Computers + Storytelling → Teaching + Learning

Teaching and Learning With Interactive Fiction

Brendan Desilets

Second Edition
May 2015

Preface

Are you one of those fed-up English teachers who can't wait for the standardized testing boom to go bust? While you're waiting, would you like to try a highly-motivational literary form that can help students to think more clearly, build their reading skills, and even enable them to write better? And, with all those improved skills, might you even hope to see those nasty test scores go up a bit?

Or are you a university instructor, looking for a way to integrate science and the humanities, without abandoning either one? Meanwhile, would you like to explore a medium that comes with a broad array of sophisticated writing-process tools?

Or are you simply interested in teaching and learning in the Information Age, without any social-media hype?

If you're in any of these categories, this book may be for you. It introduces a form of computer-based literature called interactive fiction, and shows you how this form dovetails with the goals of most students, teachers, and parents. It shows you how to get started with this challenging form, and it provides lots of instances of the form, most of them free of charge.

What is Interactive Fiction?

Interactive fiction, sometimes called text-adventure gaming or IF, is a form of narrative literature in which the reader plays the part of a character in a story. In interactive fiction, as it was originally defined in the 1980’s, the reader indicates what she wants the character to do by typing ordinary sentences at a computer keyboard.

By far, the most difficult aspect of interactive fiction, both for educators and for students, is learning the form itself. Applying interactive fiction to teaching and learning is relatively easy. As a result, this book will offer lots of guidance on getting started with interactive fiction. It will include a getting-started section for adults and another such section for younger people.

IF and Thinking Skills

Perhaps the most obvious way in which interactive fiction, also known as IF, promotes learning is that it stimulates and demands critical and creative thinking. This book will offer some ideas about improving thinking and will show how IF promotes problem-solving skills in uniquely effective ways.

Elements of Literature

Though interactive fiction always requires some level of gamelike problem-solving, it is also a true form of literature. Works of IF, or “interactive fictions,” feature plots, characters, setting, points of view, themes, tones, and all manner of stylistic variation. And, because of the unusual way in which IF readers make their way through stories, interactive fiction offers, and even enforces, a natural way to pause and consider these literary elements. This book will offer practical suggestions for using interactive fiction to study important literary concepts, the kinds that build better readers and even (sometimes) enhance test scores.

Fluency

Reading and literature teachers sometimes fight about the precise role of oral fluency in building good reading skills. It's clearly possible to overstress the importance of fluency, especially in the classroom teaching of students who have gone beyond early reading. Still, it's important for students to read with a reasonable level of quickness and fluidity, and there are ways to improve fluency in readers of all ages. As it happens, interactive fiction works extremely well for reading aloud. And this book will offer a specific technique that uses IF to build fluency in a way that few other literary forms can match.

Interactive Fiction and the Writing Process

Most young readers like interactive fiction a lot, especially when they have mentors to help them with the genre. Often, these readers want to start writing IF, even before they've achieved much skill in reading it. Fortunately, it's possible for people of almost all ages to write interactive fiction to write interactive fiction, using a number of “authoring systems,” which take some of the sting out of creating computer-based literature. In this volume, we’ll offer introductions and tutorials to three authoring systems. Two of these, Adrift and Quest, use menus and forms to make IF writing easier. A third system, known as Inform 7, allows students to work on their expository writing skills as they craft interactive narratives. This book will have suggestions about using Inform 7 with students aged eleven to eighty (more or less).

Teaching and Learning Content with Interactive Fiction

Interactive fiction, then, is a great way to build a variety of reading, writing, and thinking skills. But it can present content, too. This book will focus mostly on skill building, but it will also look at several stories that present worthwhile content, ranging from the 1893

World's Fair, to the early days of nuclear weaponry, to the delivery of candy grams in a middle school.

Some Ground Rules

Mostly because this book has separate sections for kids and adults, some of the material seems, and is, repetitious. However, some of the redundancies are adapted for their particular purposes. For instance, the example stories used to illustrate tutorials are often very similar to each other, but the versions for young people are actually somewhat different.

This book is offered under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. In other words, you’re more than welcome to use whatever you find here. However, you must attribute what you use to its author, and you must use it in a non-commercial way. Also, if you build material of you own from what you find you here, you must share what you build freely.

Updates of This Book

This book, in a frequently-updated form is available at the website “Teaching and Learning with Interactive Fiction” (http://bdesilets.com/if). Like the website, the current volume has two large sections. The longer first section, “Teaching with Interactive Fiction,” is for adults.

The second section, “Fun and Learning with Interactive Fiction,” is for younger people. The author is glad to respond to queries, at bdesilet@yahoo.com.

“Teaching With Interactive Fiction” deals with reading interactive fiction in a way that will probably appeal most to teachers of younger students, aged ten to sixteen, or so. Some of its sections on writing IF focus on younger kids, too. The sections about writing interactive fiction with a system called Inform 7 are a bit different, though. These chapters are more relevant to university teaching.

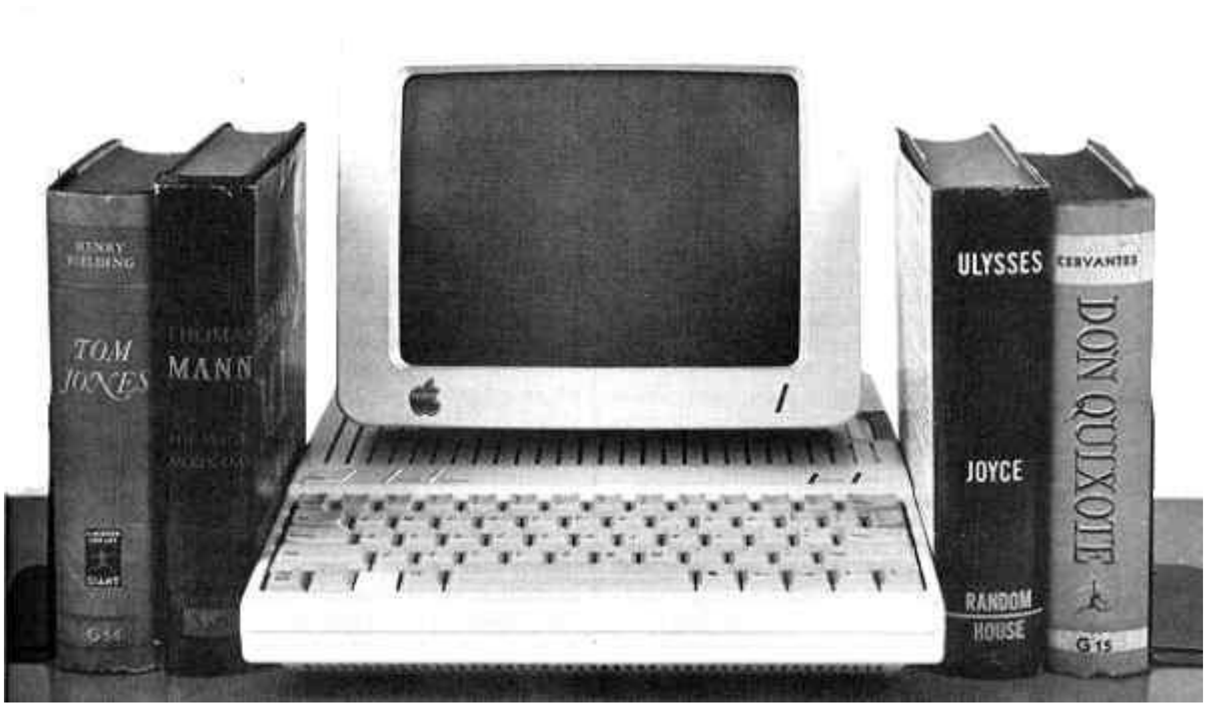
This book offers tutorials on three different systems for writing interactive fiction. Readers have often praised the usefulness of such tutorials, but they do have at least one clear disadvantage. Since the authors of systems for writing IF often refine and change their systems, it is impossible to guarantee that each tutorial will work as well tomorrow as it does today. Still, the systems are stable enough to make the tutorials potentially helpful, if not perfect.

Table of Contents

Teaching With Interactive Fiction (Mainly for Adults)

Teaching With Interactive Fiction

Chapter 1 – What Is Interactive Fiction?



Defining interactive fiction can be a tricky business, but, for our purposes, it shouldn't detain us long. Interactive fiction is a computer-based form of literature in which the reader plays the part of an important character, deciding, within limits, what action that character will take. By typing ordinary natural-language sentences at the keyboard, the reader decides where the main character will go, what objects she will pick up and use, how she will solve problems, and how she will interact with other characters. Some works of interactive fiction include pictures and sounds, but the stories communicate mostly through words. Many students find interactive fiction, also known as IF or adventure gaming, an enjoyable way to gain experience with all of the major elements of literature (though point of view takes an unusual twist or two), and teachers who are comfortable with it soon find that it grows well in the classroom, even if there's only one computer available.

In current practice, writers often use the adjective “parser-based” when discussing the kind of interactive fiction that we're discussing here, since IF stories use a “parser” to translate the reader's typed input into a form that the story can use. However, not all computer-based stories let the reader type commands in the form of natural-language sentences, and not all computer-based stories rely mostly on text. Stories that are presented through pictures and invite viewer input through mouse clicks or button presses are sometimes called “graphical interactive fiction.” One notable example of this genre is the story called *Myst*, and its several sequels. Other stories rely on text, but do not allow the reader to offer input through full sentences, opting, instead for a “choose from a list” system. These stories are often called “choice-based interactive fiction” or “hypertext literature.” For an example, have a look at “The Matter of the Monster” by Andrew Plotkin. It's available at http://www.eblong.com/zarf/zweb/matter/.

The Twine Logo

In recent years, hypertext literature has attracted the attention of a good many educators. It's generally easy to read, and, with the advent of an authoring system called “Twine,” has become somewhat easy to write. However, the parser-based input system is much more open-ended than any point-and-click approach can possibly be. It challenges the reader to compose text in way that is both disciplined and creative. Parser-based IF will be our main focus in this book, and, for our purposes “interactive fiction” and “text-adventure gaming” will refer to parser-based interactive fiction.

The phrase “text-adventure gaming” raises another issue about the nature of interactive fiction. Some works of IF seem more like games than stories. Especially in the early days of the genre, in the late 1970's and early 1980's, most works of interactive fiction were long on problem-solving and maze-mapping, and short on plot, character, tone, and theme. However, the best interactive fictions have always included various elements of literature. Today, even an IF story that qualifies as a “puzzle fest” usually offers a substantial narrative. For instance, *Bronze* by Emily Short, has often been described as “puzzle-oriented” work, but it's also a finely-crafted tale that seldom fails to entrance, and to shock, student readers.

So far, we've been using the term “reader” to refer to the person who is reading an interactive fiction and also directing one (or occasionally more than one) of the story's characters. Since this person actually does more than reading, some writers refer to him or her as an “interactor.” The character that the interactor “runs” is often called the “player/character,” and any other character is usually dubbed a non-player character or NPC, a term borrowed from role-playing games.

Chapter 2 – The Pain and Promise of the Parser

Readers love and hate interactive fiction largely because of its challenges. In early interactive fictions, the most renowned (and sometimes infamous) of these challenges took the form of complex, difficult, multi-step puzzles, and puzzles remain important in most IF stories. Students often cite the gamelike, problem-solving elements of interactive fiction as their favorite facets of the genre, and teachers usually like IF puzzles, too, as a gateway to improved critical and creative thinking. In the chapter that follows this one, we'll take a close look at interactive fiction as a tool for developing thinking skills.

A T-Shirt, Commenting on a Very Complex IF Puzzle

However, even in IF stories that eschew puzzles entirely, parser-based literature offers real challenge. It's “ergotic”; that is, the reader has to make a significant effort just go get through the text. In parser-based interactive fiction, for inexperienced readers, much of this effort focuses of wrestling with the story's parser.

The Parser

In IF, the parser is the part of the story's computer code that translates the reader’s input into something that the story can deal with. Early interactive fiction, such as *Adventure,* by Will Crowther and Don Woods, generally used a “two-word” parser, which could respond only to particular sentences of one or two words. Usually, the first of the two words would have to be a verb, but the reader could type “south,” and might get a more or less reasonable response. “Get lamp” would work too, especially if the story included an available lamp. “Put the candle in the lantern” would confound the two-word parser. But lots of two-word combinations would go just as far beyond the parser's ken. “What's up,” “Hello, world,” the widely popular “Stupid parser” would all fail miserably to communicate with the story.

During the 1980's when interactive fiction became an important commercial product, an American company called Infocom introduced a more complex parser, which could recognize and respond to a wide variety of sentence types, though a vast majority of natural-language utterances were not within its grasp. This new parser could recognize all of the following sentences.

Tie the rope to the mast.

Put the blue scroll into the box.
Ask the troll about the hobbit.
Tell the hobbit about the troll.
Take all.
Give the sword to the gnome.
Ask the elf for a jewel.
Hit the shiny nail with the hammer.
Where is the silver coin?
What is a zorkmid?

Two Parser Problems

The improvement here is pretty obvious, but, in a way, the new parser is more deceptive than the old one. With its expanded power and openness, it gives the false impression, to some people, at least, that the interactor can expect to type just about anything into the story and get a sensible response. As the renowned IF author and theorist Emily Shot has noted, this “false promise” leads new readers to lots of frustration. Consider this sequence of commands, from a video blog on IF by Jason MacIntosh.

Interactor: Where am I?
Story's Response: That's not a verb I recognize.

Interactor: You don't understand “where”?
Story's Response: That's not a verb I recognize.

Interactor: Boy, are you stupid!
Story's Response: You seem to want to talk to someone but I can't see whom.

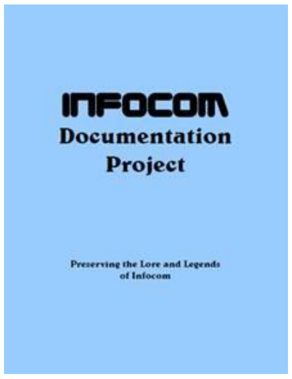
Interactor: :(
Story's Response: That's not a verb I recognize.

This sort of tangle can be frustrating, but it can be misleading, too. As IF author and polymath Andrew Plotkin has noted, when new readers of interactive fiction encounter problems with the parser, “That's a real experience. The misconception is that that's the intended interaction of the game, and that's what the author has spent all of his time thinking about.” In truth, the IF author's intent is far more benign. He or she almost always spends lots of time and effort in trying to make the reader's experience of the parser go a smoothly as possible.

Taming the Parser

For all its foibles, the interactive fiction parser is a wonderful tool. Even its limitations, are, in a way, advantages, since they tend to funnel the interactor's input in a somewhat predictable and positive direction. If the reader really could type any sentence at all into the parser, he or she might well find that puzzles would become almost impossible to solve. The parser, at its best, functions as a set of “rules” that limits the number of possible solutions.

The key to taming the parser is understanding its limitations. In truth, even the most sophisticated IF programs can deal with only a few kinds of English sentences. In the 1980's, during the commercial heyday of interactive fiction, the best of the available stories came with elaborate printed documentation, much of which dealt with the quirks of the parser in general and with particular tweaks of vocabulary and grammar that a particular story offered. Outstanding examples of this documentation are still available, free of charge, at the “Infocom Documentation Project” website, at http://infodoc.plover.net/.



If you are reading more recent works of IF, the corresponding documentation is likely to be included as part of the story itself. Many such stories advise the reader to type “About” or “Intro” or “Help” to read the docs.

Here are some general guidelines about the parser.

All works of interactive fiction, even the very earliest ones, can recognize sentences like “take coin,” which the story will consider to mean, “I want to take the coin.” Most IF stories can recognize many more kinds of sentences, but experienced readers often keep the two-word pattern in mind, anyway. For example, IF stories can now recognize “Take the gold coin,” “Take the gold coin from the fountain,” or “Take the gold coin and give it to the librarian.” However, an experienced reader may type “Take coin,” before trying a more complex formulation. If the story then asks, “Which coin do you mean, the gold coin or the copper coin?” the player will then clarify with the word “gold.” Once the player/character has the coin, the interactor might try, “Give the gold coin to the librarian.” In other words, when in doubt, it’s often best to try the simplest possible formulation.

The same simplicity principle applies, in spades, to the use of adverbs in readers' input, though IF stories themselves use all parts of speech in their text. If experienced readers want their characters to run really fast to the west, they'll probably type “West” or “Go west,” and get the desired result. “Run west” might also work, but “Run really fast to the west” will surely confound the parser.

Many works of IF can recognize at least some simple questions that begin with “who,” or “what.” Most stories also use a variety of useful abbreviations, including “x” for “examine,” “g” for “again,” “z” for “wait,” “i” for “inventory of what I’m carrying,” “l” for “look,” “n” for “go north,” “s” for “go south,” and “u” for “go up.”

“Look at the book” is equivalent to “examine the book.” However, “look,” with no object means “look around and report on what's nearby.” If the player/character is in a particular place, such as a kitchen, “Look at the kitchen,” will usually not produce useful results. The word “look,” by itself, will work fine.

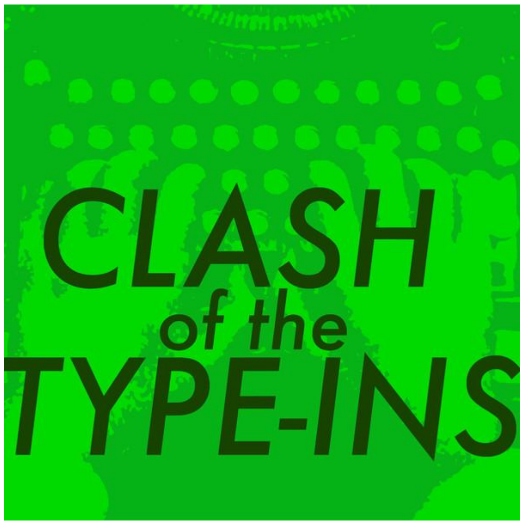
Here are some of the more popular transitive verbs that IF stories use.

Examine
Take
Open
Drop
Put
Give
Eat
Drink
Fill
Climb
Wear
Break
Burn

Popular intransitive verbs include “look,” “listen,” “jump,” and “undo,” which “undoes” the player's last move.

Individual stories frequently use unusual verbs. Try the “About” or “Help” command to find out about nonstandard verbs in the story that you're reading.

Conversing with other characters in IF can be lots of fun, but it can be frustrating, too, mainly because different stories use widely different conversation systems. That's one reason the read the “About” or “Help” file that comes with practically every IF tale. It also helps to keep in mind several patterns that most stories can understand. Directly talking to a character with a command will often work; for example, “Miss Voss, tell me about the magic stone.” Also, a reader can often make progress by asking or telling a character about something, as in “Ask the bartender about the vampire.” “Talk to the bartender” will also produce good results in some stories. Frequently, a story will interpret a single word to mean that the character says the word. In other words, “hello” will often mean the same as “say ‘hello,’” though it is sometimes necessary to type out “say 'hello.'”



Clash of the Type-Ins

The best way to learn to use the IF parser is to work through a story with an experienced reader of the genre. However, not everyone has such a tutor on call. Still, almost everyone does have a way to listen to experienced interactors as they make their way through a variety of interactive fictions. This free-and-easy tutor takes the form of a hilarious podcast called “Clash of the Type-ins” (http://rcveeder.net/clash/). In each episode, hosts Ryan Veeder and Jenni Polodna, both experienced IF readers and writers, read a story together. Most episodes include a third participant, the author of the story that's being read.

From the “Clash” website, here's a list of the first eleven episodes. You can't go wrong with any of them, but listening to them in order is probably the most fun.

EPISODE ONE: *You've Got a Stew Going!* (February 6,

2014)
In this, the first episode, Jenni Polodna plays Ryan Veeder's first game, a game about rats. Jenni says the word 'titular' kind of a lot. If horses ran the world, keyboards would be weird.

EPISODE TWO: *Dinner Bell* (February 12, 2014)
Ryan plays a game based on a They Might Be Giants song, a game that Jenni wrote for Apollo 18+20, a game that has plagued Ryan's nightmares for years. Jenni reveals that she has never played Tetris Attack. .yadot suoiciled etsat uoy teb I

EPISODE THREE: It (April 16, 2014)
Jenni and Ryan play Emily Boegheim's *It. It* is the name of the game. Cultural gaps are bridged. Murder is attempted.

EPISODE FOUR: *Taco Fiction,* Part I (May 2, 2014)
In this one we play Ryan's IF Comp-winning *Taco Fiction,* which is a game about crime. Emily and Jenni do some heinous things in this game, but Ryan doesn't even try to stop them. And you're about to listen to the whole thing! Nobody is blameless.

EPISODE FIVE: *Taco Fiction*, Part II (May 15, 2014)
The thrilling conclusion of Emily and Jenni playing *Taco Fiction*, which is a game about crime. Ryan unmasks all the game's secrets, expounds on all of its social and

ethical implications, and basically never stops talking. Even this sentence is a form of Ryan talking. You can't escape.

EPISODE SIX: *Violet,* Part I (July 11, 2014)
Jeremy Freese brings us his IF Comp-winning *Violet,* a game painstakingly calculated to drive Ryan to the brink of insanity and then drive him over that brink. And Jenni just smiles and laughs! While her friend Ryan is falling apart! Clearly whoever is writing this summary hasn't completely recovered.

EPISODE SEVEN: *Violet,* Part II (July 14, 2014)
With Jenni's help, Ryan achieves self-actualization. His mental energies finally suppress his somatic anxieties, and he succeeds in escaping from the cramped cell of repression and into the light and fresh air of true awareness. He achieves this by solving *Violet*, by Jeremy Freese.

EPISODE EIGHT: *The Statue Got Me High* (July 14, 2014)
Jenni and Jeremy play a game that Ryan wrote based on a They Might Be Giants song. Hey, just like that other game! But that game was weird, and creepy. This one will be normal, and pleasant.

EPISODE NINE: *A Day for Fresh Sushi* (January 9, 2015)

Jenni and Ryan play Emily Short's Speed-IF about an angry fish.

EPISODE TEN: *Bronze* (January 10, 2015)
Emily Short brings us one of her fractured fairy tales and Ryan works himself into a fanboy frenzy. Jenni tries to remember Dom DeLuise's first name. Shameful lunchtime secrets are revealed.

EPISODE ELEVEN: *Nautilisia* (January 20, 2015)
Crazed with power, Ryan demands that Emily Short and Jenni play a game he wrote about a person who keeps yakking about being in a coma. The person in the game does not necessarily depict Ryan. I want to make that clear.

Joys of the Parser

The interactive fiction parser is worth all the hassles it introduces. For one thing, it creates the impression that the interactor is, in some odd way, writing the story, even though the tale has an author who has provided its plot, setting, tone, mood, characters, and themes. Is this impression that the reader creates an already-authored story just an illusion? No, it's not.

To illustrate the truth of this seeming contradiction, let's consider two transcripts. Each of these transcripts shows one person's way through the first few moves of a famous IF story called *Lost Pig,* in which the player/character is an orc named Grunk. The first transcript is more typical of what experienced IF readers would to with the story, but the second one represents another approach that is not uncommon. In both transcripts, the user input is marked with the “greater than” symbol (>), as is traditional in IF.

Transcript One – A Very Common Approach

```
Lost Pig
And Place Under Ground
Release 1 / Serial number 070917 / Inform
v6.30 Library 6/11 S
(For help, use "HELP".)
Standard interpreter 1.1 (4F) / Library
serial number 04022


Pig lost! Boss say that it Grunk fault.
Say Grunk forget about closing gate.
Maybe boss right. Grunk not remember
forgetting, but maybe Grunk just forget.
Boss say Grunk go find pig, bring it
back. Him say, if Grunk not bring back
```

```
pig,not bring back Grunk either. Grunk
like working at pig farm, so now Grunk
need find pig.

Outside
Grunk think that pig probably go this
way. It hard to tell at night time,
because moon not bright as sun. There
forest to east and north. It even darker
there, and Grunk hear lots of strange
animal. West of Grunk, there big field
with little stone wall. Farm back to
south.

>Inventory
Grunk have:
  torch (on fire)
  pants (Grunk wearing them)

>Go east.
Forest look dark. Pig probably some place
in there, but Grunk not know which way
to go. Not want end up lost in forest
with no pig.

>Go north.
Forest look dark. Pig probably some place
in there, but Grunk not know which way
to go. Not want end up lost in forest
```

with no pig.

>Go south.
Grunk get in big trouble if Grunk go back with no pig.

>Listen.
Grunk get quiet. Listen for noise in dark.

Grunk hear noise! It come from some place in bushes, but Grunk not sure which way it come from.

>Listen.
Grunk get quiet. Listen for noise in dark.

When Grunk really quiet, try looking around some more. There, hear noise again!
It come from northeast. Must be pig, hiding in bushes.

[Grunk score go up one.]

>Go northeast.
If that where pig go, that where Grunk go. It hard to see where Grunk going to,

because there so much dark and leaves and branches here. Then suddenly, there not ground here any more!

Oof! Ow! Umf! Ugh!

Grunk fall down deep hole! Drop torch. Torch go out when Grunk land on it, so now it dark, too.

Hole
Grunk at bottom of deep, dark hole. It even smell like deep, dark hole. There some light that come from tunnel to east; that how Grunk see where Grunk at.
Look like there maybe stairs at west side of hole that go up to outside, but them all broken now. There big crack in ground near stairs, too.

Grunk see torch (black and sooty) here.

Transcript Two – A Fairly Common Approach, With "Attitude"

Pig lost! Boss say that it Grunk fault. Say Grunk forget about closing gate. Maybe boss right. Grunk not remember

```
forgetting, but maybe Grunk just forget.
Boss say Grunk go find pig, bring it
back. Him say, if Grunk not bring back
pig, not bring back Grunk either. Grunk
like working at pig farm, so now Grunk
need find pig.

Lost Pig
And Place Under Ground
Release 1 / Serial number 070917 / Inform
v6.30 Library 6/11 S
(For help, use "HELP".)

Outside
Grunk think that pig probably go this
way. It hard to tell at night time,
because moon not bright as sun. There
forest to east and north. It even darker
there, and Grunk hear lots of strange
animal. West of Grunk, there big field
with little stone wall. Farm back to
south.

>Inventory
Grunk have:
  torch (on fire)
  pants (Grunk wearing them)

>Scream.
```

RAAARRRR!

That fun! Grunk like shouting.

>Take off pants.
OK, pants off.

>Burn pants.
(with torch)
Grunk set pants on fire. Mmm. Grunk like fire.

>Look at forest.
Many tree and bush and leaf and branch and other plant like that. That what forest mean. It dark, too. Pig probably some place in there, but Grunk not know which way to go.

>Jump.
Grunk jump and jump, but moon still too high to reach.

>Look at the wall
Wall made of many big rock. It not very tall; Grunk lots taller than wall. Maybe it too hard, find enough big rock to make big wall. There field on other side of wall.

>Kick wall.
Grunk remember last time Grunk try hitting rock. It hurt Grunk more than it hurt rock.

Grunk hear noise! It come from some place in bushes, but Grunk not sure which way it come from.

>Kiss wall.
That only make sense for person and animal and thing like that.

>Burn forest.
(with torch)
That maybe dangerous. But then if forest all burned up, pig not have any place to hide, so maybe it really good idea. Grunk not sure. That really what Grunk should do?

>Burn forest.
(with torch)
Grunk hold torch up to tree. Tree start to burn. Then other tree start to burn. And other tree. And other tree. Then Grunk hear pig squeal! Pig run out of forest, and it not even see Grunk until

after Grunk grab it. Silly pig!

Ha ha. Grunk smart, catch pig. Grunk get to go back to farm now. Maybe boss say that Grunk do good job. Fire go toward farm, too. There lots of tree to burn. Fire go even faster than Grunk.

Uh oh. There lots of tree near farm, too. Fire get there before Grunk. Farm burn like tree. Grunk think that maybe boss not say Grunk do good job, if boss find out that Grunk make fire. Maybe it better if Grunk not go back to farm after all.

Oh well. At least Grunk get to keep tasty pig.

    *** Grunk not bring pig back to farm ***

Grunk have 0 out of 7 that time.

Time for Grunk to RESTART or RESTORE a saved story or UNDO what Grunk just do or tell FULL score or just QUIT?

> Undo.
Outside
[OK, Grunk undo that.]

>Listen.
Grunk get quiet. Listen for noise in dark.

When Grunk really quiet, try looking around some more. There, hear noise again!
It come from northeast. Must be pig, hiding in bushes.

[Grunk score go up one.]

>Go northeast.
If that where pig go, that where Grunk go. It hard to see where Grunk going to, because there so much dark and leaves and branches here. Then suddenly, there not ground here any more!

Oof! Ow! Umf! Ugh!

Grunk fall down deep hole! Drop torch and pants. Torch and pants go out when Grunk land on them, so now it dark, too.

```
Hole
Grunk at bottom of deep, dark hole. It
even smell like deep, dark hole. There
some light that come from tunnel to east;
that how Grunk see where Grunk at.
Look like there maybe stairs at west side
of hole that go up to outside, but
them all broken now. There big crack in
ground near stairs, too.

Grunk see pants (black and sooty) and
torch (black and sooty) here.
```

Looking at the Transcripts

These are, of course, transcripts of the same story. They have the same opening text, they introduce the same character, and they eventually respond to the same required moves; namely, listening and going northeast. However, the transcripts are significantly different, too, reflecting the different approaches and attitudes of the readers who contributed to them. The first transcript proceeds in a straightforward, problem-solving way, and, not surprisingly, it comes to the required moves much faster than the second transcript. It also creates a different impression of the player/character, since he, too, operates in direct and reasonable way.

The second transcript suggests that the interactor has a bit of an “attitude,” and, as a result, Grunk seems less sensible, too. The second transcript is also longer and more meandering, but it uncovers more of the comic nature of the story. The less rational approach also reveals an important fact that turns out to be essential for later problem-solving: the fact that the torch is just as good for burning things as it is for lighting.

These transcripts highlight one way in which one person's (or group's) reading of “Lost Pig” can differ significantly from another. Particularly in stories that require the reader to explore a large number of locations, readers often differ even more markedly. Some readers, or classes, stop exploring new locations as soon as they find a puzzle to solve, moving on to new places only when they're solved the puzzle or determined that they can solve it only by finding new tools in new locations. Other interactors take the opposite approach, exploring as far as they can before trying to solve puzzles. Teachers who are guiding multiple classes through the same IF story at the same time often find students asking which class is farthest “ahead” in the narrative. The answer, if there is one, often hinges on how the classes are approaching the story, more than on a counting up of places visited or problems solved.

Without the open-ended parser, which invites the reader to bring his or her own personality to the story, the sort of variation that we see among these approaches and their resulting narratives could not happen, at least not to the same degree. And, without this sort of variation, students would not experience the highly-motivational sense of agency that interactive fiction offers.

The Teacher as Parser

In a classroom, the teacher serves as the students' first parser, translating suggested student input into a form that the story's parser can understand. Suppose, for example, that uninitiated students find that their player/character is threatened with capture in a churchyard that features a large tombstone. In this situation, a student might suggest that the teacher type, at the story's command prompt, “Get out of sight behind the gravestone.” This suggestion, of course, makes good sense, but it would only confuse the parser, resulting in an error message. The teacher might translate this request into the typed command, “Hide behind the tombstone,” which would produce the desired result. In addition, the teacher might recognize a “teachable moment” for helping students to understand the nature of the parser, with its many weaknesses and its considerable value.

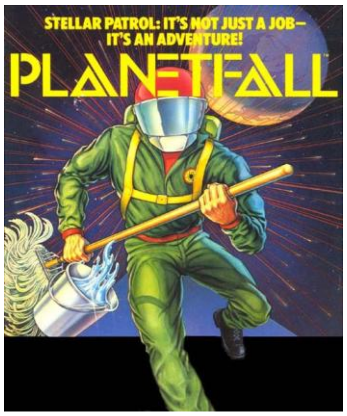
Chapter 3 – IF and Critical Thinking

Effective Thinking

Critical and creative thinking are about as central to any form of real education as any skills can possibly be. Even standardized tests claim (not too convincingly) that they want to test thinking, though it's pretty hard to figure out how bubbling the in right circle can really show good thinking in an strong sense of the word. So, what is good thinking, in a strong and clear sense?

We can understand the thinking processes involved in interactive fiction and in many other contexts as an application of Robert J. Sternberg's componential theory of intelligence. Expressed in a 1984 article in *Educational Leadership* ("How Can We Teach Intelligence?" Sept., 38-48) and elaborated on in his book *Intelligence Applied* (1986, New York: Harcourt), Sternberg's theory tries to understand intelligence partly in terms of three kinds of component processes. "Metacomponents" control intelligent behavior by Tanning, monitoring, and evaluating it. "Performance components," such as inferring similarities and differences, actually carry out the plans for thinking that the metacomponents decide on. And "knowledge acquisition components" enable the thinker to gain new information, including information that the other kinds of components may use. For example, if I were deciding to buy one of two automobiles, the metacomponents of my intelligence would enable me to choose comparing and contrasting as part of my strategy for making a good decision. I would also use metacomponents to monitor my strategy as I used it and to evaluate its outcome. The actual comparing and contrasting, though, would be performance components; and my techniques for gathering information about the cars, such as reading about them or directly inspecting them, would be knowledge acquisition components. Interactive fiction, like any kind of literature, involves all three kinds of components, but it offers an especially compelling approach to metacomponents in that it forces readers to think about how they are controlling their thinking.

Recognizing Problems in *Planetfall* and *Aotearoa*



At the beginning of one classic work of interactive fiction, Steve Meretzky's *Planetfall* (1983, Infocom), the reader must exercise a metacomponent that textbooks seldom ask students to use but that Sternberg considers essential to intelligent behavior, the ability to recognize and define the nature of a problem. Here is the beginning of the story:

"Another routine day of drudgery aboard the stellar patrol ship Feinstein. This morning's assignment for a certain lowly ensign seventh class: scrubbing the filthy metal deck at the port end of Level Nine. With your patrol-issue self-contained multi-purpose scrub-brush you shine the floor with a diligence born of the knowledge that at any moment dreaded Ensign First Class Blather, the bane of your shipboard existence, could appear."

What exactly are the problems here? Well, one problem seems to be to get the deck clean, a problem that seems to call for persistent scrubbing, and so we might try typing in at the computer keyboard, "Scrub deck." If we do so, the program responds with the not-too-exciting, "The deck looks a little cleaner now." But perhaps a more important problem we face here is a lack of knowledge of our surroundings. Maybe we should activate a knowledge-acquisition component and explore. Our location at the start of the story, Level Nine, offers two exits, a corridor to starboard and a gangway leading up. If we go up to Level Eight, we meet another character, the aforementioned "dreaded Ensign

Blather," who assigns us twenty demerits and belligerently orders us to return to our post. Here we have another apparent problem, which may suggest solutions such as obedience, arguing, or a punch in the jaw. In interactive fiction, we can try out any or all of these approaches. If we do so, we find that punching Blather causes him to dismember us (not to worry, though; in interactive fiction, death is just the program's way of telling us we've taken a misstep), that arguing with Blather at length causes us to be thrown into the brig, and that going back to work leads to a comic encounter with a broccoli-like alien ambassador who drips slime all over the deck we're shining. Eventually, though, we learn that the last of these approaches works best, since the ship we are on soon explodes and our position on Deck Nine proves especially convenient to an escape pod. In the pod, of course, we must again identify the nature of the problem(s) we face and try to deal with them; and only after we land on the planet to which the pod takes us, will we figure out that the central problem of the story is to bring the planet's inhabitants back to life and to return to the stellar patrol.

In *Aotearoa,* by Matt Wigdhal (2010), we find a similar problem-recognition challenge that confronts a younger protagonist on a

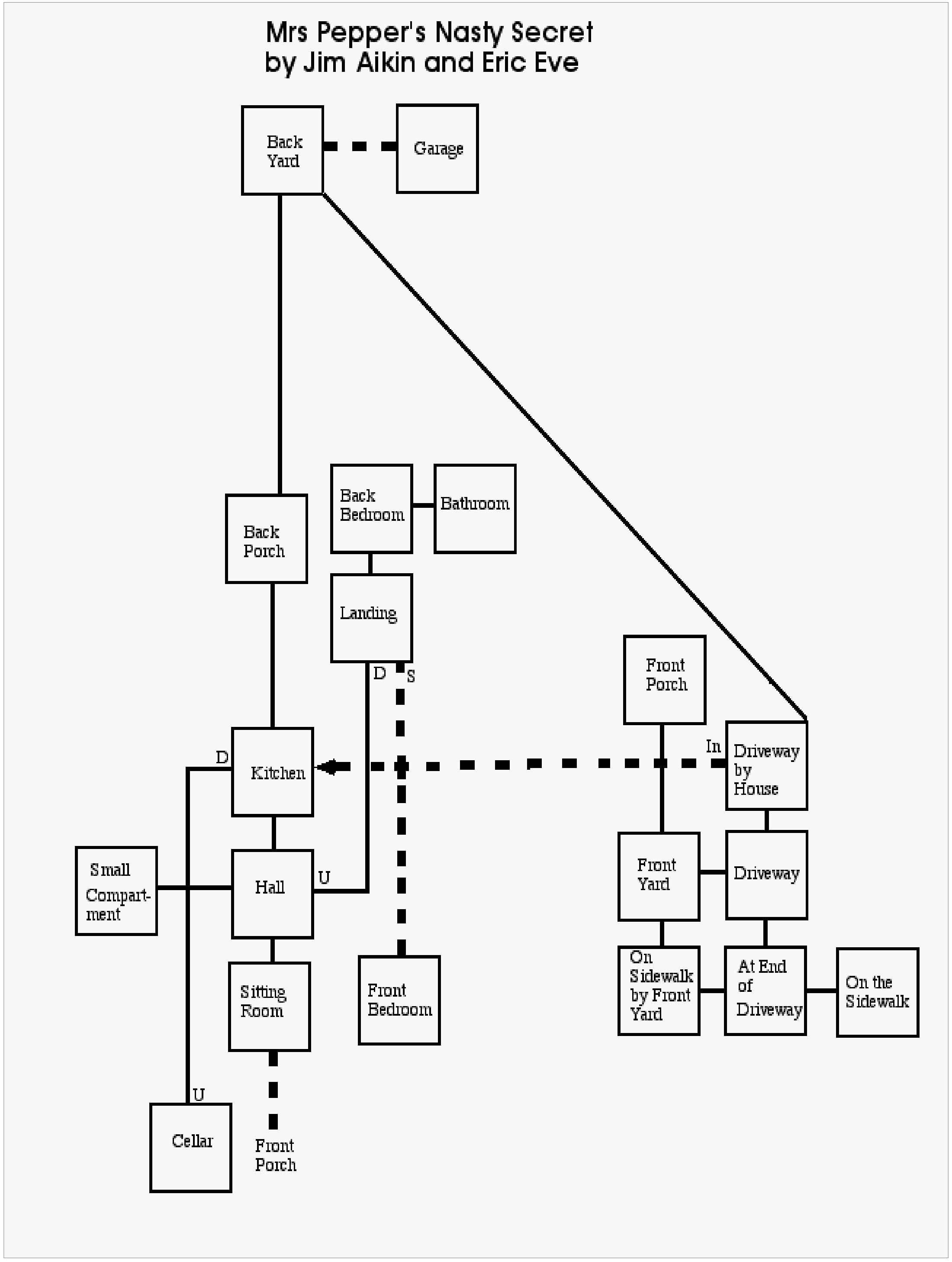
Conservation Service vessel off New Zealand. This time, Tim, a twelve-year-old contest winner, facing no obvious problem, must travel around the boat on which he finds himself, conversing with other characters and gathering information that will help him later in the story. In other words, the interactor who's running Tim's character must, at least tentatively, identify a minimum of one problem, a problem that hinges finding useful information. Eventually, as in *Plantefall*, a disaster about ship forces Tim to confront some more obvious problems and to use the information that he has gathered.

For IF beginners *Aotearoa,* offers some special advantages, in addition to its kid-friendly themes, which include family relationships and dinosaur conservation. The story, like a significant number of contemporary IF works, includes a “novice mode,” which displays helpful suggestions, based on the user's input.

Representing Problems in *Wishbringer* and *Mrs. Pepper's Nasty Secret*

Wishbringer (1985, Infocom), an award-winning work of IF by Brian Moriarty, offers an amusing instance of a second ability which, according to Sternberg, underlies intelligent behavior, the ability to decide on how to represent problem information. In this relatively easy interactive novel, the main character eventually finds himself in the lair of a dangerous, light-hating monster known as a grue, as he searches for, among other things, grue's milk. In the lair, he finds a sleeping baby grue and a refrigerator. When he opens the refrigerator door, he notes that a small light goes off, but since he's carrying his own source of light, he can see a bottle of milk inside. Unfortunately, though, because of the light, the baby grue wakes up and howls like "the screeching of a subway," summoning a horrible monster with "a calico apron and slavering fangs," which promptly dispatches the protagonist. Now, after using a few simple keystrokes to recall to the computer's memory a "snapshot" of the game just before the opening of the door, the readers have a problem to solve, a problem that will probably lead to the question, "How can I get the grue's milk out of the refrigerator?" However, with (or better, without) a little coaching, readers may see that other, more complete representations of the problem can facilitate a solution. This new formulation may be something like, "How can I kill the baby grue in order to get the grue's milk safely?" but, since there are no weapons in the story, this representation doesn't help much. Sooner or later, though, student/readers will probably try, "How can I keep the baby grue asleep in order to get the grue's milk safely?" a very helpful version in that the protagonist has easy access to a blanket. Of course, other readers may solve the problem through other sorts of alternative representations, as by drawing mental or physical pictures of the scene, thus using visualizing techniques that English teachers often urge students to try.

At the beginning of *Mrs. Pepper's Nasty Secret* by Jim Aiken and Eric Eve (2008), we find a different sort of problem-restating challenge. Here's a slightly edited version of the story's beginning.



Walking home from school is mostly okay, except for one big problem: Every day you have to pass right by Mrs. Pepper’s house. She seems to go out of her way to cause trouble for you. Once, she came at you with the garden rake, swinging it like it was a giant claw ... you still shudder when you think of it.

Last Friday was a new low. You would never, ever skateboard in her driveway — that would be practically suicidal. But somehow when you got to the driveway your skateboard swerved, all by itself, as if somebody had put a spell on it.

And then you fell off.

While you were picking grit out of the ugly scrape on your skinned elbow, Mrs. Pepper appeared out of nowhere and snatched up your skateboard! She screeched something about rowdy children, trespassing, and needing to be taught a lesson.

She swung the skateboard at you like it was a bat, and then ran off with it, cackling. Afterward you rang her doorbell for what seemed like an hour, begging her to give the skateboard back, and she wouldn't even come to the door.

And now it's Monday afternoon, and here you are, on your way home from school as usual (but with no skateboard). Just up ahead is Mrs. Pepper's driveway.

Here, the authors have given us two clear problems to consider. We apparently have to avoid more trouble with Mrs. Pepper, and our player/character would clearly like to get his or her skateboard back. Very soon, though, using a trope favored by many IF authors, Aiken and Eve offer an overriding problem. Here's the relevant passage from the story.

```
>Go west.

On the Sidewalk by the Front Yard

To the north, across a sadly dried-up and
decrepit yard, stands Mrs. Pepper’s
house.  You can also go east or west
along the sidewalk.  The busy street is
to the south.  A telephone pole plastered
with the scraps of old posters stands
here.

>Go north.

You take a cautious step onto Mrs.
Pepper’s property.  When nothing terrible
happens, you take another, your curiosity
overcoming your apprehension of the batty
old woman.

Front Yard

The front yard is mostly bare dirt and
burned-out brown lawn, except for a few
```

hardy weeds. A paved walkway crosses it, leading from the sidewalk on the south up to the front porch on the north. The driveway runs past the yard on the east side, and a tall fence guards what must be the neighbors' yard on the west. Growing next to the walkway is a scraggly, half-dead tree.

As you're crossing the front yard, a momentary lull in the traffic noise allows you to hear something very odd. Weakly, somewhere in the near distance, a voice is crying, "Help me! Oh, please, help me!" The voice is high-pitched and very hoarse, as if whoever is calling for help can barely speak at all.

You stop dead in your tracks, looking around to see where the voice might be coming from. It seems to be coming from the upstairs window in Mrs. Pepper's house. You stare hard at the window, and for a moment it seems something might be moving there — but you can't make out what it might be. After a moment the movement stops, and the voice falls silent.

Could it have been your imagination? No, you're certain you heard something.

```
And it was coming from inside the house.
```

Now, we have a new problem, perhaps related to the others in some way, but definitely more pressing. Perhaps with a bit of guidance from their teacher, students will soon determine that the problem of helping Mrs. Pepper's prisoner will almost surely have to be divided into smaller problems if it is to be solved. Even before experimenting further, thoughtful student problem-solvers might come up with a list of barriers that their player-character is likely to face. This list will, in effect, restate the overriding problem into a series of more manageable issues. These sub-problems might include determining whether Mrs. Pepper is at home, getting into her house, finding the prisoner, and helping the prisoner to escape. As students work through the story, they will almost surely find that this sort of problem-dividing skill will stand them in good stead.

Other Thinking Skills

Like the problem-recognition instances in *Planetfall* and *Aotearoa*, these problem-representation examples are by no means unusual in interactive fiction. In fact, every good piece of IF challenges the reader to use these abilities and many more that Sternberg stresses. IF readers must carefully monitor their solution processing because surprising, and sometimes even random, events can occur at unexpected times in familiar settings, as in *Wishbringer*, when the Boot

Patrol, a magical police troop consisting of gigantic, smelly boots, suddenly threatens. Readers must evaluate their solutions, since some apparently good results, such as the capture of the apparently larcenous title character in Bonnie Mongomery's *The Firebird* (1999), may turn out to be serious mistakes. Mental and physical resources must be allocated to various problems, as by deciding which of the many available objects to carry around in Laura Knauth's *Winter Wonderland (1999)*. Readers must apply old relations to new situations, as in deciding whether to respect the orders of Perelman, an important character in *A Mind Forever Voyaging* (Meretzky 1985, Infocom), a work of serious science fiction that many readers regard as one of the finest pieces of IF yet written. Likewise, readers must make automatic some elements of their information processing, as by mapping the twenty-five locations in *Robin & Orchid* by Ryan Veeder and Emily Boegheim *(2013).*

How Do We Know That IF Works?

IF is a form of text literature, a form that happens to be particularly motivational for many students. Like other forms of literature, it offers opportunities for students to study a variety of literary themes and techniques, including character, setting, and tone, though a few such techniques, especially plot and point of view, get new twists in IF.

Still, a teacher might wonder whether IF can really help students to achieve goals that may seem a little more arcane, such as controlling their own thinking more effectively. In my own school, we try to help students develop a variety of metacognitive techniques, with special emphasis on planning; and so, several years ago, I initiated a field study of IF and student planning.

How does one measure a student's ability to plan well? Such a measurement would have to be unobtrusive, since we would want to know, among other things, whether a student could recognize an appropriate situation for planning without prompting from a teacher. Further, the student would have to have a chance to show her own ability to create a good plan, without the kind of prompting that multiple-choice or short-answer instruments usually provide.

My colleagues and I decided, then, that we would need a brief essay test of some sort, a test that could pass for a simple journal-writing assignment, while enabling us to reliably observe students' planning skills. After a course with Edys Quellmalz, a nationally-prominent designer of essay tests of critical thinking, I developed two prompts for journal writing assignments that, answered in a complete and thorough way, would require the student to set out an explicit plan. Then, with the help of my colleagues, I worked out and tested a scoring rubric.

In 1989, I tried out the instrument with a real class, using one prompt before the students studied planning through IF and another after they did so. The pre-test showed little sensitivity to the need for planning, though students who planned at all showed some skill at it. The post-test revealed a clear improvement, statistically signigicant at the p<.003 level.

I do not claim much for this study. Indeed, I suspect that one could obtain similar results though skillful teaching for planning without any use of IF at all. Nevertheless, IF offers, for students of literature, a unique and appealing vehicle; and this study offers evidence that it works, as part of a well-conceived effort to help people think more effectively.

And what about those unwelcome standardized tests? As you might expect, there are no rigorous studies on interactive fiction and high-stakes testing. However, my middle school students, who received a substantial part of their English language arts instruction, over several years, did very well on the Massachusetts (USA) Comprehensive Assessment System (MCAS) tests, which were among the most rigorous in the nation. In 1999, when the standardized-testing boom was first gathering momentum, only one grade in each middle school had to take the MCAS Language Arts Test. In that year, as it happened, a significant percentage of the tested students worked with interactive fiction, and the school achieved the second highest MCAS Language Arts scores in the state. In 2000, the tested grade, many of whom had extensive interactive fiction instruction, tied for the best language arts scores in Massachusetts. Through the subsequent years of the deceptively-named “No Child Left Behind” initiative, students who studied interactive fiction continued to do well, even in the context of a high-achieving school that reached every “Adequate Yearly Progress” target through 2008.

Try It!

With one student at the computer, even an older PC or Mac, typing what a class wants to try and reading the results aloud, and the rest of the class actively engaged in mapping, keeping track of problems, and generating suggestions, interactive fiction can become an engaging experience for groups of almost any size, an experience that involves students in the essential kind of thinking that we call reading. With a large-screen display, the group reading of IF stories can be even more dramatic. IF encourages the kind of systematic, consciously planned metathinking that Sternberg advocates and that English teachers know their students need. And at its best, it offers a surprise that is of great value: as a new and vibrant (if not yet fully developed) form of literature, it can spark a renewed sense of wonder at the power of the written word.

Chapter 4 – Interactive Fiction and the Reading Process

The Pause Problem

Suppose that you and your seventh-grade class are reading aloud, in Chapter 9 of Catherine Patterson’s famous young adult novel, *Lyddie*. Perhaps you're using old-school round-robin techniques, or maybe you've moved on to approaches like choral reading or echo reading. In any case, you know that most of the students in the class don’t especially like to read, but, for now at least, the group seems quite caught up in the narrative, as the main character goes to visit her new friend, who, according to her housemates, is a notorious labor agitator. Around the middle of the chapter, the class encounters the word, “phrenologist,” complete with sufficient context to enable the students to make a very reasonable guess at its meaning. You know that, according to recent high-stakes, high-profile statewide testing, students in your school are not especially good at determining meaning from context; and so you are tempted to interrupt the reading process here to do a little direct teaching, or perhaps just to remind students to use techniques that they already know to glean the meaning of the unfamiliar word. If you don’t stop now, you know that you’ll lose an opportunity to do some good teaching – the occasion just won’t be as fresh if I wait until you reach the end of the chapter. But, of course, if you stop the process now, when the students are enjoying a real aesthetic encounter with the printed word, you may lose in motivation more than you gain in skill-building.

"Wouldn't it be great," you might muse, "if the author of this story had built into the story's design lots of good stopping places – places that occur frequently, not just at the end of chapters, and places that work aesthetically, not just pedagogically?" But, of course, novels are not built that way.

Perhaps you decide to take the phrenology pause. The kids are pretty patient and attentive, and you manage a bit of clear, direct teaching, but nobody likes breaking the story up this way, and the rest of the chapter is just not as powerful, though you may decide to keep subsequent interruptions to a minimum.

The next day, the same class is reading a passage in *Arthur: the Quest for Excalibur* (1989), a novel-length work of computer-based interactive fiction by Bob Bates. As the main character, the youthful, pre-coronation Arthur, approaches a peasant's cottage, he finds something called a "slean." Does the class pause in its reading process to figure out what a slean is? In the context of interactive fiction, the question is absurd. Of course we have to figure out what a slean is. If we don't, we probably won't be able to continue reading the story at all, at least not for very long. Unlike the *Lyddie* class, this group doesn't mind the pause at all. Indeed, the author, in constructing his story, has made an aesthetic judgment that just such a pause belongs at this point in the tale.

