instance). Also, somewhat confusingly, the stories you build using the online tool are accessible only in the browser and computer you built them from. They are not stored on a server and the link cannot be shared, which can be problematic for newcomers.

Twine is available from Twinery at no charge, with versions for Windows (32- and 64-bit), Mac OS, and Linux. Installation is straightforward: download and run the appropriate installer. Twine will set up necessary files and permissions. You'll find a new folder named "Twine" in your Documents directory, where data files associated with your various Twine projects (called "stories" by default) will reside. An icon for the Twine application will appear in your list of programs and in the appropriate system folder where applications are stored.

On rare occasions, things don't go so smoothly. Depending on how your system is configured, you may encounter pushback from antivirus software. You might, for instance, see a warning about a supposed vulnerability called "WS.Reputation.1." Appearances to the contrary, this is not the name of a malicious virus code lurking in the Twine installer. "WS.Reputation.1" is a designation applied by makers of antivirus software to programs that serve small or niche audiences. Such programs, they imply, aren't circulated widely enough to have a reliable reputation. There are usually straightforward ways around any obstacles your antivirus software presents. At worst, you might have to open a quarantine folder, click on the Twine installer, and give it an exemption. You should only have to do this once.

We considered not mentioning antivirus problems. Most people will never run into them. If you do, you should feel safe using

installers downloaded from Twinery. The Twine developers understand the risks of malicious software and maintain their code responsibly. It is at least ironic, and not a little insulting, to question their reputation. We tell this story because it highlights Twine's identity and ethos. You pay nothing for Twine—though you should consider making a contribution for its support. The program is nonproprietary, open-source software, supported and advanced by expert volunteers and a community of users. Twine belongs to a culture sharply different from that of giant corporations with hundreds of millions of sales. While you can use Twine for all sorts of things—journalism, education, research, and so on—it is designed for one basic task: telling complex, interesting stories. The rest of this chapter will help you get started with that.

## Interface and Controls

Launch the Twine application by clicking on its icon. Since the installer may not place the icon on your desktop, you may need to look for it in the appropriate Applications folder on your system. After launching, you should see something like this:

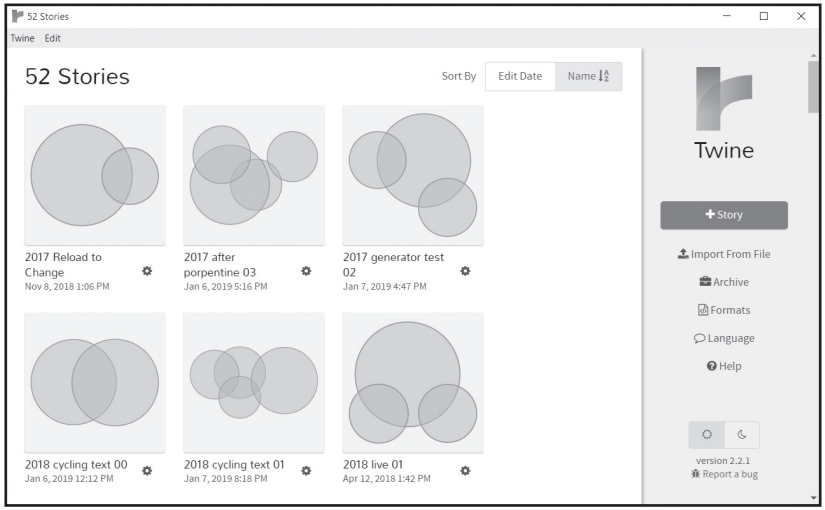


**Figure 5:** Twine library view

You're looking at a Twine library. This one has contents, but yours will contain no stories if you have done a fresh installation. The larger panel on the left will include a visual representation or thumbnail for each story you create. You can sort this collection in various ways. Each thumbnail is a square containing one or more circles. The circles stand for the subdivisions or *passages* in your story. They are arranged in a way that roughly imitates their layout in the Twine structure editor. (More about both passages and structure is just ahead.) Thumbnails can be helpful if you are trying to find a specific Twine story in a crowded library, but they are impressionistic.

To the right of the library panel, you'll see a stack of operators, or function buttons. The most important of these is the prominent green button labeled "+ Story." You'll use this operator to begin a new story. Next in the stack is "Import from File," which is a way to find and import existing Twine stories that may be located outside of your Twine folder. This function will be useful later in your Twine career, especially if you change computers or collaborate with other writers.

Below the import operator is "Archive," a function that creates a copy of your entire current library. You should use this tool early and often. Don't worry too much about storage space, unless your system is unusually constrained. Today's storage devices are designed with graphics and video in mind, and in many cases, Twine stories, which are mostly made up of words, take up only a tiny share of available space. Feel free to archive as often as you like. Remember to use a sensible naming convention for the resulting files.

Below "Archive" is the "Formats" operator, which shows all the *story formats* available in your Twine application. Story formats are presentational interfaces for Twine stories. Interfaces probably need less of an explanation today than they did in the last century. You're probably familiar with the way web pages can change as you view them on different devices and browsers. In effect, browsers and devices are interfaces. The contents of the web page exist in a file that various interfaces process for display. Something similar happens with Twine. Your story contents go into an output file, which Twine processes using a designated story format to determine what appears on the user's screen.

You are not required to choose a story format. The default format for Twine at this writing is Harlowe. A newer format called Chapbook will install as an inactive alternative. Chapbook is a simple, readable scheme designed to help people use Twine without encountering too much complexity—though it is quite good at supporting more advanced techniques. In our opinion, Chapbook provides excellent ways to move from simple to more ambitious creative uses. The material covered in the present chapter is the same for both formats, though we will rely largely on Chapbook in later practical chapters. Switching to Chapbook and making it your default story format is quite easy. We'll discuss it in the next practical chapter.

The "Formats" operator offers a choice of formats built into the Twine application. At this writing, that set includes Chapbook, Harlowe, and two even earlier formats, Snowman and SugarCube. You can find documentation and projects using these schemes online. Because Twine is an open-source application based on the core technologies of the web, anyone who wants to can build improvements and offer them to the world. Software never sleeps. The "Formats" operator allows you to add new formats as they are developed. Many Twine users do this seldom or never. At some point, though, a remarkably elegant and useful new format may give you the itch to switch.

Below "Formats" is an operator called "Language," which lets you localize Twine to any of thirteen national languages. Below this is a "Help" operator, which takes you to the wiki at Twinery. Continuing down the screen, after a small gap, we find two buttons marked by a representation of the sun (left) and moon (right). These buttons toggle the background and text colors used in any story format. In the dark theme, which is active by default, words appear in a light color against a dark background. The light theme reverses this arrangement. The choice of theme is largely a matter of preference. Some find the dark theme more dramatic, maybe suited to dystopian or gothic moods. Those less aesthetically inclined may find the dark theme hard to read in low light, preferring dark text against a bright background. This setup has worked pretty well for books, after all.

There are two text indicators at the very bottom of the stack. One identifies the version of Twine you are using, with a link to the credit screen for that build. Note—and please use—the included link for donations. Like public radio and TV, Twine depends on a combination of pride, love, and guilt. Every donation helps. The final item is a link to a bug-reporting channel, should you encounter anything in Twine that seems clearly dysfunctional. Use this link by all means—bug reports help everyone—but always ask yourself if the trouble could have been caused by some mistake in your use of Twine rather than the program itself.

Twine is nowhere near as complicated in its interface as some commercial products we use regularly, but it does have more than one menu of functions. We've just discussed the one that appears at the top level of the library view. Another will show up after you have opened a story file. You can access this menu by clicking on the name of your story, which appears at the lower left of your screen, or on the black triangle that may be visible next to your story name. (Longer story names make the triangle invisible.) When you unfold this menu, you will see nine options. We'll only discuss the last two, "View Proofing Copy" and "Publish to File." In fact, we'll defer "Publish to File" to the end of this chapter. "View Proofing Copy" produces a very useful printout of your story with the contents of each passage, including the script elements we will describe in later chapters. We're not quite sure how the passages are sorted in this report, possibly from graph position or order of creation, but they're all included, and each is set off by a dotted line. This draft view can be of great help if your story has very many passages or if you've lost your way in the process of composing.

## Key Terms

Before we start doing things with Twine, we need to define some basic terms. A *story* is a Twine work, indicated in the library by a title and thumbnail. It is interesting to think of possible alternatives to this word. The strongest tension, of course, is between story and *game*, as we discuss in the theory chapters, but we could also consider other metaphors for branching texts. Labyrinths? Mazes? Webs? Weaves?

However, *story* is the word Klimas chose because Twine is first and foremost a storytelling tool. You may take this term loosely and think of your work as a news or feature story, or the kind of instructional story used in teaching, or the unfolding story-of-play that belongs to games, but everything tends to be some kind of story.

As we have seen in chapter T-1, Twine has a story of its own. Twine's relationship to earlier hypertext systems is complicated but close enough for some comparisons. Hypertext programs generally adopt a scheme called a *directed graph*—typically, a stylized tree made of boxes and lines—in which the reader's attention is meant to move from one division or *node* to another. Adapting this idea to literature, George P. Landow, the first theorist and rhetorician of hypertext, renamed nodes as *lexias*, deriving the term from the French literary theorist Roland Barthes, in whose work Landow found a conceptual basis for hypertext (Landow 2–3). Like early hypertext itself, the term *lexia* was eventually eclipsed by other usages, such as *page*, *post*, and *tweet*.

In many ways, Klimas's term for a textual unit, *passage*, represents a second coming of the lexia. A passage is a discrete body of material that may contain words, still or moving images, and sound cues. Passages are displayed one by one as a reader moves through a Twine story. (We don't yet know of a Twine story format that displays more than one passage at a time, or a good reason for building one, but never say never.) You can put as much or as little information into a passage as you like. Twine passages tend to be relatively terse, though some writers put multiple paragraphs into their passages. Long before Twine, the first hypertext writers faced a similar aesthetic problem: Why put a lot of text into a lexia if the point is to replace it with something else? A famous early hypertext paper contrasted "holy scrollers," who preferred longer, unbroken texts, to "card sharks," who thought the contents of a lexia should fit the dimensions of a file card (Halasz 838). Though the sharks still dominate, both traditions are still with us, often within single works. As Twine writers like Porpentine and Anna Anthropy show, varying the length of passages can be highly effective. There are no absolute rules about passage length. If you need a long passage, write one. Scroll if you want to—but don't consider it mandatory.

The last of our three crucial terms is *link*, the active aspect of any directed-graph system. We'll begin our practical work with Twine by describing how to make links, but before we get there, we need to explain what links essentially are. Links have become an invisible part of everyday life in the internet age. You use one kind of link, the type described by Hypertext Transport Protocol, or HTTP, every time you move from one web page to another—but what exactly are you doing?

The World Wide Web has led us to identify links with words or phrases set off in special colors or images that invite a click or tap. This sense comes into play every time we say something like "Go to the main Twinery page and follow the third link on the left." Such expressions may be inevitable, but they're also inaccurate. Visual traces are only one aspect of hypertext linking. We could also describe a link in terms of its underlying code—in, for instance, HTML:

```
<A HREF="www.twinery.org">Get your Twine
here!</A>
```

If we keep in mind the infrastructure that underlies any visible trace of a hypertext link, we begin to understand that links are, like all programming code, *hyperlinguistic* (Galloway 165). They are simultaneously meaningful in more than one mode. The HTML anchor tag (<A> . . . </A>) usually surrounds some readable text. That text is often meaningful as part of a sentence. The anchor tag itself has meaning as a bit of web coding, and that code in turn is significant to another piece of software, a browser application, which converts it into an expression in machine language. When your computer receives this machine instruction, it retrieves and displays the indicated information.

Links are more than just markers on a screen. In the most complete sense, a link is a transition—an action or event—associated with multiple signatures or traces: the immediate words of the text, to which the link is *anchored*; the underlying code that specifies what the link will do; and a third aspect that we have not yet explored, which we will discuss in greater detail later in this chapter.