How IF Pauses

Interactive fiction, of course, is a literary form in which authors must build pauses into their stories. Since the reader plays the part of an important character, deciding, within limits, what action that character will take, the story must pause, quite frequently, to give the reader a chance to contribute to the narrative.

Motivation

The way kids take to interactive fiction is really quite striking. Since 1985, the author of this book has introduced about a thousand students, mostly aged eleven through nineteen, to the genre. A clear majority of them like it. In fact, it quickly becomes the most popular form of literature with most of them, especially when it's read orally, in a large group. Students like interactive fiction mainly because it's an exciting way to read a story, a way that lets them feel very active and involved. They enjoy using IF to gain experience with all of the major elements of literature, such as plot, setting, and point of view. Many young people also like the problem-solving that comes with the IF experience. These kids appreciate interactive fiction because it challenges them to recognize and solve problems in ways that no textbook seems able to match.

Given the choice of reading conventional literature or interactive fiction, most of the students in the author’s classes choose IF, especially for reading aloud. This result may seem unremarkable at first. After all, most people, old or young, like to try something new. However, even after numerous opportunities to choose over a period of months, and even when the available hard-copy reading includes highly motivational books, magazines, and newspapers, most students prefer to work with the always-challenging computer-based form.

Elements of Conventional Literature

Beyond its motivational effect, a second important advantage of IF is that it offers a straightforward way for students to learn about the elements of conventional literature. For example, though the IF reader has great control over what the main character of a stories tries to do, a work of IF still has a largely conventional plot, with an exposition (often in the form of conventional text), a rising action (albeit one in which the order of events can vary somewhat from one reading to another), a climax, and a denouement (or, occasionally, more than one possible ending). Unlike many other electronic storytelling environments, such as Multi-User Simulated Environments or Habitats (MUSEs or MUSHes), interactive fiction does not present itself as a way for students to create their own stories by interacting with other authors within a digital mini-world. In IF, students do get some sense of building their particular readings of the stories. However, IF remains quite close to the experience of reading a well-constructed novel or short story.

Pausing in Interactive Fiction

An important advantage of IF in the classroom, as outlined at the beginning of this chapter, is its way of providing – and, indeed, forcing – aesthetically valid pauses in the reading process. Of course, not all interactive fiction works equally well in offering the pedagogically-best pausing points. In truth, many early, and, in some ways, primitive works of IF, such as a well-known series by Scott Adams (not the cartoonist) offer little evidence that the author has made literary calculations about the placement of pauses for puzzles. In these stories, there are no extended passages of text to interrupt, just a series of interrelated problems, connected with a tight little plot; and the reader soon comes to expect that the solution to one puzzle will lead immediately to the next problem-solving exercise.

Several of these pieces, such as Pirate Adventure and Adventureland (1978), can certainly engage and entertain a puzzle-loving reader, but they offer little in the way of theme and character development.
But works of interactive fiction in its more mature variations offer a dramatically different set of opportunities for literature teachers. Some of these, such as Adam Cadre's brilliant interactive short novel *Photopia* (1998) move away from lengthy problem-solving altogether.

In literarily complex stories like *Photopia,* the reader must still pause often, sometimes briefly and occasionally at greater length, to decide on what action a character should take, but the appeal of the tale stems almost entirely from conventional literary elements, especially an intricately woven plot and highly engaging characters. In one scene, for example, a father and his precocious little daughter look up at the sky outside their garage and talk about one of their common interests, astronomy. The reader has no real problems to solve, but must stop to make some choices about what one of the characters, in this case the father, will say, and these pauses offer a literature teacher some remarkably teachable moments. Gradually, the thoughtful student reader, with the right kind of help, comes to see that the astronomical concepts that emerge from the a touching father-daughter dialogue illuminate another subplot of the story, one in which the daughter, some years later, weaves a tale of space travel for a younger girl who idolizes her.

Other mature IF stories, however, such as *Arthur: the Quest for Excalibur* and *Once and Future* by Kevin Wilson (1998) take a different approach, maintaining an extensive puzzle-solving dimension, but adding rich narrative elements. Often, in these stories, some of the puzzles are far less mechanical than those in the earliest IF, depending more on a good, clear sense of plot, character, and theme. At one point in *Arthur*, for instance, the reader, in the role of the title character, encounters a knight who challenges the young Arthur to a joust. Before the combat begins, the knight shows a gentlemanly sense of fairness, insisting, for instance, that Arthur wear the appropriate protective equipment ("Knight in shining armor and all that, don't you know?"). But as the mock combat progresses, the knight feints in a way that suggests that he may be about to cheat. If the reader accepts the feint as sufficient evidence of duplicity, the knight will always win the joust. If the reader understands the knight’s character well enough to see that he would probably not cheat, Arthur will win, gaining fighting skill and a useful trophy. Here, once again, the teacher has an excellent opportunity to guide students in the operation of an important literary element, with the help of a pause that the author has structured into his narrative.

IF, then, helps teachers to focus on whatever instruction they need to be doing, whether in response to read needs of students or to the more artificial and oversimplified demands of standardized tests.

Getting Organized for Teaching with IF

Later chapters in this book discuss, in some detail, particularly useful interactive fictions. These chapters offer ways to organize instruction to maximize the educational effects of these specific works, with their highly-productive pauses. For now, let's have a look at some more general tools for structuring our work with students.

Generally, it is helpful for each student to have an interactive fiction folder to keep together print materials, such as background information and maps, which are necessary for most of the stories. Of course, electronic folders will work fine for this purpose, when the necessary technology is available. If, however, like most of us, the teacher has limited access to computers, manila folders are excellent, too. The cover of a manila folder makes a good place to have each student record class goals for studying IF, such as learning about a new form of literature, learning to manage his or her thinking more effectively, and learning about a variety of literary techniques, such as plot, character, setting, and point of view.

Some of the numerous pauses in a typical interactive story introduce puzzles or problems for the reader to solve. Typically, a classroom full of interactors will be working on several of these problems at the same time, and so will need to keep track of which problems are current and which are solved. For this reason, each student's IF folder should contain lists of problems or puzzles encountered in each story, perhaps in a small "blue book" of some sort. Each puzzle should probably have a page of its own, to allow room to record restatements of the problem, possible solutions, and confirmed solutions.

If students are to work on stories individually, whether in or out of the classroom, they will generally need a more generous supply of printed aids than those who have a teacher always at hand. These materials may include more detailed hints, maps, or even "walkthroughs," which present solutions to a story's problems. It's often quite fascinating, and surprising, to see how much careful reading and problem-solving a student must do to complete a work of IF, even if he or she has a walkthrough in hand.

Self-Evaluation in IF

IF, then, with its unique structure of narrative pauses, offers special opportunities for direct teaching. However, it also adds an evaluative dimension of considerable instructional power, an element that operates even when the teacher isn’t around. How many teachers have felt exasperated at a student’s declaration that he or she has completed the reading of a work of literature without understanding it at all? And how many students, at least the more conscientious ones, have felt even more frustrated in the same circumstance? With most IF, though, it is simply impossible, short of getting the problem solutions from someone else, to finish a story without understanding it in some depth. The careless or unskilled reader will become “stuck” on one or more of the problems and will thus be unable to continue beyond a particular point. The aesthetically-placed pauses for problems thus become, among other things, compelling and integrated reading comprehension tests, perhaps the only such tests that most students will take voluntarily.

Chapter 5 – Building Fluency with Interactive Fiction

As we've seen, interactive fiction is a form of text literature in which the reader plays the part of an important character, deciding, within limits, what action that character will take. By typing ordinary English sentences at the keyboard, the reader or, frequently, a group of readers, decides where the main characters will go, what objects they will pick up and use, how they will solve problems, and how they will interact with other characters. Many students find interactive fiction, also known as IF or adventure gaming, an enjoyable way to gain experience with all of the major elements of literature (although point of view takes an unusual twist or two), and teachers who are comfortable with it soon find that it works well for oral reading in a one-computer classroom, largely because it encourages students to work collaboratively in solving the problems that the protagonist encounters.

Interactive fiction offers lots of instructional advantages, including its motivational effects; its usefulness in teaching conventional literary elements such as plot and theme; its unique qualities as a problem-solving tool; and its natural inclusion of helpful stopping places for instruction. But interactive fiction has a less obvious advantage, too. It’s a uniquely powerful tool for helping students to read more fluently.

Building Fluency

Fluency, the ability to read to read aloud quickly, accurately, and expressively, is a quality that all teachers would like their students to have. Fluent readers enjoy reading more than other students, and they have fewer comprehension problems. Fluency helps children to move into the world of literate adults.

Fortunately, we know a good deal about how to teach fluency. The National Reading Panel, in *Teaching Children to Read* (2000), concludes, from an analysis of seventy-seven research studies, that “guided repeated oral reading procedures” have a positive impact on fluency. These procedures take a variety of forms, but all of them require students to read the same passage a number of times as a way of increasing fluency, and most culminate in opportunities to show off newly-developed fluency by reading for classmates. Some educators place special stress on the opportunities for relatively public performance, as a way to motivate students to engage in repeated reading.

Repeated Reading

Guided, repeated oral reading, then, works as a teaching tool. However, it’s not the way literate adults and kids usually read. The literature that we find in books and magazines, with the possible exception of a poem here and there, does not ask the reader to peruse the same text over and over again. But what if there were a narrative form of literature that was especially well suited to classroom performance in the form of guided oral reading? And what if the same form required repeated reading as an integral part of its storytelling? And what if it were available free (or almost free)? Computer-based interactive fiction offers all of these qualities.

IF and Reading Aloud

In my twenty years of work as a middle school teacher, I found that students generally enjoy interactive fiction a great deal. In fact, about thirty percent of my students choose IF, over all other options, for individual silent reading. However, they like IF much more when they can read it together, with the level of preference rising to about eighty-five percent.   Why does this form of literature motivate oral reading more than silent reading? Three factors conspire to produce this result: collaborative problem solving, reading text in appropriately-sized chunks, and a unique sense of creating a story while reading it.

Guided Reading

In order to foster reading fluency effectively, it’s not enough to offer motivational oral reading–we have to offer guidance as well. Sometimes teachers offer coaching in oral reading “on the fly,” reminding students of the need to pay attention to pitch, stress, and juncture as they read; but, on other occasions, more formal approaches, such as phrased-text lessons, with groups of related words marked off for the reader, yield dramatic results. In any case, explicit guidance requires pausing in the reading process.

An important advantage of IF in the classroom is its way of providing–and, indeed, forcing–aesthetically valid pauses in reading. Every work of interactive fiction, regardless of its level of sophistication, must wait for the reader’s input; and every author of an interactive story has to make aesthetic judgments about the placement of the waiting. Typically, even when a reader is making rapid progress through a story, these periods of waiting chunk the text into pieces that are seldom more than a hundred and fifty words long. In other words, interactive fiction stories naturally fall into brief episodes that are ideal for guiding a novice reader.

IF and Repeated Reading

How does interactive fiction incorporate repeated reading? We can best answer this question by considering a typical example of middle-school-appropriate interactive fiction, *Wishbringer*, by Brian

Moriarty. In this tale of fantasy and adventure, the main character, a young postal clerk, must save his or her town from the effects of some rather elaborate, transformative, evil magic. Like most works of interactive fiction, *Wishbringer* requires the main character to visit a variety of locations. When the character visits a location, the reader sees a description like this one:

```
Rotary East
You're on the eastern side of the
Festeron Rotary. A street branches off to
the east, towards the bay.

On the corner nearby stands a charming
little movie theater. Showtimes and
admission prices are listed on a schedule
near the closed entrance, and a marquee
announces the current feature.
```

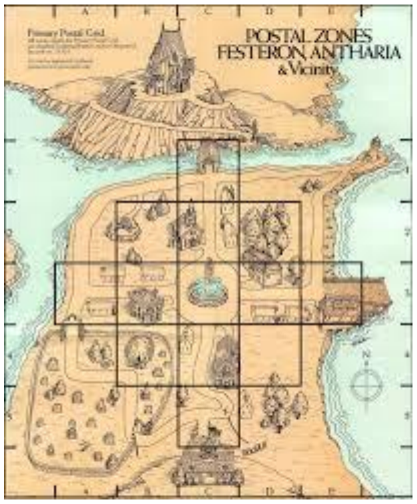
If the protagonist goes east from Rotary East, he or she comes to another location, the Pleasure Wharf, described like this:

```
You're standing near the entrance to the
Pleasure Wharf, the town's most popular
tourist attraction. The Wharf extends
eastward into Festeron Bay, and a tidal
beach curves north along the shore.
```

```
To the south stands a ramshackle old
building. Colorful lights, curious
electronic sounds and a neon sign beckon
you through the open entrance.
A big mailbox is nearby.
```

If the player/character enters the ramshackle old building, he or she reads the description of a video arcade:

```
This old building is the home of a sleazy
arcade, lined with coin-op video games.
The machines are all deserted and quiet,
except for one in the corner that emits a
feeble "wokka-wokka" sound.
A sign on the wall says, "All Games One
Token."
```



It should be clear, then, that, in *Wishbringer*, the player/character spends a lot of time moving through various locations. He or she does lots of other things, too, some of them much more interesting than mere movement, such as talking to other characters and solving problems. However, in order to make progress, the protagonist must move around the tale’s map, reading and understanding the descriptions as he or she goes. Most works of IF make similar use of various locations. In fact, a typical interactive story is often described as a “romp around a map.”

In any given story, some of the locations will require only one or two visits, but others will require many more, commonly at least a dozen, by the time a typical story, which is around the length of a short novel, reaches its conclusion. The location called “Rotary East,” as described above, for instance, will almost surely be visited by the player/character early in the story, as he or she heads off to deliver a letter to the tale’s northernmost locale. Later, the character will pass the location again, as he or she tries to reach a mysterious tower at the southern end of the story’s map. Later still, the player/character will visit Rotary East on his or her way to the Video Arcade, after obtaining a token to play one of the games. At another time, the protagonist will return to enter the theater. On another occasion, the player/character will pass Rotary East on route to a lighthouse that is located in the northeast corner of the story’s territory. And all of these particularly purposeful visits do not include many incidental passings, as the protagonist is chased by the antagonist's police patrol or as the player/character simply looks around without any particular object in mind.

Near the end of *Wishbringer,* the player/character makes a climactic visit to a location called Rotary South, a visit that underscores the importance of reading the description of a location carefully and repeatedly. Here's the description of Rotary South:

```
This is the south side of the Festeron
Rotary. A road branches south, towards
Post Office Hill.

The Festeron Public Library, famous for
its museum of local historic artifacts,
stands proudly on the nearby corner.
```

On this occasion, the library, which has seemed like a bit of permanently-locked scenery throughout the story, has suddenly become accessible because the protagonist has obtained a key from the librarian's cottage. If the readers miss the importance of the library, simply because they've read its description a dozen times or more, they will not be able to solve the story's climactic problem.

Approaches in the Classroom

One of the principal advantages of interactive fiction as a tool for building fluency is its flexibility. For some students and classes, the repetition experienced in a straightforward, guided, oral reading of a story will suffice. Other learners require procedures that are a bit more scripted and formal, though still unobtrusive. Still others may need more rigorous and immediate repetition.

As one might imagine, students, who are presumably reading each work of IF for the first time, will have little idea, at first, which map locations require frequent visits. But any teacher who has read an IF story even once will have a very strong feel for which locations require frequent visits. Using this information, the teacher can develop a plan to help students with moderate problems in fluency. For these learners, a teacher can model good oral reading of each description that will be repeated, by reading aloud the initial instance or two of such descriptions. Then, the teacher can assign the oral reading of each particular, frequently-visited location, and perhaps its adjacent venue, to a student who needs some fluency development. From then on, the assigned student will have an opportunity for repeated reading of the description, with any guidance the teacher chooses to give, whenever the main character enters the assigned location. Eventually, the student will, very likely, be able to read the description with some smoothness and expression.

In some classes, a teacher may choose to keep the assignment of students to particular descriptions rather discreet, revealing the assignments to each student in private. In other cases, as with most of my sixth graders, students may be quite happy to “own” Rotary South or the Video Arcade. In extremely sensitive situations, such as those that I’ve occasionally encountered with low-achieving eighth graders, the teacher need not reveal the assignment at all. He or she can simply remember to call on the particular student whenever the appropriate destination is reached.

Of course, interactive fiction is not a one-size-fits-all solution for students who need work in fluency. Some students with more severe fluency problems, for instance, will need several repeated readings of a passage that occur in rapid succession, in order to improve fluency. For this sort of rapid repetition, IF is not dramatically better than other forms of literature, but it’s no worse, either. If a student needs immediate repetitions, and the teacher can provide opportunities for this sort of work, the well-rehearsed reader will find no more striking opportunity to show off his or her practiced fluency than an interactive story.

Issues and Problems

Most middle school teachers do not have daily access to the latest computer technology. Fortunately, though, all-text interactive fiction makes only the lightest demands on computer hardware. A Pentium II-based computer can run all of the stories listed in this article with no delays at all. When they originally appeared, most of the commercial stories ran perfectly well on Apple II machines.

Large-screen displays that a whole class can see at once are a bit more of a problem; but, since IF stories can be run with large font sizes, they can be displayed without any sort of expensive projection equipment. A large television/monitor, attached to the computer with a VGA cable, with a Chromecast device, or with a similar interface, does the job perfectly well.

The real challenge of using interactive fiction in the classroom is not technological at all — it’s literary. Most adults have not learned a truly new form of literature since they were introduced to novels and plays as children; and IF is designed to be a challenging genre. Learning the kinds of sentences that interactive stories can “understand” is tricky in itself; and most IF deliberately confronts the reader with problems that are somewhat difficult, even though walkthroughs and hints are readily available. Because teachers are usually accomplished readers, we may come to assume that, for the most part, the reading of good literature is easy. Interactive fiction challenges that assumption.

Still, help is available. The welcome page for the newsgroup rec.games.int-fiction (http://www.faqs.org/faqs/games/interactive-fiction/part1/) is very useful. The Interactive Fiction Forum (http://www.intfiction.org/forum/) is even more active.   “A Beginner’s Guide to Interactive Fiction” (http://www.brasslantern.org/beginners/beginnersguide.html) by two outstanding IF authors, Stephen Granade and Emily Short, offers succinct and clear suggestions. For a young person’s perspective, try “Fun and Learning With Interactive Fiction,” a section of this book designed for kids. Fredrik Ramsberg’s “Beginner’s Guide to Playing Interactive Fiction (http://www.microheaven.com/IFGuide/) takes a systematic, step-by-step approach that appeals to many IF neophytes.   It is distinctly helpful with the process of downloading and using the software that’s needed to run interactive fiction.

The difficulties in learning IF, then, are certainly real, but so are the resources; and the people of the international interactive fiction community, a very active and smart group of real enthusiasts, are always ready to help “newbies.” For teachers who go to the considerable trouble of learning this new form of literature, IF will serve as a unique, perhaps even indispensable, tool for building reading fluency.

Chapter 6 – Creating Interactive Fiction With Adrift

Once students have tried and enjoyed interactive fiction, they often inquire about writing their own works of IF. Frequently, students are impatient to get on with the IF-writing adventure, and a realistic teacher may have to rein them in a bit. It doesn't make much sense to try to write in a genre that you don't really understand.

Still, creating interactive fiction can be enjoyable and educational. The necessary first step, if a real work of fiction is the goal, is for the student-authors to produce a non-interactive script of the works they hope to produce. Creating a good script will require students to apply the usual stages of the writing process: prewriting, drafting, revising, editing, and publishing.

Here's the script of a very simple interactive story.

```
“Lost Chicken” Script

You've made it to your home, as usual,
but it seems that you've forgotten your
key.

Lost Chicken
An Interactive Fiction by Brendan
Desilets

Outside Front Door
```

```
You're outside your front door.  The door
is to the west, and your front yard is to
the south.

You can see a doormat and an oak door
here.

>i
You are carrying:
  a pet treat

>x doormat
You find a note under the doormat and
pick it up.

>read note
The note reads, "The chicken hides the
spare."

>s

South Yard
This is the lovely south yard of your
home.

You can see Bronson Alcatt and a plaster
chicken (closed) here.

>look at Bronson
```

```
Your very old cat, always a formidable
presence.  He looks particularly grumpy
right now, as he rubs against his
favorite lawn ornament, a plaster
chicken.

>open chicken
Your irascible cat scratches you,
preventing you from getting at the
chicken.

>give treat to Bronson
Bronson gobbles up the treat. He looks
less grumpy now.

>open chicken
You open the plaster chicken, revealing a
key.

>take key
Taken.

>n

Outside Front Door
You're outside your front door.  The door
is to the west, and your front yard is to
the south.
```

```
You can see a doormat and an oak door
here.

>w
You unlock the door and enter your house.

Front Hall
You have made it inside, where Bronson
happily joins you.


    *** The End ***
```

Of course, the script is just an early step. Even with the best of scripts in hand, a writer needs a way to create the interactivity that students enjoy so much. Fortunately, though, a number of excellent authoring systems for interactive fiction are available.

ZIL and TADS

In the commercial heyday of IF (the 1980's), professional writers and programmers produced the works with authoring systems that looked like other programming languages. The most prominent publisher of interactive fiction at this time was an MIT spinoff called Infocom. The writers who worked at Infocom used a proprietary system called Zork Implementation Language (ZIL). ZIL is available today, but it's not a system that current writers use.

At present, the most popular authoring system of the programming-language type is TADS 3 by Mike Roberts. This is an outstanding authoring system in many ways. It has excellent documentation, great flexibility, and an active community of users. When TADS first appeared in 1988 as shareware, it was the only fully-featured system of its kind. Now, TADS is completely free of charge.

Of course, if you're not a teacher of computer science, you're probably not interested in showing your students how to program in a language that looks like C, with a few bits of Pascal. The whole approach is just too arcane for most of us. As an example, here's a bit of TADS 3 code, courtesy of the “Get Me Writing” website (http://www.getmewriting.com).

```
entryway: Room ‘Entryway’
“This large, formal entryway is slightly
intimidating:
the walls are lined with somber portraits
of gray-haired
men from decades past; a medieval suit of
armor<<describeAxe>>
```

```
towers over a single straight-backed
wooden chair. The
front door leads back outside to the
south. A hallway leads
north. ”
describeAxe
{
if (axe.isIn(suitOfArmor))
“, posed with a battle axe at the
ready,”;
}
north = hallway
south = frontDoor
out = frontDoor
;
+ frontDoor: Door ‘front door’ ‘front
door’
“It’s a heavy wooden door, currently
closed. ”
initiallyOpen = nil
dobjFor(Open)
{
action() { “You’d rather stay in the
house for now. “; }
}
;
```

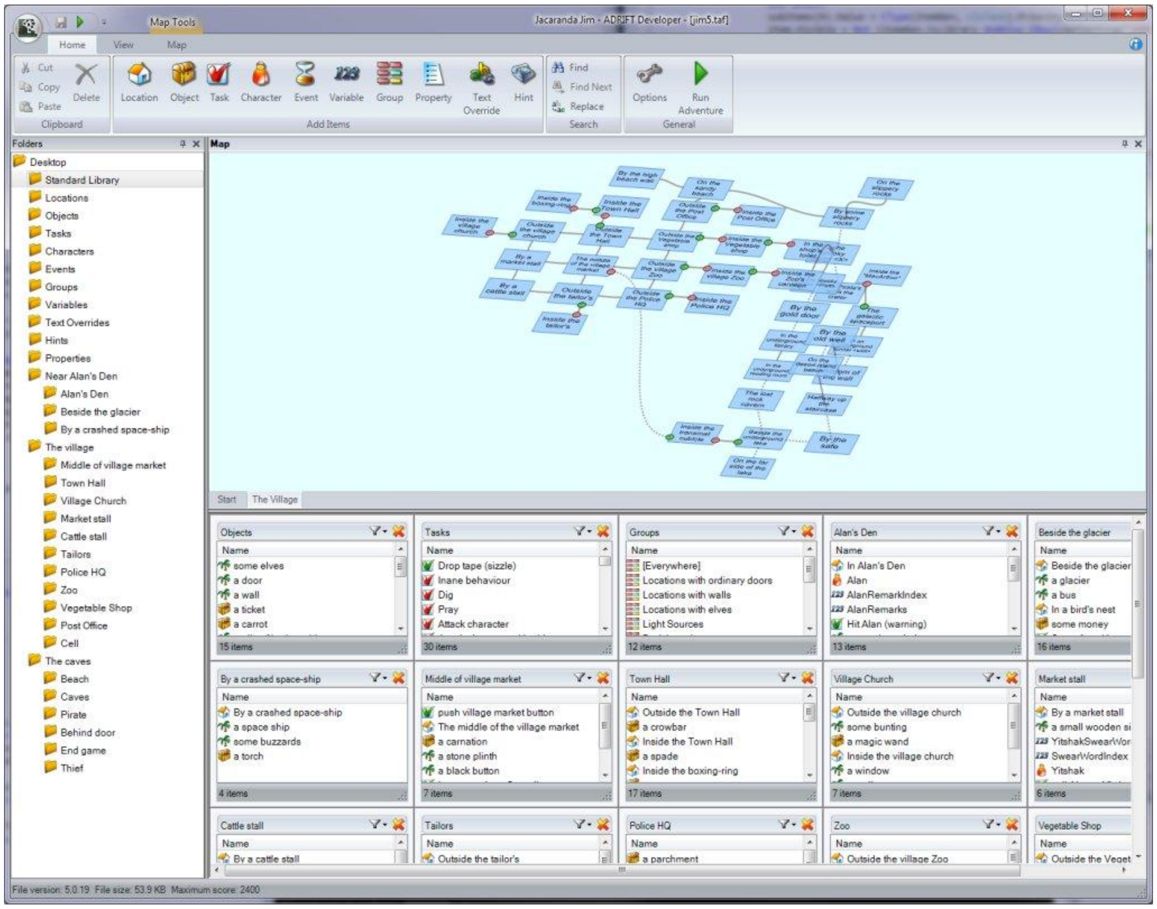
Inform

In 1993, the British polymath Graham Nelson introduced Inform, another authoring system of the programming-language sort. By the end of 1996, the system had developed into a very stable version, Inform 6, which came to rival TADS in popularity. In 2006, Nelson published Inform 7, ushering in a dramatic change. Inform 7 does not look very much like a programming language at all. In fact, its source code is, more or less, a subset of ordinary “natural language.” Inform 7 is a tool of immense educative power, and it's the topic of three separate chapters in this book.

Like TADS, Inform, in its varying versions, runs on Windows, MacOS, and Linux, and features remarkably thorough documentation. All versions of Inform are free.

Adrift and Quest

In 1988, English developer Campbell Wild introduced Adrift Developer, a Windows-only tool for making IF stories with a minimum of programming.



Adrift tries to provide most of the functions of TADS and Inform, through a series of forms and menus. Adrift lacks the power and flexibility of TADS and Inform, and it cannot approach the thoroughness of documentation that characterizes the other systems. However, it is probably the fastest way for students to produce simple works of interactive fiction – and more advanced stories, too -- complete with pictures and on-screen mapping. Adrift started as commercial software, but is now free of charge. An Adrift tutorial appears later in this chapter.

In 1998, Alex Warren, a British developer, published Quest, another user-friendly way to write interactive stories. Quest started out as a Windows-only platform, but it now offers a Web version as well. Like many Web versions of PC software, Quest on the Web is somewhat slow, buggy, and truncated, but it's usable. Initially, Quest may involve a slightly steeper learning curve than Adrift, but it has an advantage for educators in that offers ActiveLit, a private place on the Web for classes to use in working with interactive fiction. The Quest website includes a helpful series of tutorials, at http://docs.textadventures.co.uk/quest/tutorial/tutorial_introduction.html. These tutorials focus mainly on the Windows variation of Quest. In the next chapter, we'll offer a tutorial on the Web version.

ActiveLit

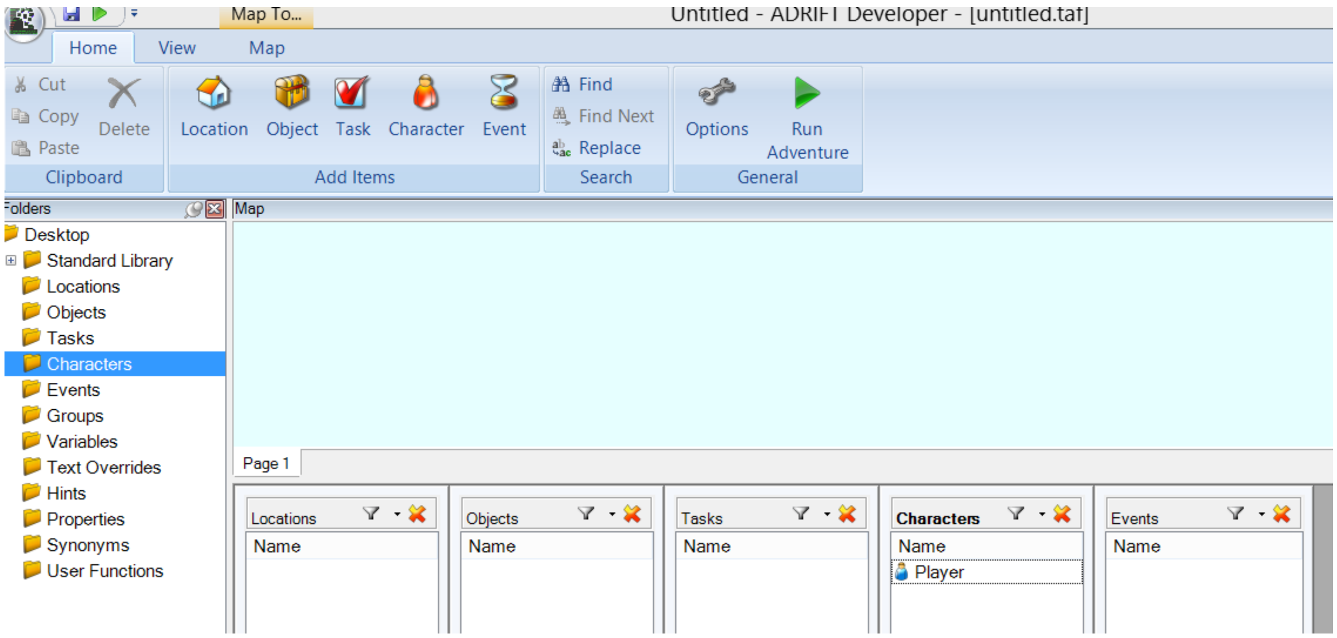
A Tutorial for Adrift, Version 5

Since Adrift is probably the fastest way for students to get their own IF stories up and running, we offer this rather lengthy tutorial.

Adrift Developer costs nothing and is available on the Web at
http://www.adrift.co.

For this tutorial, we'll use the story “Lost Chicken” as an example. A transcript of this story appears earlier in this chapter.

When Adrift Developer starts, it usually displays five windows, using its “Simple Mode,” which is adequate for this tutorial. The windows are labeled “Locations,” “Objects,” “Tasks,” “Characters,” and “Events.”



Use these steps to get started with Adrift Developer:

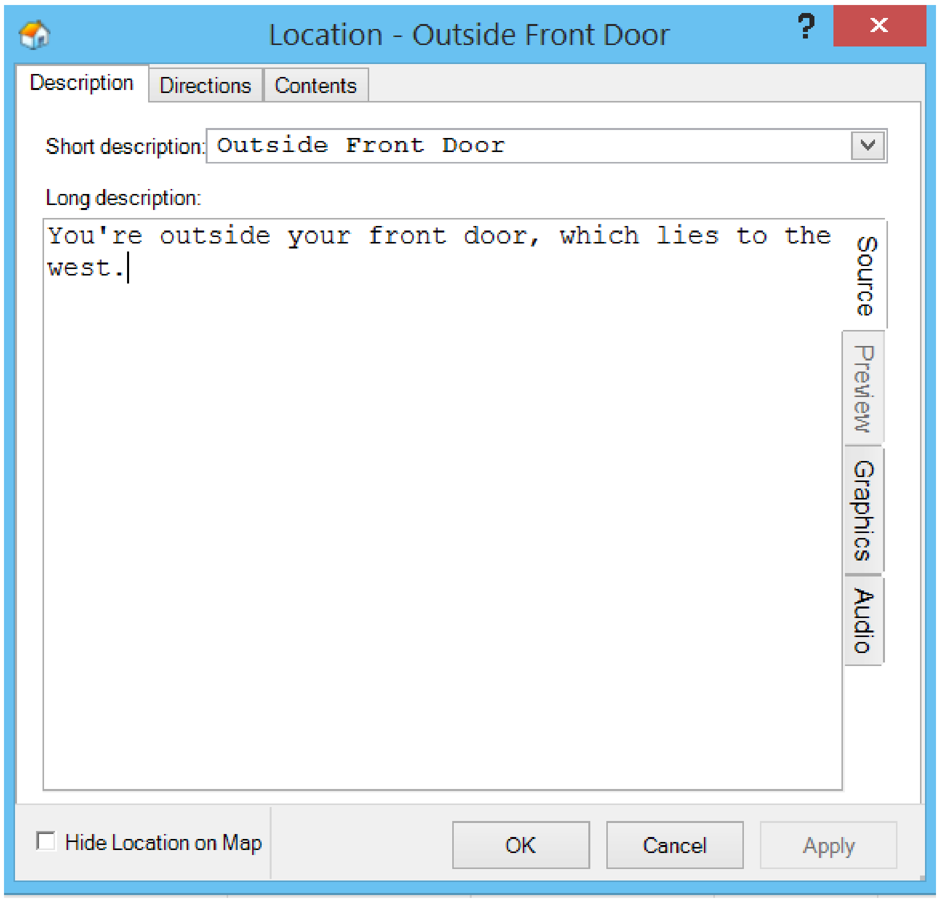
1. Start the computer program called Adrift Developer. Click on the “Home” tab at the top of the screen, if that tab is not already selected.
2. Click on “Options” and then on “Bibliography.” Type in the title of your piece of writing and your name, in the appropriate spaces.   Then, click on OK.
3. In the “Characters” section of the Adrift Developer screen, you'll notice that one character, the player/character, has already been created. Double-click on the word “Player,” and type in the name of your player/character. Then type a brief description of your player/character in the appropriate space. Click on the appropriate gender for your player/character, male or female. Then click on OK.

Create your first location.

1. On the Adrift Developer screen, locate the

section labeled “Add Items.” In this section, click on “Location.”

2. In the form that opens, fill in the “short name” of the location. This name should be three words or fewer in length. For our “Lost Chicken” story, we'll use “Outside Front Door” as the short name of our first location.
3. Then, in the appropriate space, fill in the “long description.” You can copy and paste the long description from your word processor, if you like. For our long description of “Outside Front Door,” we'll use “You're outside your front door. The door is to the west, and your front yard is to the south.”
4. Click on “OK” when you have finished with this form.



Short description: Outside Front Door

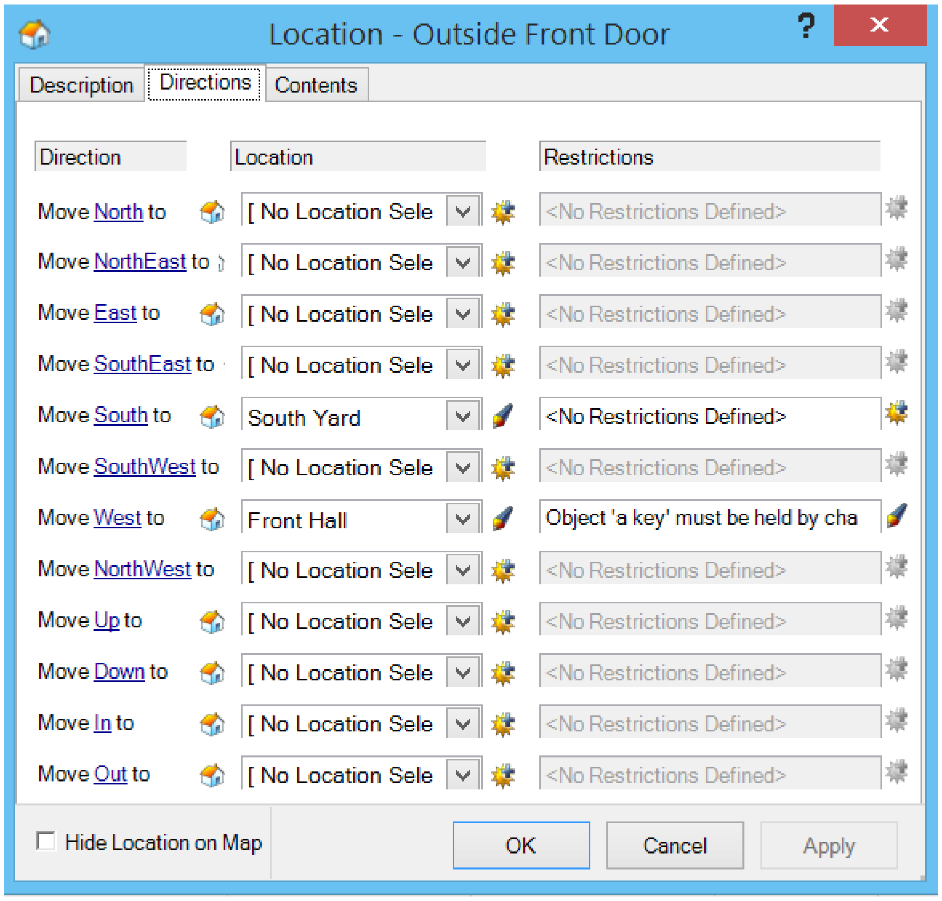
Long description:

You're outside your front door, which lies to the west.

☐ Hide Location on Map

5. You can use the same procedure to make more rooms. For “Lost Chicken,” we'll create two more locations, “Front Hall” and “South Yard.” We won't need a long description for “Front Hall.” Our long description of “South Yard” will be “This is the lovely south yard of your home.”
6. Double-click on “Outside Front Door,” from your list of locations. In the form that opens, click on the “Directions” tab. Using dropdown menus that appear, indicate that going west from “Outside Front Door” will take the player/character to “Front Hall” and that going south from “Outside Front Door” will take the

player/character to “South Yard.” When you've indicated these directions, Adrift may ask if you want these directions to work both ways. In other words, Adrift may ask whether going east from Front Hall should lead to “Outside Front Door” and whether going north from “South Yard” should bring the player/character to “Outside Front Door.” For our story, we can allow both of these two-way pairs.


Location - Outside Front Door

Direction	Location	Restrictions
Move North to	[No Location Sele	<No Restrictions Defined>
Move NorthEast to	[No Location Sele	<No Restrictions Defined>
Move East to	[No Location Sele	<No Restrictions Defined>
Move SouthEast to	[No Location Sele	<No Restrictions Defined>
Move South to	South Yard	<No Restrictions Defined>
Move SouthWest to	[No Location Sele	<No Restrictions Defined>
Move West to	Front Hall	Object 'a key' must be held by cha
Move NorthWest to	[No Location Sele	<No Restrictions Defined>
Move Up to	[No Location Sele	<No Restrictions Defined>
Move Down to	[No Location Sele	<No Restrictions Defined>
Move In to	[No Location Sele	<No Restrictions Defined>
Move Out to	[No Location Sele	<No Restrictions Defined>

☐ Hide Location on Map

Save and Test Your Story, So Far

1. Though Adrift is a solid and mature program, it is important so save your work

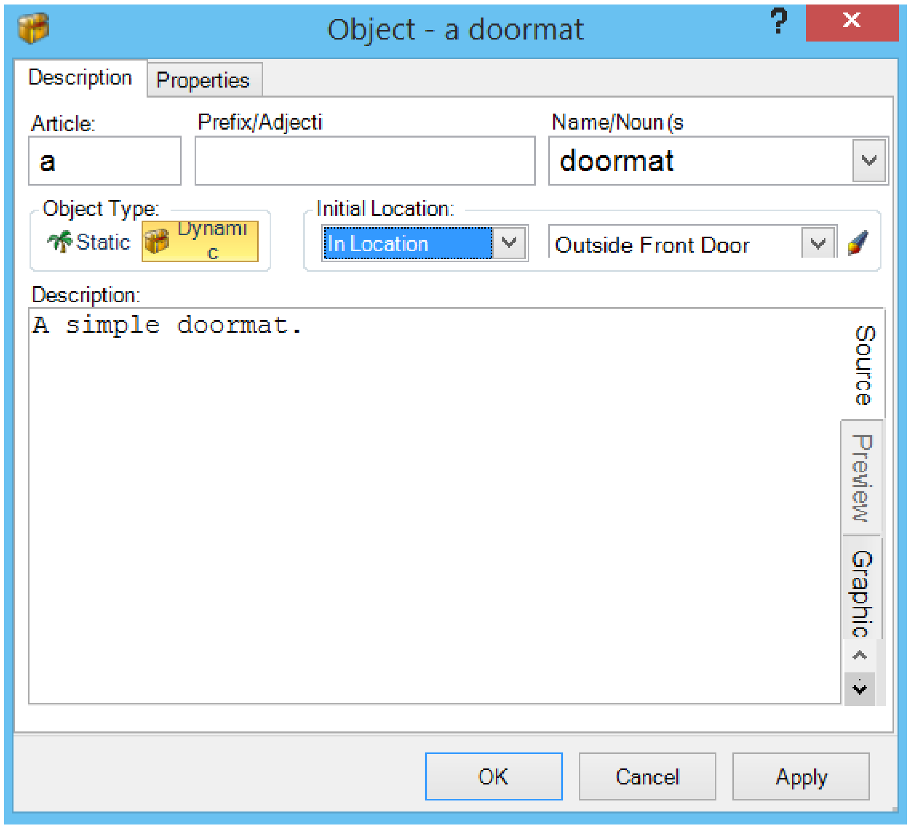
frequently. It's a good practice to save several versions of your work, in case
something goes wrong with your latest saved version.

2. To save your story, click on the circular icon in the top left of the screen, and
choose "Save As."
3. To test your story, click on the green triangle in the top center of the screen. This triangle is labeled, "Run Adventure." Adrift Runner will open and display
your story. Your player/character should be able to move between the rooms you've created.

Create your first "object."

1. From the "Add Items" list, choose Object.
2. In the form that opens, type in the name of the object. In our example, we'll create the doormat as our first item. Using the buttons and dropdown menus, indicate that this object is "dynamic" and that its initial location is "Outside Front Door." A dynamic object is an item that the player/character can pick up.
3. Then, in the "name" box, type "doormat" and then press the Enter key.
Next, type "mat." Now, we've signaled our intention to create an object named
"doormat," which the user can also refer to as "mat."
4. Next, type in the description of the object.
5. Click on the Properties tab, and indicate that the

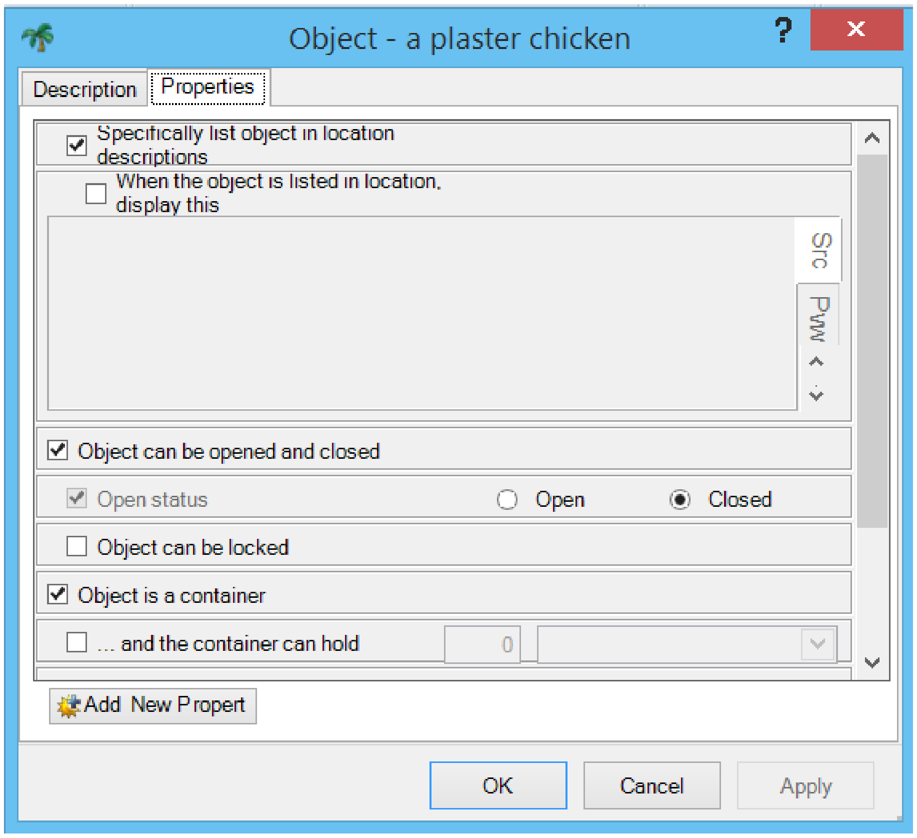
object should be mentioned in room descriptions.
6. Click on “OK.
7. You can use the same procedure to make more objects.



Create more objects.

1. Next, let's create the object called the “note.” This object will be similar to the doormat, except that its location will be “hidden.”
2. Now, let's create, or “implement,” the plaster chicken, which will contain the key. This object will be similar to the others, with several differences.
   1. It will be static, rather than dynamic.
   2. It is in the South Yard.

2. Its properties, accessed through the "properties" tab, will specify that
    1. it is a container,
    2. it can be opened and closed,
    3. it is closed, and
    4. it should be included in room descriptions.


Object - a plaster chicken

Description | Properties

[✓] Specifically list object in location descriptions
[ ] When the object is listed in location, display this
[✓] Object can be opened and closed
[✓] Open status ( ) Open (●) Closed
[ ] Object can be locked
[✓] Object is a container
[ ] ... and the container can hold 0

3. Finally, let's implement the key. It will be a dynamic object, and its initial location will be inside the plaster chicken.

Set the Opening and Closing Text

1. Click on the circular icon that appears in the top left corner of the Adrift Developer screen. From the menu that opens, choose “Introduction & End of Game.”
2. The page that opens will have two tabs, one for the opening of the game and one for its ending.
3. The “Introduction” tab allows for the creation of opening text, for specifying the story's first location, and for displaying, or not displaying, the description of the opening location.
4. Fill in the opening text for your story, something like, “You’ve made it home as usual, but you’ve forgotten your key.
5. Indicate that your opening location is Outside Front Door and that the story should display the description of the opening location.
6. This would be a good time to save and test your story, so far.

Create your first “task.”

1. A task is an action that the player/character must, or may, take to produce a certain result. Adrift has lots of built-in tasks, such as picking up an object, but you'll want to create your own, too. Tasks that you create generally override Adrift's built-in tasks.
2. Our first example of a task will implement what happens when the player/character first takes the doormat, thus revealing the note. Recall that, when

we created the note, we placed it in the “location” called “hidden.” In other words, it's not anywhere in the story's world until we bring it in.

3. In the “Add Items” section at the top of the screen, click on “Task.” A fairly complex window will open.

4. With the Description tab selected, type in a name for the task. We'll use “Take Doormat.”

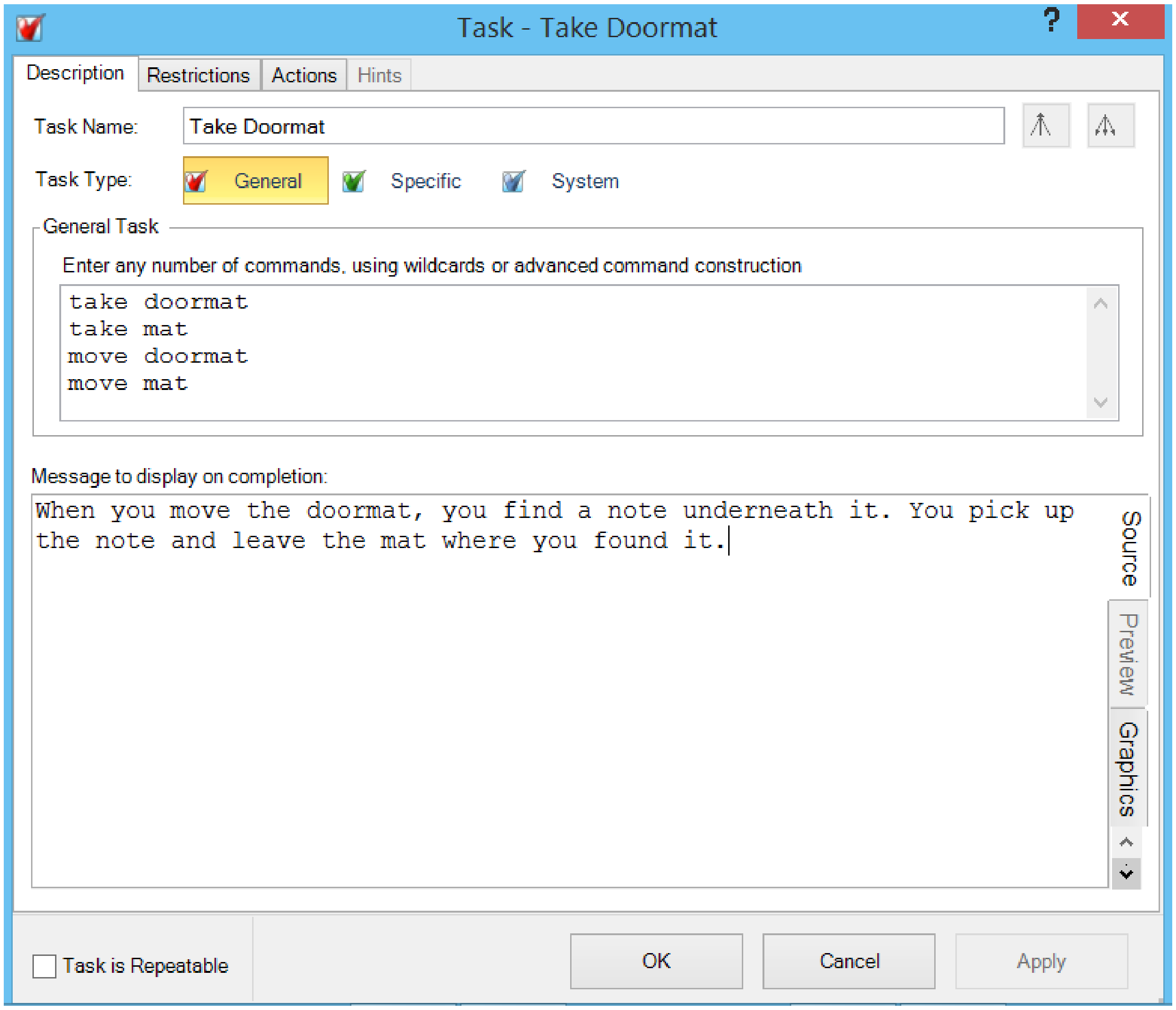
Our “Task Type” will be “General.” (We could also implement this task as “Specific,” but, if we did, we would not be able to introduce the action called “move,” which Adrift does not normally understand.)

5. In the box labeled “Enter any number of commands,” type in all of the commands that the reader will be able to use to activate the task. Put one command on each line. In our case, we'll list these commands:

```
take mat
take doormat
move mat
move doormat
```

6. In the box labeled “Message to display on completion,” type something like “When you move the doormat, you find a note underneath it. You pick up the note and leave the mat where you found it.”

7. Since we want the note to be revealed only once, uncheck the box that's labeled “Task is repeatable.” Then click on “Apply.”