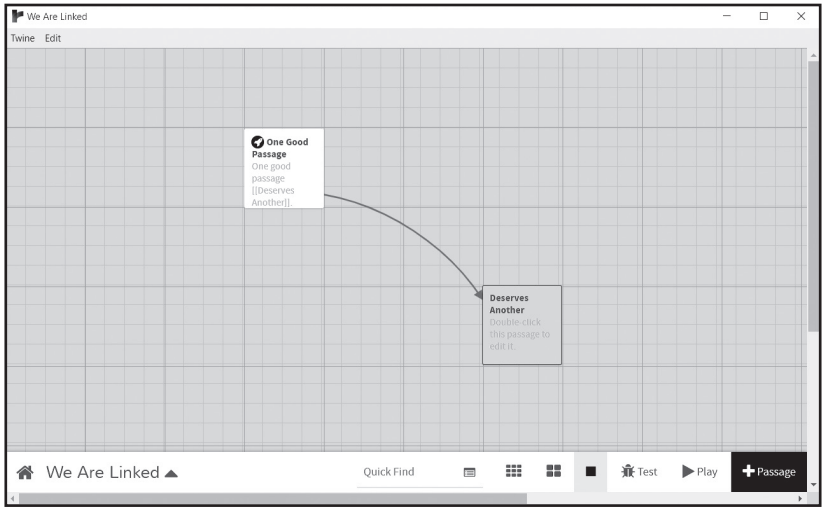


**Figure 6:** Two passages connected by a link

In terms of the hypertext graph, which we will call *structure*, the link is represented by a line with an arrow at one or both ends, showing the possibility of transition from one passage to another. Hypertext systems do not always display graph structure—consider the World Wide Web—but graphical mapping can be very important in building complex narratives like hypertexts and games.

To summarize, when we say *link*, we may refer to any or all these aspects:

1. Some anchoring text or image
2. Underlying software code
3. A representation in the story's structure map
4. An action, typically replacing one passage with another

The most important item in this list is the last. As the poet and designer Johanna Drucker says, digital writing is always "more event than entity" (Drucker 31). The point of a hypertext or other kind of digital fiction is that it allows things to happen, often in response to choices by a reader/player. In a sense, all language—certainly all storytelling—is a happening of some sort. In Twine and systems like it, this active aspect

of the text is particularly central. This distinction will be important as we shift our conception from literary texts to games, which must be actively played.

## Example 1.1: A Simple, Circular Story

Action requires planning. Stories need to be written, and digital stories have to be designed or structured as well. You'll eventually divide your time between the words of your story and its logical layout, but everything begins at that primary, verbal level—what you choose to say—so let's start there.

◊ Launch Twine if you haven't already.

A dialogue box will open asking for a title. You can call this story anything you like, though we recommend calling it *The Ostrich*. Click that green button on the right labeled "+ Story." At the title prompt, enter "The Ostrich."

Here's what your screen should look like—a view into the Twine structure editor:

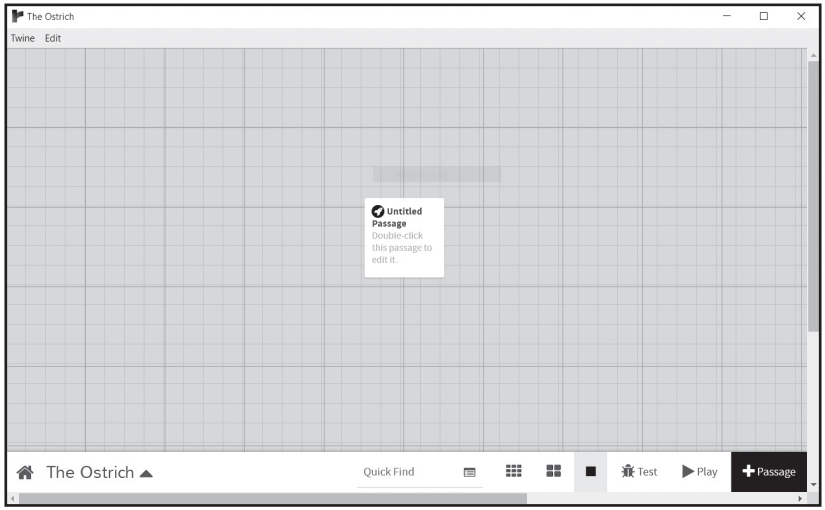


**Figure 7:** Beginning a new story

The background of the structure editor is a grid system with two scales, or rules, bold and fine. These lines are just a visual convenience for people who like to line things up neatly. They don't affect the function or underlying code of your story. Within this grid is "Untitled Passage." Below the title, you can faintly see the message "Double-click this passage to edit it"—which is a bit like the label on that bottle Alice finds in Wonderland—*Drink me*. Who could resist?

◊ Double-click the untitled passage.

When you double-click the passage, the structure editor is replaced by the passage editor, which is a specialized text processor. It looks like this:

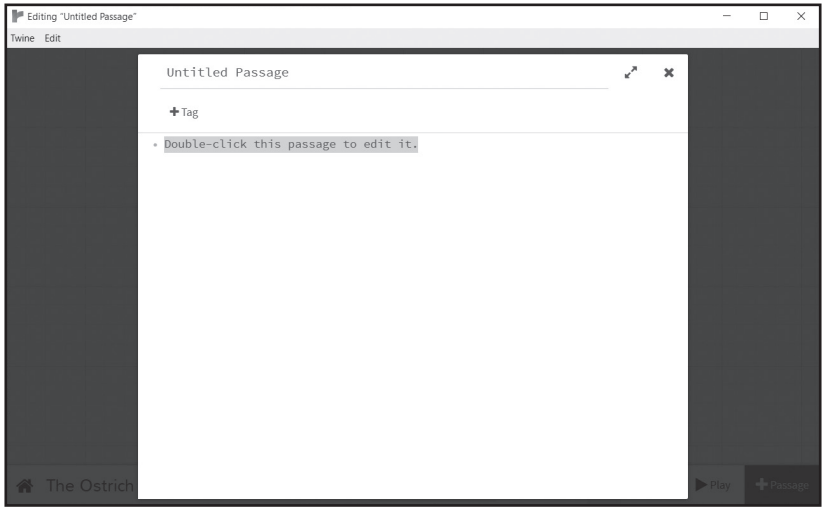


**Figure 8:** Passage after opening

Here's where we'll begin writing, but first have a look at the elements that sit above the editing window. The first is a title bar. Every passage needs a title, preferably a *unique* title, for reasons that will become apparent shortly. (In fact, you can give two passages the same name in Twine, but this is a bad idea unless you know what you are doing.)