Task Name: Take Doormat

Task Type: [✓ General] Specific System

General Task

Enter any number of commands, using wildcards or advanced command construction

```
take doormat
take mat
move doormat
move mat
```

Message to display on completion:

```
When you move the doormat, you find a note underneath it. You pick up
the note and leave the mat where you found it.
```

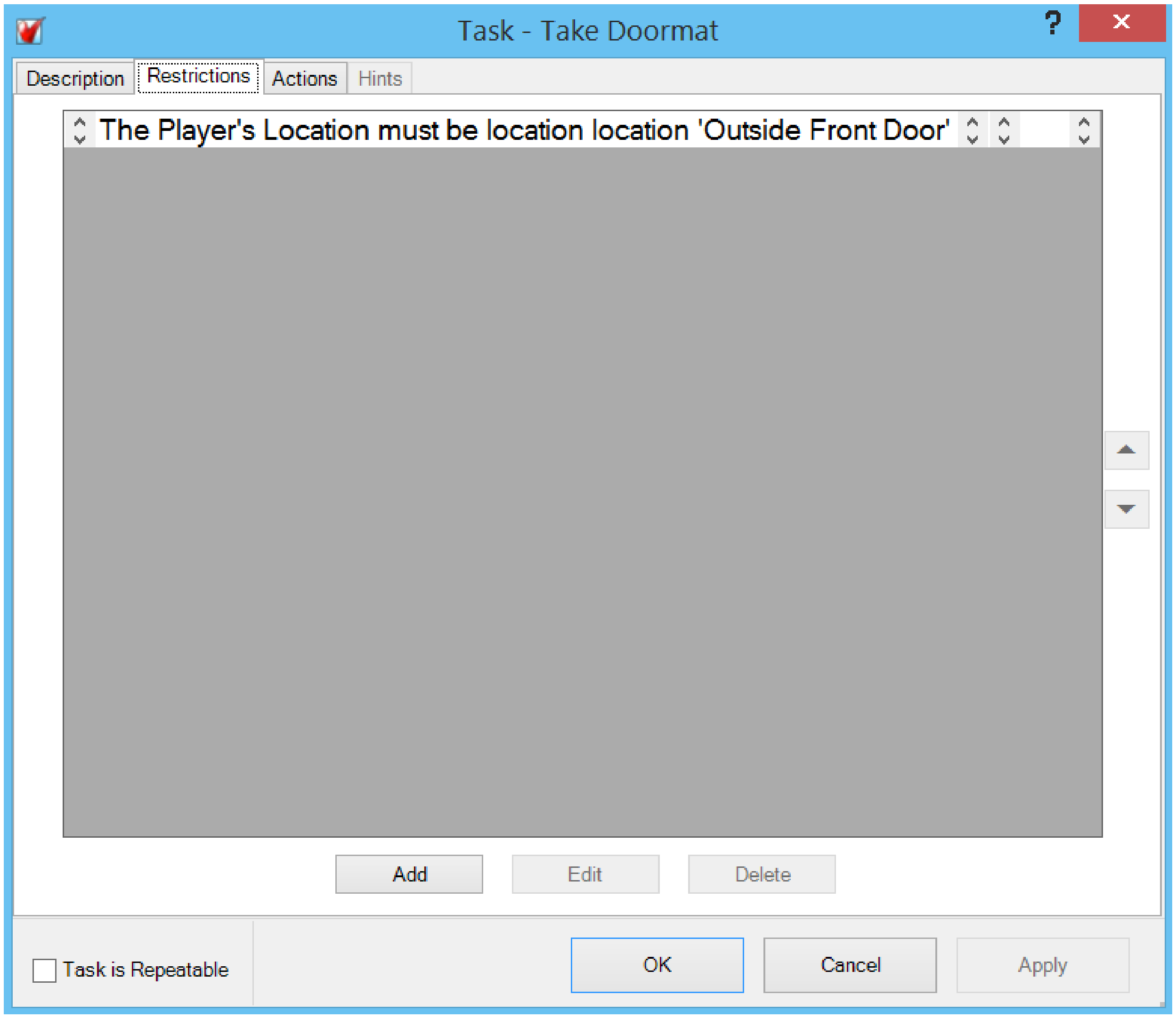
Task is Repeatable

8. Click on the “Restrictions” tab. Here we'll indicate what conditions must be met for the task to execute. In this case, our only restriction will be that our character must be in the location called “Outside Front Door.”

9. However, as of this writing, the screen that you're looking at right now is a little buggy. It displays a large box, currently blank. When you create restrictions, the box is supposed to display them. It

should also allow you to change the order in which the restrictions are applied and to edit each restriction by clicking on it and then on an “Edit” button. However, when you've created a restriction, even if you make no mistakes in doing so, the restriction usually does not appear in the large box. To find out if the restriction is really in effect, you usually have to close the whole task-creation window by clicking on the “OK” button at the bottom of that window. When you do, you'll see that the name of your task appears on the “Task” list on the main screen of Adrift Developer. If you double-click on the name of your new task on the task list, and then on the “Restrictions” tag, the large box will list all the restrictions that you've created, just as it was supposed to all along. A bit later in this tutorial, when we create actions for a task, you'll work with a very similar big box that lists the actions that your task triggers. This box exhibits the same bug.

9. With the “Restrictions” tab open, click on “Add.” Using the dropdown menus that appear, create the restriction that the player/character must be in the “Outside Front Door” location. Click on “OK” and then on “Apply.” You should now see your newly-minted restriction in the previously-blank list on the “Restrictions” tab. Your restriction should read, somewhat awkwardly, “The Player's Location must be location location 'Outside Front Door'

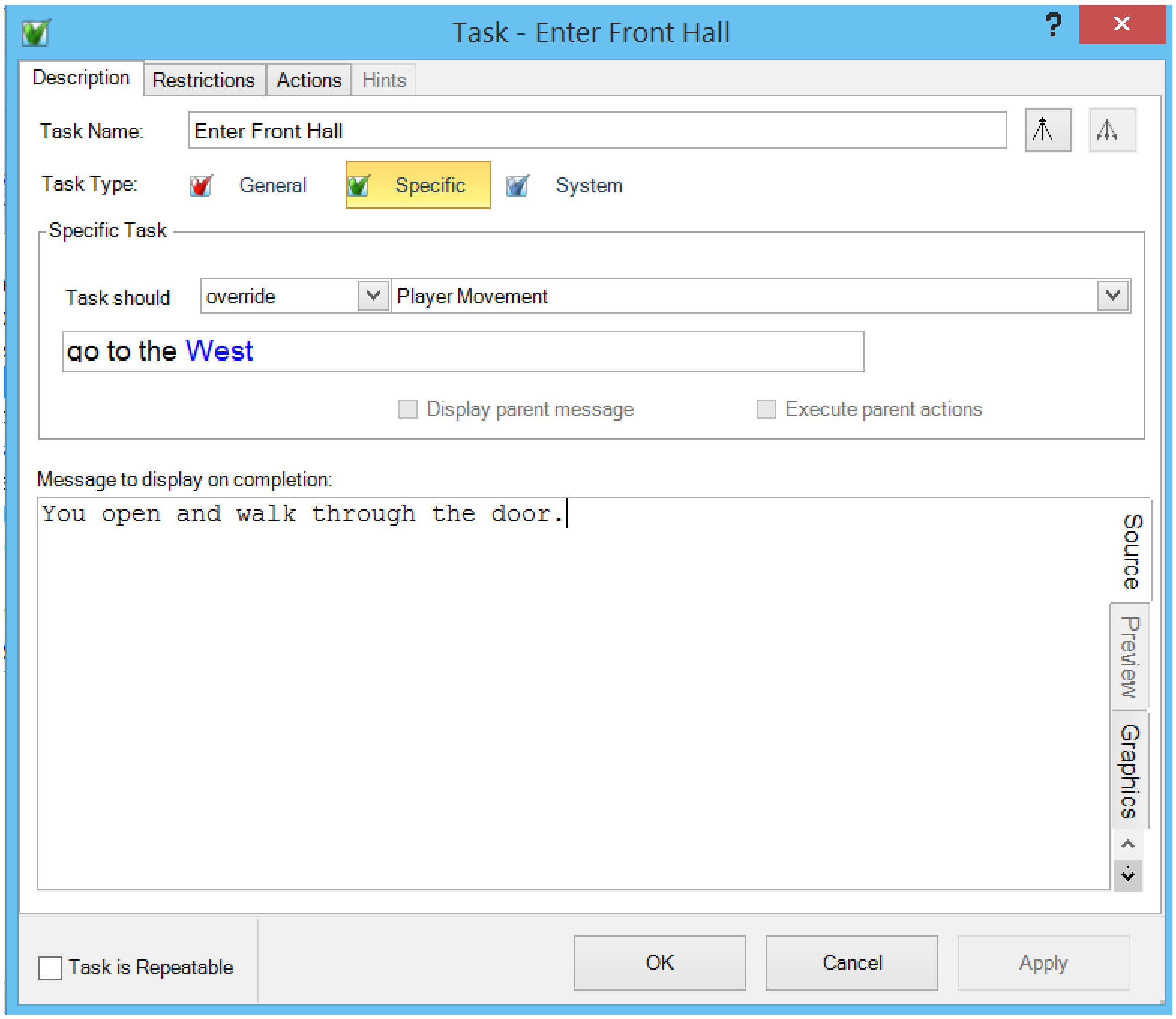


The Player's Location must be location location 'Outside Front Door'

10. Click on the “Actions” tab and then on “Add.”
11. Use dropdown menus to construct an action that moves the note from “Hidden” to “Held by the player.” Click on “OK” and then on “Apply.”
12. Your action should now appear on the previously blank list of actions, in the nonstandard but functional form, “Move object 'a note' to held by character 'Player'
12. Click on “OK” until the “Task – Take Doormat” window closes.

Create a task that ends the story

1. From the “Add Items” Section, click on “Task.” We'll call this task “Enter Front Hall.”
2. This task will not require that we introduce any new verbs, and so our task type will be “Specific.”
3. Using dropdown menus, indicate that the new task should “override” “player movement” “go west.”
4. As a “Message to display on completion, use something like “You open the door and walk through it.”



Task Name: Enter Front Hall

Task Type: General [✓ Specific] System

Specific Task

Task should override Player Movement

go to the West

☐ Display parent message ☐ Execute parent actions

Message to display on completion:

```
You open and walk through the door.
```

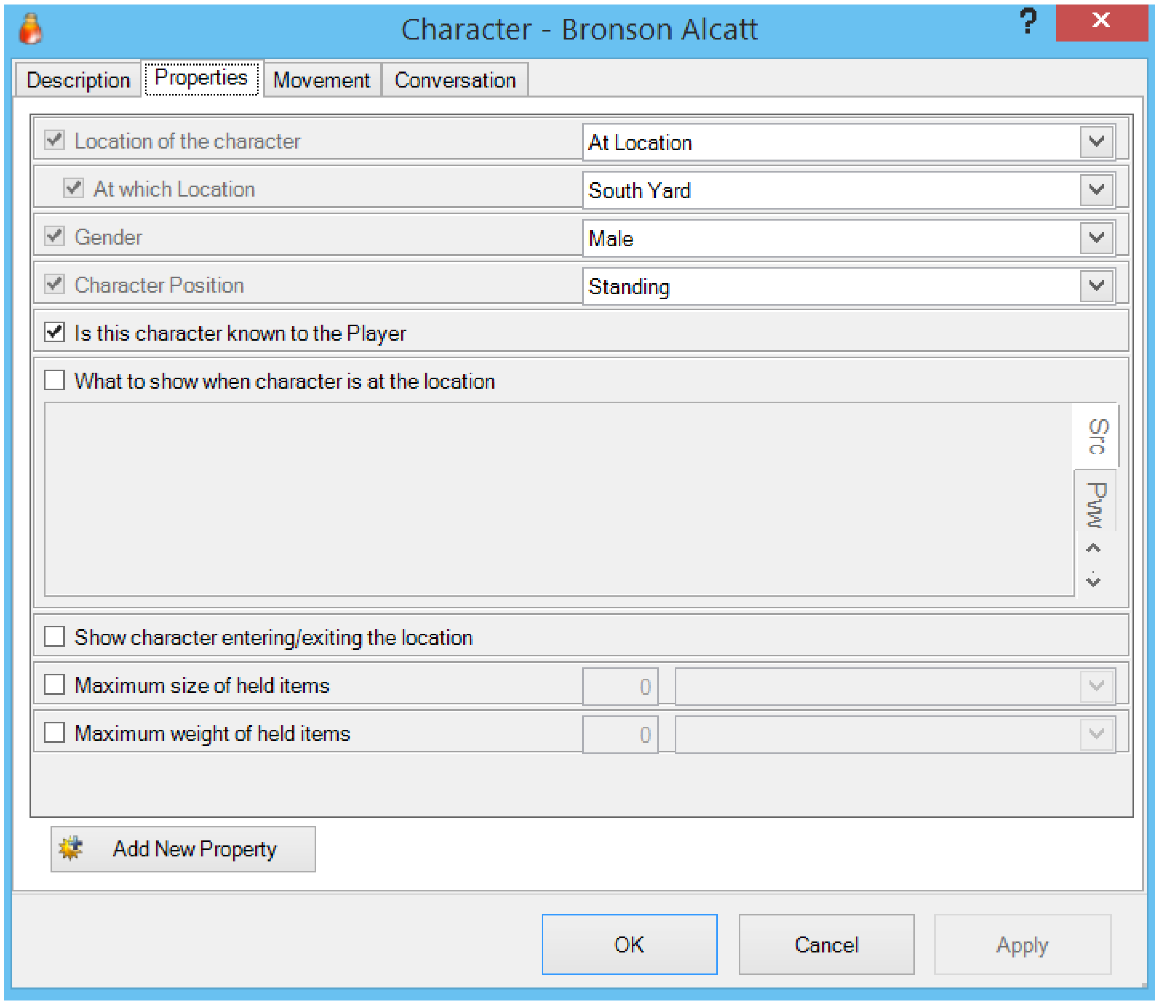
☐ Task is Repeatable

5. Click on the “Restrictions” tab. Click on “Add.”
6. Using the dropdown menus create the restriction, “The Player's Location must be location location 'Outside Front Door'
7. Click on “OK,” and “Apply.”
8. Again, click on “Add” in the “Restrictions” tab. Using the dropdown menus, create the restriction, “Object, 'a key' must be held by a character 'player'”
9. Click on “OK” and “Apply”
10. Click on the “Actions” tab.

11. Click on the “End Game” tab, and choose “In Victory.”
12. Click on “OK” and “Apply.” Click on “OK” again.
13. Save your work, and try out your story by pressing the green button.

Create your first character.

1. In the “Add Items” section, click on “Character.”
2. In the “Proper Name” box, type your character's proper name, “Bronson Alcatt,” in our example. In the “Descriptor/Noun” box, type “cat.” In the “Description” box, type what you want the player to see when she types, “examine cat.”
3. Click on the “Properties” tab.
4. Using the first dropdown menu, change the character's location from “Hidden” to “At Location.” Then choose “Front Yard” as Bronson's location.
5. Use similar dropdown menus to indicate that Bronson is “male” and “standing.”
6. Click in the checkbox after “Is the character known to the player?”
7. Click on OK.
8. Once again, save your work, perhaps using a new filename, and click on the green button to try out your story.


Character - Bronson Alcatt

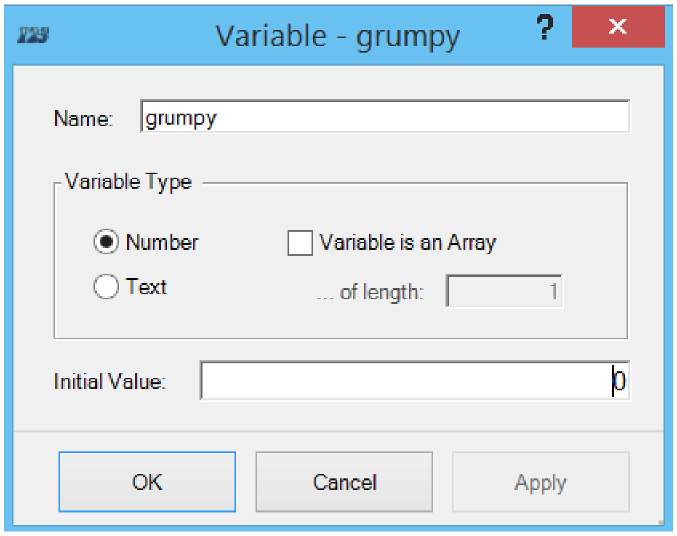
Description | Properties | Movement | Conversation

- [✓] Location of the character: At Location
- [✓] At which Location: South Yard
- [✓] Gender: Male
- [✓] Character Position: Standing
- [✓] Is this character known to the Player
- [ ] What to show when character is at the location
- [ ] Show character entering/exiting the location
- [ ] Maximum size of held items: 0
- [ ] Maximum weight of held items: 0

Create your first variable.

1. Variables enable you to achieve a huge variety of effects in interactive stories. We'll use one to block the player's opening the plaster chicken when Bronson Alcatt is feeling hungry.
2. In the list of folders that appears on the left side of the Adrift Developer screen, click on “Variables.” You should now see a list for variables beside the “Events” list on your screen.
3. Right-click on a blank area in the new “Variables” list, and choose “Add Variable.”
4. In the box that opens, indicate that your variable

will be called “grumpy, that it will be a number, and that its initial value will be 0 (zero).



Name: grumpy

● Number ○ Text

☐ Variable is an Array ... of length: 1

Initial Value: 0

Create a task that uses a variable.

1. In the “Add Items” group, click on “Task.” Name the new task, “Open Chicken.”
2. Indicate that your new task will be of the specific type. Using the dropdown menus, indicate that your new task should override “Open Objects.” Click on the word “object,” which appears in blue, and choose “a plaster chicken.”
3. As a “message upon completion, type in something like “You open the plaster chicken, revealing a key inside.”
4. Uncheck the box beside “Task is repeatable.”
5. Click on the “Restrictions” tab and add a restriction, using the dropdown menus. This first restriction should read, “Character 'Player' must be in same location as object 'a plaster chicken'”

6. Now we'll make a restriction that uses our new variable. With the “Restrictions” tab open, click on “Add.”
7. Click on the “variable” tab. Then, using the dropdown menus, create a restriction that reads, “Variable 'grumpy' must be equal to 1”
8. In the box labeled “This restriction must be passed. Otherwise the following will be displayed,” type “Bronson tries to scratch you when you reach toward the chicken. Maybe he's grumpy because he's hungry.”
9. Click on “OK” and “Apply.”
10. Click on the “Action” tab, click on the “Set Properties” tab. Use the dropdown menus to create an action that reads, “Set Property 'Open status' of object 'a plaster chicken' to Open”

Review the status of your variable

1. Since the use of variables can be a bit confusing, let's review the current state of
the variable that we've created.
2. The purpose of our variable is to make Bronson Alcott prevent the player/character from opening the plaster chicken, as long as Bronson is hungry. Once the player/character feeds Bronson the cat treat, the chicken should be available for opening.
3. Our variable is called “grumpy,” and it's numerical; that is, it represents a number.
At the start of the story, this number is zero.

4. We have created a task called “Open Chicken,” which prevents the opening of the plaster chicken unless the value of our “grumpy” variable is one.
5. So far, we have not implemented any way in which the value of the “grumpy” variable can change to one. Therefore, as our story now works, the player/character will never be able to open the plaster chicken.
6. What we need now is a new task that will include a change in the value of “grumpy.” This new task should involve feeding Bronson.

Create a task that changes the value of a variable.
1. In the “Add Items” group, click in “Task.”
2. Give your new task a name like “Give Treat,” and make it a specific task.
3. Un-check the box labeled “Task is repeatable.”
4. Using the dropdown menus, indicate that the task should override input of the pattern “give object to character.” Click on the blue-colored word “object” and choose “a pet treat.” Then click on the blue word “Character” and choose “Bronson Alcatt.”
5. In the box labeled “Message to display on completion,” type something like, “Bronson accepts the treat and gobbles it up. He looks less grumpy now.”
6. Click on the “Restrictions” tab and create the following restrictions:

1. The 'Player's Location' must be location location 'South Yard'

In the box labeled, “The restriction above must be passed. Otherwise the following will be displayed,” type, “Bronson doesn't seem to be here.”

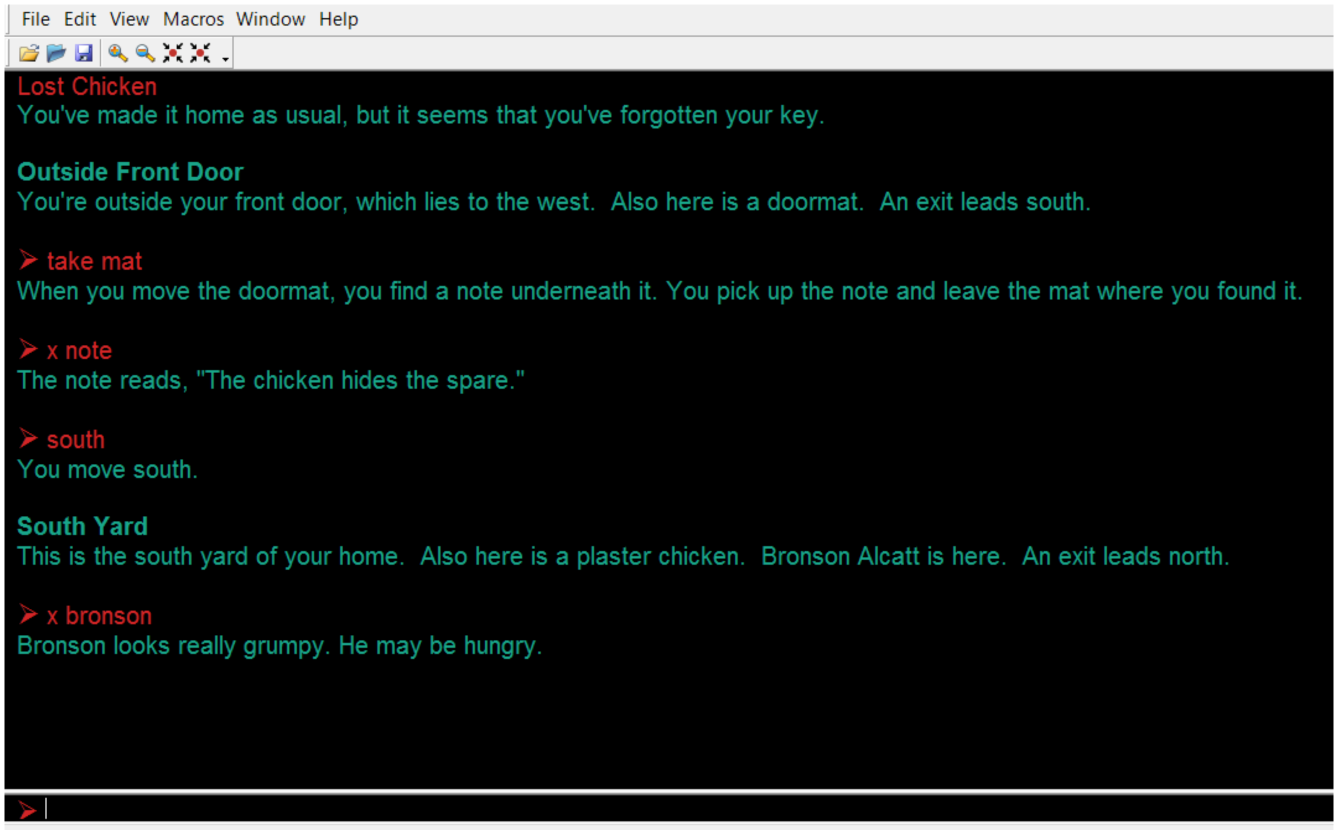
2. Object 'a pet treat' must be held by character 'Player'

In the box labeled, “The restriction above must be passed. Otherwise the following will be displayed,” type, “You don't have the pet treat.”

7. Click on the “Actions” tab and then on “Add.”
8. Click on the “Variables” tab. Using the dropdown menus, choose “Set” and “grumpy.”
9. If you're looking carefully, you'll see a small symbol that looks like this: *123.* Click on that symbol. In the box that opens, type the numeral 1.
10. Click on “OK.” As you've probably already realized, you have created an action that changes the value of the variable. On the list of actions, this action should come first.
11. Create another action. After clicking on “Add” use the “Move Objects” tab.

The finished version of this action should read, “Move object 'a pet treat' to held by character 'Bronson Alcatt'

12. Save your changes and try out your story.



```
Lost Chicken
You've made it home as usual, but it seems that you've forgotten your key.

Outside Front Door
You're outside your front door, which lies to the west.  Also here is a doormat.  An exit leads south.

> take mat
When you move the doormat, you find a note underneath it. You pick up the note and leave the mat where you found it.

> x note
The note reads, "The chicken hides the spare."

> south
You move south.

South Yard
This is the south yard of your home.  Also here is a plaster chicken.  Bronson Alcatt is here.  An exit leads north.

> x bronson
Bronson looks really grumpy. He may be hungry.

> |
```

A challenge to try on your own

1. Implement the door, as mentioned in the transcript.
2. The Adrift Manual Wiki (http://wiki.adrift.co) explains how to create a fully-featured door, but you could get away with something much simpler in this story.

Another challenge

1. Create a new specific task, overriding Adrift's usual response to “Examine Bronson Alcatt.”
2. This task should produce two different descriptions of Bronson, one which appears before he gets his treat and one which appears after.

Chapter 7 -- Creating Interactive Fiction with Quest on the Web

Quest offers a way to create interactive stories with very little programming. Like Adrift, Quest works mainly though a series of windows and drop-down menus. Originally, Quest was, like Adrift, a Windows-only application, and, as of February 2015, its Windows version remains its most complete variation. However, Quest comes in a Web version, too, and we'll use the Web variation for this tutorial, since it's more widely available. However, if you have a Windows computer, you should use the Windows version of Quest. It's more mature, less buggy, and more complete, but it works very much like what you'll see in this tutorial. In our Quest tutorial, we'll implement the super-simple story “Lost Chicken,” which we also used in our chapter on writing IF with Adrift. Here's a transcript of the story:

```
“Lost Chicken” Script

You've made it to your home, as usual,
but it seems that you've forgotten your
key.

Lost Chicken
An Interactive Fiction by Brendan
Desilets

Outside Front Door
You're outside your front door.  The door
is to the west, and your front yard is to
the south.

You can see a doormat and an oak door
here.

>i
You are carrying:
  a pet treat

>take doormat
You find a note under the doormat and
pick it up.

>read note
The note reads, "The chicken hides the
spare."

>s
```

```
South Yard
This is the lovely south yard of your
home.

You can see Bronson Alcatt and a plaster
chicken (closed) here.

>look at Bronson
Your very old cat, always a formidable
presence.  He looks particularly grumpy
right now, as he rubs against his
favorite lawn ornament, a plaster
chicken.

>open chicken
Your irascible cat scratches you,
preventing you from getting at the
chicken.

>give treat to Bronson
Bronson gobbles up the treat. He looks
less grumpy now.

>open chicken
You open the plaster chicken, revealing a
key.

>take key
Taken.
```

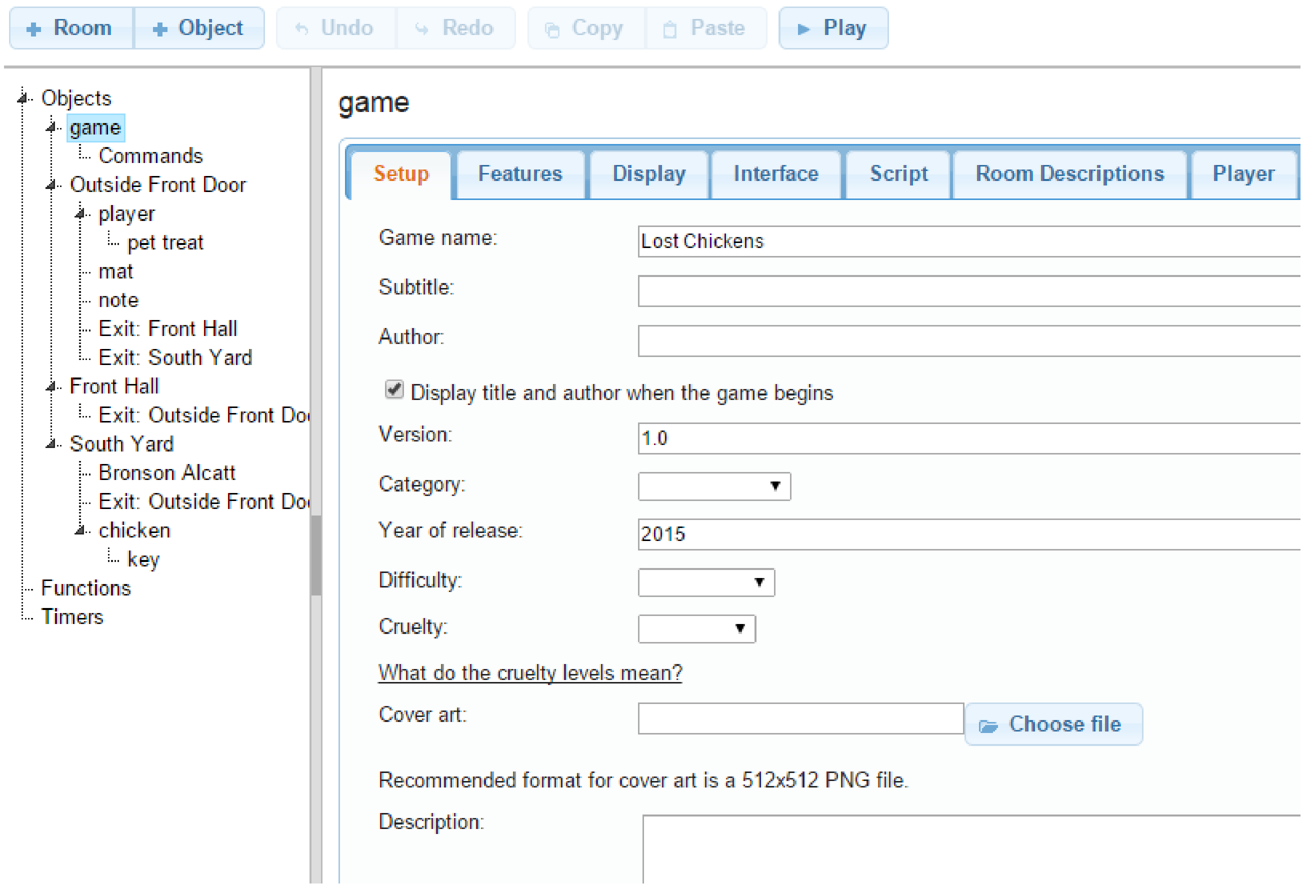
```
>n

Outside Front Door
You're outside your front door.  The door
is to the west, and your front yard is to
the south.

You can see a doormat and an oak door
here.

>w
You unlock the door and enter your house.

Front Hall
You have made it inside, where Bronson
happily joins you.


    *** The End ***
```

A Tutorial for Quest on the Web

Quest may require a slightly steeper learning curve than Adrift, but Quest offers significant support for educators at its website, http://textadventures.co.uk/quest. In order to create a story with the Web version of Quest, you first have to set up an account at the Quest website.

Set Up Your Story

1. Log in to your Quest account and click on the "Create" button at the top of your screen.
2. Click on "Create a New Game."
3. In the "Game name:" box, type in the title of your story. Your "Game Type" is "Text Adventure."
4. Click on "Create."
5. Click on "Start editing!"
6. Wait around for a minute or two. Like many Web applications, Quest can be a bit slow to execute.
7. Quest will open with a screen that looks like this:



Game name: Lost Chickens

Subtitle:

Author:

✓ Display title and author when the game begins

Version: 1.0

Category:

Year of release: 2015

Difficulty:

Cruelty:

Cover art:

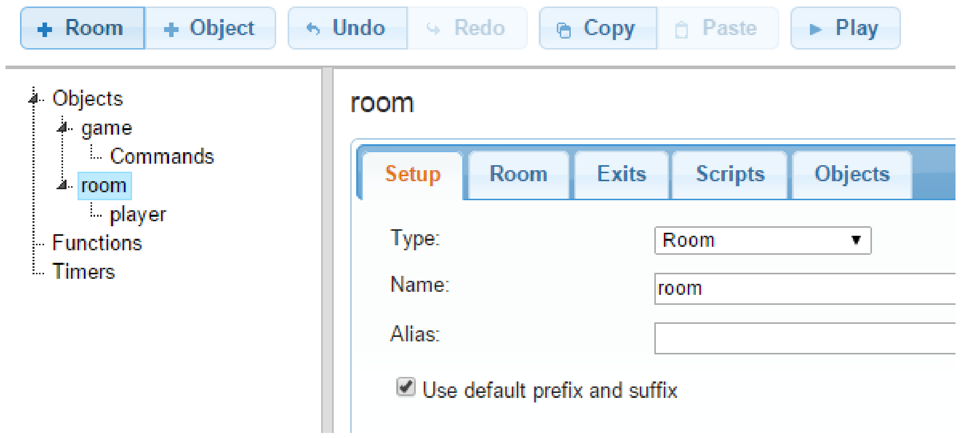
Recommended format for cover art is a 512x512 PNG file.

Description:

8. Notice that the word “game” is highlighted on the diagram that appears on the left side of the screen. That diagram is important, and we'll have to monitor it carefully as we go along.
9. Fill in the information that's asked for on the “Setup” tab. Feel free to examine and even experiment with the other tabs if you wish, with the exception of the “Player” tab. It's best to leave that one as it is, in order to head off possible buggy behavior.

Enhance Your First Room

1. You may have noticed, in the chart on the left of the screen, that Quest has already created a room for you and placed the player/character in that room.
2. Click on the word “room” in the chart at the left. Do not click on the button labeled “+ Room” at the top of the screen. We'll use that one later.
3. A screen that looks like this will open:



room

Type: Room

Name: room

Alias:

☑ Use default prefix and suffix

4. In the screen that opens, fill in the name of the opening room, in our case “Outside Front Door.”
5. As an “alias,” use something like “outside your front door, which lies to the west.”
6. Click on the “Room” tab and fill in a description, such as “This is a grassy area outside your front door, which lies to the west.”
7. We'll use the other tabs later. For now, click on the “Save” icon at the upper right corner of the screen, unless Quest has already greyed it out, indicating that the story is already saved.
8. In addition to saving your story, you'll want to download it occasionally as your project gets larger. Downloaded versions of you story are excellent insurance, in case something goes wrong with your on-line copy.
9. The downloaded stories will require the full Quest program, a Windows-only tool, for editing them. However, in the case of a complex story, most users will be better off with the Windows program anyway, as it's faster, more complete, and more reliable.

Add Two More Rooms

1. Click on the word “game” in the chart at the left

of the screen. Make sure that the word “game is highlighted, as we want our additional rooms to be inside the game, but not inside anything else.

2. Click on the “+ Room” button at the top of the screen.
3. You'll see a familiar room-edition screen. Fill in the “Name” as “South Yard” and the “Alias” as something like “a lovely yard, south of your home.”
4. Click on the “Room” tab and fill in the description text, “This is the lovely South Yard of your home.”
5. Click on the “Exits” tab. Since we want the player to go north from here to Outside Front Door, click on the circle beside the word “North.” When you've done so, you'll see a dropdown menu labeled “Create an exit to:” Use this dropdown menu to select “Outside Front Door.”
6. We want this exit to work in both directions, so check the box that is labeled “Also create exit in other direction.”
7. Once again, click on the word “game” in the chart that appears at the left of the screen. Click on the “+ Room” button at the top of the screen.
8. This time, we'll name the room “Front Hall,” and we'll give it the alias “cozy front hall of your home.”
9. We'll not use the “default prefix” for this room.

Instead we'll use the prefix “the.”

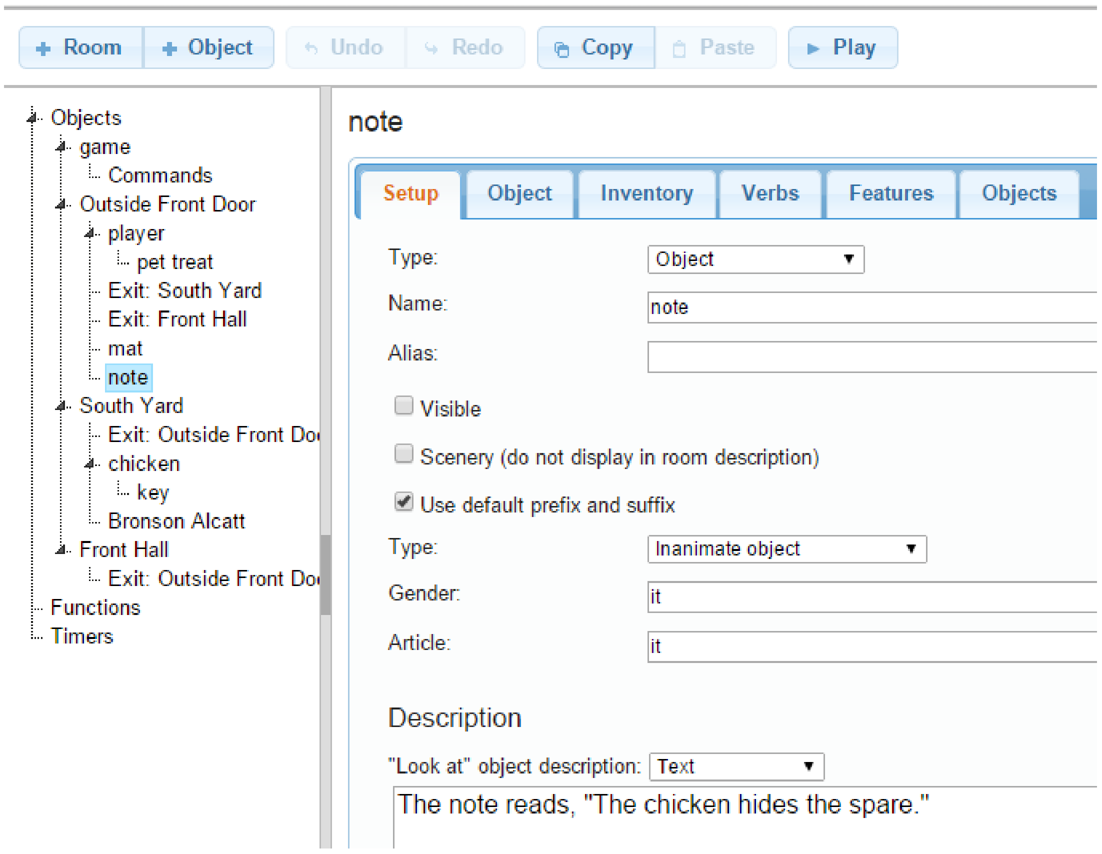
10. We won't use the “Room” tab here, but we will click on the “Exits” tab.
11. With the “Exits” tab open, create a two-way exit that leads east to “Outside Front Door.” Later, we'll create a restriction on when the player can go west from Outside Front Door to Front Hall.

Try Out Your Story, So Far

1. Click on the “Save” button in the upper right part of the screen, unless the button is already greyed out.
2. Click on the “Play” button at the top of the screen. After a few seconds, you should see you story running, with the player in the room called “Outside Front Door.”
3. You should be able to move around through the three rooms you've created.
4. Occasionally, the Quest interpreter seems to generate odd error messages, in response to common IF commands. Often, simply trying your command a second time will solve the problem.

Create Two Objects, the Doormat and the Note

1. New writers of interactive fiction often ask about how a writer can cause a concealed object to appear, the way the note appears in “Lost Chicken,” when the player/character takes the doormat. Here's one of the several possible ways to get this effect.
2. First, let's create the note.
3. Click on “Outside Front Door” on the chart that appears at the left of your screen. You will see the familiar screen which you used to provide information about the “Outside Front Door” room.
4. On this screen, click on the “Objects” tab. A screen will open, showing that one object, the player, is already in “Outside Front Door.”
5. In the “Objects” tab, click on “Add.” A screen will open, allowing you to provide information about your new object.
6. Give your new object the name “note.” Make its description, “The note reads, 'The chicken hides the spare.'''
7. Click on the “Inventory” tab, and indicate that the new object can be taken.
8. Make sure that the box beside “Visible” is not checked. The note, in its initial form, will not be visible to the player, but we'll make it visible soon.



note

Type: Object

Name: note

Alias:

☐ Visible

☐ Scenery (do not display in room description)

☑ Use default prefix and suffix

Type: Inanimate object

Gender: it

Article: it

Description

"Look at" object description: Text

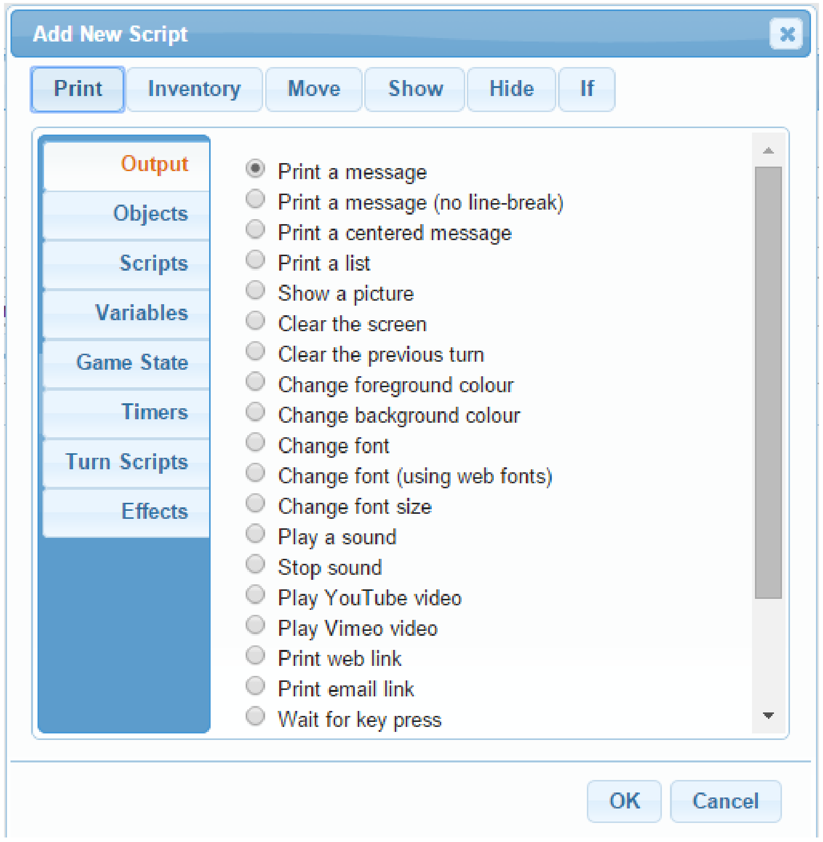
The note reads, "The chicken hides the spare."

9. Once again, click on “Outside Front Door” on the chart at the left of the screen. Then click on the “Objects” tab.
10. Click on Add.
11. This time, call your object “mat,” and make its alias “doormat.” Its description should be something like “an ordinary doormat.”
12. Click on the “Object” tab and then click on the “+ Add” button that appears beside “Other Names.” Add the new name “mat.”
13. Click on the “Inventory” tab and indicate that the mat can be taken.
14. Click on the “Save” button if it's not already greyed out.

Create Your First Script

1. Some Background Concepts
    1. Scripts are central to Quest, in pretty much the same way that tasks are central to Adrift. In Inform 7, which we’ll consider later in this book, rules serve more or less the same purpose as scripts.
    2. Scripts can be quite complex, but they are essential.
    3. Scripts allow an author to change the default behavior of the game world. For example, in our “Lost Chicken” story, we can change what happens when the player/character takes the mat, causing the note to appear.
    4. As we build a script, it we will often add other scripts to it. Thus, it is extremely common for an author to start creating a script and then to add other scripts to it.
2. Our First Script – the Goal
    1. In order to make the note appear when we want it to, we'll build a script. These are the characteristics of that script.
        1. It takes place only after the player/character takes the mat when the note has not yet been revealed. We don't want the note to mysteriously reappear every time the player/character drops mat and picks it up again.

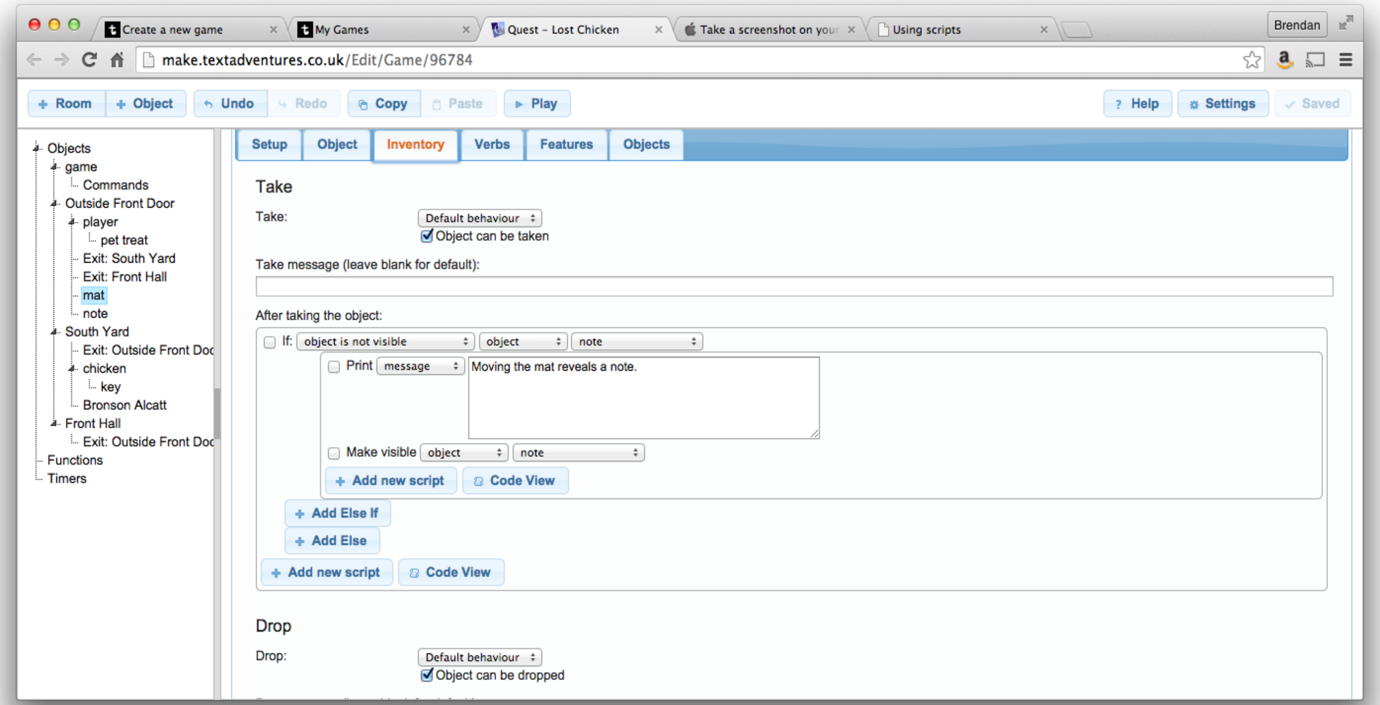
        2. It causes the note to appear.
        3. It tells the reader what the previous elements of the script caused to happen.
    2. We'll build our script, using the “Inventory” tab for the object called “mat.”
3. Our First Script – the Building Process
    1. On the chart that appears at the left of your screen, click on “mat.”
    2. Click on the “Inventory” tab.
    3. Under “After taking the object,” click on the button labeled “+ Add new script.” A popup window that looks very much like this will open:



4. Click on the word “If,” which is in the upper right part of the popup window.
5. The word “IF” will appear, followed by a series of dropdown menus.
6. Using the dropdown menus, create an “if” clause that reads, “IF: object is not visible object note.”
7. By now, you've probably noticed that the screen you're looking at has become pretty complicated. For one thing, it contains at least three identically-labeled buttons that read “+ Add new script.”
8. One of these “+ Add new script” buttons is directly under “If: object is not visible object note.” And this particular button is indented a bit.
9. The indenting is intended to show that whatever command (AKA “script”) you add using the indented button will be carried out only if the note is not visible.
10. This sort of indenting is of paramount importance in creating good scripts. Such scripts often require nesting groups of commands within other groups of commands, and the indenting is very helpful in keeping all the nests sorted out.
11. Click on the indented button that's labeled “+ Add new script.” A popup window will open.”
12. The defaults that are already selected in the popup window will probably indicate that you want to print a message. As a matter of fact, that's a good idea here, so just click on “OK.” If the defaults do not have “Print” and “Print a message”

already selected, select them before you click “OK.”

13. Indicate that the message you want to print is something like “Moving the doormat reveals a note.”
14. Notice that there's now a “+ Add new script” button directly under the word “Print.” Click on that “+ Add new script” button.
15. This time, on the popup window, select the “Objects” button and choose “Make object visible.” Click on “OK.”
16. Now using the dropdown menus, assemble this command, “Make visible object note.”
17. Your screen should now look like this:



Take

Take: Default behaviour

☑ Object can be taken

Take message (leave blank for default):

After taking the object:

If: object is not visible | object | note

Print message | Moving the mat reveals a note.

Make visible object | note

Drop

Drop: Default behaviour

☑ Object can be dropped

18. Click on “Save” if it's not already greyed out. Then try out your story so far, using the “Play” button.

Create Your First Container

1. IF authoring systems usually have some built-in object types. One of these is the container, a type of object that can hold other objects. In “Lost Chicken,” the plaster chicken is a container that has a key in it.
2. To create the plaster chicken, begin by clicking on “South Yard” in the chart on the left side of the screen. Then click on the “Objects” tab.
3. Click on “+ Add.”
4. A familiar screen for creating an object will

appear, with the “Setup” tab selected. Fill in the name “chicken” and the alias “plaster chicken.” For a description, type something like “An old plaster chicken, used as a container.”

5. With the “Object” tab depressed, add “chicken” as an “other name.”
6. Click on the “Inventory” tab. Make sure that “Object can be taken” is not checked.
7. Click on the “Features” tab and select “Container: object is a container or surface, or can be opened and closed.”
8. A new tab, labeled “Container,” will appear. Click on this tab, and chose, as the “Container type,” “Openable/Closable.” Then, check “Can be opened,” and “Can be closed.”
9. Click on the “Objects” tab and then on “+ Add.” Add the object “key,” and fill in its details in our now-familiar fashion.
10. Try out your story. It should be possible for the player/character to go south to the South Garden, open the plaster chicken, and take the key.

Create a Script That Ends Your Story

1. Our story ends when the player/character gains entry to the Front Hall. It should be impossible for the player/character to get to the Front Hall unless he or she is carrying the key.

2. We'll use a script to implement this behavior.
3. In the chart at the left of the screen, locate “Outside Front Door,” but don't click on it.
4. Under “Outside Front Door,” locate “Exit: Front Hall,” and click on it.
5. Put a checkmark beside “Run a script (instead of moving the player automatically).”
6. As we did on our previous script, start by choosing “If'” from the upper right part of the popup menu, and then build a statement, using the dropdown menus that will appear. The statement should be, “If player is not carrying object object key.”
7. Once again, you'll see a fairly complex screen with a number of buttons labeled “+ Add new script.” Pick out the “+ Add new script” button that is indented under your “If” clause, and click on that button.
8. Again, a popup window will open, and, as it happens, the default behavior, printing a message, is what we want. Click on “OK,” and fill in the message, “You can't get through the door without the key.”
9. Now, we'll add some unfamiliar features to our script. First, click on the “+ Add Else” button, which appears below your “If” clause.
10. Identify the “+ Add new script” button that is immediately under the word “Else” and slightly indented. Click on that button.

11. A familiar popup window will open. Click on “Move” at the top of the window.
12. Using dropdown menus, build the command “Move object object player to object “Front Hall.”
13. Identify the “+Add new script” button that appears immediately under the command that you just created. Click on that button.
14. Click on “OK.” Then fill in the “Print” box that appears with a message like, “Congratulations! You've finished the story.
15. Press the “+Add new script” button that appears below your “Print” statement. A familiar popup window will open.
16. Click on the “Game State” button that appears along the left side of the popup window. Then choose “Finish the game” and click on OK.
17. Save your story and try it out. You should be able to reach the end of the story.