Below the title bar is a space where you can add *tags* to your passage. Tags are further ways of identifying the passage and can be used to sort, group, and otherwise process them. If the main title is a tag's given name ("Chris"), a tag might be its family name ("Klimas"). So if our Chris passage has relations, they might all be tagged as Klimases. There are other ways to use tags, including in more sophisticated, code-intensive operations, but this is a starter example, so we won't add any tags.

◊ Type "the last thing" into the title area.

Now that we know what we're writing, let's get ready to enter some text. As you'll see, there's a small twist to negotiate here, but we can begin sensibly enough by blanking out the existing text before adding our own.

◊ Select (drag over) the phrase "Double-click this passage to open it," then press "Delete" on your keyboard.

You might expect this operation to yield a nice, blank writing area. It doesn't. Out of their desire to make their program superfriendly, Twine's developers have decided to display a page of hints every time you begin editing. This quick reference includes directions about formatting text, working with special symbols, making links—the subject to which we are coming—and some advanced topics. This is all very useful information, though we wonder if you really need to see it every time you start to write. Go ahead and erase these notes before entering your own text—and don't worry, you can find all this information on the Twine wiki by clicking the "Help" button.

◊ Place your cursor to the right of the bullet (•) in the text editing area. Type any character.

The helpful page disappears, and you have a blank space in which to write. Well, almost blank—for some reason, that bullet character remains. Don't worry, though, it won't appear when your text is displayed.

◊ In the writing space, enter the following text:

```
. . . on this of all mornings, the last thing
anyone wants to see is an ostrich.
```

This text is a recommendation, not a requirement. Feel free to write other words as you move through this example. Just be sure to put in links where they are specified. We're coming to those.

◊ Click on the "X" at the top right of the editing window.

The editing window will close, returning you to the structure editor, where you should see a passage named "the last thing," with the sentence you just typed visible inside the square. In the upper left corner of the square, you should see a small, circular icon in green and white. This icon, which looks to some people like a rocket ship, marks the passage from which play will begin.

With only one passage, we don't have anything like a real Twine story or a hypertext yet. The magic begins with a second passage. We're going to add one now. You might expect to repeat the process we used to create our first passage, and in fact, you could do so—but if we use that method, we'll miss an elegant and charming aspect of Twine.

◊ Double-click your passage. In the text editor, add a pair of square brackets on either side of the word *ostrich*, making your text look like this:

```
. . . on this of all mornings, the last thing
anyone wants to see is an [[ostrich]].
```

Use right and left square brackets—the keys to the right of the letter *P* on your keyboard—not parentheses or curly braces. Twine looks for these specific characters and won't recognize substitutes. Also, if you are following closely, you'll put the period outside of the right set of brackets. There's no great harm if you get that detail

wrong. (In later examples, however, precise punctuation may matter quite a bit.)

◊ Click the "X" to close this passage . . . and behold!

Treasure this moment. It's your first step into hypertext. It is also your first experience of a very beautiful thing: Twine automatically creating a new passage to complete a link. Most people take this little transaction for granted, but some early hypertext systems did not include this feature. The World Wide Web makes no attempt at all to manage structure. For these reasons, Twine's automatic link creation makes an old hypertext writer smile. It's a sweet hack. More important, creating passages automatically makes building complex structures a fluid, coherent process, giving a major boost to creativity.

If you look at the structure editor now, you'll see two passages (boxes), with a curved, arrowheaded line between them. Structurally speaking, you now have the beginnings of a genuine Twine story.

◊ Double-click on the "ostrich" passage to edit its text. Set your cursor to the left of the first line and type the following:

```
But here it is, maybe slightly larger than
life, in the middle of your Auntie Integer's
sunroom. A flightless bird with eyes the size of
[[gumballs]].
```

As you may guess, typing those double brackets around the word *gumballs* automatically links the "ostrich" passage to a third passage called "gumballs."

◊ Return to the structure view; open this new, blank passage; and enter a third paragraph:

```
Once, for an entire week between the ages of
two and two-point-one, your entire vocabulary
```

```
consisted of the word "gumball," which became the
name of every person and object as well as
the lone verb in your dramatic revision of the
human language. Now, somehow, the occurrence of
this word makes you vaguely [[uncomfortable]].
```

If you do this right, you'll summon up a third link to a fourth passage, which is automatically titled "uncomfortable."

◊ Open that passage and enter a last chunk of text:

```
"!!!" says the ostrich, also apparently
unsettled.

But you speak Human, not Ostrich. You take a
step closer.

"!!!," the ostrich reasserts.

You carefully blink each of your eyes in
succession, an old trick for stabilizing
realities. It is unmistakably an ostrich;
```

This time we didn't end with an automatic link. That's because we're going to join this fourth passage back to the first one—ironically called "the last thing." Doing this will let us demonstrate a second powerful technique for making links in Twine.

◊ At the end of your text in the current passage, following the semicolon, add the following text:

```
[[and . . .->the last thing]]
```

Here you see a second way to define a link in Twine: by adding the symbol -> (a two-character rendering of an arrow) plus the name

of an existing passage. Links made in this way can run anywhere you like, not just from new to newer but into and among existing passages as well. The name of the destination passage ("the last thing") must be spelled exactly as it appears in the structure graph, and unlike destinations in HTML links, it is not placed in quotation marks.

Our first story, *The Ostrich*, is now complete. Properly assembled, it forms a simple loop. To see how the loop works in practice, we'll need to play it through.

## Playing through the Story

Twine provides two main ways of checking the operation of a story, each associated with a button you will see to the left of the "+ Passage" button. Moving from right to left, the first of these is "Play." To its left is "Test." The "Play" button shows the story pretty much exactly as it will appear to your reader. "Test" adds debugging tools, which become useful as you begin building more ambitious things.

◊ Click the "Play" button.

You should find yourself looking at the text you wrote in the first passage, called "the last thing." If you are using the Harlowe story format (still the default at this writing) and did not change from dark to light themes, the letters will be light-on-dark. If you have switched to Chapbook, you'll see something resembling a printed page. The final word of the passage, *ostrich*, should appear in a style that indicates the starting point of a link: blue in Harlowe or underlined in red for Chapbook. If any of these details are wrong, click the "X" in the upper right corner of the window to return to the structure view. Reopen the problematic passage and check what you wrote.

Suppose you found a mistyped character in the third of our four passages ("gumball"). After you make the correction, you can press the "Play" button again and move through the story from the first passage. With only four passages, this is just mildly annoying, but once your

stories stretch to dozens of passages, you won't want to return to the top. Fortunately, Twine allows you to make any passage the start of the story. Let's make "gumball" the beginning.

> ◊ Return to structure view if you are not there already. Hover over the "gumball" passage. A row of buttons appears. Slide your cursor down and to the right, then click once on the button marked with three dots ( . . . ). A menu appears. Choose the first option, "Start Story Here."

The "gumball" passage now has the green rocket ship, indicating that it is the start of the story. If you use the "Play" or "Test" buttons now, you'll automatically begin with "gumball." Resetting your start passage can be essential in building longer stories—but be careful. If you change the start passage for editing purposes, *remember to change it back*. When you export or publish your Twine story, whatever passage is currently marked as the start will become the entry point. If you send out a story and readers complain that it seems to start in the middle, that might be because you forgot to reset the start passage. (With hypertext, though, you can always say you wanted things that way.)

> ◊ Use the "Start Story" procedure to set "the last thing" as the start passage again. Then play through the entire story, following the single link at the end of each passage. Visit all four passages and make sure the final link takes you back to "the last thing."

Our *Ostrich* story shows one legitimate use of Twine, but not the best or most interesting. *The Ostrich* has only one reading sequence. Even if we move the start point, the reader will always follow the same path around the loop. Every passage has only one exit, leading inevitably to the next. *The Ostrich* is the sort of story we might find on printed pages, which is fine, but Twine can do more than imitate print.

It is tempting to say this story is not a hypertext, but that claim could be controversial. Theodor Holm Nelson, who invented the term, insisted that "hypertext is the most general form of writing" (Nelson 3/2). According to Nelson, writing in fixed succession—the paragraphs of a newspaper story, for instance—artificially limits language. Even without computers, writing tends toward multiple arrangements or sequences. You can open a book to any page you like. To return to newspapers or web pages, think of the way your eye might drift from one story to an item in another column or space, then back again. For Nelson, multiple sequences are the natural order; linear chains, like our *Ostrich* story, bury their heads in constraint, ignoring other possibilities.

What happens if we explore those possibilities? Doing so would lead us away from the conventions of single-stream media (books, film, video) in the direction of other things, including hypertexts and computer games. This turn has obvious creative consequences, but it can also be a technical matter. We can measure how hypertextual a story is in terms of *link density*, the ratio of passages to links. *The Ostrich* has a link density of 1.0, with exactly one link per passage. Other values are possible. Consider a story with five passages and a total of seven links among them. That story has a link density of 5 to 7, or 1.4. We might say that a true hypertext should have a link density greater than 1. We might also say that link density will generally fall somewhere between 1.0 and 2.0—which may seem strange, considering there is no formal constraint on the number of links you can put into a passage. However, not all constraints are formal. Consider the following example.

### Example 1.2: *Overflow*

◊ Start a new story in Twine. Name it *Overflow*. Create a new passage, title it "Overflow," and type in the following text:

```
Let me make one thing perfectly clear: I am
in no way responsible for whoever or whatever
devoured the sun.
```

◊ Type double square brackets around each word in the sentence. (You can either include or exclude the colon and period, as you like.) You should end up with something like this:

```
[[Let]] [[me]] [[make]] [[one]] [[thing]]
[[perfectly]] [[clear:]] [[I]] [[am]] [[in]]
[[no]] [[way]] [[responsible]] [[for]]
[[whoever]] [[or]] [[whatever]] [[devoured]]
[[the]] [[sun.]]
```

◊ When you are finished typing, click the "X" in the upper right corner to return to structure view. Consider the results.

Twine will happily anchor a link on every word in a sentence or every word in a passage. You could even . . .

```
[[l]][[i]][[n]][[k]] [[e]][[v]][[e]][[r]][[y]]
[[c]][[h]][[a]][[r]][[a]][[c]][[t]][[e]][[r]]
```

. . . if you were entirely mad. Of course, Twine will generate a destination passage for everything you link. In the case of *Overflow*, we end up with a total of twenty-one passages—the original plus twenty possible successors. Think for a moment about the time it will take to write unique text for each of those twenty passages (as we've actually done in the completed version in the digital version of this book). Now consider writing at least one outward link from each of those passages. And what if every successor passage needed *more than one* link out? Before you knew it, you'd have a completely unmanageable project. One early hypertext writer, Shelley Jackson, encountered this problem while working on her celebrated fiction *Patchwork Girl* in 1995. She began by making links at will, starting threads and branches that expanded in all directions. After a while, she found the proliferation of links downright monstrous. In a later interview, she called the resulting structure map a "Brillo pad" of tangled lines. So much for the first draft. "I erased all the links," she said. Then she started over with a more careful approach (Jackson).

Link explosions can be troublesome. Yet strangely, there are at least two important digital fictions in which each word in every lexia behaves like a link: Michael Joyce's *afternoon* (1990; the first thing called a hypertext fiction) and Judd Morrissey and Lori Talley's *The Jew's Daughter* (2000). In the Twine era, Porpentine's *Howling Dogs* contains at least one passage in which every word is linked. How did these writers manage to avoid the Brillo pad of madness?

If every link system implies a conceptual tree or bush, the answer to explosive growth is simple: cut back the excess. Trim judiciously. Think topiary gardens, not jungles. Let's consider a more sensible example.