Create Your First Non-Player Character

1. The only non-player character in our story is cat named Bronson Alcatt. He starts out in the South Yard, so let's start out by clicking on “South Yard” on the chart at the left of the Quest screen.
2. Click on the “Objects” tab and then on the button labeled “+ Add.” Let's call our new object

“Bronson Alcatt” and use “Bronson Alcatt” as his alias. Bronson's “Type” will be “Male character (named).”

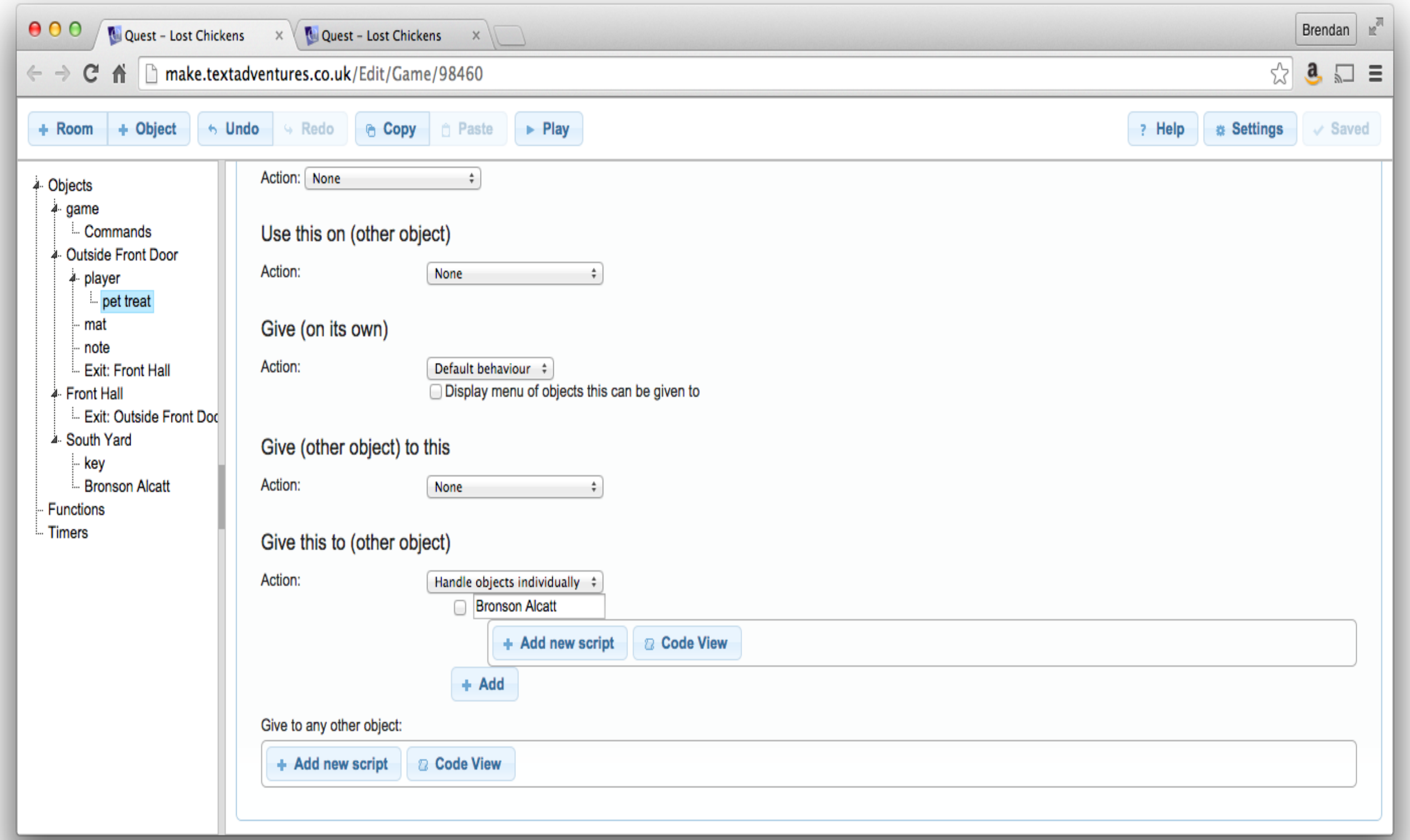
3. As a “Look at” description, let's choose “Text” from the dropdown menu and type something like, “Your ancient and formidable feline.”
4. Using the “Object” tab, let's add some names for the cat. These might be “Bronson,” “Alcatt,” and “cat.”
5. Save you progress, unless Quest has already saved it for you. Try out your story once more.

Create the Pet Treat

1. Quest considers the pet treat to be a bit different from the other objects we have created because we want to be able to give it to Bronson.
2. Click on the word “player” on the chart at the left of the screen. Then click on the “Objects” tab.
3. Click on “+ Add,” and give your new object the name “pet treat.” Using the dropdown menu, indicate that the “Parent” of the pet treat is the player, not the room in which the player starts out.
4. Give the “pet treat” the alias “treat.”
5. Click on the “Object” tab and add the name “pet treat.”
6. Click on the “Features” tab and check the box

beside “Use/Give: object can be used on its own or with other objects/characters or given to other objects characters.” You'll notice that a new tab, labeled “Use/Give” will now appear.

7. Focus on the section of the screen labeled “Give this to (other object).” Beside the word “Action,” you'll see a dropdown menu. From this menu, choose “Handle object individually.”
8. Click on “+ Add” and choose “Bronson Alcatt.” Note that, slightly indented under “Bronson Alcatt,” there is a familiar “+ Add new script” button.
9. We are now going to build a script will accomplish two goals.
    1. It will tell the reader what happens when the player/character gives the treat to Bronson.
    2. It will actually transfer the treat from the player/character to Bronson.



Action: None

Use this on (other object)

Action: None

Give (on its own)

Action: Default behaviour

Display menu of objects this can be given to

Give (other object) to this

Action: None

Give this to (other object)

Action: Handle objects individually

Bronson Alcatt

Give to any other object:

10. Click on the "+ Add New Script" button that appears, slightly indented, under "Bronson Alcatt."
11. In the popup menu that opens, you'll notice that printing a message is already selected. As a result, simply click on "OK" at the bottom of the popup.
12. In the "Print message" box, type something like "Bronson accepts the treat and devours it greedily. He looks less grumpy now."

13. Locate the “+ Add new script” button that appears directly below the word “Print.” Click on that button.
14. In the popup window that opens, click on “Move,” which appears on the top row of options. The popup will disappear, and you'll find that you're back on the previous page. However, you'll note that a new line has been added under your “Print” box. This line offers a series of dropdown menus.
15. Using this new line and its dropdown menus, create the command, “Move object pet treat to object Bronson Alcatt.”
16. Save your work and try out your story.

Implement Bronson's Gumpy Behavior

1. We've now made it possible for the player/character to give the pet treat to Bronson, but we have not yet provided any motivation for doing so. We now need to create a way to implement Bronson's mood and to change it once he gets the treat.
2. In the various IF authoring systems, the most obvious way to implement this sort of change is to use a variable. If we were working with the Windows version of Quest, we would use a similar approach, but the Web version of Quest does not give us this option.

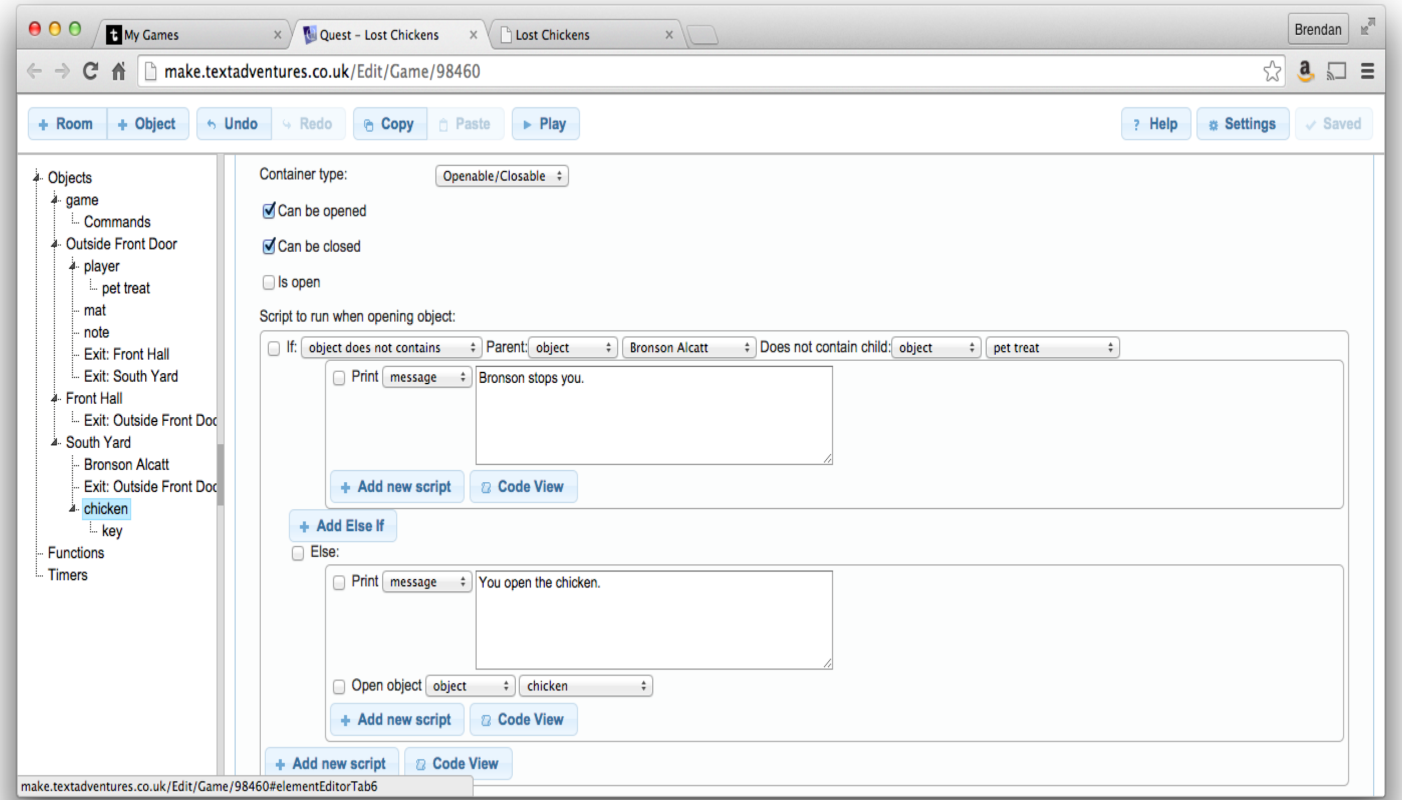
3. Instead, we'll implement Bronson's mood by keeping track of whether or not he has the pet treat.
4. Click on the word “chicken” in the chart that appears on the left of the screen. Click on the “Container” tab.
5. Under “Script to run when opening object,” click on the button labeled “+ Add new script.”
6. Using the dropdown menus, construct the following, not too grammatical, clause:

   If: object does not contains Parent: object Bronson Alcatt Does not contain: Child object pet treat.

7. The meaning of your odd-sounding “if” clause is, “If Bronson Alcott is not carrying the pet treat.”
   1. Note that, when the player/character gives Bronson the pet treat, the reader is told that Bronson ate it.
   2. However, we never actually implemented anything that would make the pet treat go away.
   3. We did implement the transfer of the pet treat to Bronson.
   4. By default, when the player/character looks at another character, the player/character does not see what the other character is

carrying. For this reason, the reader has no way of knowing that Bronson is actually carrying the treat.

8. Locate the button labeled “+ Add new script,” the one that is slightly indented under the “if” clause you just created.
9. On the popup window that opens, click on “Print” in the top left part of the popup. Then click on “OK.”
10. Fill in the resulting print box with something like, “When you reach toward the chicken, Bronson tries to scratch you, preventing you from opening it. Perhaps he's grumpy because he's hungry.”
11. Click on “Add Else.”
12. Identify the “+ Add new script” button that appears below “Else:” and slightly indented. Click on that button.
13. In the popup window that opens, click on “Print.” Fill in the print box with something like, “You open the plaster chicken.”
14. Locate the “+ Add new script” button that appears directly below the word “Print.” Click on that button.
15. In the popup window that opens, click on the word “Objects,” which appears on the left side of the popup. Then select “Open object” and click on “OK.”



Container type: Openable/Closable

✓ Can be opened

✓ Can be closed

Is open

Script to run when opening object:

If: object does not contains | Parent: object | Bronson Alcatt | Does not contain child: object | pet treat

Print message | Bronson stops you.

Else:

Print message | You open the chicken.

Open object | object | chicken

16. Then using the dropdown menus that appear, construct the command, “Open object object chicken.”
17. Save your story and try it out.

Some Further Challenges

1. Try to make Bronson's description more flexible, using a script rather than plain text.
2. Implement the front door. There's a way to do this sort of implementation in the Quest documentation at http://docs.textadventures.co.uk/quest/tutorial/using_lockable_exits.html. However, you probably don't need a fully-implemented door for our story.

Chapter 8: Inform 7 and the Writing Process

Can students improve their writing and thinking by programming computers? In his seminal book *Mindstorms: Children, Computers, and Powerful Ideas* (1980), Seymour Papert argues that, within a computer-rich environment, or “microworld,” students can construct knowledge in uniquely exciting ways, largely through the use of the user-friendly programming language called Logo. Taking their cue from Papert, teachers of writing have developed ways to apply Logo to the writing process, even in conventional instructional settings that do not offer a great deal of computer access. In one instance, teachers have used the recursive power of Logo to help students develop their ideas through extended definition.

Still, Logo has its obvious drawbacks as a tool for teaching writing. To begin with, Logo programs, or “source text,” don't have much in common with actual essays and stories. For example, two Logo programs, or “procedures,” which can be used in teaching students to develop essays through extended definition, look like this:

```
TO SQUARE :SIZE
    FD :SIZE
    RT 90
    FD :SIZE
    RT 90
    FD :SIZE
    RT 90
    FD :SIZE
    RT 90
    END

TO GROWSQUARES :SIZE
    SQUARE :SIZE
    RT 20
    GROWSQUARES :SIZE + 5
    END
```

Beyond Logo: What Would Writing Teachers Want?

This Logo code is easy enough to teach, and it includes some “powerful ideas,” such as the use of a variable (:SIZE) and the odd notion of recursion, through which the procedure called “GROWSQUARES” starts increasingly large iterations of itself (GROWSQUARES :SIZE +5). However, if we want to instruct students the writing process without having to teach for transfer in a very vigorous, time-consuming way, we would be better off with a programming language whose code looks more like an essay. In addition, Logo code generally does not produce output that looks like an essay or narrative. In the example we've reproduced above, the output is an ever-increasing spiral drawn on the computer screen—an admirable way to show what recursion is, but a long way from a written text. For teaching purposes, most of us would prefer a programming language that can produce an interesting prose output.

Is there a programming language that uses essay-like source code to produce useful prose output? Yes, but it hasn't been around for long, only since 2006. The language is called Inform 7, and it's used to create interactive fiction. Inform 7 (http://inform7.com) is free of charge and runs on Windows, Macintosh, and Linux computers. There's a truncated, but very useful Web version called Playfic (http://playfic.com)

Interactive Fiction

Here is a transcript of a session with “A Narrative About Virtue,” a work of interactive fiction (AKA, an interactive fiction). This story includes a bit of conversation with a character, and so it is not quite so simple as “Lost Chicken,” which we used in the previous chapter. This new example story will prove useful when we look at the sort of writing students might do with Inform. The reader's input, which would vary greatly from session to session, appears in boldface type.

```
It's another typical day in College
Writing II.  The professor is
illustrating the use of thesis
statements, using a whiteboard, when she
drops her marker on the floor.  She
appears to be a little hesitant to bend
down and pick it up.

A Narrative About Virtue
An Interactive Fiction by Brendan
Desilets
Release 1 / Serial number 101106 / Inform
7 build 6E72 (I6/v6.31 lib 6/12N)

Coburn 301
A typical college classroom, with lots of
desks, a whiteboard, and a dozen or so
students.

You can see the professor, a marker, and
Matt here.

>Look at the desks.
Ordinary desk-chair furniture.

>Look at the whiteboard.
An ordinary whiteboard, which the
professor has been using in her
presentation.
```

>**Look at the professor.**
A smart and kindly lady, but elderly and not too spry.

>**Pick up the marker.**
As you pick up the marker, you notice that there's a Zune MP3 player on the floor, under your chair. The Zune isn't yours, though you wish it were. In fact, you recognize the Zune as the property of Matt, who's slouched beside you.

>**Take the Zune.**
Taken.

>**Give the marker to the professor.**
The professor thanks you with notable sincerity. You have demonstrated the virtue of thoughtfulness.

[Your score has just gone up by five points.]

>**Look at the professor.**
A smart and kindly lady, but elderly and not too spry. The professor looks pleased with you.

>**Give the Zune to Matt.**
Matt nods. He looks a little surprised at getting his toy back so easily. You have demonstrated the virtue of honesty.

The professor turns to you and asks, "Now, Jamie, can you tell us who wrote Plato's *Republic*?" Unfortunately, you've never heard of this work.

The professor says, "Jamie, I know you can figure this one out. Who wrote Plato's *Republic*?"

[Your score has just gone up by five points.]

>**Look at Matt.**
A sincere, if slightly lethargic young man, wearing a UMass Lowell sweatshirt. Matt looks quite happy right now.

The professor says, "Jamie, I know you can figure this one out. Who wrote Plato's *Republic*?"

>**Say Aristotle to the professor.**
The professor says, "No. That's not it."

```
The professor says, "Jamie, I know you
can figure this one out. Who wrote
Plato's Republic?"

>Say Plato to the professor.
The professor expresses appreciation of
your insightful response.  You have
demonstrated the virtue of intelligence.



    *** You have won ***


You scored 15 out of a possible 15, in 13
turns.
```

(An interactive version of this very brief story is available on the Web, at http://tinyurl.com/virtueif Some examples of real, fully developed interactive fiction are at http://pr-if.org/play/.)

Inform 7

How does Inform 7 enable a writer to produce an interactive story? Like all IF authoring systems, Inform 7 provides a parser that can accept a reader's input. In addition, Inform offers ways for the author to manipulate the parser for the purposes of a particular story. Further, Inform offers a model world that the writer can use, altering it as he or she will. This model world includes locations (called “rooms”), doors, items that can or can't be carried, containers, locks and keys, and characters. The great innovation that Inform 7 offers to IF authors is the ability to implement all of the above, while staying, pretty much, within the parameters of readable English prose, with typical sentences, punctuation, and paragraphs.

In earlier chapters, we looked at two authoring systems that use menus and forms to produce interactive stories. But Inform, TADS, and most other IF-making systems work differently. Using these systems, when an author produces a work of interactive fiction, he or she writes “source code,” a set of instructions for the computer to use in presenting the story to a reader. The writer then tells the authoring system to transform the source code into a form that a computer can understand. This transformation is called “compiling.” The Inform 7 source code for an interactive story takes the form of a recognizable essay developed through process analysis. Using English sentences, the source code tells the computer, step by step, how to present the story to the reader.

The process of compiling source text offers some useful opportunities for the teaching of writing. In order to compile source code, the Inform system has to check the code for a variety of features, including spelling, punctuation, vocabulary, transitions, and completeness. In other words, the compiler checks for textual features that relate directly to what writing teachers call clarity, coherence, and development. If the source code fails to live up to the compiler's expectations, Inform 7 will fail to produce a work of interactive fiction, issuing an error message. Inform 7's compiler acts somewhat like a very strict teacher of writing.

A Look at Inform 7 Code

Let's have a look at some source code, to see how these ideas work out in practice. Once we have set up an Inform 7 project, by specifying its title and author, the system offers us a nearly blank pane in which to place our code. The pane looks like this:

```
"A Narrative About Virtue" by Brendan
Desilets
```

In this pane, let's create our first room, using English sentences.

"A Narrative About Virtue" by Brendan Desilets

```
Coburn 301  is a room.  The description
```

```
of Coburn 301 is "A typical college
classroom, with lots of desks, a
whiteboard, and a dozen or so students."
```

At this point, if we tell Inform to compile our code, Inform will do so, placing our player/character of the story in a room called “Coburn 301” and described as “A typical college classroom, with lots of desks, a whiteboard, and a dozen or so students.” The reader won't be able to do much of anything yet, but the story will, at least, start. Suppose, though, that we misspell “description.” In this case the compiler will refuse the create a story, complaining as follows:

```
Problem. You wrote 'The desciption of
Coburn 301 is "A typical college
classroom, with [...] board, and a dozen
or so students."'  but this seems to say
that a thing is a value, like saying 'the
chair is 10'.
```

The compiler thus has a role in enforcing clarity by insisting, up to a point, on correct spelling. Similarly, it will insist on a controlled vocabulary (“room”) and completed punctuation, such as the closing quotation marks. What's within the quotation marks, on the other hand, is, for the most part, none of Inform's business. The compiler will not complain if a student misspells “whiteboard,” for example.

Next, let's create the character of the professor. Here's some code that will get us started:

```
The professor is a woman.  The
description of the professor is "A smart
and kindly lady, but elderly and not too
spry."  The professor is in Coburn 301.
The indefinite article of the professor
is "the".
```

Here, some writing teachers will see an opportunity to teach about articles. Since “the professor” is more appropriate for our story than “a professor,” we add an assertion that controls the article. Note that, in specifying the article “the,” Inform does not follow the usual English convention for the period and its placement relative to closing quotation marks. If Inform did not make this variation, “the professor” would come out, in the story as “the. Professor.”

There's also an opportunity for work with clarity and development here, since student writers will often forget to specify a character's location. In other words, they will fail to use an assertion like “The professor is in Coburn 301.” If a student leaves out the professor's location, the story will still compile, but the professor will not appear anywhere. She will be “off-stage,” to use Inform's terminology. Of course, we might, under some circumstances, want a character to be off stage at first, so that we can bring her into the story later on.

Now, let's create, or “implement” the marker and the Zune.

```
The marker is a thing in Coburn 301.  The
description of the marker is "An ordinary
black dry-erase marker."

The Zune is a thing.  The description of
the Zune is "A slick new music player."
Understand "ipod" and "mp3 player" as the
Zune.
```

Inform's assumptions about these objects, since we haven't specified otherwise, is that they are small enough to pick up and carry around, though we can easily override these assumptions when we want to. Notice that the Zune is initially off-stage, since we don't want the player/character to see it until she picks up the marker for the professor. Note, also, an important consideration for clarity and development in interactive fiction: we have provided synonyms for the reader to use in referring to the Zune. Readers of IF are famously appreciative of such consideration.

Now that we've implemented some objects, let's set up our first rule for governing how the story is to be presented.

```
When play begins, say "It's another
typical day in College Writing II.  The
professor is illustrating the use of
thesis statements, using a whiteboard,
when she drops her marker on the floor.
She appears to be a little hesitant to
bend down and pick it up."
```

This rule will take effect only once in the story, at the very beginning. In Inform 7, “say” is an instruction to the computer to display some text on the screen. This particular rule causes only one action to happen, the “saying” of some words. We'll soon see other rules that cause a series of events to take place.

An interactive fiction does not have to include a “When play begins” rule, but nearly all IF stories do. The inclusion of such a rule helps a student writer to see the importance of a coherent beginning. We'll soon see a rule for the story's ending.

Next, let's implement a more complex rule to describe what happens when the player/character gives the marker to the professor.

```
Instead of giving the marker to the
professor:
```

```
say "The professor thanks you with
notable sincerity. You have
demonstrated the virtue of
thoughtfulness.";
move the marker to the professor;
increase the score by 5.
```

This time, we're using an “instead” rule. Instead of what? Instead of what the model world of Inform would otherwise do. The Inform model world supports the notion of giving something to someone. It understands the verb “give” as a transitive verb that is typically followed by an object and by a prepositional phrase starting with “to.” However, since it has no way of knowing what any given author wants to do with the action of giving, the model world, as a default, responds, to an attempt to give, with a plain-vanilla piece of text, proclaiming that the intended recipient “doesn't seem interested.”

In our case, we want to make three assertions that will take effect when the player/character gives the marker to the professor. In order to signal that we want more than one assertion to attach itself to this rule, we use a colon, rather than a comma, after “Instead of giving the marker to the professor.” Then we make our three assertions, each indented to show that all three are to follow from our “instead” rule. The first of our three assertions prints some text to the screen. This text expresses the professor's gratitude, but it doesn't do anything to move the marker. Like all assertions in a list of this sort, our “say” assertion has to end in a semicolon, odd as it may look. The second assertion handles the actual transfer of the marker to the teacher, and the third increases the player/character's score, which we use, in this story, to mark the character's demonstration of various virtues.

Now, let's implement the player/character's picking up the marker.

```
After taking the marker:
   say "As you pick up the marker, you
notice that there's a Zune   MP3 player
on the floor, under your chair.  The Zune
isn't    yours, though you wish it were.
In fact, you recognize the Zune  as the
property of Matt, who's slouched beside
you.";
   move the Zune to Coburn 301.
```

In its structure, this new rule looks a lot like the “instead” rule that we created earlier. This time, however, the rule does not substitute itself for Inform's default action. Instead, the rule takes effect **after** the player/character has taken the marker. Inform's default action of “taking” transfers the marker to the player's possession automatically. We don't need to create our own assertion for the transfer. The first assertion that we do make prints some text to the screen. The second assertion moves the Zune from off-stage to Coburn 301, where the player/character can now interact with it.

Some writing teachers, since they are not really trying to build better programmers, might omit the idea of the “after” rule altogether, since the same result can be achieved with an “instead” rule, like the one that follows.

```
Instead of taking the marker:
   say "As you pick up the marker, you
notice that there's a Zune
     MP3 player on the floor, under your
chair.  The Zune isn't
     yours, though you wish it were.  In
fact, you recognize the
     Zune as the property of Matt, who's
slouched beside you.";
   move the marker to the player;
     move the Zune to Coburn 301.
```

Creating the character Matt is essentially the same process as implementing the professor.

```
Matt is a man in Coburn 301.  The
description of Matt is "A sincere, if
slightly lethargic young man, wearing a
```

```
UMass Lowell sweatshirt."
```

And the rule for giving the Zune to Matt is similar to the rule for giving the marker to the professor.

```
Instead of giving the Zune to Matt:
   say "Matt nods. He looks a little
surprised at getting his toy
    back so easily.  You have
demonstrated the virtue of honesty.";
   move the Zune to Matt;
   increase the score by 5.
```

Now we need rules for a limited sort of conversation with the professor. Inform's built-in conversation system allows for asking, telling, consulting, and answering. First, we'll create a rule for the professor's posing of her question.

```
An every turn rule:
   if the score > 8:
      say "The professor turns to you and
      asks, 'Now, Jamie, can you tell us
      who wrote Plato's Republic?'
      Unfortunately, you've never heard
      of this work."
```

Here, we've implemented an “every turn” rule. Inform will check, at the end of every turn, to see whether the conditions imposed by the rule have been met. In our case, the only condition is that the player's score be greater than eight. If the condition is met, the professor will pose her question.

Now, we need a way to implement the player's answer. We'll use an “instead” rule to handle the correct answer. Using this rule, we'll print some text to the screen, increase the score to its maximum, and end the story.

```
Instead of answering the professor that
"Plato":
   say "The professor expresses
appreciation of your insightful
response.  You have demonstrated the
virtue of intelligence.";
   increase the score by 5;
   end the story saying “Congratulations!
You have proven to be virtuous.”
```

To handle incorrect answers, we'll create a similar rule that is deliberately more general than the rule for the correct answer. Inform applies a more general rule only if a more specific rule does not already apply.

```
Instead of answering the professor that
something:
   say "The professor says, 'No. That's
```

```
not it.'"
```

Our source code has now reached the point at which, when compiled, it produces a readable story, though not quite the story that our transcript shows. In addition, with some appropriate explanatory text, our source code would constitute an essay developed by process analysis. Still, there's one more powerful idea that we will need in order to produce the tale that our transcript tells. We need a variable. Inform allows for various sorts of variable, but let's start with a straightforward adjective.

```
A person can be pleased. A person is
usually not pleased.
```

Here, we have created the adjective variable “pleased.” We've restricted this adjective to people, and we're provided a default value for it, “not pleased.”

What can we do with this variable to shape our story? Almost anything we want. For example, if Matt is “not pleased” when the player/character has held his Zune for a certain number of turns, he might attack the player/character. On the other hand, if he is pleased, he might offer the player/character a reward for returning the MP3 player. In our transcript, though, we make somewhat subtler changes based on our variable—we change the characters' descriptions when they become pleased, by changing the source code for their descriptions as follows. Note the use of brackets to create conditions under which new text will be displayed.

```
The professor is a woman.  The
description of the professor is "A smart
and kindly lady, but elderly and not too
spry. [if the professor is pleased] The
professor looks pleased with you[end
if]."  The professor is in Coburn 301.
The indefinite article of the professor
is "the".

Matt is a man in Coburn 301.  The
description of Matt is "A sincere, if
slightly lethargic young man, wearing a
UMass Lowell sweatshirt. [if Matt is
pleased] Matt looks quite happy right
now[end if]."
```

As our code stands now, however, neither Matt nor the professor will ever become pleased. We'll have to change our rules for giving them their objects to include a change in pleased-ness. Our assertion for making this change reads “now the professor (or Matt) is pleased.”

```
Instead of giving the marker to the
professor:
    say "The professor thanks you with
    notable sincerity. You have
    demonstrated the virtue of
    thoughtfulness.";
    move the marker to the professor;
    now the professor is pleased;
    increase the score by 5.

Instead of giving the Zune to Matt:
   say "Matt nods. He looks a little
   surprised at getting his toy back so
   easily. You have demonstrated the
   virtue of honesty.";
   move the Zune to Matt;
   now Matt is pleased;
   increase the score by 5.
```

Let's add one more tool to our programming repertoire, a numerical variable. We may decide not to use this tool in connection with the teaching of writing, but it does become useful when students try to produce more complex stories. In our example, we'll make a numerical variable to help us vary what the professor says when she asks her *Republic* question.

```
The professor has a number called
questioning. The questioning of the
professor is 0.
```

Next, we create an "every turn" rule. At the end of every turn, Inform will check to see if it should carry out this rule. We'll use our numerical variable to vary what the professor says.

The first time the professor asks her question, the value of the numerical variable will be zero, its default. However, after she asks, we'll increase the value of the variable to one.

Note, also, the use of single quotation marks within the double quotation marks. For the player, Inform will display the single quotation marks as double quotes. We've added some italics here, too.

```
An every turn rule:
   if the score > 8:
      if the questioning of the professor
      is 0:
         say "The professor turns to you
         and asks, 'Now, Jamie, can you
         tell us who wrote Plato's
         [italic type]Republic[roman
         type]?'  Unfortunately, you've
         never heard of this work.";
         increase the questioning of the
         professor by 1;
      if the questioning of the professor
      is greater than 0:
         say "The professor says, 'Jamie,
         I know you can figure this one
         out. Who wrote Plato's [italic
         type]Republic[roman type]?'"
```

We've implemented nearly all of our story, as it appears in the transcript. All that's left to add are some objects and people that aren't important to the story's interaction. These include the whiteboard, the student desks, and most of the students. We implement these elements largely to achieve a sense of completion in our interactive world. If we did not implement these objects and the player tried to examine them, the response would be a nonsensical “You can't see any such thing.” By declaring that our new items are “scenery,” we prevent the player/character from taking them. Also, we'll use the Inform concept of “understanding” to implement some synonyms to help the reader along.

```
The whiteboard is scenery in Coburn 301.
The description of the whiteboard is "An
ordinary whiteboard, which the professor
has been using in her presentation."

Some desks are scenery in Coburn 301. The
description of the desks is "Ordinary
desk-chair furniture." Understand "desk"
and "chair" as the desks.
```

```
Some students are scenery in Coburn 301.
The description of the students is
"Ordinary collegiate scholars."
```

Finally, we'll establish two useful controls on the way the story will work. One of these controls declares the maximum score the player can achieve. The other tells Inform to punctuate series in the American style.

```
The maximum score is 15.

Use the serial comma.
```

A Process-Analysis Essay

Now the source code is complete, we can produce an essay developed through process analysis. All we'll have to do is add appropriate text to comment on the source code. The finished essay might look something like the following.

"A Narrative About Virtue" by Brendan Desilets

[How can we write a computerized interactive story? Using the authoring system called Inform 7, we can give the computer plain-Englsh instructions about how to present almost any story. Here, we'll develop some “source text,” which will tell the computer, step by step, how to tell a simple interactive story. Obviously, we've started with some text in brackets. Usually, text that appears in brackets is not part of the instructions for the computer. Instead, it is text intended for a human reader.]

[Our story is about a young woman in a college class. In the course of the story, she'll have a chance to display three traditional virtues, thoughtfulness, honesty, and intelligence.]

[First, we set the maximum score that the player can achieve in this gamelike story. The score will help to document virtues that the player/character displays.]

```
The maximum score is 15.
```

[Next, we tell Inform to use a comma between the last two items in a series. This use is typical in America, but not in Inform's homeland, England.]

```
Use the serial comma.
```

[Here we create our first noun, a room.]

```
Coburn 301 is a room.  The description of
Coburn 301 is "A typical college
classroom, with lots of desks, a
whiteboard, and a dozen or so students."
```

[Here we create an adjective variable. Once we make our variable, any person can be either pleased or not pleased. The default, as we declare it, is "not pleased."]

```
A person can be pleased. A person is
usually not pleased.
```

[Here we create our first character, the professor. Her description changes, depending on whether or not she is pleased.

Notice that we have overridden the article that Inform would normally use in referring to the professor. We changed the article from the default "a" to "the." In this case, in our source code, we cannot follow the usual rule of putting a period inside quotation marks. If we put the period inside the quotation marks, the story would say, "You can see the. professor."

Note, also, that, within the description of the professor, we use brackets in a way that we have not seen before. Within quoted material, such as the professor's description, brackets can be used to specify a condition under which some text will be shown to the reader.]

```
The professor is a woman.  The
```

```
description of the professor is "A smart
and kindly lady, but elderly and not too
spry. [if the professor is pleased] The
professor looks pleased with you[end
if]."  The professor is in Coburn 301.
The indefinite article of the professor
is "the".
```

[Next, we create an object, in the form of a noun. The default qualities of this object allow the play to pick it up and carry it around. These defaults also cause the item to mentioned in room descriptions.]

```
The marker is a thing in Coburn 301.  The
description of the marker is "An ordinary
black dry-erase marker."
```

[Here we create another object, but this one is an example of scenery. If an object is scenery, the player cannot pick it up, and it is not mentioned separately in the description of a room.

The reader will not be able to interact with the whiteboard, except by looking at it.]

```
The whiteboard is scenery in Coburn 301.
The description of the whiteboard is "An
ordinary whiteboard, which the professor
has been using in her presentation."
```

[Here we create some scenery with a plural name, "desks." Also we create two synonyms, "desk" and "chair."]

```
Some desks are scenery in Coburn 301. The
description of the desks is "Ordinary
desk-chair furniture." Understand "desk"
and "chair" as the desks.

Some students are scenery in Coburn 301.
The description of the students is
"Ordinary collegiate scholars."
Understand "student" as the students.
```

[Rules constitute a major component of any IF story's source code. Rules control, to a great extent, how the story unfolds. This is our first rule, carried out when the story starts.]

```
When play begins, say "It's another
typical day in College Writing II.  The
professor is illustrating the use of
thesis statements, using a whiteboard,
when she drops her marker on the floor.
She appears to be a little hesitant to
bend down and pick it up."
```

[Next, we create an "instead" rule. This rule causes something interesting to happen when we try to give the marker to the professor. Notice the precise use of parallel structure in this rule.]

```
Instead of giving the marker to the
professor:
    say "The professor thanks you with
    notable sincerity. You have
    demonstrated the virtue of
    thoughtfulness.";
    move the marker to the professor;
    now the professor is pleased;
    increase the score by 5.
```

[Here we create another rule. This one takes effect after Inform has allowed the player to pick up the marker.]

```
After taking the marker:
    say "As you pick up the marker, you
    notice that there's a Zune
    MP3 player on the floor, under your
    chair.  The Zune isn't yours, though
    you wish it were. In fact, you
    recognize the
    Zune as the property of Matt, who's
    slouched beside you.";
    move the Zune to Coburn 301.
```

[Here we create the Zune, in the form of a noun. Notice that the Zune isn't anywhere when we create it. We move it to Coburn 301 when the player takes the marker.]

```
The Zune is a thing.  The description of
the Zune is "A slick new music player."
Understand "ipod" and "mp3 player" as the
Zune.

Matt is a man in Coburn 301.  The
description of Matt is "A sincere, if
slightly lethargic, young man, wearing a
UMass Lowell sweatshirt. [if Matt is
pleased] Matt looks quite happy right
now[end if]."

Instead of giving the Zune to Matt:
   say "Matt nods. He looks a little
   surprised at getting his toy
   back so easily.  You have demonstrated
   the virtue of
   honesty.";
   move the Zune to Matt;
   now Matt is pleased;
   increase the score by 5;
```

[Next, we create a numerical variable called "questioning." We'll use this number to vary what the professor says when she questions the player/character.

The first time the professor asks her question, the value of the questioning variable will be zero, its default. However, in a rule we'll create later, we'll increase the value of the variable when she speaks.]

```
The professor has a number called
questioning. The questioning of the
professor is 0.
```

[Here we create an "every turn" rule. At the end of every turn, Inform will check to see if it should carry out this rule. We'll use our numerical variable to vary what the professor says.

The first time the professor asks her question, the value of the numerical variable will be zero, its default. However, after she asks, we'll increase the value of the variable to one.

Note, also, the use of single quotation marks within the double quotation marks. For the player, Inform will display the single quotation marks as double quotes.]

```
An every turn rule:
   if the score > 8:
      if the questioning of the professor
      is 0:
         say "The professor turns to you
         and asks, 'Now, Jamie,
         can you tell us who wrote
         Plato's [italic
         type]Republic[roman type]?'
         Unfortunately,
         you've never heard of this
         work.";
         increase the questioning of the
         professor by 1;
      if the questioning of the professor
      is greater than 0:
         say "The professor says, 'Jamie,
         I know you can figure
         this one out. Who wrote Plato's
         [italic type]Republic[roman
         type]?'"
```

[Here we create a rule to enable the player to answer the professor's question. This rules also ends the story/game with the success of the player. Notice that Inform can understand the verb "answer." In Inform, it is also possible to introduce new verbs. We can even create verbs that can be used with particular prepositions.]

```
Instead of answering the professor that
"Plato":
   say "The professor expresses
   appreciation of your insightful
   response.  You have demonstrated the
   virtue of intelligence.";
   increase the score by 5;
   end the story saying “Congratulations!
   You have proven to be virtuous.”
```

[Finally, we create a rule that applies when the player/character gives any wrong answer to the professor's question. We make this rule more general that the previous rule because Inform will pass over a more general rule if a more specific rule applies.]

```
Instead of answering the professor that
something:
   say "The professor says, 'No. That's
   not it.'"
```

[Now, once we compile our story, it will be ready for our readers.]

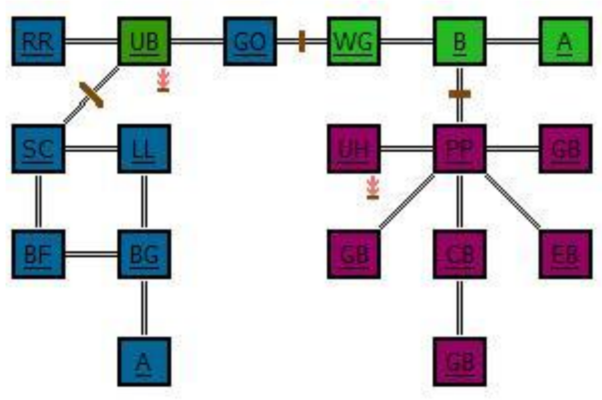
Inform 7 and Qualities of Good Writing

Inform 7 can help writers to produce good prose—prose that exhibits widely-accepted qualities of good writing, such as unity, coherence, development, and clarity.

Unity

Inform 7 helps to encourage, and even to enforce, unity in some ways that we have already seen. For example, it nudges students toward giving each story a beginning, marked by a “when play begins” rule, a middle, and at least one ending, indicated by an “end the story” assertion. In addition, Inform allows a writer to divide source code into larger and smaller pieces, called, form largest to smallest, "volumes," "books," "parts," "chapters," and "sections." The Inform compiler points directly to the names of these pieces when it complains about problems in the source text.

Unlike our sample story, many works of interactive fiction require the user to move through a great many locations. Inform helps the writer maintain the physical unity of his or her setting by generating a simple map of the story's locations. If the author has made a mistake in connecting locations, the map will usually show the error. Here, for instance, is the Inform-generated map of a fairly complex interactive fiction.



The eighteen rooms on this map connect in clear ways, with problem-solving required to get through the passages marked with a brown bar. If the writer had left a room unconnected with the rest, the map would make the error clear.

Also, Inform indexes all the people and objects in a story, showing which encloses which. Writers often check Inform's index to see how the story's parts relate to the whole. In our brief story, for example, the index looks like this:

Coburn 301 - *room where play begins*

professor - *woman*

marker

whiteboard

desks

students

Matt - *man*

yourself – *person*

Zune

This chart clearly shows an item that might violate the unity of our story, the Zune, which is not inside the story's single room. In our case, we have a good plan to add the Zune to our tightly-unified tale; but, if, in fact, we had inadvertently created an off-stage music player, the index would alert us to the problem.

Coherence

Creating an interactive fiction requires great attention to connectedness. A story that is not coherent in its use of rules and variables will often fail to compile and will never execute well. In our sample story, for example, we create the adjective variable “pleased.” If we are to use this variable, we must do so in such a way as to effectively connect with our story's rules. In our rule for making the professor pleased by giving her the marker, we must specify that “now the professor is pleased,” not “happy” or “grateful.” Further, when we show the professor's pleased state in her description, we must use the bracketed text “if the professor is pleased” to connect the description with the variable, and we must complete the connection by showing the end of the text to which our condition applies, using the bracketed text “end if.”

Earlier, we noted that Inform automatically generates a story map and an outline of the story's elements. In addition to helping with unity, these devices help the author to see how coherently a story's elements connect with one another. Inform also helps the author to manage a story's locations by facilitating the grouping of the rooms into regions. In the sample map that appears above, regions are color-coded.

Just as helpful is the structure known in Inform as the “scene.” Inform scenes are like the scenes of a play in that they usually have clear beginnings and endings, but these starts and stops generally depend on which actions the player/character has always taken. Suppose, for example, our “Narrative About Virtue” had a second location, called the Hallway. Suppose, further, that we want to start a new scene when the player/character enters the Hallway, but only if she has returned the Zune to Matt, who will help her out if she has helped him. Let's say that, in this scene, the player, with Matt's help rescues a cat that is stranded on top of some scaffolding. We could create our scene with source code like this.

Rescue is a scene. Rescue begins when the player is in the Hallway and the Zune is carried by Matt. Rescue ends when the player carries the cat.

Some IF writers make excellent use of scenes to establish coherence, even in relatively simple stories. Inform helps writers to manage scenes well by providing a command that allows an author to see exactly when a scene starts and ends during a trial of his or her story. In addition, Inform offers an index of scenes that shows how each scene begins and ends, and indicates whether or not each scene can recur.

Development

The creation of a readable IF story requires an unusually studied sort of development, in at least five areas, characters, actions, objects, rules, and synonyms. The development of multi-dimensional characters and the creation of new actions (and the verbs that activate them) goes beyond the expertise that we would expect students to develop, if we are using Inform only as a tool to help them build their essay-writing skills. However, even in our very brief example story, we have seen the importance of developing effective objects, rules, and synonyms.

Development Through Implementing Objects

As we've noted in our creation of a sample source text, it is important for IF writers to implement the objects that they mention in their descriptions. Otherwise, their stories will generate inappropriate responses that break the immersion of their readers in the model world. Normally, the Inform compiler cannot help much with this problem; but acclaimed interactive fiction writer Aaron Reed has developed a system to customize Inform code to ensure thorough implementation of objects. Under Reed's system, when an IF author creates text within quotations marks, he or she put brackets around the noun that names each important object. If the author has already implemented the bracketed object, the brackets will have no noticeable effect. However, if the author has not implemented the bracketed object, the compiler will complain, thus reminding the writer that one of the story's elements needs more development.

Using Reed's “Bracket Every Notable Thing” procedure, our description of Coburn 301 would look like the following.

```
Coburn 301 is a room.  The description of Coburn 301 is "A typical college classroom, with lots of [desks], a [whiteboard], and a dozen or so [students]."
```

In the final version of our sample story, we have implemented the desks, whiteboard, and students. As a result, the bracketing will have no effect on the compiler's output. However, if we had forgotten to implement the desks, for instance, the computer would produce no output and would complain that it cannot understand “[desks].”

Developing Effective Rules

In the authoring of interactive fiction, developing rules sufficient presents a real challenge, especially to inexperienced writers. The Inform compiler will not generally signal problems with rule development, but even superficial reader-testing of a story usually will.

For example, to new authors, our first rule for answering the professor's question may look quite sufficient.

```
Instead of answering the professor that
"Plato":
   say "The professor expresses
   appreciation of your insightful
   response.  You have demonstrated the
   virtue of intelligence.";
   increase the score by 5;
   end the story saying “Congratulations!
   You have proven to be virtuous.”
```

However, in testing the story, a reader will soon see that the rule is insufficient to deal with an incorrect answer. In evaluating a rule, especially an “instead” rule, an experienced IF writer will immediately check to see whether the rule covers all possible situations, not just the one that seems most likely. In the case of our rule, as it appears above, we have accounted for only one possibility, the correct answer. In our completed source code, we have handled the wrong answers in a separate, more general rule.

```
Instead of answering the professor that
something:
   say "The professor says, 'No. That's
   not it.'"
```

We could also have dealt with the possibilities in a single rule, but such a rule might be more complex that we want, in modeling Inform code for our composition students.

Developing Synonyms

Interactive fiction is a difficult form of literature for most readers. One of the ways an IF author can help the reader along without damaging the reader's immersion in the story is by providing a ready supply of synonyms for important words. A lack of such synonyms often produces frustrating and unfulfilling “guess the verb” or “guess the noun” puzzles. The Inform compiler can't really help with needed synonyms, but test readers certainly can; and Inform provides a quick and easy way to create synonyms, as we did in our description of the Zune.

```
The Zune is a thing.  The description of
the Zune is "A slick new music player."
Understand "ipod" and "mp3 player" as the
Zune.
```

Clarity

Clarity may be the most elusive quality of good writing. The Inform compiler cannot approach the effectiveness of a careful reader in suggesting ways to clarify a text, but it can help in a number of ways, when a human reader isn't available.

Clarity in Quoted Text

In general, Inform does not actively intervene in material that appears within quotation marks. The writer usually asks only that Inform print such material on the screen, warts and all. However, in our example story, we can see two kinds of text with quotes that the Inform compiler does check. First, consider bracket material within quotation marks, as in our description of Matt.

```
Matt is a man in Coburn 301.  The
description of Matt is "A sincere, if
slightly lethargic young man, wearing a
UMass Lowell sweatshirt. [if Matt is
pleased] Matt looks quite happy right
now[end if]."
```

The compiler will point out any spelling errors within the brackets, and it will report on anything that may be missing, such as a bracket or a needed phrase “[end if].” (However, in this particular case “[end if]” is not strictly necessary because the conditional expression comes at the end of the quoted material.) It will check to see that a variable like “pleased” has been previously defined and that it is applicable to Matt.

The compiler will also report problems with quotations with quotations, which must be marked with single quotation marks, just as they would be in standard English. If we had made the following error, our text would not compile.

```
say "The professor turns to you and asks,
“Now, Jamie, can you tell us who wrote
Plato's [italic type]Republic[roman
type]?”  Unfortunately, you've never
heard of this work.";
```

We would need single quotation marks around the professor's words to solve this problem.

Clarity in Other Text

The Inform compiler really shines (or, some would say, becomes really picky) in checking the clarity of source code that is not within quotation marks. It will reject most spelling errors, and will require end punctuation according to rules that mimic standard English, except that Inform allows the semicolon to end an assertion. Inform checks for a comma at the end of an introductory adverbial clause, though it requires the use of a colon if such a clause is followed by more than one assertion. The Inform compiler also insists on parallel structure in lists of assertions. Such lists are very common in Inform code, as in this example from our sample story.

```
Instead of giving the Zune to Matt:
   say "Matt nods. He looks a little
   surprised at getting his
   toy back so easily.  You have
   demonstrated the virtue of honesty.";
   move the Zune to Matt;
   now Matt is pleased;
   increase the score by 5.
```

On the other hand, Inform does not check for capitalizing, though it almost always allows conventional use of the upper case, and it does not use the conventional comma to separate quoted text from quoted text, as in:

```
Some desks are scenery in Coburn 301. The
description of the desks is "Ordinary
desk-chair furniture."
```

Inform's built-in text editor helps with clarity, too, in that it color-codes words that appear in quotation marks and in brackets.
Here's an example.

```
An every turn rule:
   if the score > 8:
      if the questioning of the professor
      is 0:
         say "The professor turns to you
         and asks, 'Now, Jamie, can you
         tell us who wrote Plato's
         [italic type]Republic[roman
         type]?'  Unfortunately,
         you've never heard of this
         work.";
         increase the questioning of the
         professor by 1;
      if the questioning of the professor
```

```
    is greater than 0:
        say "The professor says, 'Jamie,
        I know you can figure this one
        out. Who wrote Plato's [italic
        type]Republic[roman type]?'"
```

“Commented” text, which the compiler ignores, appears in green, as in

[This is our first rule. It is carried out when the story starts.]

Inform and the Writing Process

In addition to providing students with some insights into the quality of the writing, Inform can help them to understand the phases of the writing process, though some of these phases take on some new wrinkles in the creation of interactive fiction.

Prewriting

Even the simplest of IF stories, such as the one we've used as an example here, require some explicit planning. This planning has to address the usual concerns of narrative writing, such as plot, setting, and character, and some concerns that address the peculiarities of interactive fiction.

In planning a first work of interactive fiction, it makes sense to look for brevity. Even our super-brief example story requires 575 words of source text, not counting comments, and getting source text to work can be a real challenge for any new author. If we add enough commenting to make the story comprehensible to an intelligent, but previously uninformed, reader, we approximately double the length. In addition, a new IF writer should think about what does, and does not, work well in an all-text interactive medium. IF stories do well with exploration, discovery, and problem-solving. They are not well suited to quick-twitch reactions.

After finding a suitable story topic, a new author generally does well to outline, or map, the basics of the story's plot, characters, objects, puzzles, and setting. The rigors of writing a story that requires computer programming work strongly against the composing styles of students who resist planning. An unplanned story is virtually impossible to implement. Often IF planning goes beyond an outline. If the story has a half dozen or more rooms, the writer will probably produce a map as part of his or her prewriting. In a longer story, the author may create a list of scenes.