### Example 1.3: *The Reign of the Two Doors*

◊ Start a new story called *The Reign of the Two Doors* (holding nose for pun). Enter the following text in a new passage also called "The Reign of the Two Doors":

```
You find yourself in the two-door universe. It
was slightly less expensive than the three- or
four-door models and all we could afford.
```

◊ Below the text you just entered, enter the following:

```
[[Go through the left door->Not Right]]

[[Go through the right door]]
```

Before going on, let's introduce a useful Chapbook feature designed especially for the kind of story we're telling here. It's called a *fork*. A fork is a visual device for presenting a small set, usually a pair of links. If you're using Harlowe, don't worry—the fork is convenient but not essential.

◊ Add a greater-than sign, or right angle bracket, before each link, like so:

```
>[[Go through the left door->Not Right]]

>[[Go through the right door]]
```

The angle bracket notation creates the fork. Its visual effect is subtle but pleasing: a fine line appears between the two links. The online Chapbook guide (https://klembot.github.io/chapbook/guide/) provides advanced information about restyling the appearance of forks. You can include this effect or not, depending on your taste.

As you can see, this story is written in an idiom many Twine fictions share with parser-based interactive fictions and the kind of multipath novel usually called "choose your own adventure." The reader/player is addressed in the second person. For scene-setting, we use the present progressive tense. Link anchors, which substitute for command-line typing in interactive fiction, use the imperative mood and describe some action—in this case, movement through space carried out by the reader/player's persona.

Each of our twin links uses one of the main linking styles available in Twine. The leftward exit names a specific destination passage that, since it does not previously exist, will now be created. The rightward link calls into being a passage named for its anchoring text.

Looking at the structure graph, you'll see that we have two fresh passages to deal with: "Not Right" and "Go through the right door." Let's handle the second of these ("Not Right") first:

◊ Open the passage called "Not Right" and enter the following text:

```
You find yourself in the Place of No Winning. It
is a simple room with, of course, two doors.
```

```
[[The Init Door->The Reign of the Two Doors]]

[[The Exit Door->The Reign of the Two Doors]]
```

The point here is that both doors from this passage lead back to the start point, closing a loop. There are two doors because this is a two-door universe. If you'd prefer one, that's fine. The reference to winning (or its opposite) is a matter of judgment. Maybe the player wants to stay in the loop. Who are we to say?

◊ Open the passage called "Go through the right door" and enter the following text:

```
Advancing boldly through the dexterior portal,
you find yourself in another version of the same
stupid room. Someone is trying to make a point,
you suppose.

[[Left! Maybe it will work this time->Not
Right]]

[[Right!]]
```

For the record, *dexterior* is not an actual word, though maybe it should be. By now, you should grasp the general design of this story: there are two links (or doors) from every passage. So far, at least, one of them always leads to the so-called fail passage, locking the player into the loop. However, there is a bit more to the story.

◊ Return to structure by closing the current passage. Open the new passage called "Right!" and type the following text:

```
Right. Always take the door on the right. You
get it now.
```

```
[[Always right->The Reign of the Two Doors]]

[[Left Behind]]
```

In this passage, we do the perhaps all-too-predictable thing, capriciously breaking the left/right pattern. The first link leads back to the beginning, while the second, left-hand door leads on. This is an entirely voluntary decision, of course. You could be kinder to your player/reader and avoid such perversity. When it comes to link patterns, the rules are up to you.

◊ Return to structure. Open the new passage called "Left Behind" and type the following text:

```
Moving at last through the door the writer
apparently doesn't want you to take, you begin
to float above the confines of the labyrinth,
leaving fools behind.

Rise up, you lovely winner.
```

Players of Davey Wreden's metagame, *The Stanley Parable*, will recognize the two-door controversy (Wreden, *Stanley Parable*). This rising-up business is an abject steal from Wreden's next offering, *The Beginner's Guide*, which we'll address in the conclusion of this book, even though it is not a Twine work. The player's upward motion expresses a universal figure or trope. Given a loop or labyrinth, there are three possible actions: make your way to the center, find some way out, or rise above the whole thing. It is no coincidence that we find ourselves referring to a video game and reaching beyond the realm of hypertext fiction (and indeed Twine). The current generation of Twine creators think of themselves as game developers as well as storytellers, and they occupy the same social and economic space as independent game developers. As we'll see in chapter T-2, they're part of the conflicts that come with that

contested space. Many Twine stories are explicitly designed as games, with rules, consequential decisions, winning and losing outcomes, and even scoring systems. Porpentine's *Ultra Business Tycoon III* (Porpentine, *Ultra Business Tycoon III*) and Seth Alter's *RocketJump-ification* (Alter) are excellent examples.

For some, Twine works belong entirely within the game world. Others see Twine works as hypertexts encompassed within a larger group of creative products called *cybertexts*. That term was coined many years ago by Espen Aarseth, who went on the become one of the founding theorists of computer games. Cybertexts include games but also any other undertaking without a fixed sequence of presentation, where "non-trivial effort" is required to experience the work (Aarseth 2). There's much more to say about Twine stories and games both here and in the chapters that follow, but for the moment, there's more to say about our example, both as story and as hypertext.

*The Reign of the Two Doors* has a respectably hypertextual link density of 2.0: there are two ways out of every passage. Yet it requires no more than six passages, since, in five of those passages, only one link runs to a nonexisting passage, expanding the structure. (In the "Not Right" passage, both links bend back to the beginning.) You can build as many links as you want, provided many or most do not expand your inventory of passages. This is the technique used by both Porpentine in *Howling Dogs* (Porpentine, *Howling Dogs*) and Joyce in *afternoon* (Joyce). In the former, the overlinked passage presents a field of linked words that mainly go to the same place, except for the one that doesn't. Joyce uses a different but similar technique in which a few words in each lexia will "yield" a connection to a specific other lexia (Joyce). Every other word in the lexia is implicitly linked to a default destination. This design was made possible by a clever feature of Storyspace (Bolter, Joyce, and Smith), the early hypertext system for which *afternoon* served as the test file.

The third example mentioned earlier, *The Jew's Daughter* (Morrissey and Talley), arrives at universal link coverage very differently. In Morrissey and Talley's story, which is, in fact, less a hypertext fiction than an example of digital text generation, clicking any word on the current

screen feeds the word to a program that composes a new passage beginning with that word. This revolutionary technique goes beyond predefined passages and links, but you may want to keep it in mind even so. Because it offers access to programming resources like JavaScript, Twine allows you to work with dynamic, variable, and even logically generated text. These are more advanced subjects, so we reserve them for later chapters, beginning with P-3.

Our *Reign of the Two Doors* example shows that it's possible, even with basic tools, to manage links and story structures, avoiding explosive overload. Links and linking strategies take a wide variety of forms. *The Reign of the Two Doors* shows what we might call *navigational* linking, tied to the movement of a virtual character or point of view through a described space. A close cousin of this approach is *procedural* linking, where the anchoring text describes an action involving the persona: "You shut the door"; "The ostrich says nothing"; "The Twinebot emits another burst of story," and so on. Also quite popular is *conversational* linking, where the anchors are options for responsive speech. For instance,

```
The high commissioner shoots you an arctic stare
and says, "Twine. Really?" You answer:

[["It is the way among my people."->Way]]
[["Who said anything about Twine?"->No Way]]
[["Hey, is that an ostrich?!"->Way Out]]
```

All these linking strategies—navigational, procedural, and conversational—share a common feature of composition. They divide the visible space of the passage into two parts: an upper section that advances the story and a lower part that contains the link anchors (perhaps set off as a fork). We could say the upper part is definitive or diegetic, reporting what happens or has happened in the world of the fiction, while the lower portion is hypothetical, consisting of language still in play. We'll call this arrangement a *bifold* construction.

The alternative, which we'll call a *unified* construction, brings the links directly into the diegetic text. Here is a thumbnail example:

```
Every morning, [[the old man->Hubert]]
comes to search our [[trash bins]]. He is
impeccably dressed and obviously from the
[[Ministry->Darkness]].
```

As you can see, this passage consists only of narration—strictly speaking, the direct report of an unidentified narrator. There is no second-person address and no reader/player persona. There are link anchors, but they fit into the diegesis instead of pulling away from it, as is often the case in the bifold scheme. These features give the example a stronger resemblance to conventional literary fiction than to interactive fiction or choose your own adventures. The unified or in-line treatment of links was a signature of early hypertext fiction, whose writers sometimes set themselves (perhaps regrettably) against the older interactive fiction tradition of parser-based games.

Very roughly speaking, bifold construction accentuates the game-like qualities of Twine stories, while the unified approach plays to literary interests; but this distinction can never be absolute. In the 1980s and '90s, some hypertext writers said of their work, "This is not a game" (see McDaid). In the following decades, however, stories and games inevitably converged. In a later hypertext from the web era, one of us revised the claim, declaring, "This is not not a game" (Moulthrop). Today's Twine writers dispense with single and double negatives alike. Klimas's use of "story" notwithstanding, many Twine creators call their products games and even "videogames," as in merritt k's groundbreaking and essential anthology, *Videogames for Humans* (merritt k).

However controversial the claim to game identity may be, it will not go away. Twine stories can be games, and Twine games tell stories. One interest or the other may dominate, but both will be present. In fact, many Twine writers exploit this dynamic, alternating the two types of construction, writing some passages in the double-decked way and others with the all-in-one pattern. Two of the most impressive Twine stories, Porpentine's *With Those We Love Alive* and *Howling Dogs*, display this strategy. Know and consider your options. Nothing requires you to address a player persona (the eponymous "you"). Likewise, no law

says that Twine stories have to imitate print fiction. At its best, Twine allows us to explore the spaces between those alternatives, refining a new art form as we go.

### Example 1.4: *Don't Think of an Elephant*

Here's a fourth example exploring what can happen if you let your links mingle with the rest of the text.

◊ Start a new Twine story. Title it *Don't Think of an Elephant*.

◊ Add a new passage. Title it "Don't Even Think." In this passage, type the following:

```
At dawn, the [[Elephant Men->Elephant]] will
come for your skull. But meanwhile, as Uncle Jed
always told you, [[the night]] is as long as you
want it to be.

Just, you know, don't think. You know. Of it.
```

◊ Close the text editor and return to the structure editor. You should see two links running from your first passage—one to a new passage called "Elephant" and the other to "the night."

◊ Open the "Elephant" passage and enter the following text:

```
Hyperintelligent pachydermatoids from an
exoplanet we haven't found yet are here to
avenge humanity's crimes against the elephants.
Evidently, they will be satisfied with just one
trophy. That would be you.

Why they chose you remains a mystery, though
it could have something to do with the illegal
```

```
safari they caught you on. And that elephant gun
with the smoke coming out of it.

The senior Elephant Man asks if you have any
[[last words]].
```

Yes, this is one of those tales about exoplanetary pachydermatoids. This story also appears to have a link density of 2, like another example we might recall. If this story follows the earlier pattern, we might expect it to have two tracks: one leading to happiness and the other, otherwise. Let's finish the darker destiny first.

◊ Open the passage called "last words" and enter the following:

```
Evidently, you don't.
```

There are no links from this passage. It is in every sense a dead end. With the less fortunate outcome covered, let's see what lies along the other track.

◊ Return to the structure editor and find the passage titled "the night." Open it and enter the following:

```
As, for instance, that first night in the
Algarve, when Georges-Marie said, "La, but it is
[[so big->Elephant]]!"

Meaning the room, or the bed, possibly. But you
[[flattered]] yourself.
```

◊ Back to structure. Two links, as always, one already pointing conveniently back to "Elephant." The other runs to a new passage called "flattered." Open that one and enter this text:

```
"Not the [[Hermes->Elephant]]," Georges-Marie
objects. "It flatters not the slightest. Goes
immediately into wrinkles. And the gray does
nothing for you."

This was on the night train to St. Petersburg.
You remember the cocktails with prices in
Korean, the waiters in their tricorn hats, the
endless fields of [[elephant grass]].
```

As in *The Reign of the Two Doors*, we're throwing some curves. The general rule here is to avoid any word or phrase that makes us think of an elephant—references to things of a large scale or a gray and baggy suit. But now there's this link to "elephant grass." Remember, in *Two Doors*, this was where we challenged the player/reader to win by reversing logic. Think we'll do the same now?