Drafting

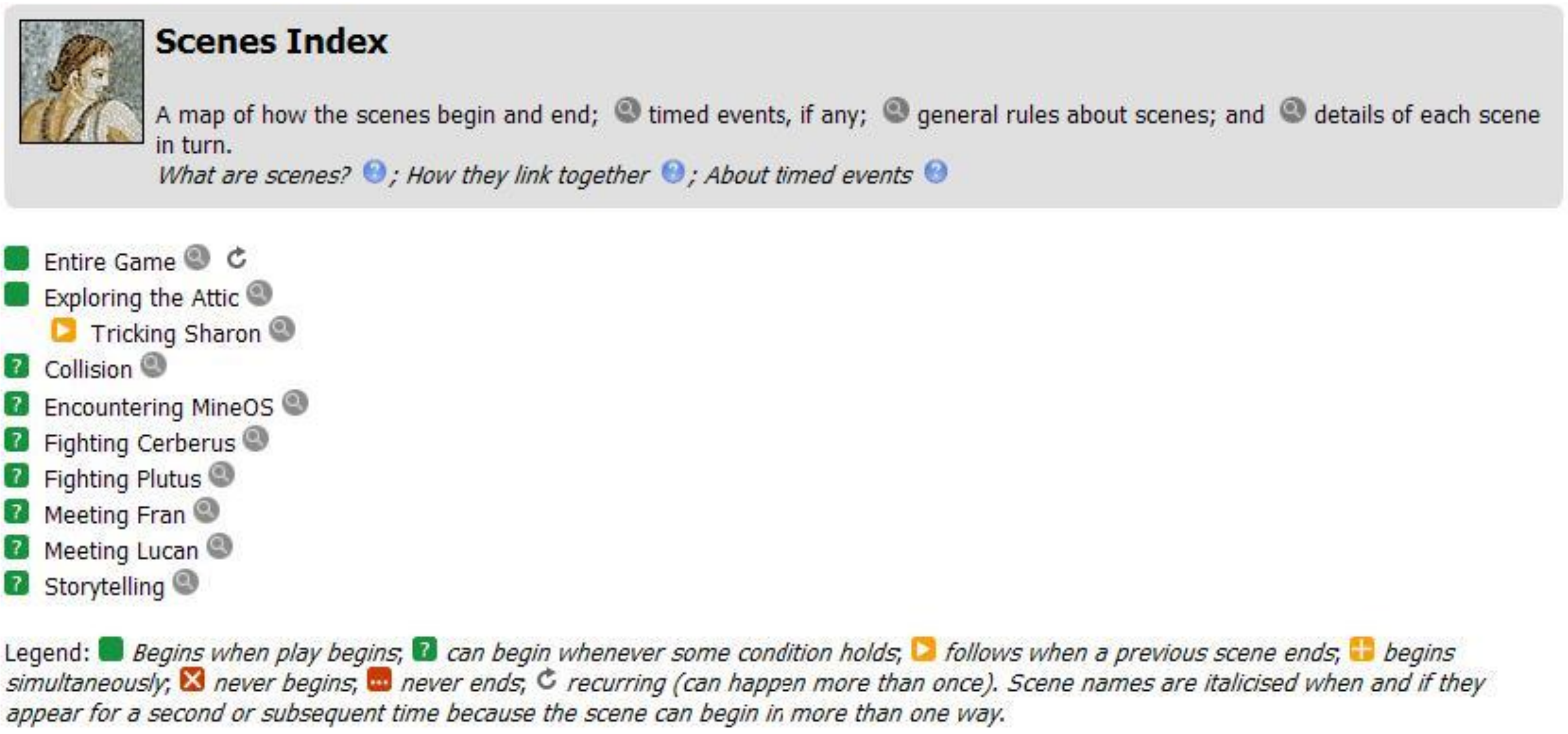
After prewriting, an IF writer often produces a detailed prose version of the story, or at least of one possible run-through of the piece. This non-interactive draft usually takes the form of an imagined transcript of a session with the story. In the case of a longer story, the author may draft a scene and implement that scene in Inform, before going on the next scene.

Of course, the first draft of an interactive fiction is not complete until it has taken an interactive form. For this reason, drafting in IF requires creating source text and testing the source text by trying to compile it. When the compiling fails, the writer will typically jump ahead to the revising and editing stages of the writing process until the text compiles, before getting back to drafting. This sort of alternating among drafting, revising, and editing may run against the grain for many teachers of the writing process, while others will find this sort of shifting quite natural. In any case, no one can deny that producing an interactive fiction requires active revising and editing. Without revising and editing, the source code will simply never compile.

Revising

Even after the source code compiles, the revising stage of writing an IF story has a long way to go. “Alpha” testing, by the writer herself invariably reveals many problems that need correcting, and “beta” testing, by others, will reveal even more issues, ranging from difficulties in character development to rules that don't work as intended. Inform offers some tools that can help.

As noted earlier, Inform generates a map of the story's model world to help with revisions, and it also creates an outline of all the story's rooms, characters, and objects. To help with scenes, Inform indexes all of a story's scenes and their properties, as in the following example.



Scenes Index

A map of how the scenes begin and end; timed events, if any; general rules about scenes; and details of each scene in turn.

What are scenes? ; *How they link together* ; *About timed events*

- 🟩 Entire Game ↻
- 🟩 Exploring the Attic
  - ▶ Tricking Sharon
- ? Collision
- ? Encountering MineOS
- ? Fighting Cerberus
- ? Fighting Plutus
- ? Meeting Fran
- ? Meeting Lucan
- ? Storytelling

Legend: 🟩 *Begins when play begins;* ? *can begin whenever some condition holds;* ▶ *follows when a previous scene ends;* ✚ *begins simultaneously;* ✖ *never begins;* ⋯ *never ends;* ↻ *recurring (can happen more than once). Scene names are italicised when and if they appear for a second or subsequent time because the scene can begin in more than one way.*

Inform also provides a testing command, “scenes,” which causes the playing of a story to note each scene change when it occurs.

Rules often require revision, partly because they are difficult to craft in themselves, and partly because they often interact with one another in complicated ways. The “rules” testing command is, therefore, very useful. This command causes the story's output to list, every turn, each rule that applies. The results of the “rules” command can be a bit confusing at first, because it lists not only the applicable rules that the author has created but also the underlying, “standard” rules of the Inform system, with which the writer's rules may conflict. Here's an example from our sample story.

```
>rules
Rules tracing now switched on. Type
"rules off" to switch it off again, or
"rules all" to include even rules which
do not apply.

>take marker
[Rule "can't take yourself rule"
applies.]
[Rule "can't take other people rule"
applies.]
[Rule "can't take component parts rule"
applies.]
[Rule "can't take people's possessions
rule" applies.]
[Rule "can't take items out of play rule"
applies.]
```

```
[Rule "can't take what you're inside
rule" applies.]
[Rule "can't take what's already taken
rule" applies.]
[Rule "can't take scenery rule" applies.]
[Rule "can only take things rule"
applies.]
[Rule "can't take what's fixed in place
rule" applies.]
[Rule "use player's holdall to avoid
exceeding carrying capacity rule"
applies.]
[Rule "can't exceed carrying capacity
rule" applies.]
[Rule "standard taking rule" applies.]
[Rule "After taking the marker" applies.]
As you pick up the marker, you notice
that there's a Zune MP3 player on the
floor, under your chair.  The Zune isn't
yours, though you wish it were.  In fact,
you recognize the Zune as the property of
Matt, who's slouched beside you.
```

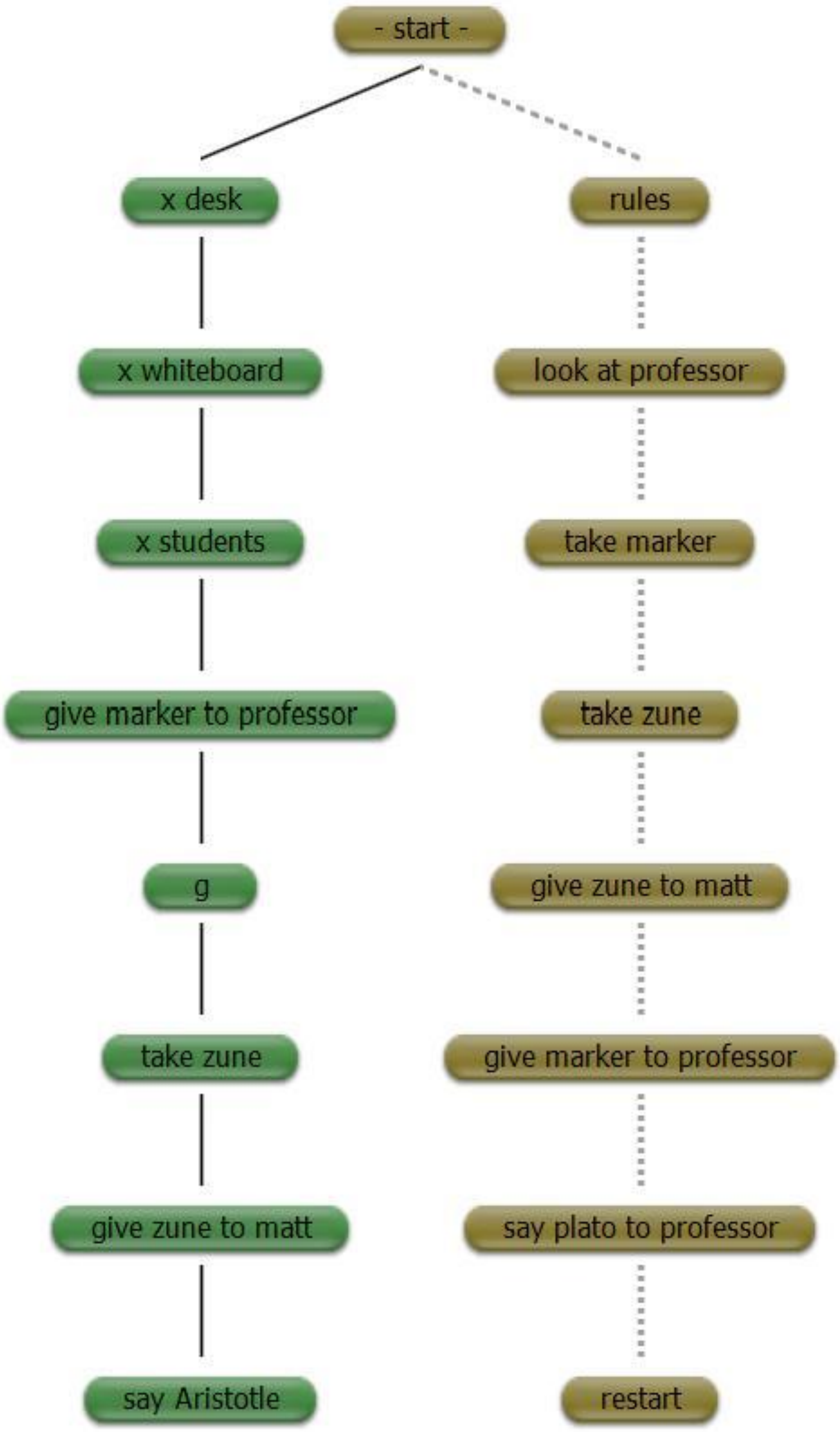
Variables often require revising, too, since it's easy to make mistakes in applying them to people and objects. Frequently, an author may think that a variable adjective, such as our “pleased” should be applying when the execution of the story indicates otherwise. To help with this sort of revision, Inform provides the testing command “showme,” which reveals all of an object's properties. Consider the following transcript, which shows a change in whether the professor is pleased and in what she carries.

```
>showme professor
professor - woman
location: in Coburn 301
unlit; inedible; portable
female
printed plural name: women
carrying capacity: 100
printed name: professor
indefinite article: the
description: A smart and kindly lady, but
elderly and not too spry.
questioning: 0

>give marker to professor
The professor thanks you with notable
sincerity. You have demonstrated the
virtue of thoughtfulness.

[Your score has just gone up by five
points.]

>showme professor
professor - woman
    marker
```

```
location: in Coburn 301
unlit; inedible; portable
female; pleased
printed plural name: women
carrying capacity: 100
printed name: professor
indefinite article: the
description: A smart and kindly lady, but
elderly and not too spry.  The professor
looks pleased with you.
questioning: 0
```

In a typical writing class, would students use testing tools commands like “scenes,” “rules,” and “showme”? Probably not, until (or unless) they needed them. However, students would, almost to a person, use the most useful revising tool of all, Inform's “skein.” In revising an interactive fiction, writers often need to replay the story to a particular point. This sort of replay can be repetitive and exhausting, but not if one uses the skein. Surprisingly, the skein keeps track of all the moves of every playing through of a particular story. These attempts at working through the story can become so numerous that Inform allows for trimming the skein from time to time.

Here's a look at a trimmed version of the skein for our example story.


- start -
x desk
rules
x whiteboard
look at professor
x students
take marker
give marker to professor
take zune
g
give zune to matt
take zune
give marker to professor
give zune to matt
say plato to professor
say Aristotle
restart

When a writer is viewing the skein in Inform, he or she can double click on any node (or “knot”), causing the story to play automatically to that point. If the author has been trying to remedy a problem that occurs at the chosen knot, the skein's replay will usually reveal whether the remedy has succeeded. If the remedy hasn't worked, the author can use Inform's other revision tools, or he or she can study the game's output to see where the problem first appears. Or the author can try Inform's “transcript” tab, which coordinates closely with the skein to help the author compare various replays, some of which will likely seem more correct than others.

And, best of all, an IF author can depend on the very supportive interactive fiction community for help. Some genuinely expert writers offer free and responsive advice at the Interactive Fiction Forum (http://www.intfiction.org/forum/).

Editing

We've already seen that Inform's color-coded text editor helps with paired punctuation marks, such as brackets, and that the Inform compiler flags a variety of issues that call for editing. These include spelling errors, and problems with various punctuation marks, including quotation marks, commas, and end marks. Most Inform authors, especially inexperienced ones, will feel that they have to resolve the compiler's complaints as they create source text, rather than postponing their editing until later.

Still, most of the editing that goes into an interactive story is much like the editing of any other piece of text. It requires close attention, benefits enormously from a reader's help, and occurs, most profitably, after drafting and revising.

Publishing

Inform offers exciting options for publishing, both in the sense of producing a polished final draft and in the sense of reaching readers. Inform allows a writer to produce a story that does not include testing commands, such as “rules” and “scenes.” Such commands only get in a reader's way. Quite easily, an author can include, in her final, compiled draft, cover art for her story, a booklet that introduces interactive fiction, a website about her story, a link that allows a reader to experience the story over the Web, and a walkthrough.

The interactive fiction community offers an active group of readers, too, reachable through the Interactive Fiction Forum (http://www.intfiction.org/forum/). The Interactive Fiction Archive (http://www.ifarchive.org/) houses thousands of IF stories an related material. So does the Interactive Fiction Database (http://ifdb.tads.org). New contributions to the archive often attract readers and, sometimes, reviewers.

Competitions for interactive fiction occur often and help to provide readers for new stories. The most prominent of these is the annual IF Comp (http://www.ifcomp.org/).

Advantages of Inform 7

It seems, then, that, from a writing teacher's viewpoint, Inform 7 has some real advantages of other programming languages. It is easy to learn. It works with natural-language source text that takes the form of an essay developed by process analysis. It produces interesting prose output, in the form of interactive fiction. Its compiler helps to flag a variety of problems, ranging from spelling errors to completeness of rules. And its built-in text editor, with its color-coding, helps with revising and proofreading. These advantages feed into ideas and processes that have strong support among writing teachers and researchers—ideas like the qualities of good writing and processes like prewriting, drafting, and revising.

But real teachers of writing may have reservations about the time and effort needed to get students started with Inform 7. What would constitute a truly minimalist approach that would allow students to produce a brief process analysis draft very quickly, perhaps in one class period?

One minimal sort of interactive story is the simple riddle. A transcript of such a story might look like the following.

```
A Riddle
An Interactive Fiction by Brendan
Desilets
Release 1 / Serial number 101120 / Inform
7 build 6E72 (I6/v6.31 lib 6/12N)

Riddle Room
This is a room with a big sign and a
number of toys scattered around.

The big sign poses a riddle. "What is
tall as a house, round as a cup, and all
the king's horses can't draw it up? Pick
up the correct toy to answer the riddle?"

You can also see a the rubber duck, a
Tonka truck, a Barbie doll, a miniature
robot, an old cell phone, a toy well, a
wooden plane, a plastic hammer and a
picture book here.

>take duck
```

```
    *** Sorry. That's not it. ***




Would you like to RESTART, RESTORE a
saved game, QUIT or UNDO the last
command?

> restart


A Riddle

An Interactive Fiction by Brendan
Desilets

Release 1 / Serial number 101120 / Inform
7 build 6E72 (I6/v6.31 lib 6/12N)


Riddle Room

This is a room with a big sign and a
number of toys scattered around.


The big sign poses a riddle. "What is
tall as a house, round as a cup, and all
the king's horses can't draw it up? Pick
up the correct toy to answer the riddle?"


You can also see a the rubber duck, a
```

```
Tonka truck, a Barbie doll, a miniature
robot, an old cell phone, a toy well, a
wooden plane, a plastic hammer and a
picture book here.

>take well
Well done! You've solved the riddle.

    *** You have won ***
```

This minimal story requires very few Inform 7 skills. The writer has to create a room, implement some objects, and develop an instead rule or two. The source text would look like this:

```
"A Riddle" by Brendan Desilets

Use no scoring.

The Riddle Room is a room. "This is a
room with a big sign and a number of toys
scattered around."

The sign is in the Riddle Room. "The big
sign poses a riddle. 'What is tall as a
```

```
house, round as a cup, and all the king's
horses can't draw it up? Pick up the
correct toy to answer the riddle.'"

The rubber duck, the Tonka truck, the
Barbie doll, the miniature robot, the old
cell phone, the toy well, the wooden
plane, the plastic hammer, and the
picture book are in the Riddle Room.

Instead of taking the toy well:
   say "Well done! You've solved the
   riddle.";
   end the story saying “Congratulations!
   You have won.”

Instead of taking something:
   end the story saying "No. That's not
   it."
```

After a brief study of this riddle story and its 134-word source code, students should be able to create their own riddle stories. Then, their challenge will be to add enough clear comments to make the source code intelligible. The whole essay would end up at around 300 words, quite manageable for a one-class exercise.

Some Instructional Considerations

What are the principal sticking points in teaching students to use Inform 7? In simple cases like the riddle story, there's just one — the introduction of the rule.

Seven years of working with students aged eleven to thirty have made it clear that no one has much trouble with the creation of rooms and portable objects. Even with a half hour of instruction, about ninety-nine percent of students can produce a compiled story with at least one room. With similar teaching, about ninety-three percent can produce portable objects.

However, the percentage of success drops to about two thirds when students try to produce simple, “When play begins” rules. And only about half of students can produce more complex “Instead” rules, without more extensive instruction and/or individual help.

The difficulties with rules occur in two principal areas, formatting and vocabulary. The correct formatting of rules requires a grasp of Inform's idiosyncratic (but sensible) use of colons, semicolons, and tabs. An earlier section of this essay, “A Look at Inform 7 Code,” addresses the formatting of rules in Inform. The vocabulary issues are a bit more complicated in that they demand some instruction in two kinds of word that Inform understands. Inform's documentation offers extensive lists of all the sorts of words that the authoring system recognizes. The docs also explain how a writer can expand Inform's vocabulary. Because familiarity with Inform's vocabulary is so useful for authors, Inform includes word lists in the “Index” tab on its default screen. These lists may be too elaborate for beginners to use right away, but they become indispensable resources, as students become more proficient writers. For new IF writers, the lists of words that appear below will probably prove more than adequate.

One of the word types that Inform understands consists of vocabulary that the interactor can use in reading the story. An example of such a command would be “taking,” in the rule

```
Instead of taking the scroll:
   say “The scroll seems unusually heavy
   as you pick it up.”;
   move the scroll to the player.
```

The more commonly-used words of this sort include the following:
taking (as in “taking the pistol”);
removing;
opening;
examining;
going (as in “going north in the kitchen”);
inserting (as in “inserting the pistol into the holster”);
dropping;
entering (as in “entering the booth”);
exiting;
asking (as in “asking Jeff about 'the key'” or asking Jeff for the key);

telling (as in “telling Jeff about 'the key'”)
and
answering (as in the odd-sounding “answering that 'Plato.'”)

A second type of word consists of vocabulary that the player cannot use. In the “Instead of taking the scroll” rule that appears above, the verb “move” would be example of this second type of word.

Commonly-used words and phrases of this type include the following:
now the ______ is ______ (as in “now the player carries the knife,” or “now the player is in the kitchen,” or “now the door is open”);
move (as in “move the pistol to the kitchen”);
say (meaning “print words on the computer screen,” as in “say 'Congratulations! You have obtained the babel fish.'”);
end the story (as in “end the story saying “The Sorcerer transforms you into a newt. Thus you have failed in your quest. But why not try again?”);
stop the action (referring to the action described in the current rule); and
continue the action.

We’ll deal, at greater link with this question of the author’s vocabulary in our next chapter, when we discuss the distinction between an action and a verb.

Obtaining Inform 7

Users of MacOS, Windows, and Linux can download Inform 7, free of charge, at http://inform7.com/download. The Mac and Windows versions are a bit more complete than the Linux variation. Web users can find a limited, but very convenient, version of Inform 7 at http://playfic.com. The Playfic variation does not offer most of Inform's revision tools, such as the skein and the transcript, but it works well for many student projects.

Table 1--Inform 7 Versus Four Other Programming Languages

Language	Easy to Learn	Natural Language Source Code
Logo	Yes	No
Inform 6, TADS	No	No
Adrift	Yes	No
Inform 7	Yes	Yes

Inform 6 and TADS (The Adventure Development System) are programming languages that help authors to produce interactive fiction. Adrift is a menu-based system for producing works of interactive fiction.

Chapter 9 --Why Inform 7?

How can writing teachers use Inform 7? Chapter 8 of this book tries to answer this question. It offers a tutorial on Inform 7, too. For most readers, Chapter 8 is a prerequisite for the essay you are reading now. Readers who are unfamiliar with Inform 7 really **must** start with Chapter 8, or something very much like it. But **why**, exactly, should teachers choose Inform 7? The current chapter will look at this question in two ways. First, it will answer in terms of two broad advantages. These advantages are the development of clear thinking through programming, and the creation of essays that simultaneously represent two different modes of discourse, exposition and narration. Then, "Why Inform?" will describe a series of tools for writers that Inform 7 provides, showing how these tools correspond to familiar writing-process implements, such as outlines and storyboards.

Clear Thinking Through Programming

Inform 7 is a programming language, though it's much friendlier and easier to learn than other programmers' utilities. For many years, scholars have noted that the process of writing good computer code is similar to that of writing good stories and essays, and have wondered whether the right sort of programming experience might improve students' composition skills.

Perhaps the most prominent of these scholars is Seymour Papert, whose seminal book *Mindstorms: Children, Computers, and Powerful Ideas (1981)* captured the imaginations of many educators. Papert advocated the use of child-friendly programming language called Logo to help students apply “powerful ideas,” such as constrictive repetition (recursion) and the organizational relationship of parts to wholes. (For an example of how Logo can help writing teachers, have a look at “Logo and Extended Definition,” at http://bdesilets.com/if/Logo.pdf). Today, Linda Sandvik and her associates teach programming to ten-year-olds in the UK, following a similar rationale and a variety of coding software, including Scratch, which is especially relevant for very young children. Inform 7 requires, and, to some degree, teaches powerful ideas, too. However, for learners who are in their teens and beyond, Inform 7 can be more useful for writing teachers than other programming environments because it requires, and enables, students to write computer code that consists of ordinary English sentences. These sentences add up to an expository essay of the process-analysis type. In addition, Inform 7’s code results in output that takes the form of another familiar mode of discourse, a prose narrative.

“Two Birds” (Two Modes of Discourse)

One reason to use Inform 7, then, is its unique way of providing students with rigorous practice in at least two of the traditional modes of writing, exposition and narration. As a very simple example, consider this ridiculously-brief interactive story.

```
A Riddle
An Interactive Fiction by Brendan
Desilets
Release 1 / Serial number 101120 / Inform
7 build 6E72 (I6/v6.31 lib 6/12N)

Riddle Room
This is a room with a big sign and a
number of toys scattered around.

The big sign poses a riddle. "What is
tall as a house, round as a cup, and all
the king's horses can't draw it up? Pick
up the correct toy to answer the riddle."

You can also see a the rubber duck, a
Tonka truck, a Barbie doll, a miniature
robot, an old cell phone, a toy well, a
wooden plane, a plastic hammer and a
picture book here.
```

(At this point, the reader sees a prompt like this “>,” indicating that he or she can type in any response that he or she likes. Let’s say that the reader types the following.)

```
>take well

“Well” done! You’ve successfully solved
the riddle, thus completing the story.
```

In order to produce this very simple narrative, the student would have to write “source code” consisting of a brief process-analysis essay, which tells the computer how to present the story. The source code would look something like what follows.

```
"The Riddle Room" by Brendan Desilets

The Riddle Room is a room. "This is a
room with a big sign and a number of toys
scattered around."

The sign is in the Riddle Room. "The big
sign poses a riddle. 'What is tall as a
house, round as a cup, and all the king's
horses can't draw it up? Pick up the
correct toy to answer the riddle.'"

The rubber duck, the Tonka truck, the
Barbie doll, the miniature robot, the old
cell phone, the toy well, the wooden
plane, the plastic hammer, and the
```

```
picture book are in the Riddle Room.

Instead of taking the toy well:
   say "Well done! You've solved the
   riddle.";
   end the story saying “Congratulations!
   You have won.”
```

In order to get even a very simple story to work, an author has to think through the narrative elements that teachers often ask their students to consider, such plot and setting. However, Inform 7 has its own unique way of pushing students in the right direction. With respect to setting, for example, Inform 7 insists that the author create at least one location. Inform does not allow a simple example story without an assertion like “The riddle room is a room.” Without such a sentence, Inform will not create a story that “compiles” or runs at all. With respect to plot, Inform requires that the author declare explicitly when an ending of the story has been reached. In our simple story, we need a statement like “end the story.” Otherwise, the story continues forever, in a blatantly unsatisfactory way, without any clear resolution.

In helping their students to write process analysis, or “how to” essays, teachers often stress the need for writers to create a series of clear, articulated steps. Inform 7, to a considerable degree, mandates such steps. In our “Riddle” example, for instance, the story would be utterly incomprehensible with the step that creates the sign and provides its description. Without the step that creates the toy well, the rubber duck, and the other playthings, it would become quickly obvious that the reader would be unable to make the slightest progress. Without such progress, the story really would have no plot at all.

There’s an example of a process-analysis essay that uses Inform in the previous chapter of this book.

Familiar Tools for Writing -- the Outline

Another way to understand Inform 7's usefulness for writers, and for teachers of writing, is to consider some traditional tools for composition and Inform's corresponding utilities. One such tool is the outline. Writers use outlines in various forms for varying purposes. Sometimes outlines are quite formal; at other times, they are “quick and dirty.” In the course of prewriting, some authors make outlines to organize their material before drafting. Other writers favor outlining as a revision technique, used to check a draft’s unity and/or organization. Some use outlining in both ways. Teachers of writing sometimes like outlining because it helps students to understand what we mean by “unity,” “coherence,” and “organization.”

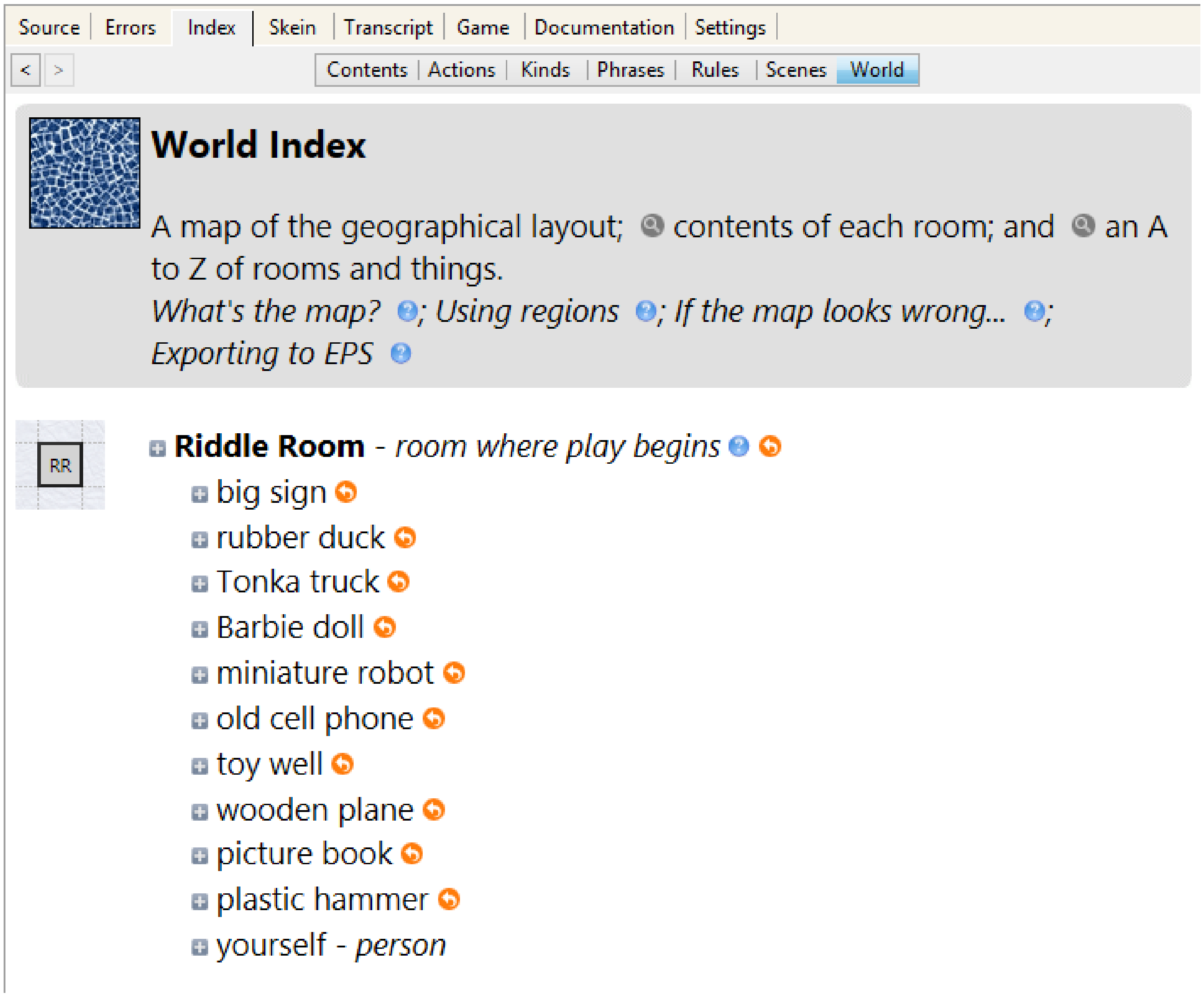
Inform 7 provides several different outlining tools, all of them useful for at least some writers. The most obvious of these allows an author to categorize the divisions of a story in a built-in hierarchical system, which writers usually employ to map a story. The system looks like this:

Volume
  Book
    Part
      Chapter
        Section

Inform lets writers use any, none, or all of these categories, but it rewards authors for using them by keying its error messages to the categories. In other words, when Inform detects an error in an author’s code, it points to the category in which the problem has occurred. Thus, Inform provides student writers with a special incentive to use a type of outlining as a prewriting technique. Of course, many student projects will not require all five of Inform’s category types, but many projects become more manageable through the use of one or two.

Another of Inform’s outlining tools enables the writer to quickly see which of the story’s objects enclose other objects. This tool appears under the Index/World tab of the Inform 7 application. Let’s think of our “Riddle

Room" example to illustrate. When the story begins, the Index/World tab reveals an outline that looks like this:



World Index

A map of the geographical layout; contents of each room; and an A to Z of rooms and things.
What's the map?; *Using regions*; *If the map looks wrong...*; *Exporting to EPS*

Riddle Room - *room where play begins*
- big sign
- rubber duck
- Tonka truck
- Barbie doll
- miniature robot
- old cell phone
- toy well
- wooden plane
- picture book
- plastic hammer
- yourself - *person*

Even in the case of this very simple story, this sort of outline can be useful as the writer revises. If, for example, the author had made the common mistake of creating the object called the "wooden plane" but had forgotten to place it in the Riddle Room, the outline would clearly show the error. Inform 7 updates this outline throughout the testing of the story. If, for example, we ask Inform to generate this outline after the player has picked up the rubber duck, we would get

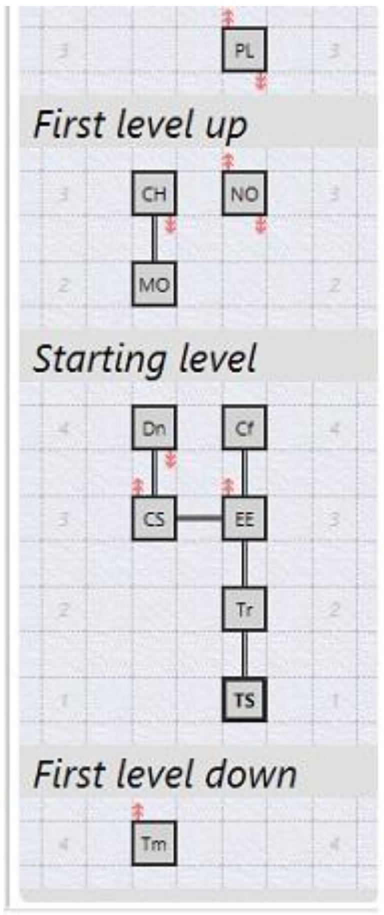
Riddle Room--*room where play begins*

```
    big sign
    Tonka truck
    Barbie doll
    miniature truck
    old cell phone
    toy well
    wooden plane
    picture book
    plastic hammer
    yourself -- person
        rubber duck
```

It's easy to imagine how useful this sort of outline can be, as an author creates more rooms and objects, some of which can contain one another.

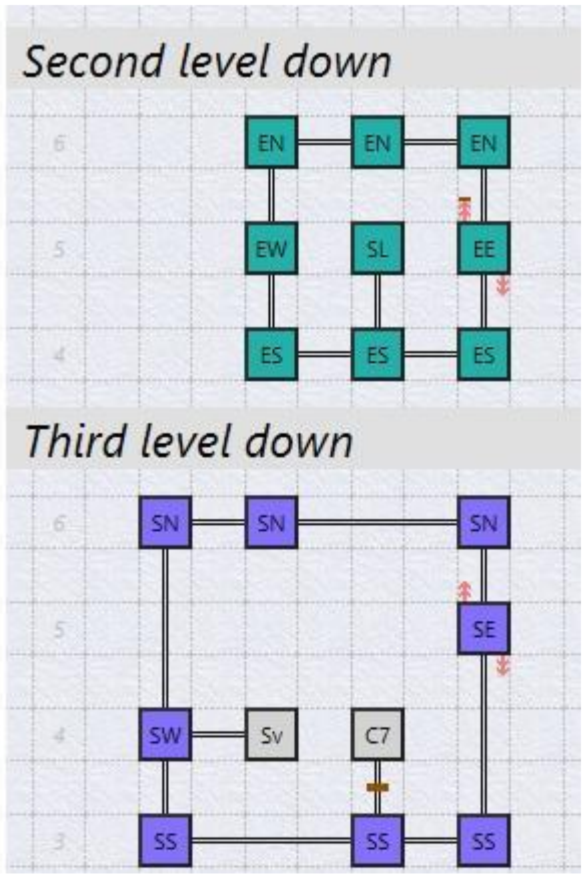
Familiar Tools for Writing -- the Map

Inform 7 offers no facility for creating idea maps, such as those generated by computer programs like Inspiration and FreeMind. However, Inform can help writers to make maps of their stories' settings. Many writers of conventional narratives, and of other works, use such maps as prewriting and revising tools. Mapping is especially helpful in interactive fiction because most works of interactive literature, unlike our ultra-quick “Riddle Room” example, require the reader to move from place to place. Inform 7 automatically creates a map of each of a story’s locations, showing the connections among them. Inform displays these maps in the “Index/World” part of the program. Here’s an instance, which represents, in part, a story that takes place on several floors of a building:



Notice that this map shows not only the locations, with their names abbreviated to two letters, but also the ways in which they are connected and the levels on which they exist. Though the locations in this story are not particularly numerous or complex, this sort of map makes it easy to identify and repair the kinds of problems that frequently occur, as students write interactive stories. The room that appears at the bottom of this map is labeled “Tm” (for “Tomb” in this case). The red arrows at the top of “Tm” indicate that the player can go in the direction “Up” from this location. But suppose, for example, that the author had erroneously created the room abbreviated “Tm” without connecting it to the room above it. The map would immediately show the error.

Inform automatically generates the “levels” that appear in our example map, whenever the author creates a location that is up or down from another location. However, Inform also allows the writer to invent his or her own regions, which can contain various locations. Inform color-codes these regions on its map, as in the following:



This map shows that all of the rooms in the “Second Level Down” (“down” from the starting level, that is) are in the same region as one another. Inform has mapped these rooms in pink. The writer has created a second region to contain most of the rooms in the “Third Level Down.” This region is mapped in purple. Two of the rooms on the “Third Level Down,” however, are not in any region at all, as shown by their white coloration.

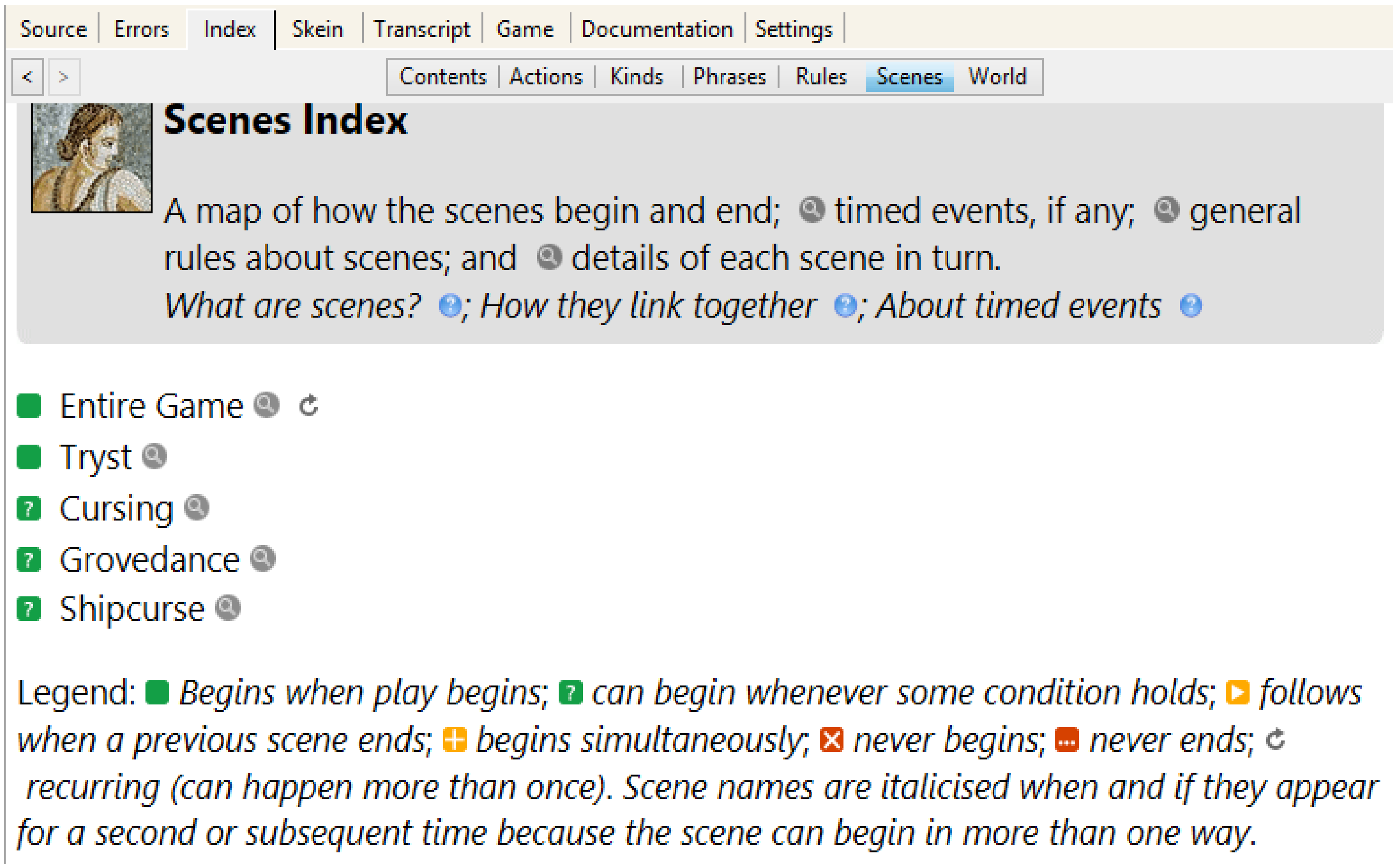
Inform’s mapping facility, then, offers student-writers a friendly mapping tool for their story planning and for their revising and debugging. This mapping system is largely automatic in its handling of levels and connections, but it’s also easy to customize through the use of author-created regions. An Inform 7 map often provides a clear and useful visualization of a story’s physical coherence (or, perhaps, its lack of coherence).

Familiar Tools for Writing -- Storyboards

Writers of stories often make storyboards to plan out the scenes of their narratives. Storyboards often depend largely on pictures, but, of course, they can make good use of text, too. Inform 7 sticks with the words. Inform doesn’t force its authors to formally create scenes, but it makes it easy for them to do so. Once scenes have been created, Inform represents them, using two different tools. The author of an extremely simple story like “The Riddle Room” would probably not employ scenes, but writers of more complex tales almost always use them, often to the extent that managing scenes becomes a major debugging (revising) task.

The Scenes Index

One of Inform’s tools for representing scenes is in the Index/Scenes part of the program. This utility lists all of the story’s scenes and offers useful information about each of them. Let’s say that a writer has created four scenes in a story based on the opera *Dido and Aeneas.* Inform’s Index/Scenes tab might reveal something like this:


Source | Errors | Index | Skein | Transcript | Game | Documentation | Settings
< >
Contents | Actions | Kinds | Phrases | Rules | Scenes | World

Scenes Index

A map of how the scenes begin and end; timed events, if any; general rules about scenes; and details of each scene in turn.

What are scenes?; *How they link together*; *About timed events*

🟩 Entire Game ↻
🟩 Tryst
? Cursing
? Grovedance
? Shipcurse

Legend: 🟩 *Begins when play begins;* ? *can begin whenever some condition holds;* ▶ *follows when a previous scene ends;* ➕ *begins simultaneously;* ✖ *never begins;* … *never ends;* ↻ *recurring (can happen more than once). Scene names are italicised when and if they appear for a second or subsequent time because the scene can begin in more than one way.*

Note that the “Entire Game” counts as a scene, which encompasses all the other scenes. The Index/Scenes tab also provides more detailed information about each scene, as appears here:

```
The Entire Game scene  recurring

The Entire Game scene is built-in. It is
going on whenever play is going on. (It
is recurring so that if the story ends,
but then resumes, it too will end but
then begin again.)
```

Begins when: the story has not ended

Ends when: the story has ended

The Cursing scene

Begins when: the player is in the Lair of the Sorcerer

Ends when: the Magician is triumphant

The Grovedance scene

Begins when: the player is in the Grove and the Magician is triumphant

What happens:

```
   (move Aeneas to the Grove; ...)
```

Ends when: the Magician is prophetic

The Shipcurse scene

```
Begins when: the player is in the Beach
and the Magician is prophetic

What happens:

   (move the enchantresses to the Beach;
...)

Ends when: the Magician is finished

----------------------------------------

The Tryst scene

Begins when: play begins

Ends when: Dido is off-stage

----------------------------------------
```

Here, we can see the events that mark the beginning and end of each scene. If a particular event happens at the beginning of the scene, we can see what that event is. If a scene can happen more than once, we can see that the scene is “recurring.” In our example, each of the scenes will occur only once, except the “Entire Game” scene.

The “Scenes” Debugging Command

When problems with scenes develop during the writing of an interactive story, as they very frequently do, the information in the Scenes Index can be quite helpful, but the “Scenes” debugging command usually works even better. The writer uses this command while testing the story, as it runs within the Inform application. For the most part, this test run of the narrative looks exactly as it would look to a reader, except that the author has some special “debugging” commands that will not be available to readers. Let’s suppose that the writer of our “Dido and Aeneas” story types the command “scenes” as soon as the story starts. The output will look like this:

```
>scenes
Scene 'Entire Game' playing (for 0 mins
now)
Scene 'Tryst' playing (for 0 mins now)
(Scene monitoring now switched on. Type
"scenes off" to switch it off again.)
```

A few turns later, the “scene monitoring” function that the author activated reports

```
[Scene 'Tryst' ends]
```

```
Now, the “scenes” command shows the
following:
```

```
>scenes
Scene 'Entire Game' playing (for 5 mins
now)
Scene 'Tryst' ended
(Scene monitoring now switched on. Type
"scenes off" to switch it off again.)
```

The information that the scene “Tryst” started and ended when it was supposed to is of considerable importance to the writer. When something goes wrong, as it often does, the “scenes” command assumes even greater importance, since it points, quite specifically, toward the problem.

In effect, then, Inform 7 automatically creates a detailed storyboard in a way that helps writers construct a clear and coherent narrative. Thus, Inform offers students a dramatic instance of the value and techniques of effective storyboarding.

Familiar Tools for Writing -- The Word Processor

Most writers prefer to write with a word processor. Interactive fiction authors, though, are often especially appreciative of the text editor that comes with Inform. Like many programmers’ text editors, the Inform utility makes the writer’s work more efficient through a helpful system of color coding. In Inform, instructions to the computer appear in black, and text that is to be printed to the screen is dark blue. Sometimes, Inform programmers include instructions to the computer within printed text. These instructions appear in light blue. The programmer's "comments," which make the source code easier for humans, including the author, to understand, are green. The example that follows may look as if it's contrived just to show off Inform's color system, but, actually, the source code comes from a real IF story and is quite typical.

```
[Rooms]

The Palace Knoll is south of the Path.
The description of the Palace Knoll is "A
secluded green knoll, with a park bench,
near the stately Palace of Dido, the
Queen of Carthage. An overgrown path
leads north."  The description of the
Path is "[if visited]A simple path,
leading toward a dark, craggy area to the
north. You can also follow another path
to the east.[otherwise]You enter a simple
path, branching toward a dark, craggy
area to the north and a lovely forest to
the east; however, immediately after
leaving the Palace Knoll, you are
attacked by a pack of vicious hounds.[end
if]"
```

At the top of this example text, we find a comment, [Rooms], suggesting that the next section of code will implement the various locations of the story. The brackets identify this text as a comment, and Inform marks the comment in green, helping the writer to avoid such common mistakes as failing to close the brackets correctly. The next bit of text is

```
The Palace Knoll is south of the Path.
```

This text appears in black type because it addresses the computer itself (or perhaps it addresses Inform, depending on how you look at the nature of a programming language). This text creates two locations, or “rooms,” called “Palace Knoll” and “Path,” the former located immediately south of the latter. A few words later, we find the description of the Palace Knoll, enclosed in quotation marks and colored dark blue. If the writer had inadvertently ended the description with an apostrophe rather than a closing quotation mark, the color coding would clearly indicate the typo.

The description of the Path is a bit more complicated, since it actually consists of two different descriptions. The descriptions are within quotation marks and are coded in blue. One description, the one that appears first in the text, is the usual description of this location. It is the description that the reader would see if he or she had visited this room before. Thus, this description is marked [if visited], with the bracketed text shown in lighter blue, since the bracketed words do not actually print on the reader's screen. The second description would appear only if the reader had never visited this location before.

Familiar Tools for Writing -- the Usage Checker

Many word processors, including Microsoft Word, include software that checks for various usage errors, such as mistakes in punctuation and sentence structure. These checkers are usually so unreliable that few writers use them. Oddly, these checkers often do not check for errors that they actually could get right consistently, such as unpaired parentheses or quotation marks.

Inform 7, though, like most other programming languages, requires the use of a specialized sort of usage checker called a compiler. The compiler carries out the essential task of translating the source code that writer creates into a form that a computer can read.

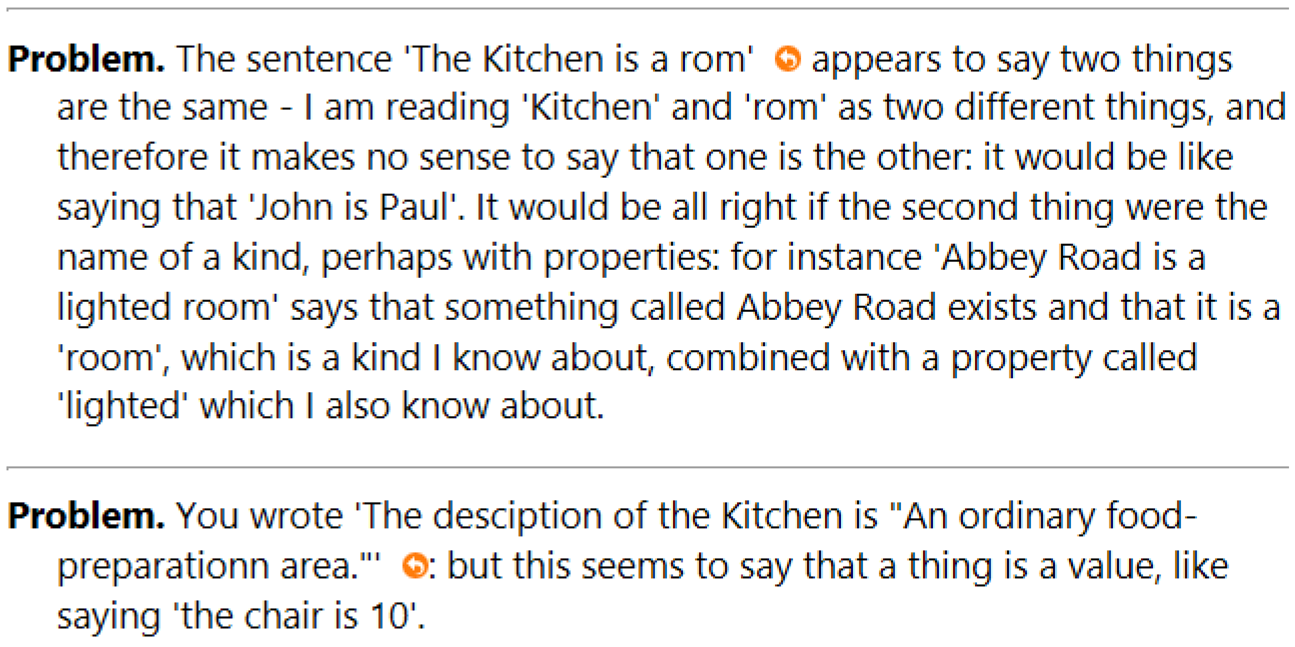
Of course, programmers, like all other writers, make mistakes. In a programming environment, many, though not all, of those mistakes cause the compiler to be unable to do its work. When the compiler finds that it cannot translate source code in a machine-readable form, the compiler tries to point the programmer toward the code that is problematic, often with a line number and a brief, cryptic description of the error.

The Inform 7 compiler is much friendlier than those of most other languages. Since Inform 7 doesn't usually use line numbers, the Inform 7 compiler provides the writer with a link to the erroneous source code. Most of the time, Inform also offers a reasonably well-developed and clear description of the problem it has found in the code.

To illustrate, let's start with a simple of example of problematic source code:

```
   The Kitchen is a rom. The
desciption of the Kitchen is “An
ordinary food-preparationn area.”
```

When we try to compile this code, we get the following result:



Problem. The sentence 'The Kitchen is a rom' appears to say two things are the same - I am reading 'Kitchen' and 'rom' as two different things, and therefore it makes no sense to say that one is the other: it would be like saying that 'John is Paul'. It would be all right if the second thing were the name of a kind, perhaps with properties: for instance 'Abbey Road is a lighted room' says that something called Abbey Road exists and that it is a 'room', which is a kind I know about, combined with a property called 'lighted' which I also know about.

Problem. You wrote 'The desciption of the Kitchen is "An ordinary food-preparationn area."': but this seems to say that a thing is a value, like saying 'the chair is 10'.