◊ Open the passage titled "elephant grass" and enter the following:

```
Oh dear. You meant to say
"[[potatoes->Elephant]]."

[[Oops->Elephant]].
```

Sometimes Twine stories are simply cruel. Save the elephants.

With a couple of exceptions—the additional passage on the way out and the tragi-farcical ending—this story is structurally identical to *The Reign of the Two Doors*. Instead of explicitly revealing the logic of its links, however, this story works by implication and association—as language tends to do, especially literary language. Perhaps this difference represents a step away from the idiom of games, at least games of a certain kind. But we could as easily say that it connects logical play with wordplay. That might be a promising

match, at least in stories more graceful and sophisticated than this one.

## Exporting and Sharing Twine Stories

One last detail needs attention before we finish with the basics: how to make your Twine story available to friends, strangers, editors, and teachers. There are many options for circulating a story, but before you can circulate, you must first export your work. *This requirement applies even if you are using the online version of Twine.* Stories you build online are accessible only using the current browser and computer: they are stored locally to your computer. If you send the web address (URL) of an online Twine story, recipients will not be able to view it, as the material is not stored online.

> ◊ In the structure editor, look at the bottom left area of the window. You should see an icon that resembles a house. Click this icon to return to your library. In the library, find the thumbnail that represents the story you wish to export. Next to the title and time stamp for each thumbnail, you'll see an icon that looks like a gear wheel. Click this icon to reveal a menu. The third item in this menu is "Publish to File." Select that item.

At this point, your operating system will go into its usual file-saving routine, asking you where you want the results to be stored. *Pay close attention.* In Windows and Mac OS, saved files usually default to a Documents folder. Be sure you know how to find that folder. When in doubt, change the destination to the desktop, where items are immediately visible.

Click the "Save" button in the file-saving dialogue to complete the process. Twine now creates a single file containing your story plus everything a web browser needs to display it. This file is saved as a web page, with the file extension *.html*.

Why a web page? you may ask. Why doesn't Twine use binary code or some arcane, proprietary format? Like the World Wide Web

and HTML, Twine is noncommercial, open-source software. It uses free, accessible resources. For all its limitations, HTML/HTTP is for most of us the most convenient hypertext platform available. Exporting Twine stories as web pages means they can be uploaded to a web server and displayed either remotely or locally in virtually any browser.

The other advantage of HTML is accessibility of code, for those who are motivated and prepared to read it. The file format for every web page is plain text, which can be read by built-in text processors such as Notepad (Windows) and SimpleText (Mac OS). You do need to know what you're looking at, which in the case of Twine stories is not just basic HTML but also quite a bit of JavaScript. JavaScript is an auxiliary coding language (or scripting language) developed to extend the function of web browsers. Much of the magic of Twine depends on JavaScript.

If you are not a programmer, you don't need to concern yourself with any of the underlying code for your story. All you need to know is that Twine pages require JavaScript to function, so your server and browser must be configured to allow for this. We've encountered at least one academic course management system that prohibits script-enabled web pages. Hopefully, that won't happen to you. Talk to your teachers or your system administrator if it does. If the prohibition has no exceptions, there are work-arounds.

With export complete, you are ready to show your file to others. The simplest way to do this is via email, again provided your email system allows you to send web pages as attachments. Any browser application can read a web page immediately without going through a server. All your friends and teachers need to do is download the attachment and open it as a local file.

If you want a wider world to experience your work, you can upload your HTML file to a web server. There are plenty of free web hosts (Wix, Weebly, etc.). Many schools offer server access for students. Always remember that information shared on the web can be seen by nearly anyone in the world, so don't include sensitive details, names of private persons, or other things that violate common sense. You might also

want to think about the audience you have in mind for your Twine story. If the story contains material that might disturb or trigger some people or might be inappropriate for young children, include a disclaimer at the beginning.

## Works Cited

Aarseth, Espen J. *Cybertext: Perspectives on Ergodic Literature*. Johns Hopkins Press, 1991.

Alter, Seth. *RocketJump-ification*. Subaltern Games, 2013. Accessed June 6, 2020. https://subalterngames.itch.io/rocketjumpification.

Bolter, Jay David, Michael Joyce, and John B. Smith. *Storyspace* [hypertext system software]. Eastgate Systems, 1990.

Drucker, Johanna. *What Is? Nine Epistemological Essays*. Cuneiform Press, 2013.

Galloway, Alexander. *Protocol*. MIT Press, 2004.

Halasz, Frank. "Reflections on Notecards: Seven Issues for the Next Generation of Hypermedia Systems." *Communications of the ACM* 31, no. 7 (1988): 836–52.

Jackson, Shelley. "Interview Part 5: Thinking outside the Screen." In *Pathfinders*, edited by Dene Grigar and Stuart Moulthrop. Nouspace Press, 2015. https://scalar.usc.edu/works/pathfinders/shelley-jackson.

Joyce, Michael. *afternoon, a story*. Eastgate Systems, 1990.

Landow, George P. *Hypertext 2.0*. Johns Hopkins University Press, 2006.

McDaid, John G. *Uncle Buddy's Phantom Funhouse* [hypermedia novel]. Eastgate Systems, 1993.

merritt k, ed. *Videogames for Humans: Twine Authors in Conversation*. Instar Books, 2015.

Morrissey, Judd, and Lori Talley. *The Jew's Daughter*. Self-published, 2000. http://www.thejewsdaughter.com/.

Moulthrop, Stuart. "Reagan Library." *Little Magazine* [CD-ROM edition], 1999.

Nelson, Theodor H. *Computer Lib/Dream Machines*. Microsoft Press, 1987.

Porpentine. *Howling Dogs*. Alien Dovecote, 2012. http://slimedaughter.com/games/twine/howlingdogs/.

———. *Ultra Business Tycoon III*. Alien Dovecote, 2013. http://slimedaughter.com/games/twine/tycoon/.

———. *With Those We Love Alive*. Alien Dovecote, 2014. http://slimedaughter.com/games/twine/wtwla/.

Wreden, Davey. *The Beginner's Guide*. Everything Unlimited, 2015.

———. *The Stanley Parable*. Galactic Café, 2011.