The compiler has found two errors in this source text. Clicking on the orange arrows that appear in the problem reports will cause the offended passages to be highlighted in the writer’s source code, showing, with come precision, where the difficulty is.

The compiler first identifies “The Kitchen is a rom” as problematic. Inform is not smart enough to guess that “rom” is just a misspelling of “room,” but it does provide an error message that would help most writers. In fact, Inform offers a highly relevant example of good code, “Abbey Road is a lighted room.”

Then the compiler finds a second mistake, “The desciption of the Kitchen is ‘An ordinary food-preparationn area.'" Once again, Inform does not realize that the problem is really a spelling error ("desciption"). This time, though, Inform's description of the error, though perfectly accurate, would be much less helpful to a novice writer."

Notice that the compiler does not find a third spelling error, "food-preparationn." Because this misspelling occurs in text that is to be printed to the screen, the compiler does not try to understand the word "food-preparationn," nor does it check the word's spelling. Instead, the compiler assumes that its job is simply to see that the text appears for the reader, exactly as the writer has typed it in. Inform does have its own spell checker, which works separately form the compiler. This spell checker would have found all three errors, though most writers run the compiler more often than the spell checker and so would have found the first two errors more quickly, when the compiler pointed them out.

Of course, many coding errors are not just spelling mistakes or typographical errors. The more advanced errors offer writers an opportunity, and some motivation, to check into Inform's well-indexed documentation, or to ask for help from other authors. In any case, the compiler enforces a certain care and precision in any writing of Inform code.

Familiar Tools for Writing -- Transitional Words and Phrases

Teachers of writing often suggest to students that they check the coherence of their writing. One tool for establishing coherence is the transitional word or phrase, such as “however,” or “on the other hand.” Most writing texts offer lists of commonly used phrases of this sort, but writing teachers usually put just as much stress on the use of transitional devices that arise more spontaneously in particular pieces of writing. For instance, in order to improve coherence, a writer may use a technical term, such as “compiler,” and judiciously repeat this word and its forms.

In interactive fiction, programming structures called “values” provide great transitional power. In fact, writers sometimes think of Inform’s values as ways for one part of an Inform story’s code to send “messages” to another part. Suppose, for example, that, in one part of an interactive story, the main character may or may not drink a triple shot of whiskey. Suppose, further, that, an hour later in game time, the character may or may not decide to drive a car. The Inform author will likely use a value to send, to the later part of the story, a message about whether the character imbibed heavily earlier in the game.

Inform tries to make the use of values both simple and powerful. In order to do so, it enables the writer to set up values in more than one way. Perhaps the most powerful way to implement a value is to create it as a noun, along with adjectives that are related to the noun. Here’s an example:

```
Sobriety is a kind of value. People
have sobriety. The sobrieties are
drunk and sober. A person is usually
sober.
```

This source text enables Inform to understand “drunk” and “sober” as adjectives that may or may not apply to any person in the story, including the “player/character,” the character that the reader controls. The phrase “A person is usually sober,” in the source text as we have written it, establishes the starting condition for this value. In other words, the player/character, and all other characters, start out sober, unless the writer explicitly declares otherwise. However, events in the story may cause the player’s condition to change.

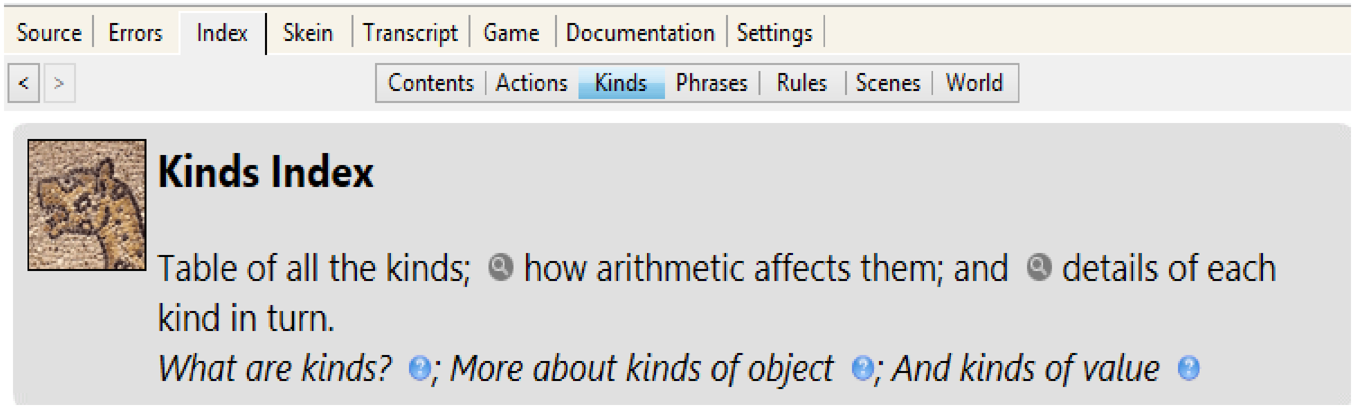
Now that we have created the adjective “drunk,” the phrase “when the player is drunk,” is a condition to which Inform can respond in whatever way the author deems appropriate. For instance, the writer could notify the reader of the player/character’s sobriety after every turn of the story, using the following source code:

```
Every turn when the player is sober,
say “You are sober.”
```

Once the sobriety value is established, the writer can change its state. For example, the following code would produce a change when the player/character imbibes.

```
The player carries some whiskey.

Instead of drinking the whiskey: say
“You drink the whiskey”; now the
player is drunk.
```

The use of values in interactive fiction is a bit abstract and elusive for many new writers. However, Inform 7 offers two tools to help. One of these is a listing of values which, like our “sobriety” example, are set out as “types of value.” This listing appears in the Index/Kinds tab of the program. For our “sobriety” example, the relevant section of the Index/Kinds tab looks like the following:



Kinds Index

Table of all the kinds; how arithmetic affects them; and details of each kind in turn.

What are kinds?; More about kinds of object; And kinds of value

Value	Default Value
Sobriety[2]	sober

Inform also offers a debugging command that an author can use while testing a story. This command, “showme,” offers a listing of information that includes the state of relevant values. In our example, “showme me” yields this information about the player/character, if he or she has not consumed the whiskey:

```
>showme me
yourself - person
    whiskey
location: Kitchen
singular-named, proper-named; unlit,
inedible, portable; male
printed name: "yourself"
printed plural name: "people"
indefinite article: none
description: "As good-looking as
```

```
ever."
initial appearance: none
carrying capacity: 100
sobriety: sober
```

After the the player/character imbibes, we’d get the following:

```
>showme me
yourself - person
location: Kitchen
singular-named, proper-named; unlit,
inedible, portable; male
printed name: "yourself"
printed plural name: "people"
indefinite article: none
description: "As good-looking as
ever."
initial appearance: none
carrying capacity: 100
sobriety: drunk
```

For almost all inexperienced writers, it’s difficult to create a strong sense of coherence in any sort of writing. For writers of interactive fiction, values can help to develop good coherence, but only if they are implemented correctly. By using Inform’s tools for creating well-thought-out values, novice writers can sharpen their understanding of just what it takes to write coherently. Indeed, writers of interactive fiction really **have** to write coherently, using values and other techniques, if they are to produce works that look like IF stories at all. In its own way, Inform 7 enforces coherence.

Familiar Tools for Writing -- Dictionaries and Thesauruses

Like all writers, IF authors have to choose the right words. For this purpose, they have to find words that are potentially “right.” However, in using a programming language, the notion of “right word” takes on a whole new meaning, because the language understands only a small subset the words from any natural language, such as English. Inform offers the writer two important tools for identifying words that the language can understand. One of these tools focuses on what Inform calls “actions,” and the other offers information about “phrases.”

A Thesaurus of Actions

In Inform, the following code will compile and work:

```
Instead of taking the anvil, say
```

```
“That’s too heavy to lift.”
```

But this code will not compile:

```
Instead of picking up the anvil, say
“That’s too heavy to lift.”
```

The problem centers on the way Inform handles actions that the reader can invoke.

Inform 7, in its native form, allows **readers** to use a total of 143 verbs. “Take” is one of those verbs, and so is “pick up.” However, Inform 7, in its native form, understands only eighty different actions. The sixty-three verbs that are not actions serve as synonyms that the reader can use. The **writer**, though, must use the single unique word that Inform recognizes as an action. Let’s look at our example rule once again:

```
Instead of taking the anvil, say
“That’s too heavy to lift.”
```

If the writer has implemented the anvil, this rule will compile and work. In a story that uses this rule, the player can type “Take the anvil,” and get the response, “That’s too heavy to lift.” The player can also type “Pick up the anvil,” and get the same response. If the player types, “Grab the anvil,” the response will be “That’s not a verb I recognize.”

If a writer wants to allow new synonyms for an action, he or she can do so quite easily. To add the “grab” synonym for “take,” the writer would add the following code:

```
Understand “grab [something]” as
taking.
```

Implementing new actions is a bit more complicated than creating synonyms, but Inform writers can make as many new actions as they like, too.

Inform 7 provides several tools to help readers with its specialized vocabulary. One of these is accessed though Inform’s Index/Actions tab. This tool lists all of the actions that Inform understands, including any new actions that the author has created, along with the synonyms of each action.

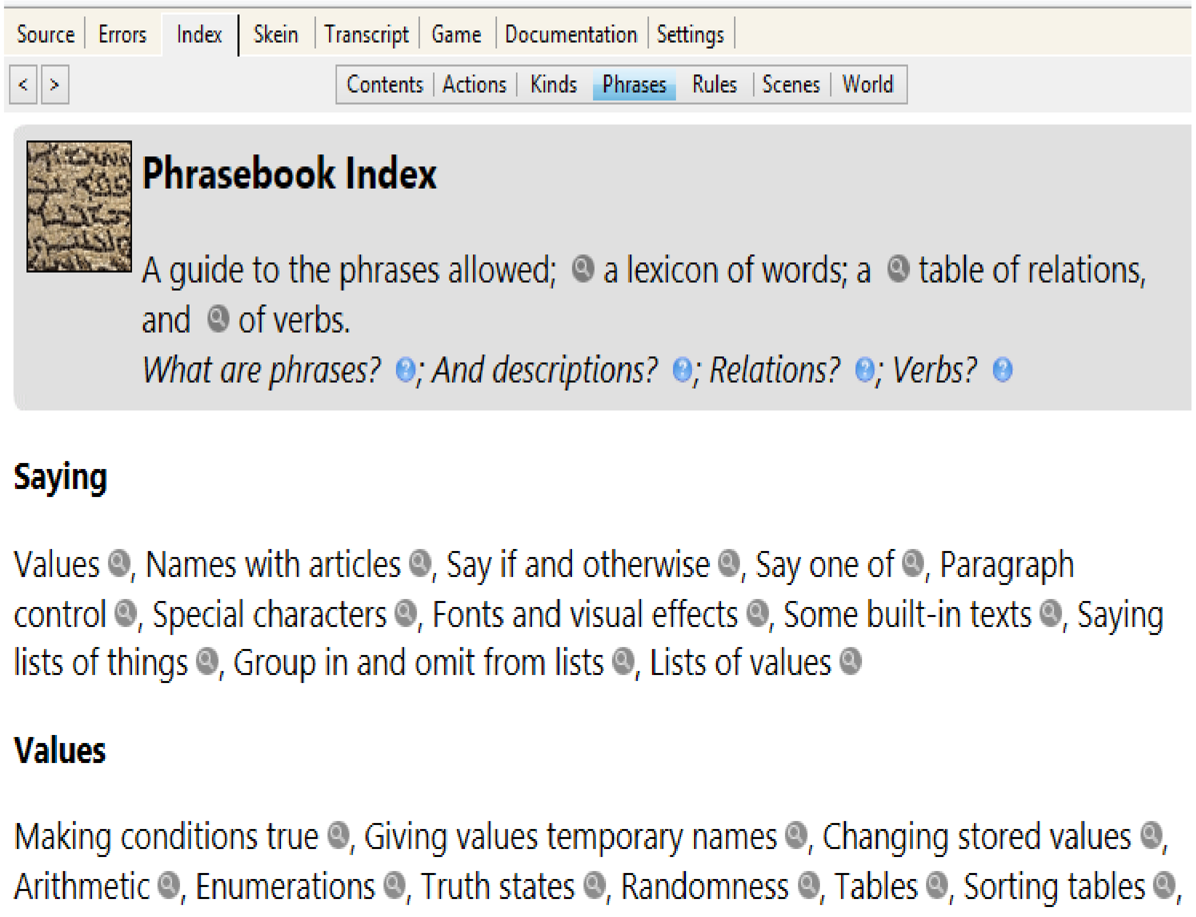
Inform 7 also offers an “actions” debugging command, which an author can use while playing through a story. With the many ways in which actions and their synonyms can interact, this command can help a writer to sort through what’s really happening behind the scenes of an IF work. The “actions” command can prove especially helpful when a story is not behaving the way the author expects.

A Thesaurus of Phrases

Of course, not all problems with Inform 7 vocabulary involve actions. Consider another bit of code that will not compile:

```
Rather than taking the anvil: say
“That’s too heavy to lift.”
```

In this instance, Inform’s rigid preference for “instead of” over “rather than” does not hinge on any particular action. To sort out this sort of vocabulary problem, Inform offers its Index/Phrasebook tab. The tab offers rather long list of phrase types, each followed by a magnifying glass icon that links of an explanation. Here’s a look at part of what this tab offers.



Phrasebook Index

A guide to the phrases allowed; a lexicon of words; a table of relations, and of verbs.

What are phrases?; And descriptions?; Relations?; Verbs?

Saying

Values, Names with articles, Say if and otherwise, Say one of, Paragraph control, Special characters, Fonts and visual effects, Some built-in texts, Saying lists of things, Group in and omit from lists, Lists of values

Values

Making conditions true, Giving values temporary names, Changing stored values, Arithmetic, Enumerations, Truth states, Randomness, Tables, Sorting tables,

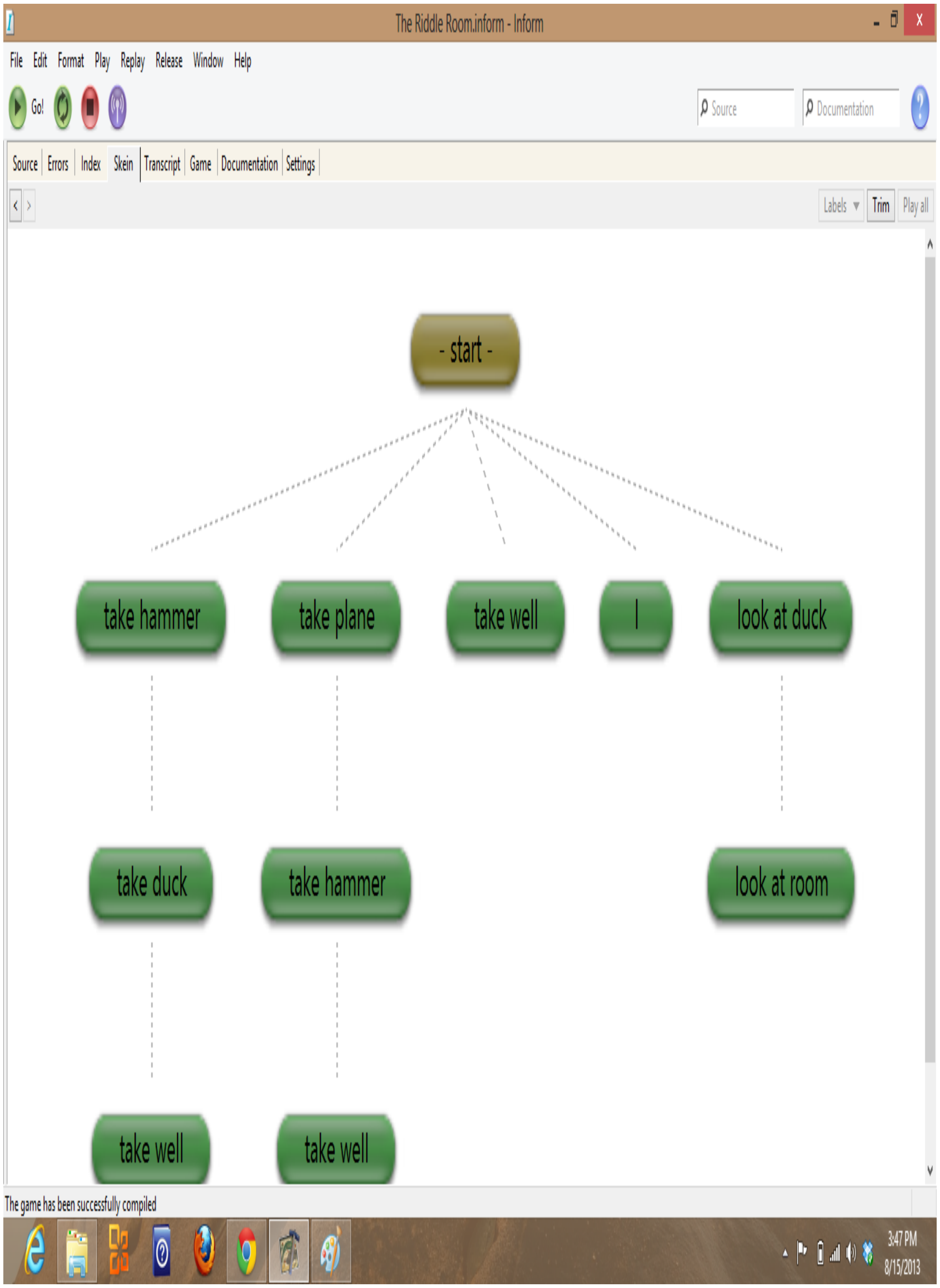
Using Inform's Thesauruses

A good dictionary or thesaurus can help students to use words more clearly and precisely. In its own, demanding way, however, Inform goes much farther. In order to get a story to compile and run, a student author of an Inform story must use the vocabulary of action words and function words precisely. Sooner or later (usually sooner), Inform writers develop a considerable facility with the tools that help them to achieve this kind of precision.

Familiar Tools for Writing: Rereading to Revise and Edit

Student writers usually learn a variety of techniques for rereading their work to spot likely sites for improvement. Some become experts at rereading once for unity, a second time for coherence, and a third time for clarity. Some use mnemonics like CUPS (capitalizing, usage, punctuation, spelling) to help them with their editing. Many learn to read the work aloud in order to “hear” problems. Writers of interactive fiction face a particularly time-intensive task when they seek to reread a story, since any particular reading requires the inputting of a series of commands that the reader might type. As a story becomes more complex, this list of commands can become very lengthy.

Inform 7 addresses this issue with a tool called the “skein,” which records all the input that the writer uses in testing a story and allows for the nearly-instantaneous playback of that input. The skein is so important that it has its own tab in the Inform application. Let’s take a look at a skein that represents five iterations of our “Riddle Room” story.



This skein shows three run-throughs of the story that reach its conclusion, which occurs with the taking of the well. However, the skein recalls all the iterations the author has tried, including, in this case, two iterations that did not reach a conclusive result. Let’s suppose that an author has made some changes to the code that involves the hammer and wants to see the result. Instead of inputting a series of commands by hand, the writer can open the skein and double-click on any of the skein’s nodes, thus causing the story to run itself up to that point. In this case, the author might click on the “take hammer” node. Then, the author could examine the output to see that her changes to the code produced the desired result.

As one might imagine, the skein can be handy in revising even a very simple story like our “Riddle Room” example. For revising a more typical story that requires fifty or one hundred commands to reach its conclusion, the skein is invaluable.

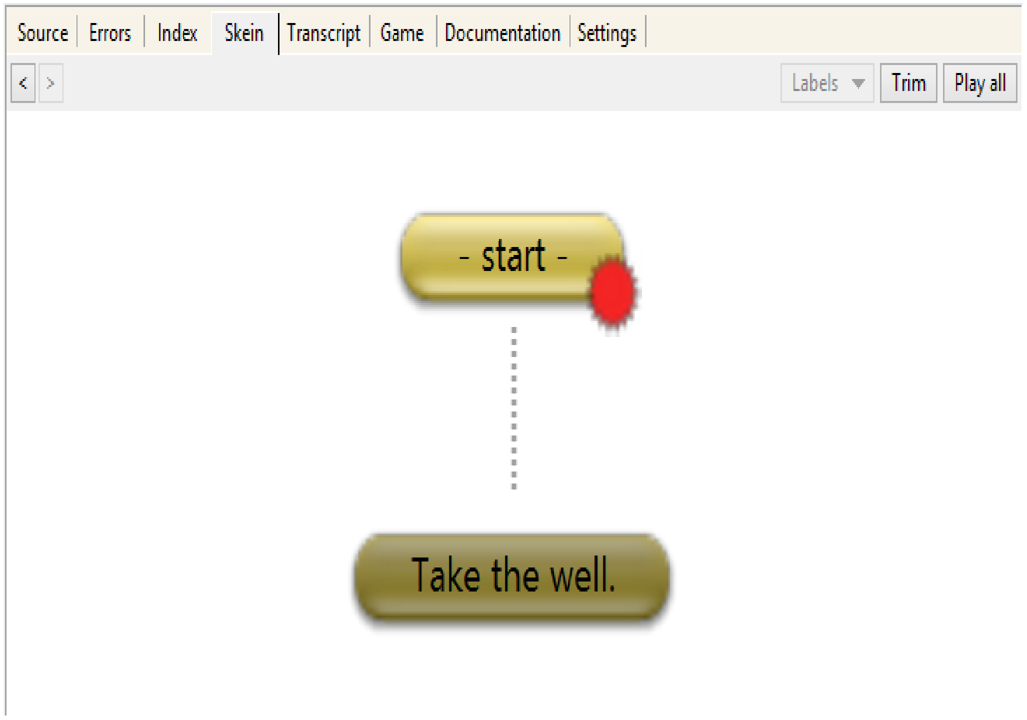
The skein offers an unusual pedagogical advantage for writing teachers in that it emphasizes an aspect of revising that students often overlook. When students make significant revisions, as they often should, students frequently fail to identify the impacts of their revisions on later sections of their essays. Even the most superficial experimenting with the skein shows how a change in one part of a piece of writing can have unanticipated effects on the rest of the text.

Familiar Tools for Writing: Side-by-Side Comparisons of Drafts

When students get into difficult tangles in revising their essays, they often find themselves wondering whether an earlier draft is actually better than a revised version. Comparing the two, side by side, can help to resolve this sort of conundrum.

Inform 7, in its Windows and MacOS versions, offers writers an extension of the skein tool that can help them to make intelligent comparisons of drafts. This tool, which does not appear in the Linux version of Inform, is called the transcript. Like the skein, the transcript has its own tab in a typical Inform window.

If a writer right-clicks on a node in any Inform skein, the resulting options will include “Show in transcript.” This option, at first, simply shows the output of the chosen skein, up to the chosen node, in the “Transcript” tab. For example, consider the screen shot of a skein that appears below. It’s from “The Riddle Room.”



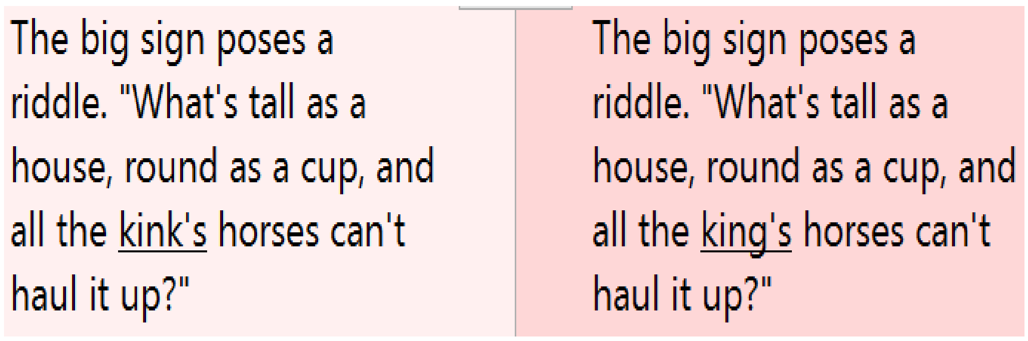
This skein would lead to the successful completion of the story. If the writer wanted to view the output of this skein in the Inform transcript, he or she would simply right-click on any node of the skein and choose the option “Show in transcript.” Now, the transcript tab would show two columns, one of which would show with the output of the skein that the author chose. In our case, this output would be as follows:

```
   Pick up the toy that solves the
riddle.

   The Riddle Room
   An Interactive Fiction by Brendan
Desilets
   Release 1 / Serial number 130816 /
```

```
Inform 7 build 6G60 (I6/v6.32 lib
6/12N) SD

   Riddle Room
   This is a room with a large sign
and a bunch of toys scattered around.

   The big sign poses a riddle.
"What's tall as a house, round as a
cup, and all the king's horses can't
haul it up?"

   You can also see a rubber duck, a
Tonka truck, a Barbie doll, a
miniature robot, an old cell phone, a
toy well, a wooden plane, a picture
book and a plastic hammer here.

   >Take the well.
   Congratulations! You've solved the
riddle.
```

In addition to showing the writer the output of this run-through of the story, the transcript offers a series of clickable buttons that let the author “bless” the skein up to any particular node. Typically, an author chooses to bless a “good” skein, a skein that represents a desirable sequence of moves that produces a useful result.

Now, let’s suppose that the author continues to work on the story, and, in the course of doing so, inadvertently changes a word in the description of the sign from “king’s” to “kink’s.” With this error in place, if the author goes to the skein and double-clicks on the “Take the well” node, the transcript will compare the new version of the story with the blessed skein, underlining all the differences between the two. The transcript would look, in part, like this:



The big sign poses a riddle. "What's tall as a house, round as a cup, and all the kink's horses can't haul it up?"	The big sign poses a riddle. "What's tall as a house, round as a cup, and all the king's horses can't haul it up?"

The highlighting of this typo could, of course, be useful to a writer, but, in a longer, more complex story, the transcript can be much more important. Typically, in Inform, as in other programming languages, a single erroneous revision in source code can result in dozens of embarrassing errors in the program’s output.

When used with a story of any significant length, the Inform 7 transcript is likely to reveal a significant, and sometimes surprising, list of differences between versions of a tale. For this reason, the transcript often underscores the story-wide implications of revision in a way that a convention comparison of drafts cannot.

Do Students Actually Use These Inform 7 Tools?

In a programming class, a teacher might well assign the use of various tools that a language offers, tools like Inform 7’s skein. However, in a typical writing class, an instructor would probably not do so. Instead the professor would likely work with two kinds of assignments, short easily-programmed exercises for all students, and optional, more ambitious projects for the ten percent of students who like an unusual challenge. In response to the first assignment, students might, after an hour or so of instruction, produce very simple stories like “The Riddle Room,” which do not require the use of many of the tools described here. The tools that students must use in even the simplest stories are still significant, however. These include the color-coded text editor (corresponding to a word processor) and the compiler (usage checker). Even in very quick projects, most students will also use the skein (reading for revising) and the actions tab (thesaurus).

Students who take on longer stories will almost always use more of Inform 7’s tools. Of course, the writers’ choice of implements will depend partly on the nature of their stories. Stories that take place in one room, for example, won’t make many demands on the mapping utility. Nearly all authors of longer stories, on the other hand, will use values and scenes.

Chapter 10 – The Writer's Self in Interactive Fiction

Yet Another Sample of Interactive Fiction

Here is a transcript of a session with a simple instance interactive fiction. The story will prove useful, later on, when we look at a surprising sort of difficulty that students have in crafting IF and in other kinds of writing. The reader's input, which would vary with each reading of the story, appears in boldface type. (A playable version of the story is available at http://iplayif.com/?story=http%3A//bdesilets.com/if/Trolley.zblorb)

```
It's another routine day of trolley
driving, in the neighborhood of your
local university.  About two hundred
yards ahead, you can see that the trolley
track splits, one branch, which you're
planning use, leading straight ahead, and
the other offering a turn to the left.
You have just one passenger in the car,
an elderly woman, and she signals that
she'd like to get off here.

Trolley
An Interactive Fiction by Brendan
Desilets
```

```
Release 1 / Serial number 121221 / Inform
7 build 6G60 (I6/v6.32 lib 6/12N)

Medford Crossing (in the Trolley)
A simple trolley stop in the suburban
community called Medford.

In the Trolley you can see an elderly
woman and a control panel (on which is a
joystick).

>Look at the woman.
A long-faced, elderly lady, perhaps
around ninety years of age. She is a
passenger on the trolley that you're
driving, and she seems to be wearing a
name tag.

>Look at the tag.
A name tag from a philosophy conference
at the nearby university. The tag reads
"Philippa Foot."

In response to the elderly woman's
request, you stop the trolley.

>Look at the joystick.
A familiar control device for your
trolley. You can steer to the east or
```

west with the device, (by simply typing "E" or "W" in this story) or use it to start the vehicle (by pushing the stick) or to stop (by pulling the joystick.)

The elderly woman gives you an odd look, perhaps an expression of pity or concern, and gets off the trolley. As you restart the trolley and it gains speed in its typical, brisk way, you notice that the controls feel a little odd.

You test the brake and find that it fails! However, a check of the steering mechanism seems to suggest that it's working.

Looking ahead, you are shocked to discover that five people have wandered onto the track ahead, where your runaway vehicle is sure to hit them in a few seconds. A glance at the track that diverges to the west indicates that things aren't much better in that direction. There's a pedestrian on the track there, too. In the next few seconds, you're going to have to decide whether to simply wait, thus letting the car go straight, or steer west.

>**Honk the horn.**
It seems that your signaling device has failed, along with the trolley’s brakes. You can still steer, though.

>**Go west.**
You find yourself on a stretch of track just west of Medford Crossing. Your runaway trolley has just rolled to a stop after colliding with an innocent bystander.

After the runaway trolley incident, you take a desk job with the local transit authority for a time. Today, three days after the accident, you find yourself approaching a footbridge as you make your way home from work. The bridge crosses a trolley track. You see that a very large man is standing on the bridge, leaning over a rather low railing.

As you near the bridge, you notice that only one other person seems to be around. She's a light-haired, bespectacled, older woman, wearing a name tag issued by a local philosophy conference. Apparently, her name is Judith Jarvis Thomson. She seems a rather anxious to get across the bridge and out of its vicinity, and she soon disappears around a corner.

Bridge
A simple footbridge, spanning the trolley tracks near Medford Crossing.

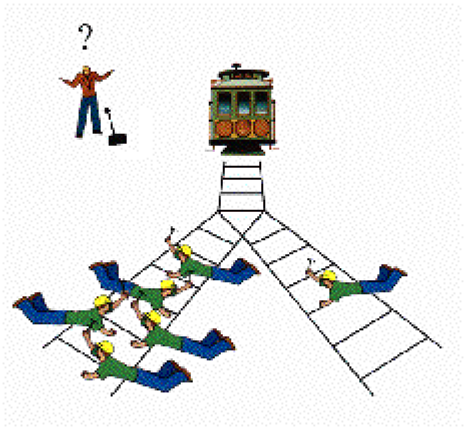
You can see a Big Man here.

>**Look at the man.**
An unusually large person, perhaps six feet eight inches tall, weighing around 400 pounds. He is leaning precariously over the railing of a footbridge, just above a trolley track.

You soon realize why the big man is leaning so awkwardly. He's observing yet another runaway trolley scene. This time, the out-of-control vehicle is headed toward five unsuspecting workers who are on its track.

It occurs to you that, if the big man were to fall off the bridge, he would land in front of the trolley and his bulk would probably stop its progress. If you were to push him, he would surely fall

```
from the bridge.

>Wait.
The big man regains his balance, and the
two of you watch the sad events that
unfold below you.


*** The thought experiment has ended. ***



Would you like to RESTART, RESTORE a
saved game, QUIT or UNDO the last
command?
```



The Trolley Problems

Our sample IF story follows two variations of a famous thought experiment, originally proposed by the British virtue ethicist Philippa Foot and much elaborated by the American philosopher Judith Jarvis Thomson. In its original version, with the listener in the role of the trolley driver, the problem leads a large majority of respondents to choose to turn the runaway vehicle, killing one innocent person rather than five. In the "Big Man" variation, a similarly-large majority choose not to sacrifice the one to save the many, usually because, in order to save the five, the respondent would have to actively and voluntarily murder the one. For many listeners, the “Big Man” version is by far the more disturbing of the two.

How do these trolley problems apply to the student writer of interactive fiction? In order to see the connection, we should first consider the nature of an IF authoring system and then look at the various “selves” of a student author.

Inform 7

An authoring system is a programming language that writers use to create their interactive stories. In most cases, at least during the last half decade, especially in universities, that authoring system is often a highly innovative tool called Inform 7.

Inform 7 produces, as its output, an interactive story that most speakers and readers of English (or any of several other supported languages) can read. In that respect, Inform 7 is just like other IF authoring systems. However, unlike other authoring systems, Inform 7 allows the writer to create his or her story in more or less plain natural-language sentences, not in arcane computer code.

Suppose that a student wants to create a room for use in an interactive fiction story. Using a conventional authoring system, the writer might create source code that looks like this:

```
Object  fac_cafe "The Faculty Dining
Room"
   with  description
         "This is a smaller version
         of the student cafeteria,
         containing the expected
         appointments for an eating
         space for teachers. A door on
         the east wall leads outside the
         building, where you're not
         supposed to be during the school
         day.",
       n_to café;
```

With Inform 7, the source code would look more like natural language:

```
The Faculty Dining Room is south of the
Café. The description of the Faculty
Dining Room is "This is a smaller version
of the student cafeteria, containing the
expected appointments for an eating space
for teachers. A door on the east wall
leads outside the building, where you're
not supposed to be during the school
day."
```

Because of its natural-language syntax, Inform 7 is much easier for students to learn than conventional programming languages. In addition, the "source text" that the student writes to create his story can take the form of a readable process-analysis essay, just the sort of expository assignment that many university instructors favor. And, less obviously, Inform 7 creates an environment in which writing an interactive story is quite closely analogous to reading one. For this reason, writing a work of interactive fiction with Inform 7 can actually help a novice to become a better reader of electronic literature.

Self and the Student Writer

For the purposes of this discussion, we need only a rudimentary understanding of the “selves” of a student writer. Let's call the actual writer the "first self." Of course, we might offer a more complex analysis of the actual author and the ways in which she might think of herself in an academic context, but, for now, let's think of the first self as an ordinary, individual person.

In creating works of fiction, the real writer (or "first self") creates characters, or "second selves." Some of these selves may be very similar to the first self, and others may be strikingly different. In the case of a narrating character who seems similar to the actual writer, readers will occasionally confuse the narrator with the author.

Works of interactive fiction feature a unique type of character, the character whom the reader controls. This very active individual is usually called the "player/character." In our "Trolley" story, the player/character is a driver who runs into some severe mechanical difficulties with his vehicle, moves into a desk job, and eventually faces a difficult decision concerning a big man on a bridge.

The player/character, who is usually referred to as "you" in an IF story, represents some unusual challenges for a writer. Perhaps the most compelling of these is that, in interactive fiction, the fundamental thrust of the genre causes the reader to *conflate herself with the player/character*. This uniquely tight identification of the reader with a character gives birth to the “third self” of the IF author. This “third self” is the reader. In interactive fiction, the reader does not merely relate to the player/character. The reader “runs” the player/character in an intensely intimate way and thus takes on a sense of responsibility for what the character does.

Because of this extreme identification of the reader with the player/character, the writer of an interactive story must be thoughtful of the reader in a very unusual way. The writer must place herself in the moral position of the reader, as the reader plays the part of the player/character. The writer must think of the reader as a “third self.”

In interactive fiction, the reader finds herself in a position similar to that of the decision-maker in the trolley problem. She has to make choices, choices which, by the author's design, may be morally difficult ones. In crafting these choices, an IF writer must exert great care to be “fair” to the reader in ways that go beyond the demands of clarity and coherence that other writers must meet. Readers of interactive fiction have a right to expect such consistent and caring treatment from IF authors. Indeed, one of the foundational documents of interactive literature is the Graham Nelson’s “Bill of Player's Rights,” including the right to “reasonable freedom of action” the right “not to depend too much on luck.”

To a novice writer of interactive fiction, this need to identify with and care for the reader as if the reader were a “third self” is not at all obvious. In truth, it often seems that new IF authors are *trying* to offend the reader, as they struggle to come up with new twists on old problems. For example, if a newby writer were to re-create the “big man” version of our trolley story, she might allow the player/character to push the giant off the bridge, only to have his body collide with the trolley and be propelled through the air, killing the bystanders.

A similar instance, based on an actual student story, involves a Space-Viking player/character who, along with a companion named Uther, is invading a monastery on a distant planet. In the interest of privacy, we’ll withhold the student’s identity here and alter the story quite a lot, even though the author has posted the piece on the Web. Here, in essence, is how one scene goes.

```
You can see a platinum book, an iridium
candlestick, an old monk, a young priest
and an apprentice priestess here.
```

`>`**`Take the book`**` (a likely choice by the reader).`

```
The pages of the book are useless to you,
```

but the cover appears to be of great value. You tear the pages from it, and throw them to the floor. An unarmed druid suddenly appears from behind the altar and runs toward you. He is very angry, and is yelling at you in a language you don’t understand. You and Uther turn continue your plundering, but the druid starts to plead with you, perhaps for the platinum cover.

You put the cover into your antigravity sack. The priest tries to grab at the bag, and as he does, Uther activates his lumensabre and slices off the priest's hand. The priest screams and falls to the floor.

Here, the writer fails to allow his reader “reasonable freedom of action,” in that, if the player/character takes the somewhat innocent step of picking up the book, he causes the priest to suffer great harm. Similarly, if the player/character chooses to try to protect the priestess by picking her up to carry her away from the carnage in the monastery, the story declares that player/character has impressed her into sexual slavery.

Three Problems

Why would an IF writer treat the reader so shabbily? There are at least three reasons. The first centers on the novice writer's inexperience in reading interactive fiction. The second involves the newby's difficulty in creating, or “implementing” the reader's options. And the third hinges on a misunderstanding of an accomplished IF writer's intentions.

Getting to Know the Third Self

First, the novice writer probably has little experience in reading interactive fiction, and the experience she does have is probably with more accomplished works of IF. She, in all likelihood, has never experienced the dismay of an interactor who issues a simple directive that results in an unpredictably complex result. In the case of a the new writer, the “first self” may not know very much about what it’s like to be the “third self.”

Implementation: “Hard Writing” in Interactive Fiction

A second reason for mistreatment of the third self is a special case of the familiar adage, “easy writing makes hard reading.” In order to give the interactor “reasonable freedom of action” in our Viking story, the writer would have to allow the reader to consider taking the book, and, whether he takes the tome or not, to make other decisions about how to deal with the pleading priest and the other characters. But giving the reader this sort of freedom would require far more difficult thinking and writing than what the author of the story has actually done. Instead of providing a single response for “take the book,” the author would have to consider a series of questions and provide programming to account for each of them. These questions would include, at least:

“What happens if the player decides to act in a way inconsistent with Viking marauding, perhaps by not taking the book?”
“What happens if the player/character takes the book but doesn't want to let his friend harm the priest?”
“What happens if the player/character wants to kill the priest, and/or all the other helpless victims?”
“What happens if the player decides to wait for a turn or two, doing nothing at all in the midst of all this chaos?”
“If the player/character's companion attacks the unarmed priest, will the player/character take action against his friend?”
“If the scene continues for several turns, what will the young priest, the old friar, and the young nun do?”

The writer, then, has a great deal of programming, or “implementing” to do, if she is to treat the reader thoughtfully. In truth, an experienced IF writer might choose a more radical solution here, concluding, perhaps, that a Viking warrior really has too many options in this scene. The author might decide to introduce fewer characters, or perhaps to use a different player/character, such as a servant of the Viking fighter, whose options would be fewer and so might require more nuanced problem-solving.

Misunderstanding the Parser

A third reason for an inexperienced author's inconsiderate treatment of the reader is a false assumption about the relationship between the reader and the writer. Interactive fiction, like video gaming and other forms of interactive storytelling, is an inherently challenging genre. Because, in a typical interactive story, a reader experiences a certain level of difficulty and frustration, the interactor may conclude that the author is deliberately taunting him, even when the writer is skillfully easing the way through the tale.

Some of the novice reader's frustration will usually involve the inability of the story's parser to interpret some of his input. For example, if the reader types, “I want to take the book,” rather than “take the book,” an IF story will typically respond, not very helpfully, “I only understood you as far as wanting to take inventory.” This odd failure to communicate results from the limitations of the story's parser and from the reader's misunderstanding of the kind of sentence she should be using. In order to avoid this sort of problem, most IF writers provide some instruction for new readers, especially on the sentences that the story will likely understand. However, many (perhaps most) new readers assume, wrongly, that an odd response from the parser results from a deliberate choice of the author. As IF author Andrew Plotkin has pointed out, they think that the writer has created this difficulty as a particularly annoying problem for the reader to solve.

An IF novice, then, will often conclude that there exists a natural enmity between the reader and writer in interactive fiction. Acting on this belief, the student writer may feel little inclination to accord the reader the status of a "third self." In fact, the newby writer often shows little or no respect for the rights and wishes of the interactor and may actually taunt the reader at times. One student writer had his story respond to the reader's forgetting his car keys with the quip, "This isn't Grand Theft Auto!"

Overcoming the Problems

Of course, with a modicum of good instruction and some constructive practice, students can substantially overcome these "third self" problems and create enjoyable IF stories. For students who are having difficulty because they lack experience with the IF genre, the obvious solution is for them to read more interactive fiction. In particular, they may benefit from some exposure to less accomplished stories, of which many appear on the Web. Christopher Fee of Gettysburg College offers a large collection of student-written stories, some quite sophisticated and others less skillful. His website is at http://public.gettysburg.edu/~cfee/courses/English4012001/topic3.htm#Playing

Frequently, the reactions of other students can help, too, especially when students are having trouble with “hard writing” problems related to implementing clear and fair options for the reader. Real readers, even novice readers, can often spot options that really should work for the interactor, but just don't. And even the most skillful IF writers invariably value the suggestions of thorough readers, or “beta testers,” in identifying points for revision.

Responding to the needs of real readers can also help new IF writers to see that more experienced authors really are trying to create stories for readers to enjoy, despite the challenges inherent in the medium. As little as an hour or so of massaging the parser to make it more reader-friendly will convinces almost any student writer that her relationship with the reader really should be a friendly one.

Back to the Conventional Essay

Writing interactive fiction, then, can help, or perhaps even require, student writers to adopt an unusually active and thoughtful stance toward their readers. And, with a little encouragement, such student writers of IF can use their newly-minted "third self" sensitivity in conventional academic writing, to the benefit of all of their readers.

Chapter 11 – Recommended Stories

New works of interactive fiction appear almost weekly, and many are suitable for use with students aged eleven through eighteen. In this chapter, we'll offer

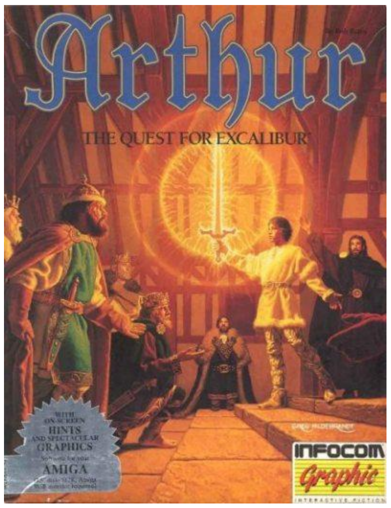
a ”Top Seventy” list and subsequent chapters will present information on teaching with six particular stories, *Arthur, The Firebird, Photopia, Winter Wonderland, The Enterprise Incidents,* and *Lost Pig.*

As you might expect, there are many more stories that are good for use with older students -- college undergraduates, for example. Among these are *Violet* by Jeremy Freese (2008), *Anchorhead* by Micheal Gentry (1998), *The King of Threads and Patches* by Jimmy Maher (2009)*, Blue Lacuna* by Aaron Reed 2008), *Coloratura by* Lynnea Glasser (2013), *Counterfeit Monkey* by Emily short, and *Hadean Lands* by Andrew Plotkin (2014).

The Interactive Fiction Database offers readers' ratings of hundreds of interactive stories. Most of these are quite reliable, though you should be skeptical of works that have high average ratings, based on a very small number of responses.

The Top Seventy (Or So) Works of Interactive Fiction (In One Person's Opinion) For Use In Middle (Students Aged 10-13) and High Schools (Students Aged 14-18)

Stories that are available in the IF Archive are also at the Interactive Fiction Database (ifdb.tads.org), where



online versions can usually be found. Many of the stories involve ancillary materials that can be obtained by Googling the stories' websites.

1. Arthur: the Quest for Excalibur by Bob Bates (1989), a well-plotted version of the story of King Arthur as a boy, excellent for middle school (Available in *Masterpieces of Infocom*, and other Infocom collections, often Offered at Ebay and Amazon)

2. Lost Pig by Admiral Jota (2007), hilarious tale of Grunk the orc, fun for readers of amost all ages. Winner of the 2007 Fall IF Competition. (Available at the Interactive Fiction Archive and the Interactive Fiction Database.

3. Wishbringer by Brian Moriatry, (1985) a gentle fantasy-adventure, excellent for middle school (Available in *Masterpieces of Infocom*)

4. *The Firebird* by Bonnie Montgomery (1998), a comic retelling of the Russian folk tale, excellent for middle school (Available at the Interactive Fiction Archive, http://www.ifarchive.org. The archive is well catalogued at the Interactive Fiction Database (http://ifdb.tads.org)

5. Winter Wonderland by Laura Knauth (1999), a finely-crafted winter solstice story, excellent for middle school (Available at the Interactive Fiction Archive, http://www.ifarchive.org)

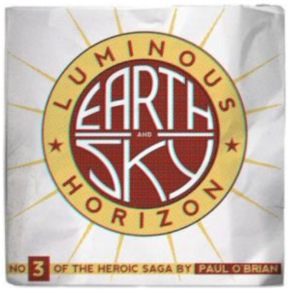
6. *Mrs. Pepper's Nasty Secret* by Eric Eve and Jim Aiken (2008), a clever and lighthearted puzzle-fest, good for beginners. (Available at http://ifdb.tads.org/viewgame?id=dcvk7bgbqeb0a71s)

7. *A Bear's Night Out* by David Dyte (1997), a funny story of a teddy bear who comes to life (Available at the Interactive Fiction Archive, http://www.ifarchive.org)

8. *The Earth and Sky Trilogy--Earth and Sky* by Paul O'Brian (2001), the first of three comic superhero stories (Available at the Interactive Fiction Archive, http://www.ifarchive.org)

9. *The Earth and Sky Trilogy--Another Earth, Another Sky* by Paul O'Brian (2002), the second of three comic superhero stories (Available at the Interactive Fiction Archive, http://www.ifarchive.org)

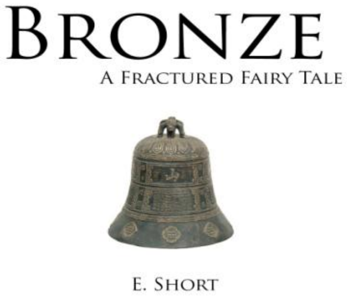
10. *The Earth and Sky Trilogy--Luminous Horizon* by Paul O'Brian (2004), the third of three comic superhero



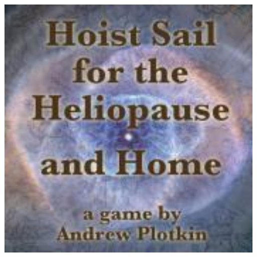
stories (Available at the Interactive Fiction Archive, http://www.ifarchive.org)

11. *Photopia* by Adam Cadre (With Censoring of Opening Scene of Some Versions, 1998), a challenging, beautiful, and very sad story of a middle school girl (Available at the Interactive Fiction Archive, http://www.ifarchive.org)

12. *The Matter of the Monster* by Andrew Plotkin (2011), an inventive variation on the "Choose Your Own Adventure" type of story. Quite brief and very enjoyable.



13. Bronze by Emily Short (2006), a Gothic retelling of Beauty and the Beast, with some references to sexuality and suicide, good for mature high schoolers. (Available



at the Interactive Fiction Archive, http://www.ifarchive.org. A slightly-edited version for younger readers is at http://bdesilets.com/if/Bronze.z8.

14. Hoist Sail for the Heliopause and Home by Andrew Plotkin (2010), a graceful space-exploration fantasy. Best for older students who have some literary-analysis skills. (Available at the Interactive Fiction Archive and the Interactive Fiction Database.)

15. *Jack Toresal and the Secret Letter* by Michael Gentry and David Cornelson, (2009) an exciting interactive novel with a suspenseful plot and wonderfully interactive characers (Available for purchase at http://textfyre.itch.io/jack-toresal-and-the-secret-letter)

16. *Robin & Orchid* by Ryan Veeder and Emily Boegheim (2013), comic story about high school students looking for a ghost in a Methodist church. Some readers may find mild irreverence here and there. (Available at the Interactive Fiction Archive, http://www.ifarchive.org)

17. *Mother Loose* by Irene Callaci (1998), an amusing retelling of some classic fairy tales, excellent for middle school (Available at the Interactive Fiction Archive, http://www.ifarchive.org)

18. *Bonehead* by Sean M. Shore (2011), the mostly-true story of Fred Merkle, who made one of major league baseball's most famous mistakes. Part of the 2011 Spring Thing Competition. (Available at the Interactive Fiction Archive, http://www.ifarchive.org)

19. Aoteoroa by Matt Wingall (2010), a lively tale of the modern world--with dinosaurs! (Available at the Interactive Fiction Archive, http://www.ifarchive.org)

20. Taco Fiction by Ryan Veeder (2011), a young man

plans on making a big mistake, but thinks better of it. Recommended for older kids, sixteen and up. (Available at the Interactive Fiction Archive, http://www.ifarchive.org)

21. History Repeating by Mark and Renee Choba (2005), a time travel story about a man who neglected an important assignment in his high school history class (Available at the Interactive Fiction Archive, http://www.ifarchive.org)

22. The Warbler's Nest by Jason MacIntosh (2010), a brief, truly creepy, though non-graphic, horror tale. Best for older students. (Available at the IF Archive and the IF Database)

23. Lost New York by Neil DeMause (1996), a detailed, well-researched time-travel story about the Big Apple; excellent for more mature middle schoolers and for high schoolers (Available at the Interactive Fiction Archive, http://www.ifarchive.org)

24. Suspect by Dave Lebling (1984), a rather challenging murder mystery, set at a high-society costume party (Available in *Masterpieces of Infocom*)

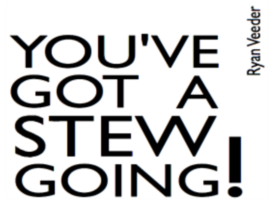
25. Moonmist by Jim Lawrence and Stu Gally (1986), a mystery for kids, set in a Cornish castle, good for middle school (Available in *Masterpieces of Infocom*)

26. Zork I by Marc Blank and Dave Lebling (1980), a fantasy treasure hunt (Available at http://www.csd.uwo.ca/Infocom/download.html)

27. The One That Got Away by Leon Lin (1995), a brief, funny story about fishing (Available at the Interactive Fiction Archive, http://www.ifarchive.org)

28. Glowgrass by Nate Cull (1997), a sad but graceful science fiction story, set in a post apocalyptic future (Available at the Interactive Fiction Archive, http://www.ifarchive.org)

29. *Small World* by Andrew Pontius (1996), a fantasy about a boy who sets a mixed up world right (Available at the Interactive Fiction Archive, http://www.ifarchive.org)



30. *You've Got a Stew Going* by Ryan Veeder (2011), funny tale with a rat progagonist. (Available at the IF Archive and at the IF Database.)

31. The Magic Toyshop by Gareth Rees (1995), an imaginative puzzle fest (Available at the Interactive Fiction Archive, http://www.ifarchive.org)

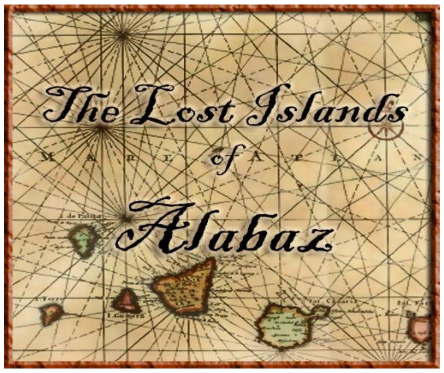
32. Mingsheng by Deane Saunders (2004), a sometimes mystical, always gentle story involving martial arts (Available at the Interactive Fiction Archive, http://www.ifarchive.org)

33. *Dragon Adventure* by William Stott (2003), an adventure story for children aged nine and up (Available at the Interactive Fiction Archive, http://www.ifarchive.org)

34. *It* by Emily Boegheim (2011). Read this appealing story and learn to play "sardines." (Available at the IF Archive and at the IF Database.)

35. The Great Xavio by Reese Warner (2004), a comic detective story (Available at the Interactive Fiction Archive, http://www.ifarchive.org)

36. *The Dreamhold* by Andrew Plotkin (2004), a difficult fantasy story with help for beginners (Available at the Interactive Fiction Archive, http://www.ifarchive.org)



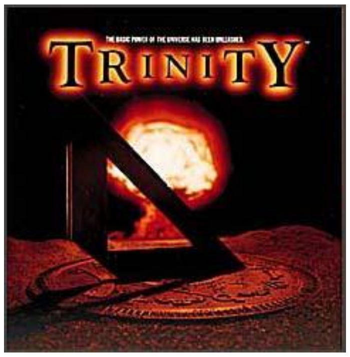
37. The Lost Islands of Alabaz by Michael Gentry (2011), an action-packed fantasy tale. Winner of the 2011 Spring Thing Competition. (Available at the Interactive Fiction Archive, http://www.ifarchive.org)

38. The Shadow in the Cathedral by Ian Finley and John Ingold (2009), a fantasy puzzle fest with a compelling plot (Available for purchase at http://textfyre.itch.io/the-shadow-in-the-cathedral)

39. The Witness by Stu Galley (1984), a mock-noir mystery, good for high school (Available in *Masterpieces of Infocom*)

40. *The Hitchhiker's Guide to the Galaxy* by Steve Meretzsky and Douglas Adams (1984), a sometimes-silly interactive story based loosely on a famous novel

(Available at http://www.bbc.co.uk/radio4/hitchhikers/game.shtml)



41. Trinity by Brian Moriarty (1986), brilliant but difficult fantasy based on the origin of nuclear weaponry (Available in *Masterpieces of Infocom*)

42. A Mind Forever Voyaging by Steve Meretzsky (1985), soaringly literary but difficult tale of a dangerous future, good for high school (Available in *Masterpieces of Infocom*)

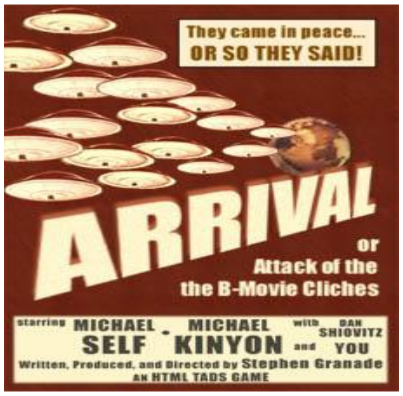
43. *Seastalker* by Stu Galley and Jim Lawrence (1984), adventure in a futuristic submarine, good for middle school (Available in *Masterpieces of Infocom*)

44. The Orion Agenda by Ryan Weisenberger (2004), a Star-Trek-like story of a distant planet (Available at the Interactive Fiction Archive, http://www.ifarchive.org)

45. Ballyhoo by Jeff O'Neill (1985), mystery-adventure, set in a circus (Available in *Masterpieces of Infocom*)

46. *At the Bottom of the Garden* by Adam Biltcliff (2000), a comic tale of an invasion by miniature dragons

(Available at the Interactive Fiction Archive, http://www.ifarchive.org)


They came in peace...
OR SO THEY SAID!
ARRIVAL
or
Attack of the
the B-Movie Cliches
starring MICHAEL SELF - MICHAEL KINYON with DAN SHIOVITZ and YOU
Written, Produced, and Directed by Stephen Granade
AN HTML TADS GAME

47. Arrival by Stephen Granade (1998), funny sendup of low-budget science fiction movies (Available at the Interactive Fiction Archive, http://www.ifarchive.org)

48. *Plantefall* by Steve Meretzsky (1983), a comic science fiction story with some tough puzzles, good for high school (Available in *Masterpieces of Infocom*)

49. *Savoir Faire* by Emily Short (2002), a beautifully written, difficult puzzle fest, with a romantic twist, excellent for high school (Available at the Interactive Fiction Archive, http://www.ifarchive.org)

50. *Zork III* by Marc Blank and Dave Lebling, (1982) a fantasy story with interesting character interaction. (Available at http://www.csd.uwo.ca/Infocom/games.html)

51. *Christminster* by Gareth Rees (1995), a difficult, detailed, finely crafted tale of alchemy and intrigue, set at a British university (Available at the Interactive Fiction Archive, http://www.ifarchive.org)

52. *Plundered Hearts* by Amy Briggs (1987), a spoof or romance novels, with some mild sexual references, good for mature high-school students (Available in *Masterpieces of Infocom*)

53. *Sherlock: the Riddle of the Crown Jewels* by Bob Bates (1987), a complex mystery with a realistic map of Victorian London (Available in *Masterpieces of Infocom*)

54. *MythTale* by Temari Seikaiha (2002), a clever blending of several Greek myths (Available at the Interactive Fiction Archive, http://www.ifarchive.org)

55. Six by Wade Clarke (2011), playful story with a six-year-old protagonist. Second Place in the 2011 Fall IF Competition. (Available at the IF Archive and at the IF Database)

56. *The Colour Pink* by Robert Street (2005), a space-exploration story with some clever twists, a mild sexual reference or two, sometimes simplistic prose styles, some avoidable violence (Available at the Interactive Fiction Archive, http://www.ifarchive.org)

57. *Galatea* by Emily Short (2000), lovely and challenging retelling of the Pygmalion myth, good for high school (Available at the Interactive Fiction Archive, http://www.ifarchive.org)

58. *Curses!* by Graham Nelson (1993), a difficult, witty, atmospheric tale of a haunted house (Available at the Interactive Fiction Archive, http://www.ifarchive.org)

59. *1893: A World's Fair Mystery,* by Peter Napstad (2002), a remarkably detailed story, set at America's most important world's fair, a bit gory at times, with one mild sexual reference (Available at the IF Database, http://ifdb.tads.org)

60. Wetlands by Clara Raubertas (2011), a rather difficult fantasy story with lots of atmosphere. Part of the 2011 Spring Thing Competition. (Available at the Interactive Fiction Archive, http://www.ifarchive.org)

61. *Worlds Apart* by Suzanne Britton (1999), an extraordinarily rich fantasy story for skilled readers (Available at the Interactive Fiction Archive, http://www.ifarchive.org)

62. The Meteor, the Stone, and a Long Glass of Sherbet by Graham Nelson (1993), a funny and cohesive fantasy tale (Available at the Interactive Fiction Archive, http://www.ifarchive.org)

63. *Zork Zero* by Steve Meretzsky (1988), a very funny fantasy adventure, good for high school (Available in *Masterpiece of Infocom*)

64. *Spider and Web* by Andrew Plotkin (1998), a spy story with an unusual plot structure and theme, ideal for skilled readers. Often appears at the top of all-time best IF lists. (Available at the Interactive Fiction Archive, http://www.ifarchive.org)

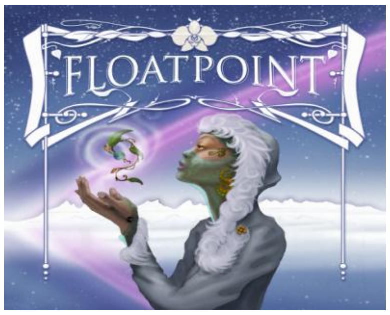
65. Moon-Shaped by Jason Ermer (2006), a compelling, somewhat Gothic fairy tale. Though sad and intense, this story is accessible to some teenagers.   (Available at the Interactive Fiction Archive, http://www.ifarchive.org)

66. Once and Future by Kevin Wilson (1998), a time-travel adventure that combines Arthurian legends and the Vietnam War. For mature readers (Available at the Interactive Fiction Archive, http://www.ifarchive.org)

67. Zork II by Marc Blank and Dave Lebling (1981), a challenging fantasy treasure hunt with an amusing wizard character (Available at http://www.csd.uwo.ca/Infocom/games.html)

68. *Enchanter* by Marc Blank and Dave Lebling (1983), a challenging tale of spells and magic (Available

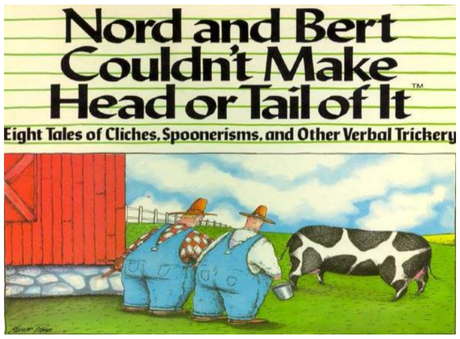
in *Masterpiece of Infocom*)



69. Floatpoint by Emily Short (2006), an artistically-crafted story of interplanetary diplomacy. Themes of genetic engineering of sentient life make this story better for older students. (Available at the Interactive Fiction Archive, http://www.ifarchive.org)

70. Beyond Zork by Brian Moriaty (1987), a detailed and difficult fantasy tale, with some options for shaping the player-character, good for high school (Available in *Masterpiece of Infocom*)

71. *The Light: Shelby's Addendum*, by Colm McCarthy (1995), a complex science fiction tale with a carefully crafted map and difficult problems (Available at the Interactive Fiction Archive, http://www.ifarchive.org)


Nord and Bert Couldn't Make Head or Tail of It™
Eight Tales of Cliches, Spoonerisms, and Other Verbal Trickery

72. Nord and Bert Couldn't Make Head or Tail of It by

Jeff O'Neil (1987), a hilarious collection of short stories based on word play (Available in *Masterpieces of Infocom*)

73. The Beetmonger's Journal by Scott Sharkey (2001), well-written, enjoyable tale of archeology (Available at the Interactive Fiction Archive, http://www.ifarchive.org)

74. *Tales of the Traveling Swordsman* by Mike Snyder (2006), a swashbuckling adventure with a twist at the end (Available at the Interactive Fiction Archive, http://www.ifarchive.org)

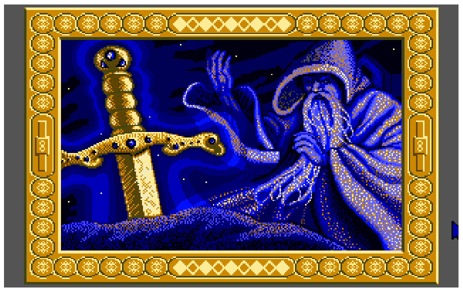
75. Metamorphoses by Emily Short (2000), a challenging story of magical transformations (Available at the Interactive Fiction Archive, http://www.ifarchive.org)

76. *Deadline* by Marc Blank (1982), a really hard mystery story with good character interaction (Available in *Masterpiece of Infocom*)

77. *Jacqueline, Jungle Queen!* by Steph Cherrywell (2014), an amusing story with some relatively easy puzzles. Includes a short, bawdy poem that can be avoided. Third Place Winner in the 2014 Fall IF Competition. (Available at

http://ifdb.tads.org/viewgame?id=h63kzv5acq9s7dak)

Chapter 12 – An Interactive Classic from the Commercial Era



Arthur: the Quest for Excalibur

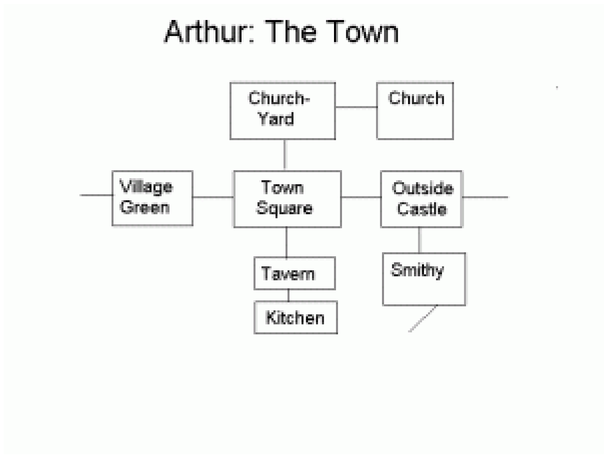
Advantages, Maps, and Walkthrough

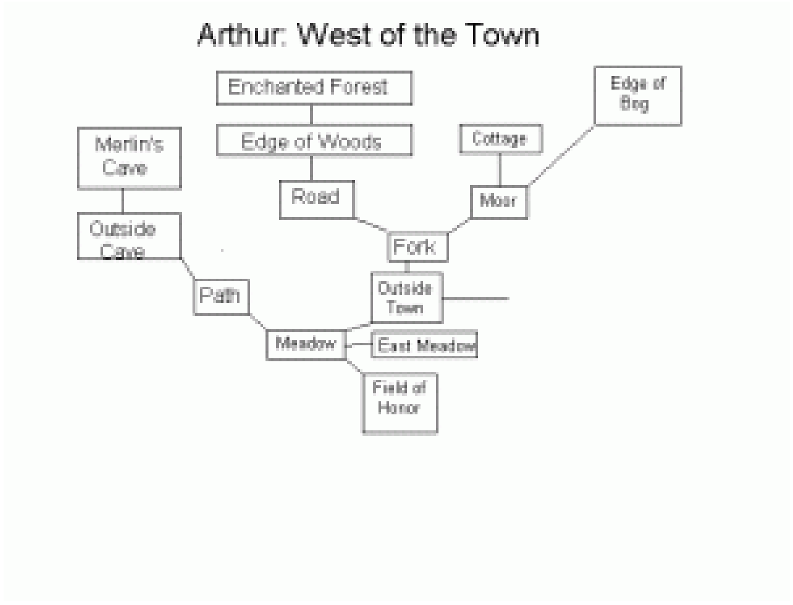
Arthur, the Quest for Excalibur by Bob Bates (Infocom, 1989) offers special opportunities for middle school teachers, in addition to those that all good interactive fiction provides. To begin with, it dovetails well with traditional middle school curriculum, which often includes the Arthurian Legends. Also, unlike most versions of the Arthur stories, it offers a look at England in the fifth century AD, the period in which the historical Arthur, if there was one, actually lived; and a set of notes that contrasts this period with the thirteenth and fourteenth-century settings of most Arthurian tales.

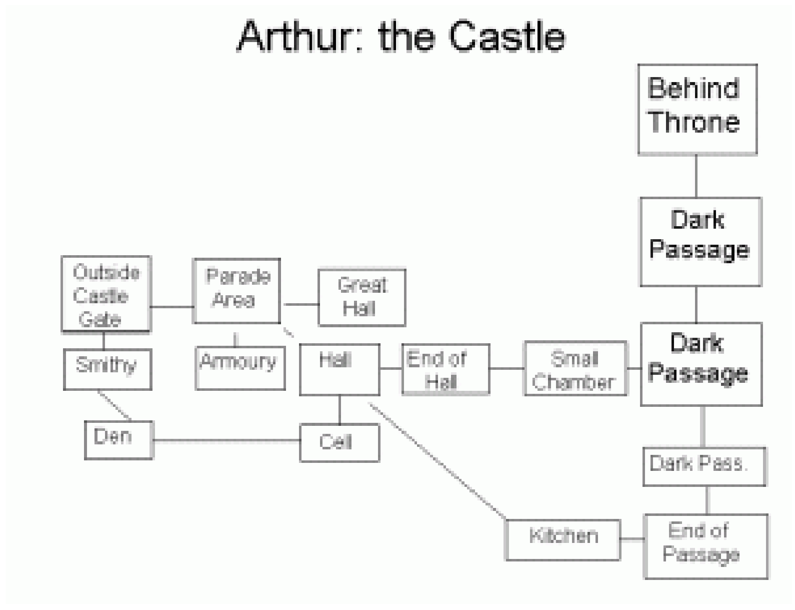
Though *Arthur* does provide on-screen mapping, teachers may find more extensive maps of stories' settings useful for their own planning, or even for use with students. Of course, distributing maps to students before they try a story will give away the solutions to some problems, but, the maps, as presented here, spoil very little of the fun and obviate a good deal of frustration.

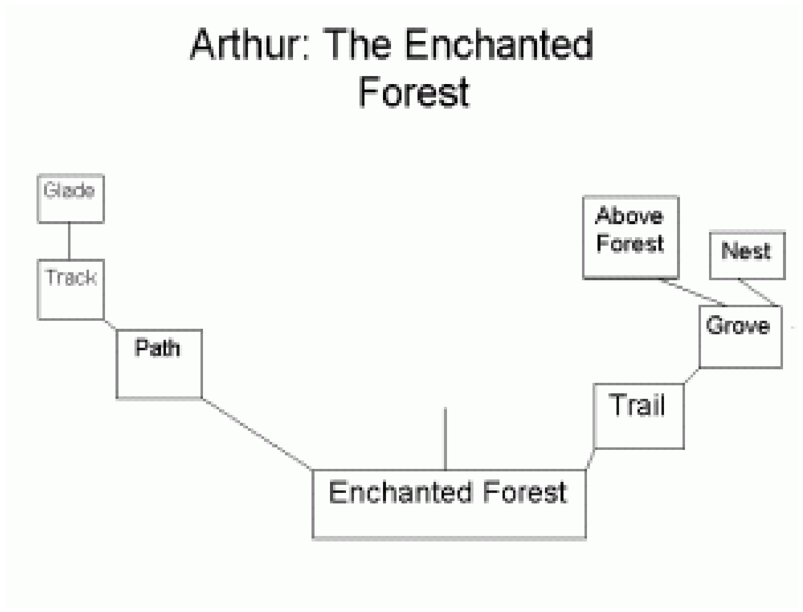
You can find a walkthrough for *Arthur* at http://www.gamefaqs.com/pc/564481-arthur-the-quest-for-excalibur/faqs/1630.

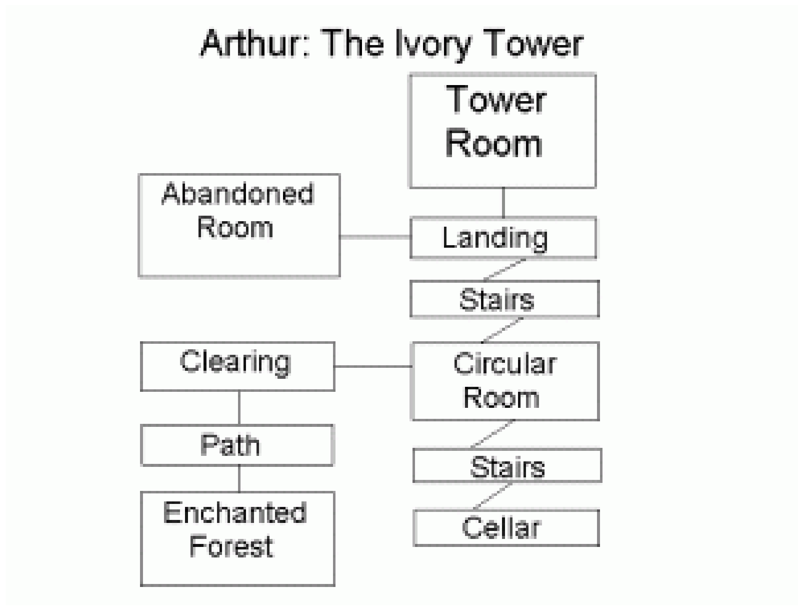
Maps for Arthur

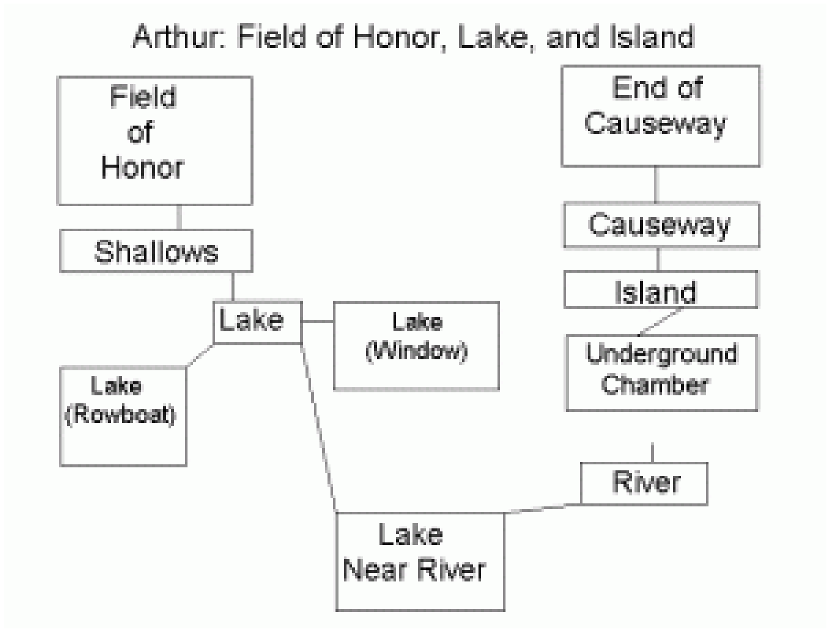


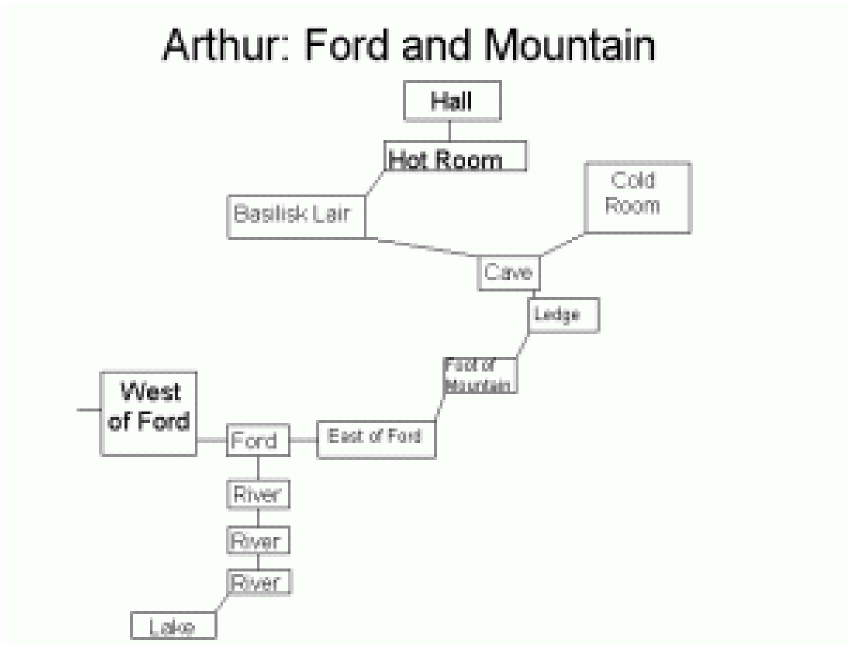



Arthur: the Castle
Behind Throne
Dark Passage
Outside Castle Gate
Parade Area
Great Hall
Smithy
Armoury
Hall
End of Hall
Small Chamber
Dark Passage
Den
Cell
Dark Pass.
Kitchen
End of Passage


Arthur: The Enchanted Forest
Glade
Above Forest
Nest
Track
Grove
Path
Trail
Enchanted Forest


Arthur: The Ivory Tower
Tower Room
Abandoned Room
Landing
Stairs
Clearing
Circular Room
Path
Stairs
Enchanted Forest
Cellar


Arthur: Field of Honor, Lake, and Island
Field of Honor
End of Causeway
Causeway
Shallows
Island
Lake
Lake (Window)
Underground Chamber
Lake (Rowboat)
River
Lake Near River



Formulating Problems in *Arthur*

Like other works of IF, *Arthur* offers some very rich opportunities for students to represent problems in a variety of ways, as they work toward solutions. Usually, my own classes maintained lists of problems and of restatements of problems, as they read works of IF. These lists helped them to keep track of the multiple problems that they were working on, helped them solve the problems, and, perhaps most important, gave them a tool for looking back at, and thinking about, their problem-solving techniques.

Let's take a look at some formulations and reformulations of problems that students typically create as they read *Arthur*.

Very early in the story, as the reader tries to start exploring, he or she encounters a problem in dealing with a curfew imposed by the evil King Lot. Here is a typical series of formulations of this problem:

- At first, students may ask, “How can I get out of the churchyard area?”

- Then, when the curfew-enforcing soldiers interfere, the problem may become, especially for relatively naive students, “How can I overcome the soldiers?”

- Soon, it becomes clear that an unarmed boy can’t defeat the soldiers, and so students usually ask, “Where can I hide from the soldiers?”

- Probably the most likely hiding place is the only nearby building, where the problem becomes, “In the church, where can I hide from the soldiers?”

- But there’s no good hiding place in the church, and this realization usually brings students to a more or less final formulation of this problem, “In the churchyard, where can I hide from the soldiers?” This question leads students to have Arthur hide, successfully, behind one of the gravestones.

Later, as he or she continues to explore, the reader encounters a deadly and treacherous bog, but a person who knows the way through turns out to be close at hand. Here is a series of reformulations of this problem.

- The problem may begin as, “How can I get through the treacherous peat bog?”

- When the readers realize that the bog is a kind of maze, in which one false step can cause disaster, they may formulate the problem in a way that takes into account the need for some sort of map: “How can I get directions for getting through the bog?”

- Once they know that they need directions, students usually ask, “Where can I find someone who might know his or her way through the bog?”

- By the time they get to the bog, students have usually found an unconscious character who apparently lives near the swamp, and so they ask, “How can I awaken the peasant?”

- But, in order to awaken the unconscious man, the reader must figure out, “What’s wrong with the peasant?”

- Some hints in the description of the unconscious man’s dwelling often lead students to ask, “What’s wrong with the peasant’s cottage?”

- And, since the cottage is described as very cold, the problem may become, “How can I warm up the cottage?”

- The cottage does have a smoldering fire, and so students may ask, “Where can I get fuel for the peasant’s fire?”

- If they have learned what peat is, the readers may now change the problem to, “How can I get some dry peat?”

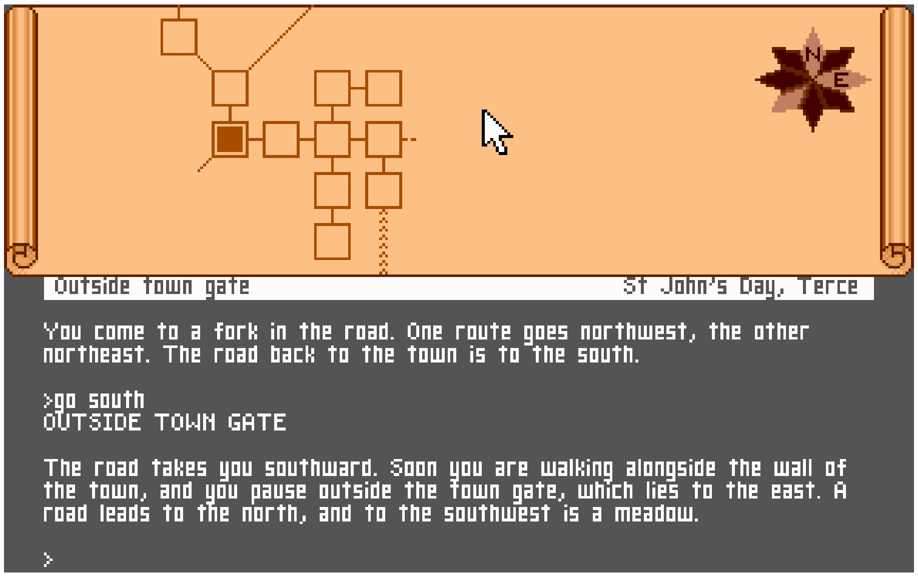
- Students find that, though dry peat is easy to find at the edge of the bog, it’s very hard to dig it out. As a result, they soon ask, “What sort of tool would be good for digging up peat?”
- Outside the peasant’s cottage, the students have already found an object with an unfamiliar name, and so they may inquire, “What is a slean?”

- Then, they may ask, “How can I find out what a slean is?” If they are catching on to the old interactive fiction rule, “Examine everything,” they will quickly solve this problem.

- Once the cottage is warmed up, the peasant regains consciousness and reveals that he does, indeed, work in the peat bog, but it’s clear that he is suffering from a leg injury. As a result, student will often ask, “How can I help the peasant further?” though this question may not seem directly related to getting through the bog. It turns out that Arthur can help the man by giving him his crutch, which has been left outside the cottage.

- Once Arthur is in the peasant’s good graces, the students may ask, “How can I find out what the peasant knows about the bog?” When they ask the peasant about the bog, he gives them directions to get through it.

Supplementary Materials for *Arthur*



```
Outside town gate                                    St John's Day, Terce

You come to a fork in the road. One route goes northwest, the other
northeast. The road back to the town is to the south.

>go south
OUTSIDE TOWN GATE

The road takes you southward. Soon you are walking alongside the wall of
the town, and you pause outside the town gate, which lies to the east. A
road leads to the north, and to the southwest is a meadow.

>
```

Arthur includes maps and hints, which are incorporated into the computer program itself. However, like other Infocom stories, it requires additional supplementary materials, which, originally, came with the story in hard-copy form. Today, you can find these materials at the website called “The Infocom Documentation Project,” which resides at http://infodoc.plover.net. The manual for *Arthur,* which you'll find at the Documentation Project, offers some important poetic materials, including a poem in which the narrator is Merlin, who appears briefly in the story. Another bit of verse, which appears at the beginning of a *Book of Hours,* praises King Lot, the villain of the tale. Each poem contains information that the reader needs in order to solve a particular problem in the narrative.

Chapter 13 – An Interactive Classic From the Modern Era: *The Firebird*

1998 was a banner year for interactive fiction, especially notable for three extraordinary works for adults, *Spider and Web* by Andrew Plotkin, *Once and Future* by Kevin Wilson, and *Photopia* by Adam Cadre. Plotkin had already established himself as one of the most brilliant of contemporary IF writers, but, with *Spider and Web*, he produced a story that many readers found more accessible than his earlier works, without sacrificing the complexity of his other stories. Wilson's interactive novel, the first major commercial work of text-based IF in many years, told a richly detailed story, based on a compelling synthesis of the Arthurian legends and the Vietnam War. And *Photopia* won a major interactive fiction competition, pushing the notion of "puzzle-free" interactive fiction to a new level, and deeply moving practically everyone who read it.

But, fortunately for kids, the good news didn't stop there. Bonnie Montogmery's *The Firebird,* a great story for young and old, appeared in the same year.

The Firebird retells a famous Russian folk tale – the same tale that Stravinski used in his
ballet – in an adventurous and often hilarious way. It offers puzzles that are fair, enjoyable, and not too difficult; and it gives the reader several possible ways to a favorable ending. In addition, it develops some memorable characters, especially the Firebird herself; raises interesting questions about the roles of women in fairy tales; and, perhaps, broadens our cultural horizons a bit.

You can download *The Firebird* and learn about running it on your local computer at the Interactive Fiction Database. You can learn more about downloading files and using interpreters in another chapter in this book, the chapter called "Acquiring Interactive Fiction."

Walkthrough for *The Firebird*

Walkthrough by Brendan Desilets, with help from Bonnie Montgomery, 1999

The Firebird by Bonnie Montgomery combines skillful storytelling, colorful characters, and rollicking humor to create a truly enjoyable work of interactive fiction. The story offers lots of entertaining problems, none of them especially difficult to solve, though, occasionally, for lack of a synonym or some such nicety, the reader may become stuck. That's where this walkthrough, which is constructed in such a way as to discourage its own use, comes in. For those who prefer a "bare bones" solution, there's such a beast at the end of this document. Many of the *Firebird* puzzles have multiple solutions and almost all are relatively easy.

Here are some simple hints for dealing with parser and synonym problems in the story. For many readers, these may be all the help that's needed.

When Ivan is trying to move some earth, the one-word command "DIG" may work better than other expressions. Similarly, the single word "ENTER" is sometimes better than more complex formulations. The key words in communicating with the allergist are "STING," or "TREATMENT," or "INJECTION," or "SHOT," or "MEDICINE."

The user's input appears in upper case letters throughout this walkthrough.

As young Prince Ivan, in this retelling of the famous Russian folk tale, you begin in a forest, hunting. Let's start by trying to get a broader view of the forest by typing U (for "up"). Bump. We didn't get very high, but we did scare up some game, a flicker. PUT ARROW IN BOW.

Whenever the quarry tries to escape, FOLLOW BIRD. AIM ARROW AT BIRD. SHOOT BIRD. TAKE BIRD. (This would be a good time to abandon this walkthrough and explore until you need more help, if you ever do.)

Now GO SOUTH and SOUTH again. (You can abbreviate “So south” with the single letter “s.

GIVE BIRD TO CATERER. There's a lovely and unfamiliar young woman here. LOOK AT WENCH. TAKE MASK. LOOK AT MASK. LOOK AT BUSBOY. TAKE ALL.

It looks as though we're finished here for now. Let's explore more of the forest. N, N, E, E. Here's a baba yaga, or witch. That bag she has looks interesting. DRINK COFFEE. PUT CUP IN BAG.

That may be a useful transformation. Let's try some others. PUT PLATE IN BAG. EAT SANDWICH. PUT PLATE IN BAG. PUT FORK IN BAG. PUT KNIFE IN BAG. PUT NAPKIN IN BAG.

Time for a bit more exploration. W, W, N, N, N, N. How can we trick those adoring groupies, who'll destroy our allergy-plagued skin with their lipstick? (This is another good time to put the walkthrough aside.)

(Spoiler Space)

WEAR MASK. N, N. Now we want to get the allergist to treat us, but his command of synonyms is somewhat limited, and we have to phrase our request with precision. ASK ALLERGIST ABOUT STING. ALLERGIST, GIVE STING. ME.

LOOK AT MASSEUR to discover his true identity. You're not the only royal brother who's hiding from the rest of the world.

Let's explore a bit further. N. TAKE COIN. LOOK AT COIN. N.

Here at the inn, we can collect a number of items, some very useful, from the trash pile, and hear a few funny stories as we wait for folks to dump the goodies. WAIT, repeatedly, or investigate the horse, the groom, or the tavern while you wait, and TAKE items from the pile, until you have collected GREENS, a WHITE FLASK, a BLACK FLASK, and a MATCHBOOK. The flasks are especially interesting. During your subsequent travels, experiment with them as much as you can, until you figure out how they work.

Now it's time to explore the forest further. S. S. S. S. S. S. SW. W. W. W. At the wall, READ PLAQUE for a really broad hint, and then throw away this walkthrough, because you're really ready to have some fun now if you work on the problems yourself.

Go SW. Then, in the newly-dug bed, DIG. On the other side of the wall, go NORTH, where you'll find the classic (more or less) fairy-tale frog. READ FROG. KISS FROG. KISS FROG. KISS FROG. Oops, no amphibian prince or princess here. KILL MANIAC WITH SWORD.

Now for some more exploring of the land encircled by the wall. Go SOUTH, then EAST, then UP into the tree. Go UP again, and then UP one more time. WAIT until the Firebird appears, and then, to effect a gentle capture, PUT BLANKET ON BIRD.

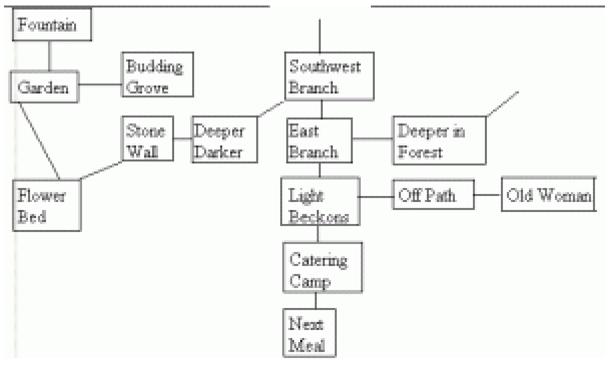
Climb DOWN, TAKE FRUIT, and then go DOWN again to the ground. Could any folk hero resist the pleas of the poor Firebird? You certainly shouldn't. RELEASE BIRD. If you do not have everything you need for your further adventures, the firebird will ask you a question. Answer YES and READ LIST.

It's time to go back under the wall, and to head for the caterer's camp again. W. D. NE. E. E. NE. S. S. S. After collecting what the Firebird requested, we'll be ready to go on to the eastern part of the story, which has been unavailable to us until now. N. N. N. E. NE. When the story asks a witty question about saving your soul, you might as well answer YES.

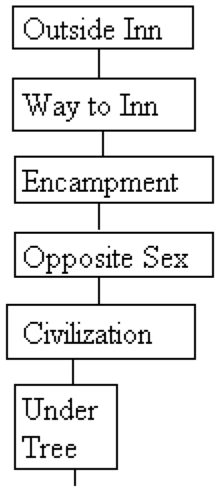
Now we must deal with the ferrymen. They represent a problem that is not too difficult and great fun to solve, so go ahead and solve it without this walkthrough. (Spoiler Space)

Maps for *The Firebird*

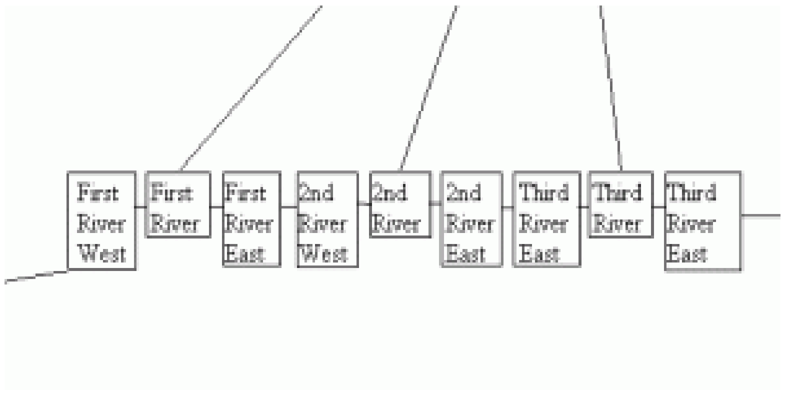
The Forest



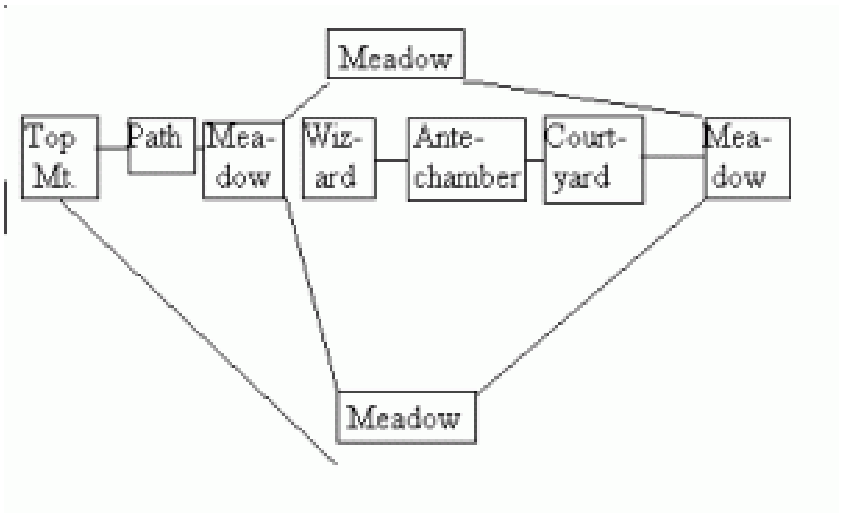
Civilization



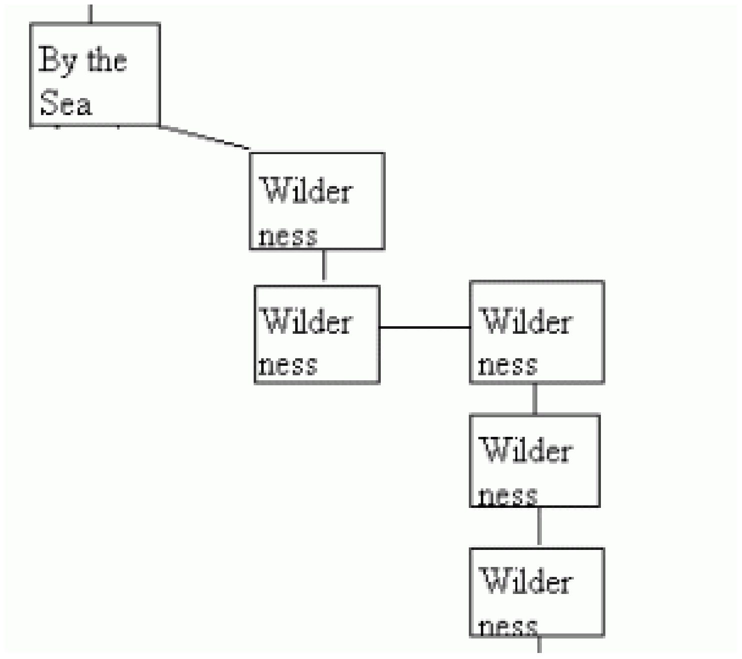
The Rivers



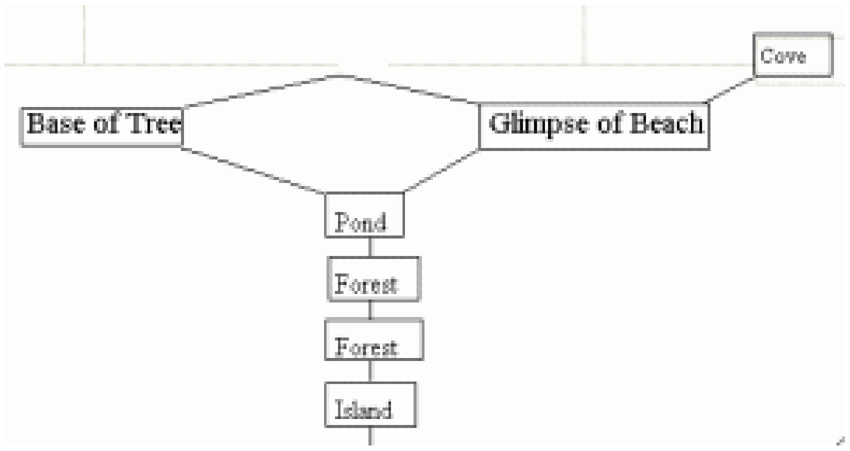
Top of the Mountain



Wilderness Maze



The Island


Cove
Base of Tree
Glimpse of Beach
Pond
Forest
Forest
Island

At the first river, GET IN FERRY. WAIT, then WAIT again. Now you'll have to pay the ferryman, one way or another. GIVE COIN TO FERRYMAN. But, as it turns out, you'll need that coin later, so KILL FERRYMAN WITH SWORD. TAKE COIN.

In order to follow a rather direct route through the story, we'll now look for a way to circumvent those other ferrymen. It's more fun, though, to try to deal with them one by one. Go ahead. Try it.

How can we get around those other grim toll-takers? Go WEST. SWIM IN RIVER. On the new shore, we find a fish out of water, a pike. TAKE PIKE. LOOK AT PIKE. PUT PIKE IN SEA.

Now we enter the world's easiest maze. Go SE. NOTE 2. READ TRAIL. S. SE. S. S. S.

Emerging from the maze, we come upon our murderous brother Vasilii. KILL VASILII WITH SWORD. LOOK AT VASILII. Now, let's see. How might we win over a dead brother? SPRINKLE WATER OF DEATH ON VASILII. SPRINKLE WATER OF LIFE ON VASILII. (But there's another, less messy way to deal with Vasilii. See if you can find it.)

It's time to head for the evil wizard's mountain. Go EAST, then SOUTHEAST. There's a strange-looking rock here. LOOK AT ROCK. OPEN DOOR. ENTER. Inside the cave, LOOK AT DEBRIS. Then, DIG, DIG, DIG, DIG. Now we have the right tools for mountain climbing.

EXIT. LOOK. Go NORTHWEST, and WEAR CLAWS. Time to do some climbing. Go UP. Here's a very strange building. How might we get in? Note the color of the place. (Spoiler Space)

REMOVE CLAWS. N. LOOK AT DOOR. PUT COIN IN SLOT. Here's a character worth chatting with. BIRD, HELLO. ASK WOMAN ABOUT FIREBIRD. ASK WOMAN ABOUT WIZARD.

Back outside the edifice, WEAR CLAWS and go UP.

REMOVE CLAWS and go NORTH. LOOK AT DOOR. LOOK AT UTENSIL. PUT UTENSIL IN SLOT. Once again, we encounter someone with a real story to tell. ASK WOMAN ABOUT FIREBIRD. ASK WOMAN ABOUT WIZARD.

We find ourselves outside again, ready for more climbing. WEAR CLAWS. U. REMOVE CLAWS. LOOK AT DOOR. PUT NUGGET IN SLOT. And here's the Firebird once again! ASK FIREBIRD ABOUT FIREBIRD. ASK FIREBIRD ABOUT WIZARD. Keep the conversation going until the Firebird makes it clear that she has nothing more to say.

One more climb. WEAR CLAWS. U. REMOVE CLAWS. E. E. NE. SE. W. It looks as though the giant serpent is blocking our entry into the palace, but we have the wherewithal to deal with the monster. Take an inventory (I), and figure it out. (Spoiler Space)

PUT CHAIN ON CENSER. PUT HERBS IN CENSER. LIGHT MATCH. PUT MATCH IN
CENSER. SWING CENSER AT SERPENT. Now we can pass. Go W, ENTER, and meet another character who has a great deal to say.

Now, let's go looking for the egg that we can use to defeat the wizard.

EXIT. E. NW. SW. W. W. WEAR CLAWS. D. D. D. D. REMOVE CLAWS. W. L. N.
READ TRAIL. N. N. N. N. NE. SW. S. W. NW.

Here we are at the sea, once again, not far from that island to the north. Is this the isle that hides the egg? Go NORTH to reach the island, then NORTH again.

We should probably try to help this unfortunate bear. MOVE TREE. FALLEN. FREE BEAR. PUSH TREE. FALLEN. Apparently, we weren't much help, but it's a small island. Maybe we'll get to try again later. In fact, if we LISTEN, we'll hear that he's not far away. (Actually, it is possible to free the bear before he loses his paw.

Check your inventory and see if you can figure it out).

Go NORTH, and find another creature in need, a baby otter. TAKE OTTER. Let's try to return the baby to the sea. S. S. PUT OTTER IN SEA. Once again, we haven't been able to help much.

N. N. N. Yet another animal in trouble, this time, a hawk. FREE HAWK. UNTANGLE VINES.

N. Here we find a pond and a log, just what we've been seeking. LOOK AT LOG. TAKE LOG. NW. TAKE LOG. CLIMB TREE. We're closer to getting the log now. Maybe the rain will help. WAIT until the pond level rises sufficiently, and then TAKE LOG.

Let's explore the island further. D. SE. NE. NE. Here's the bear again. Now we can really help the creature. How? (Spoiler Space)

SPRINKLE WATER OF DEATH ON BEAR. And we can help the otter, too, once we see that his mama is here looking for him. PUT OTTER IN SEA.

SW. SW. NW. Let's break open that log. HIT LOG WITH SWORD. Apparently, our good deeds are being rewarded. TAKE HARE. TAKE DUCK. We seem to have animal allies everywhere. TAKE EGG. LOOK IN OCEAN.

Are we finally ready to confront our nemesis, the wizard? SW. S. SW. S. S. S. S.

S. SE. S. SE. S. S. S. E. WEAR CLAWS. U. U. U. U. REMOVE CLAWS. E. E. NE. SE. W. I. PUT HERBS IN CENSER. LIGHT MATCH. PUT MATCH IN CENSER. SWING CENSER AT SERPENT. W. GO IN.

Here we are in the wizard’s castle again, but this time we have the egg and lots of enthusiastic allies. OPEN

DOOR, and confront the overconfident wizard and his bodyguards. BREAK EGG.

And, finally, we arrive at the Cathedral of Our Lady, in triumph and in charge. NOTE 6. DIMITRI, MARRY ELENA. VASILII, MARRY SILVER. (Try some other combinations, too.)

MARRY PEARL.

Now go back and try the many other variations that *The Firebird* offers!

Here’s the barebones version of this walkthrough.
U
PUT ARROW IN BOW
FOLLOW BIRD (when it moves away)
AIM ARROW AT BIRD
SHOOT BIRD
TAKE BIRD
S
S
GIVE BIRD TO CATERER
LOOK AT WENCH
TAKE MASK
LOOK AT BUSBOY
TAKE ALL
N

N
E
E
DRINK COFFEE
PUT CUP IN BAG
PUT SPOON IN BAG
I
PUT PLATE IN BAG
EAT SANDWICH
PUT PLATE IN BAG
PUT FORK IN BAG
PUT KNIFE IN BAG
PUT NAPKIN IN BAG
W
W
N
N
N
N
WEAR MASK
N
N
ALLERGIST, GIVE STING
ME
N
TAKE COIN
LOOK AT COIN
N
LOOK AT PILE

Z
TAKE GREENS
Z
Z
TAKE FLASK
LOOK AT FLASK
Z
Z
Z
Z
NOTE 1
Z
Z
Z
TAKE FLASK
LOOK AT FLASK
BLACK
Z
TAKE MATCHBOOK
S
S
S
S
S
S
SW
W
W
W

READ PLAQUE
SW
DIG
(the newly dug bed)
(with the shovel)
N
READ FROG
KISS FROG
G
G
KILL MANIAC WITH SWORD
S
E
U
U
U
Z
Z
PUT BLANKET ON BIRD
D
TAKE FRUIT
D
RELEASE BIRD
Y
READ LIST
W
D
NE
E

E
NE
S
S
S
N
N
N
E
NE
Y
GET IN FERRY
Z
Z
GIVE COIN TO FERRYMAN
KILL FERRYMAN WITH SWORD
TAKE COIN
W
SWIM IN RIVER
TAKE PIKE
LOOK AT PIKE
PUT PIKE IN SEA
SE
NOTE 2
READ TRAIL
S
SE
S
S

```
S
KILL VASILII WITH SWORD
LOOK AT VASILII
SPRINKLE WATER OF DEATH ON VASILII
SPRINKLE WATER OF LIFE ON VASILII
E
SE
LOOK AT ROCK
OPEN DOOR
ENTER
LOOK AT DEBRIS
DIG
DIG
DIG
DIG
EXIT
L
NW
WEAR CLAWS
U
REMOVE CLAWS
N
LOOK AT DOOR
PUT COIN IN SLOT
BIRD, HELLO
ASK WOMAN ABOUT FIREBIRD
ASK WOMAN ABOUT WIZARD
WEAR CLAWS
U
```

REMOVE CLAWS
N
LOOK AT DOOR
PUT UTENSIL IN SLOT
ASK WOMAN ABOUT FIREBIRD
ASK WOMAN ABOUT WIZARD
WEAR CLAWS
U
REMOVE CLAWS
LOOK AT DOOR
PUT NUGGET IN SLOT
ASK FIREBIRD ABOUT FIREBIRD
ASK FIREBIRD ABOUT WIZARD
ASK FIREBIRD ABOUT HERBS
WEAR CLAWS
U
REMOVE CLAWS
E
E
NE
SE
W
I
PUT CHAIN ON CENSER
PUT HERBS IN CENSER
LIGHT MATCH
PUT MATCH IN CENSER
SWING CENSER AT SERPENT
W

GO IN
EXIT
E
NW
SW
W
W
WEAR CLAWS
D
D
D
D
REMOVE CLAWS
W
L
N
READ TRAIL
N
N
N
N
NE
SW
S
W
N
NW
N
N

MOVE TREE
FALLEN
FREE BEAR
FREE PAW
PUSH TREE
FALLEN
N
TAKE OTTER
S
S
PUT OTTER IN SEA
N
N
N
FREE HAWK
UNTANGLE VINES
N
LOOK AT LOG
TAKE LOG
NW
TAKE LOG
CLIMB TREE
TAKE LOG
D
SE
NE
NE
SPRINKLE WATER OF DEATH ON BEAR
PUT OTTER IN SEA

SW
SW
NW
HIT LOG WITH SWORD
TAKE HARE
TAKE DUCK
TAKE EGG
LOOK IN OCEAN
SW
S
SW
S
S
S
S
S
SE
S
SE
S
S
S
E
WEAR CLAWS
U
U
U
U
REMOVE CLAWS

E
E
E
NE
SE
W
I
PUT HERBS IN CENSER
LIGHT MATCH
PUT MATCH IN CENSER
SWING CENSER AT SERPENT
W
GO IN
OPEN DOOR
BREAK EGG
NOTE 6
DIMITRI, MARRY ELENA
VASILII, MARRY SILVER
MARRY PEARL

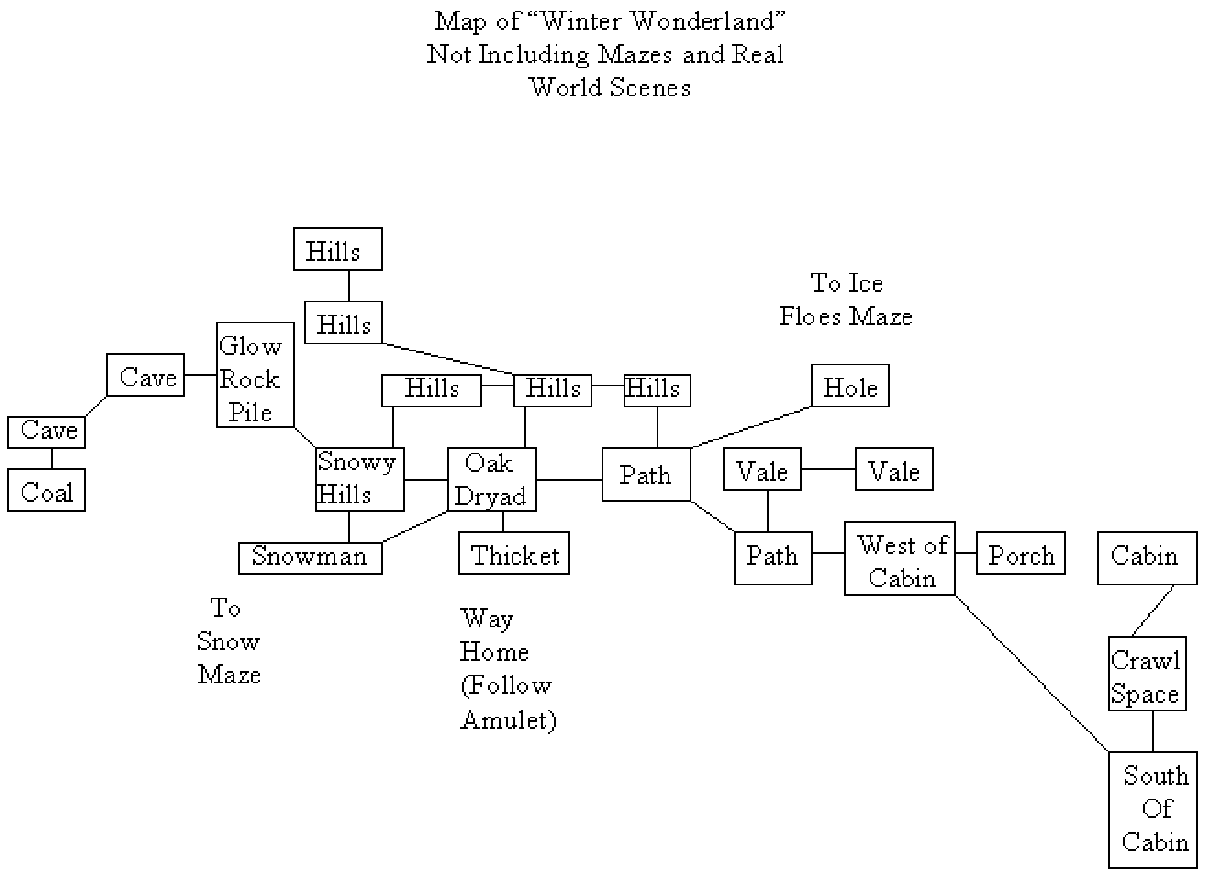
Chapter 14 – An Interactive Fiction Competition Winner: *Winter Wonderland*

Laura Knauth's *Winter Wonderland* is a beautifully plotted puzzle-fest, built around various legends of the solstice. The story won the Rec.arts.int-fiction Interactive Fiction Competition in 1999, and is highly suitable for kids.

The story includes an excellent system of on-line hints, and its mazes are quite easy to map. One problem involving snow imps is hard for many users. Just remember that, as the hints suggest, it's important to watch the imp from your hiding place for a very long time.

The map that appears below may help with some locations in the story, especially the entrance to the ice floes maze.

Laura's original map, which she used in the writing of the story, is at http://www.lauraknauth.com/Winter/WinterMap.JPG



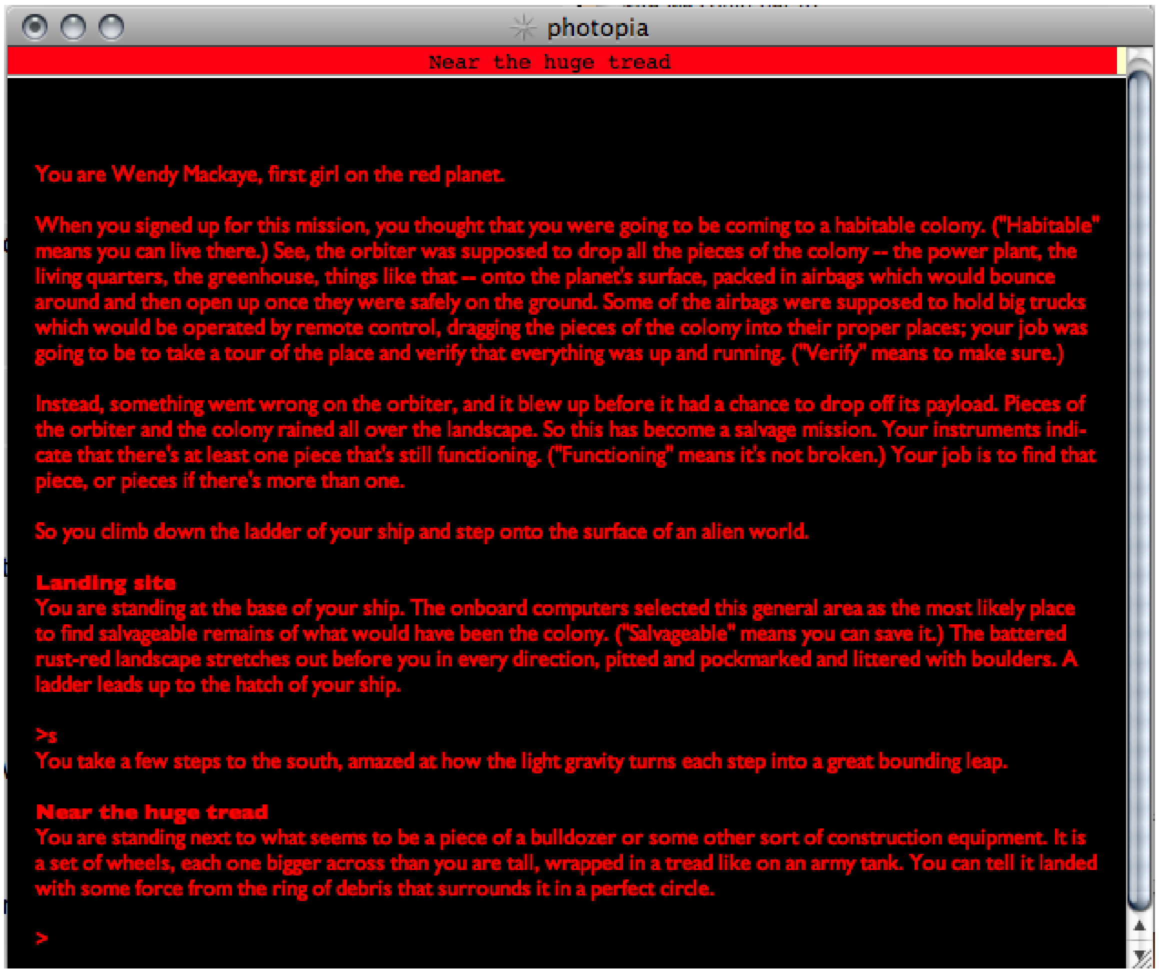
Chapter 15 – An Interactive Tragedy: *Photopia*

Many of the stories recommended on this web site are similar in a number of ways. Most include puzzles that kids like, most are not especially difficult as works of literature, and all have happy endings. Indeed, many have been crafted explicitly for young readers.

Adam Cadre's masterful IF novella, *Photopia*, is entirely different. It includes no puzzles in the usual sense of the term, drawing in students and other readers in a variety of creative and compelling ways. It is so challenging as literature that even experienced readers sometimes ask whether its final scene can really be the end of the story. It is an extraordinarily sad tale whose main character is a middle school student –- sad enough to warrant an early warning from the teacher that this is not a happy tale. And it is not targeted at kids -– in fact, some teachers who use it may choose to censor some profanity that appears in some versions of its first brief scene.

Photopia is simply inaccessible to most pre-college readers when they are working independently; but, with the right kind of guidance, the same students can enjoy the story and learn a great deal from it. The teacher's approach, however, must vary considerably from other classroom approaches to interactive fiction.

Though *Photopia* offers no puzzles of the usual sort, it is a puzzling tale, in that it challenges the reader to determine how several apparently disparate plot threads are integral parts of the same story. Students can help themselves to follow these plot threads by listing each scene of the story on paper, so that the plot threads can eventually emerge and be integrated. The story helps with this procedure by associating a series of colors with the scenes that constitute one of the principal plot threads. Also, since the story's point of view shifts from scene to scene, the author has implemented a "Who am I?" command.


photopia

Near the huge tread

```
You are Wendy Mackaye, first girl on the red planet.

When you signed up for this mission, you thought that you were going to be coming to a habitable colony. ("Habitable"
means you can live there.) See, the orbiter was supposed to drop all the pieces of the colony -- the power plant, the
living quarters, the greenhouse, things like that -- onto the planet's surface, packed in airbags which would bounce
around and then open up once they were safely on the ground. Some of the airbags were supposed to hold big trucks
which would be operated by remote control, dragging the pieces of the colony into their proper places; your job was
going to be to take a tour of the place and verify that everything was up and running. ("Verify" means to make sure.)

Instead, something went wrong on the orbiter, and it blew up before it had a chance to drop off its payload. Pieces of
the orbiter and the colony rained all over the landscape. So this has become a salvage mission. Your instruments indi-
cate that there's at least one piece that's still functioning. ("Functioning" means it's not broken.) Your job is to find that
piece, or pieces if there's more than one.

So you climb down the ladder of your ship and step onto the surface of an alien world.

Landing site
You are standing at the base of your ship. The onboard computers selected this general area as the most likely place
to find salvageable remains of what would have been the colony. ("Salvageable" means you can save it.) The battered
rust-red landscape stretches out before you in every direction, pitted and pockmarked and littered with boulders. A
ladder leads up to the hatch of your ship.

>s
You take a few steps to the south, amazed at how the light gravity turns each step into a great bounding leap.

Near the huge tread
You are standing next to what seems to be a piece of a bulldozer or some other sort of construction equipment. It is
a set of wheels, each one bigger across than you are tall, wrapped in a tread like on an army tank. You can tell it landed
with some force from the ring of debris that surrounds it in a perfect circle.

>
```

For each scene, students can profitably record the results of the “Who am I?” query, the color associated with the scene, if any, and a brief description of the action. As they look back at their writings about the scenes, the students will find it fairly easy to identify three plot threads by the time they reach the half-way point of the story. One of these will consist of just one scene, involving two drunken college students who cause a traffic accident. The second, combining several color-associated scenes, tells an apparently-unrelated science-fantasy tale; and the third offers a number of episodes from the life of Alley Dawson, a very kind and bright young girl. Eventually, students will be able to see how all three threads constitute a single tragic plot.

Photopia has gained a reputation as one of the finest work of IF ever created. It is surely challenging for both teacher and student, but it works well in the classroom, and it is more than worth the effort that it demands.

Photopia is available, in several versions, at the Interactive Fiction Database (http://ifdb.tads.org). The “Competition Version” is the only one that contains strong profanity in the opening scene.

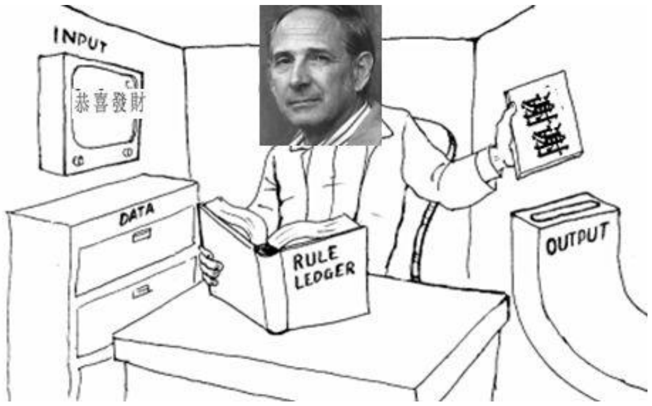
Chapter 16 -- An Interactive Fiction About Middle School Students: *The Enterprise Incidents*

IF and the Teaching of Content

This book deals, almost exclusively, with helping students to read, write, and think more effectively. Still, some very fine IF stories have focused on teaching content, too. *1893: a World's Fair Mystery* (2002) recreates of the most important events in the history of popular culture, an event that many history classes deal with, at least to some degree. The story's author, Peter Nepstad, deliberately developed this fine work to appeal to educators, though some of its plot twists may be a bit strong for younger students. A bit of traditional poetry might cause a stir in a principal's offices, too:

When Adam delved and Eve span,
Who was then the gentleman?

1893 is a formerly-commercial product, now available at the IF Database (http://ifdb.tads.org)



The Chinese Room by Harry Giles and Joey Jones (2007) offers entertaining lessons in philosophy by giving the reader interactive versions of famous thought experiments, starting with John Searle's famous critique of the Turing Test. It's pretty hard to make these experiments interactive, though, since they aren't, by nature, puzzle-oriented. Searle's “Chinese Room,” for instance, in its IF version, becomes an attempt to escape from the chamber, while Searle's experiment, as he wrote it, has nothing to do with escaping. Nevertheless, you can teach yourself, and your students, some broad and important lessons in philosophy with this game.

Schools often deal with mythology in one way or another, and so does interactive fiction. *Myth Tale,* by Temari Seikaiha (2002) presents several Greek myths in interactive form, along with some funny bits about cats. *Arthur: the Quest for Excalibur* by Bob Bates (Infocom, 1989), one of the most acclaimed If stories for young people, stays quite close to the traditional legends.

Trinity by Brian Moriarty (Infocom, 1985) has earned extraordinarily high praise from IF readers throughout its long history. Though the tale features elements of poetic fantasy, it can teach its reader about the history of atomic weaponry through dramatic scenes set at actual historical settings of nuclear detonations, in Nevada (1945 and 1974), Nagasaki (1945), Siberia (1949), and elsewhere. Trinity is a difficult story with challenging problems, but it's always rewarding.

A Story About Getting by in Middle School

The Enterprise Incidents: a Middle School Fantasy, is a story about middle school kids and teachers. Its puzzles are fairly easy, though its reading level can be challenging at times. “The Enterprise Incidents” was written by Brendan Desilets, who offers this book. Beta testers of *The Enterprise Incidents* were Sophie Fruehling, Jeff Nassiff, Matt Carey, and Sean Desilets. Hints and a walkthrough are available within the story.

The Enterprise Incidents tries to teach some content that might be of use to students aged eleven to fourteen, especially those who get involved with unusual tasks like delivering Valentine's Day candy grams to other students during the school day. It instructs students, for instance, to follow the sometimes-arcane procedures that many schools require for kids who are in the halls during classes, and it provides some puzzles that hinge on unanticipated contingencies, such as encountering a student who claims to have purchased a candy gram but is not on the list of recipients.

You can download the z-code (plain text) story file for *The Enterprise Incidents* from the Interactive Fiction Database. For more information on using a story file, read the “Acquiring Interactive Fiction” chapter in this book.

Or, if you prefer, you can probably read *The Enterprise Incidents* on line, using the “Play Online” button on the story's Interactive Fiction Database page.

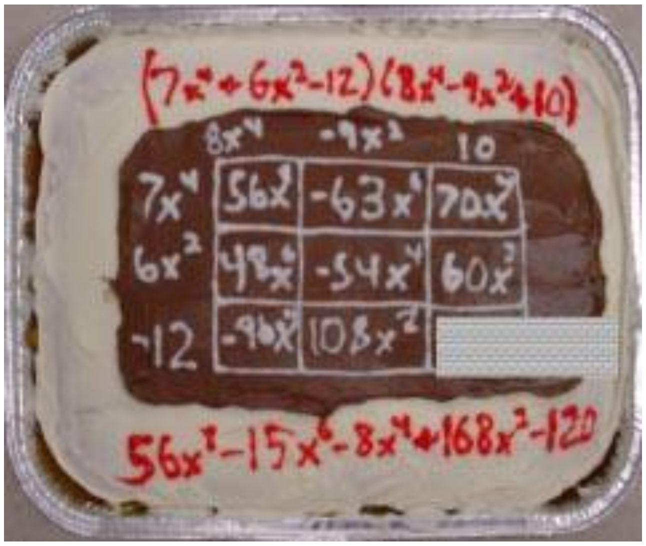
Another version of *The Enterprise Incidents* includes some on-screen maps and other graphical elements. You can download the story file for this version, from the IF Database, too. This variation of the story uses an IF-creation tool called Glulx and requires an interpreter that is different from the ones used with other stories featured in this book. The interpreter called “Gargoyle” works best for this story. Gargoyle is available at http://ccxvii.net/gargoyle/. Hints for this version of the story are at http://if1.home.comcast.net/~if1/hints.html.

The pictures and maps that appear below may add to your enjoyment of the story, if you are reading the z-code version. These graphics appear in the Glulx version.

The mural in the cafeteria, where the story begins

A candy gram card

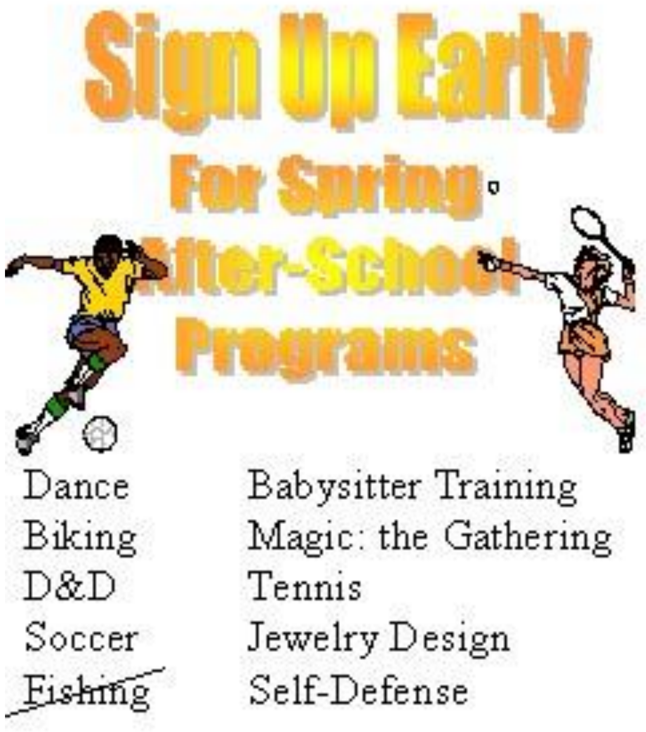
A decorated locker



A math cake from Ms. Garrulous' class

The cover of Ed Dibbles' science folder

Meghan Mascaras' science folder



Sign Up Early

For Spring

After-School

Programs

Dance
Biking
D&D
Soccer
~~Fishing~~

Babysitter Training
Magic: the Gathering
Tennis
Jewelry Design
Self-Defense

A poster from outside the office. Fishing is cancelled because of Mr. Pisces' need for a bone marrow transplant.

The
Silas
Exciter

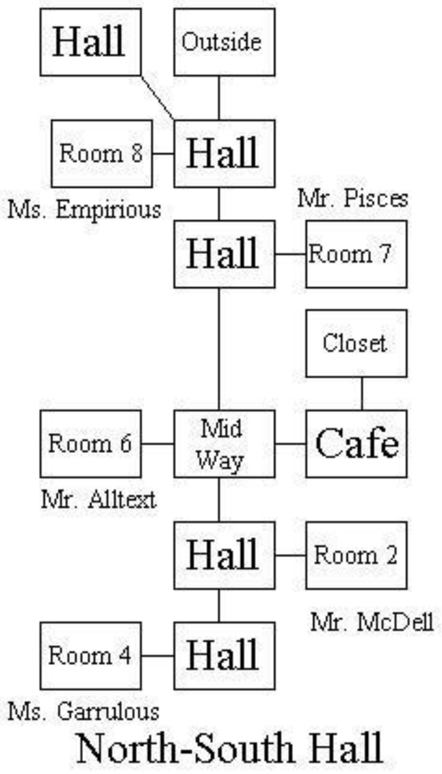
An art project by Silas Gibber

Lightning above a thistle

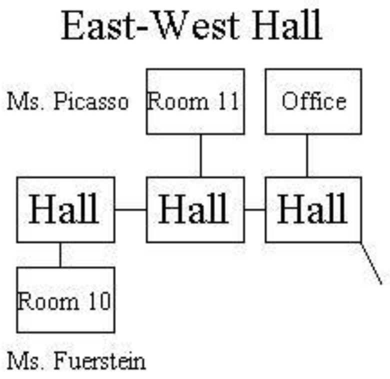
Sprung from the guts

of an old, grey stump

Jim Hastely's riddle poem



A Map of the North-South Hall



A Map of the East-West Hall

To reach the North-South Hall,
go southeast from the hall
near the Office.

Walkthrough

This is a walkthrough of *The Enterprise Incidents: A Middle School Fantasy* by Brendan Desilets, Release 2. Like many of its fellows, this walkthrough does not explore many of the avenues that make the story enjoyable. The player's inputs are preceded with the > symbol. At times, in the course of the story, characters will initiate conversations. These are good occasions for chatting with the characters in question, but the walkthrough often does not specify which conversational choices are best.

```
>open door
>talk to alltext
>3
>g
>3
>g
>1
>1
>talk to queenie
>1
```

```
>g
>1
>g
>1
>look at mural
>look at queenie
>4
>n
>take list
>take envelope
>read list
>s
>close door
>w
>n
>e
>give gram to meghan
>w
>s
>s
>e
>w
>s
>talk to Stephanie
>1
>g
>2
>look at Stephanie
>put gram in basket
```

(Queenie will initiate some conversation here.)

```
>open door
>w
>give gram to Danielle
>say -120 to garrulous
>e
>n
>n
>n
>n
>wear badge
>open door
>knock on door
>talk to empirious
>1
>w
>take jar
>give gram to ed
>e
>nw
>talk to woman
>1
>w
>w
>e
>n
>talk to reunite
>1
>g
```

```
>3
>s
>w
>w
>s
>give gram to Alicia
>give gram to andrea
(Queenie will invite a conversation here.)
>n
>e
>n
>say rat to Picasso
>give gram to silas
>s
>e
>n
>give pass to reunite
>s
>se
>s
>s
>w
>1
>give gram to jim
>say firefly to jim
>look at list
>talk to alltext
>2
>e
```

```
>give gram to queenie
>4
>e
>open door
>n
```

Chapter 17 – Acquiring Interactive Fiction

Fortunately, interactive fiction is inexpensive and fairly easy to find. Most IF stories that have been written since 1990 are free of charge and available via the Web, especially through the Interactive Fiction Database (http://ifdb.tads.org). The IF database also offers links for running many stories in a Web browser, and it presents a good set of instructions for getting the same stories running on the user's local computer. However, running stories on a local computer often works more smoothly than reading the same stories online, especially when it comes to saving one’s progress. But getting the stories to run on local PC's, Macs and Linux computers is a multi-step process.

To run an interactive fiction on a particular computer, you usually need two pieces of software on your computer. One piece of software is called an interpreter. The interpreter you need for a particular story depends mainly on two factors, the kind of computer you have and the tool the author used to create the story.

Interpreters

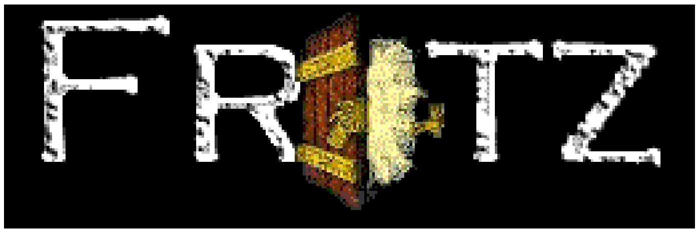
Each Internet-distributed story recommended on this site was created with one of five tools. These tools are called Inform, TADS, Hugo, Quest, and Adrift.

Authors who write with Inform sometimes create stories in a format called z-code. Other Inform writers use an extension called Glulx, if they want to create very large stories or stories with multimedia elements. Interpreters for stories made with these tools are available for just about any kind of computer, even older, more unusual ones. If you have one Inform interpreter, one Glulx interpreter, one TADS interpreter, one Hugo interpreter, and one Adrift interpreter on your computer, you have enough interpreters to run almost all the stories recommended on this site.

Quest stories are a little different. They're often read online, and their only offline interpreter is the same Quest program that's used to write the stories. This program runs only on Windows. Beyond these five authoring systems, there are other IF-development systems, with their own interpreters, but, once you get the hang of using one interpreter, you'll probably be able to handle them all.

There's an excellent list of interpreters at http://www.ifwiki.org/index.php/Interpreter

You can simplify the interpreter-gathering process by using a tool that combines a number of interpreters into one application. The most widely used of these is Gargoyle, which runs on Windows, Mac, and Linux. The Linux version is sometimes called “gargoyle-free.” Many Mac users like two other multi-interpreter programs, Zoom and Spatterlight. These multi-interpreter systems are fine for most users, though they make it a bit tricky to change font sizes and styles. In addition, they will, in general, not run one of the highly-recommended stories in this book, *Arthur: the Quest for Excalibur.* To read *Arthur,* on Windows, you'll need an easily-installed interpreter called Windows Frotz (http://www.davidkinder.co.uk/frotz.html). Frotz is also available for Linux and Mac, though the Mac version is very difficult to install successfully. *Arthur,* however, will run on Zoom for the Mac, if the user renames the story file from ARTHUR.ZIP to ARTHUR.Z8.



You can download interpreters from the IF Archive. The IF Database links to them, too.

Google Play and iTunes offer a limited spectrum of interpreters for IOS and Android.

Story Files

To run a story, you need, in addition to an interpreter, a story file. The story file you need depends only on which story you want to run – it doesn't matter what kind of computer you have.

So, if you want to read the TADS game *The One That Got Away,* using a Macintosh, you need the right story file and the right interpreter, which, in this case would be a TADS interpreter for the Macintosh, or a multi-interpreter program for the MAC. Gargoyle, Zoom, and Spatterlight will all work fine.

Where Can I Get These Things, Again?

You can find links to plenty of interpreters and story files at the Interactive Fiction Archive and the Interactive Diction Database. Downloading, at present, is usually just a matter of left-clicking or right-clicking on a link. Downloaded files are often compressed and/or archived.

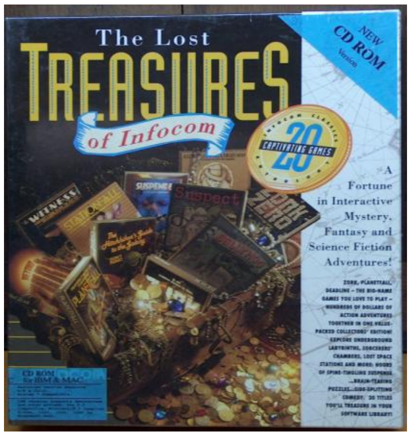
Newer computer operating systems make decompressing files quite easy, by showing compressed files as folders that can be opened like other folders or extracted to show their contents.

If For Sale

A few contemporary works of IF are commercial products, always reasonably priced. Textfyre.com sells *Jack Toresal and the Secret Letter* *(*http://textfyre.itch.io/jack-toresal-and-the-secret-letter) and *The Shadow in the Cathedral* *(*http://textfyre.itch.io/the-shadow-in-the-cathedral*.)* You can buy the brilliant *Hadean Lands* at its own website. And several works of IF are for sale at iTunes and Google Play.

Things get a bit trickier when you're looking for classic works from the 1980's. Most of the best works from this era were published by a company called Infocom. You can buy most of the Infocom stories, (thirty-three, to be exact) for both PC and Macintosh computers, in one magnificent collection called *Classic*

*Text Adventure Masterpieces of Infocom (*often called *Masterpieces of Infocom).* However, *Masterpieces* has become quite difficult to find in the last few years, and since it dates from the days of DOS, it doesn't always play well with modern computers. Fortunately, most of the story files from Infocom, the ones that have the extension .dat, will run on multi-format interpreters like Gargoyle and Spatterlight. The Infocom stories that have the .zip extension (*Arthur*, for example) will run on the z-code interpreter called “Frotz.”


The Lost
TREASURES
of Infocom
NEW CD ROM Version
20 CAPTIVATING GAMES
A Fortune in Interactive Mystery, Fantasy and Science Fiction Adventures!

Masterpieces, along with less comprehensive (and less pricey) collections like *The Lost Treasures of Infocom,* is often available at ebay and at Amazon.com. Just search ebay or Amazon, using "Infocom" as your search term. In early 2015, the lowest available price for *Masterpieces* seems to be $70.00 US.

Abandonware

It is also possible, though not really legal, to download all the Infocom stories from the Web, under the concept of abondonware. Those who offer the stories in this way often include disclaimers like this one, from Achim J. Latz:

“Copyright by Infocom, Inc. Provided for non-commercial use only, with the sole intent of making information available that would otherwise be lost. To whomever presently holds the copyright to the information contained in this page: if you think the existence of this page violates your copyright, please complain to achim@latz.org and this page will be removed.”

One site that offers the Infocom stories in this way is called My Abandonware (http://www.myabandonware.com). This site urges its users to try to purchase the software it offers, before downloading it for free. Why would one do this? In addition to staying within the law, a person who downloads from My Abandonware will probably be able to run the story on a modern computer more easily.

Most abandonware sites are oriented toward Windows and DOS, but, in the case of Infocom stories, the story files will work on all kinds of computers. DOS programs, in general, can run on practically all modern computers, using the software emulator called DOSBox. DOSBox is not the most intuitive program for users of modern graphical interfaces, but it works very well on Windows, Mac OS X, and Linux. Once you have DOSBox working, you can use it to run Infocom stories, including *Arthur,* without having to deal with an interpreter.

Raiders of the Lost Software

People who don't like obtaining software illegally have often mused, “Wouldn't it be great if a publisher like Activision, which now owns the Infocom stories, could just sell them for five dollars apiece? Acitivision, which now makes no money at all on the Infocom titles, would earn a few dollars, and readers would be happy, too.”

Yes, it would be great if Activision had a such a way to act on its own self-interest, but, in the current era of app stores and repositories, the software publishers **already have** an easy way to sell their back titles for a few dollars each.

Why don't they? Perhaps they're just a bit slow to react. By the time you read this, maybe you'll be able to buy *Plantefall*, tricked out for your great new phone or laptop, for a sum of money that you'll never miss.

Fun and Learning With Interactive Fiction (Mostly for Kids)

Chapter 18 – What is IF?

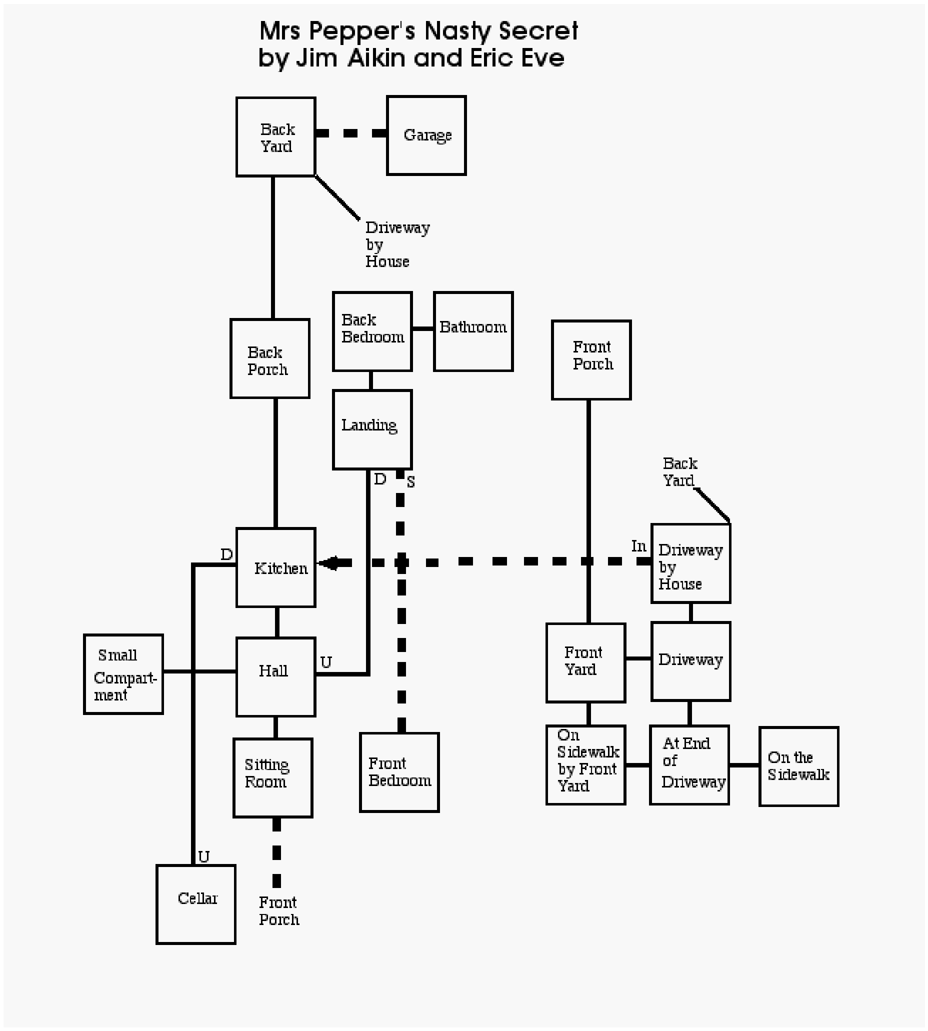
Interactive fiction, or IF, is a kind of story in which the reader plays the part of an important character, deciding, most of the time, what the character will do. By typing ordinary English sentences at a computer keyboard, the reader or, frequently, a group of readers, decide where the main character will go, what objects he or she will pick up and use, how he or she will solve problems, and how he or she will behave toward other characters. An interactive fiction story is a lot like a video game in words.

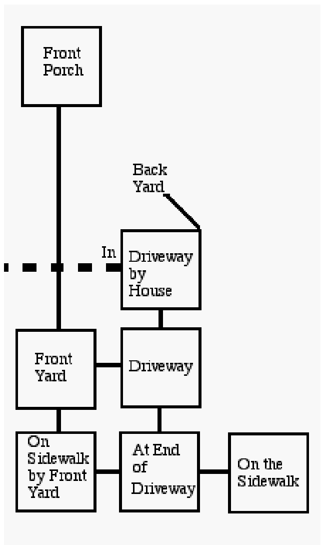
Often, when people talk about interactive fiction, also known as IF or adventure gaming, they are thinking about stories that are told mainly through words, not pictures. Sometimes, though, people use the term “graphical interactive fiction” when they mean stories that are told largely through pictures. Famous examples of graphical interactive fiction include *Myst* and *Riven*.

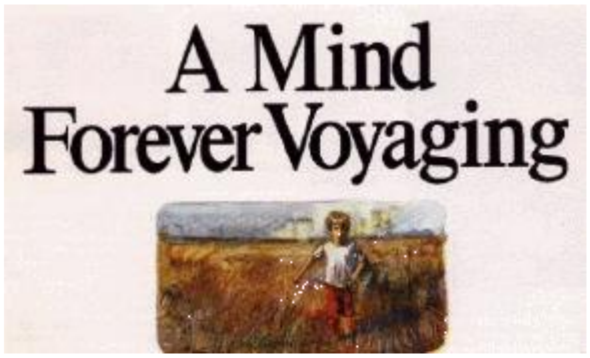
At other times, people talk about “choice-based interactive fiction.” In this kind of story, the reader makes choices by clicking on links, not by typing commands in ordinary language.

This book is about the kind of interactive fiction that uses mostly words to tell its story. It's also about the kind in which the reader types in commands, rather than clicking on links. Typing in commands allows the reader to have more freedom in deciding what her or she wants to do. It lets the writer write in a more open-ended way, too.

Would you like to try some IF right now? If your browser is Java-enabled, you can try out lots of stories online, including a really good one called *Mrs. Pepper's Nasty Secret.* One of its Web pages *is* at http://ifdb.tads.org/viewgame?id=dcvk7bgbqeb0a71s. Just click on the “Play Online” link to start. You can get hints by typing “hint,” and helpful maps appear below. However, you may want to read about communicating with interactive fiction before you attempt one of the stories. You’ll find a section about communicating with IF later on in this chapter.


Mrs Pepper's Nasty Secret
by Jim Aikin and Eric Eve
Back Yard
Garage
Driveway by House
Back Bedroom
Bathroom
Back Porch
Front Porch
Landing
D
S
Back Yard
In
Driveway by House
D
Kitchen
Small Compart-ment
Hall
U
Front Yard
Driveway
Sitting Room
Front Bedroom
On Sidewalk by Front Yard
At End of Driveway
On the Sidewalk
U
Cellar
Front Porch


Front Porch
Back Yard
In
Driveway by House
Front Yard
Driveway
On Sidewalk by Front Yard
At End of Driveway
On the Sidewalk



Communicating in Interactive Fiction

What’s wrong with this picture? If you’re tried interactive fiction and hated it, you may think you know. Actually, may kids get off to bad start with IF when someone says, “Look at this great program! All you have to do is type in what you want to do. It’s just like reading a novel!” But when you try a story yourself, you seem to get nothing but error messages that make very little sense.

In truth, even the best IF programs can deal with only a few kinds of English sentences.
However, interactive fiction authors are really trying to help. Sometimes, the help is built into the story itself. For example, the story might display this text early on: “Type 'about' for more information about this story.”

At other times, the help, or some of it, will be in separate the instructions that come with the story. If you use these aids, you’ll find communicating with IF programs much easier.

In general, try these suggestions. All works of interactive fiction, even the very earliest ones, can recognize sentences like “take coin,” which the story will consider to mean, “I want to take the coin.” The IF stories recommended here can recognize many more kinds of sentences, but experienced readers often keep the two-word pattern in mind, anyway. For example, IF stories can now recognize “Take the gold coin,” “Take the gold coin from the fountain,” or “Take the gold coin and give it to the librarian.”

In addition, many works of IF can recognize at least some simple questions that begin with “who,” “what,” or “where.”   Most stories also use a variety of useful abbreviations, including “g” for “again,” “z” for “wait,” “i” for “inventory of what I’m carrying,” “l” for “look,” “n” for “go north,” “s” for “go south,” and “u” for “go up.”

Conversing with other characters in IF can be lots of fun, but it can be frustrating, too, unless you keep in mind several patterns that most of the stories can understand. Directly talking to a character with a command will often work; for example, “Miss Voss, tell me about the magic stone.” Also, a reader can often make progress by asking or telling a character about something, as in “Ask the bartender about the vampire.” Frequently, a story will interpret a single word to mean that the character says the word. In other words, “hello” will often mean the same as “say ‘hello,’” though it is necessary to type out ”say hello.” “Talk to Mrs. Pepper (or whatever the character is called)” also works in many stories.

Getting Un-Stuck

Another objection? You learned to communicate with the program just fine but soon became “stuck” in trying to solve a problem and finally decided to give up? Well, that sort of thing happens to all of us. In fact, figuring out tough problems is part of the fun of IF. And, as a matter of fact, few readers get through a whole piece of interactive fiction without help from someone else. A problem that seems impossible to me may be easy for you, and so interactive fiction often ends up as a social activity.

What if there’s no one available to work with? Then use the hints and “walkthroughs” that you can find on the World Wide Web, especially at the Interactive Fiction Database (http://ifdb.tads.org). A Google or DuckDuckGo search for “Mrs. Pepper walkthrough” will often work, too. But try not to look at the hints until you’ve really tried to solve the problem yourself.

If you’d like just right amount of hinting, from people who know how to avoid giving away too much, try the Interactive Fiction Forum at

http://www.intfiction.org/forum/.

Here are some Web addresses for walkthroughs for stories recommended on this site:
Wishbringer – http://www.gamefaqs.com/pc/564461-wishbringer/faqs/13727
Moonmist – http://www.gamefaqs.com/pc/564468-moonmist/faqs/1619
Arthur: the Quest for Excalibur – http://www.ifarchive.org/if-archive/infocom/hints/solutions/arthur.txt
A Bear's Night Out – http://ifarchive.jmac.org/if-archive/games/competition97/inform/bear/bear.sol
The Magic Toyshop – *http://ifarchive.giga.or.at/if-archive/solutions/magictoy.sol*
Lost Pig – http://ifarchive.giga.or.at/if-archive/games/competition2007/zcode/lostpig/walkthru.txt

Clash of the Type-Ins

The best way to learn about IF is to read a story with someone who has experience with interactive fiction. But you may not know anyone who's like that. Still, you probably do have a way to listen to experienced readers as they make their way through several IF stories. This free-and-easy teacher takes the form of a very funny podcast called “Clash of the Type-ins” (http://rcveeder.net/clash/) In each episode, hosts

Ryan Veeder and Jenni Polodna, both experienced IF readers and writers, read a story together. Most episodes include a third participant, the author of the story that's being read.

From the “Clash” website, here's a list of some of the best episodes for kids. You can't go wrong with any of them.

EPISODE ONE: *You've Got a Stew Going!* (February 6, 2014)
In this, the first episode, Jenni Polodna plays Ryan Veeder's first game, a game about rats. Jenni says the word 'titular' kind of a lot. If horses ran the world, keyboards would be weird.

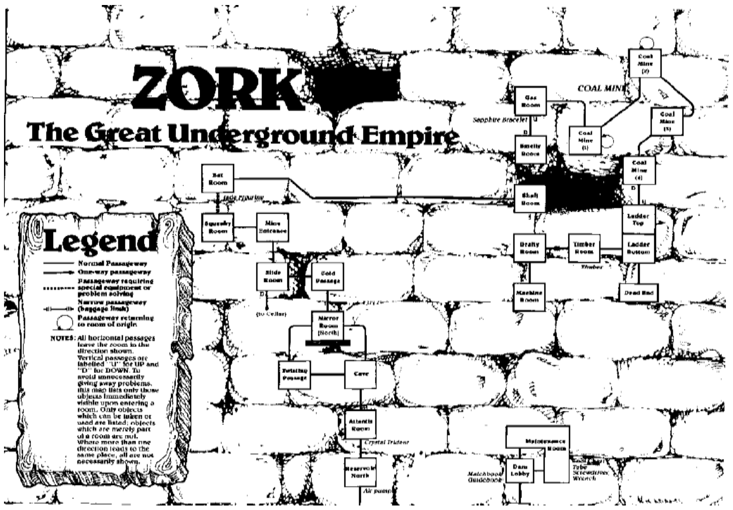
EPISODE THREE: It (April 16, 2014)
Jenni and Ryan play Emily Boegheim's *It. It* is the name of the game. Cultural gaps are bridged. Murder is considered.

EPISODE NINE: *A Day for Fresh Sushi* (January 9, 2015)
Jenni and Ryan play Emily Short's Speed-IF about an angry fish.

EPISODE TEN: *Bronze* (January 10, 2015)
Emily Short brings us one of her fractured fairy tales and Ryan works himself into a fanboy frenzy. Jenni tries to remember Dom DeLuise's first name. Shameful

lunchtime secrets are revealed. This episode includes some interesting discussion of IF in general.

Chapter 19 – Why Kids Like Interactive Fiction


ZORK
The Great Underground Empire
Legend
Normal Passageway
One-way passageway
Passageway requiring special equipment or problem solving
Narrow passageway (baggage limit)
Passageway returning to room of origin
NOTES: All horizontal passages leave the room in the direction shown. Vertical passages are labelled "U" for UP and "D" for DOWN. To avoid unnecessarily giving away problems, this map lists only those objects immediately visible upon entering a room. Only objects which can be taken or used are listed; objects which are merely part of a room are not. Where more than one direction leads to the same place, all are not necessarily shown.
Coal Mine (2)
COAL MINE
Gas Room
Sapphire Bracelet
Coal Mine (3)
Coal Mine (1)
Smelly Room
Coal Mine (4)
Bat Room
Jade Figurine
Shaft Room
Ladder Top
Squeaky Room
Mine Entrance
Drafty Room
Timber Room
Ladder Bottom
Timber
Slide Room
Cold Passage
Machine Room
Dead End
(to Cellar)
Mirror Room (North)
Twisting Passage
Cave
Atlantis Room
Crystal Trident
Maintenance Room
Reservoir North
Air pump
Matchbook Guidebook
Dam Lobby
Tube Screwdriver Wrench

Since 1985, the author of this web site has introduced about a thousand young people, aged eleven through twenty, to interactive fiction. Most of them like it. In fact, it is the most popular form of literature with most.

Lots of kids and young adults tell me that they like interactive fiction mainly because it's an exciting way to read a story, a way that lets them feel very active and involved. They enjoy using IF to gain experience with all of the major elements of literature, such as plot, setting, and point of view.

Many young people also like the problem-solving that comes with the IF experience. These folks appreciate interactive fiction because it challenges them to recognize and solve problems, in ways that no textbook seems able to match.

Here's what young people themselves say about interactive fiction and about their favorite stories.

"Interactive fiction can be fun because it is like a game. You get to be a character and explore, and figure out puzzles. It is good because it gets people to use their imaginations."

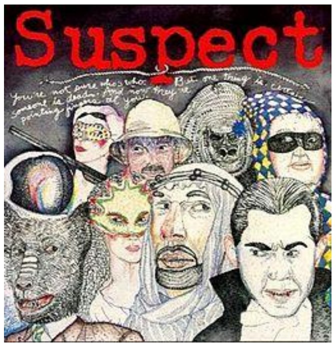
"Kids like IF because it's fun. It gets you thinking and helps you learn strategy."

"Kids choose IF because they get to decide what happens next."

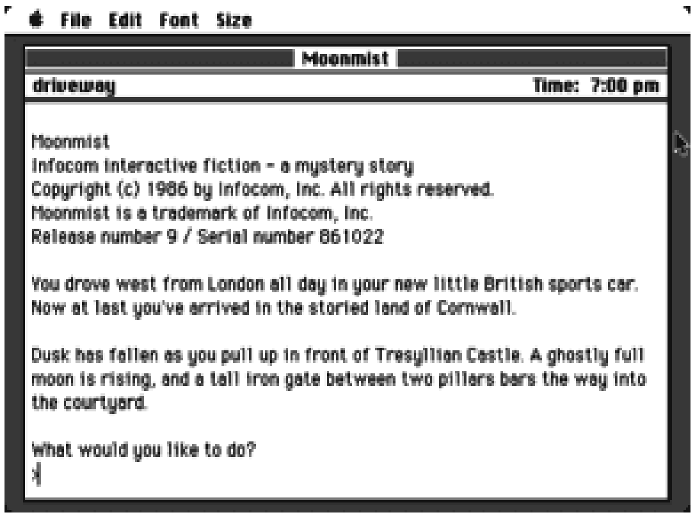
"I would recommend *Zork I* because it is somewhat challenging and you need to use lots of thought. It is a very exciting game, too. You get to try lots of things."

"The book/game I am reading, *Zork I,* is an adventure story. You start off as a regular person, and, as you get more into the game, you get more experience. You find all sorts of different treasures, and it is really FUN."

"I like the way you participate, instead of just reading a book. IF also makes you *think*."

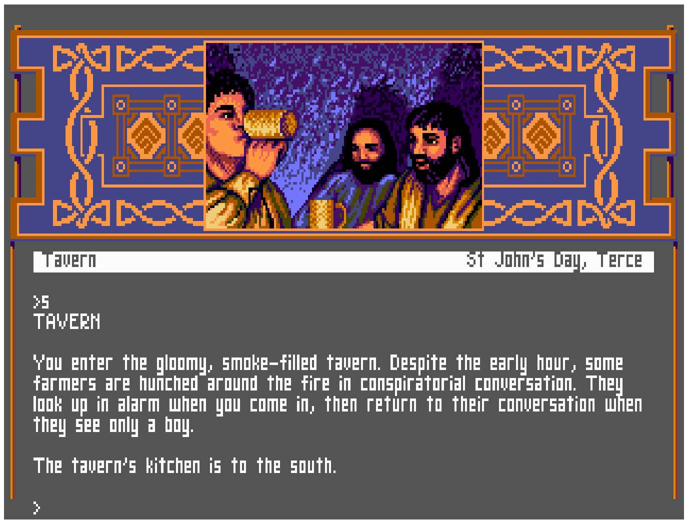


“I like *Suspect* the most of all the interactive fiction I have done so far. I like it mostly because I like mysteries and solving puzzles.”



```
Moonmist
Infocom interactive fiction - a mystery story
Copyright (c) 1986 by Infocom, Inc. All rights reserved.
Moonmist is a trademark of Infocom, Inc.
Release number 9 / Serial number 861022

You drove west from London all day in your new little British sports car.
Now at last you've arrived in the storied land of Cornwall.

Dusk has fallen as you pull up in front of Tresyllian Castle. A ghostly full
moon is rising, and a tall iron gate between two pillars bars the way into
the courtyard.

What would you like to do?
>
```

“My favorite piece of literature this year is *Moonmist.* I like it a lot because it was my first piece of interactive fiction. It also had a great plot.”



```
Tavern                                        St John's Day, Terce

>s
TAVERN

You enter the gloomy, smoke-filled tavern. Despite the early hour, some
farmers are hunched around the fire in conspiratorial conversation. They
look up in alarm when you come in, then return to their conversation when
they see only a boy.

The tavern's kitchen is to the south.

>
```

"My favorite piece of reading this year is *Arthur: the Quest for Excallibur.* I like it because it is a good adventure story and I like adventures, and because I like King Arthur stories. In this one, I like how you can change into different animals."

Reading IF Aloud

In my twenty-two years of work as a middle school teacher, I found that students generally enjoy interactive fiction a great deal. In fact, about thirty percent of my students chose IF, over all other options, for individual silent reading. However, they liked IF much more when they could read it aloud, together. When they could read IF together, about eighty-five percent of the kids like IF more than other literature. Why did this form of literature motivate oral reading more than silent reading? Three factors came together to produce this result: solving problems together, reading stories in bite-sized chunks, and a feeling of writing a story while reading it.

As this chapter has shown, students are very much aware that they like IF because of its gamelike problem-solving. They can also easily see that, with interactive fiction, there are lots of well-placed pauses for reader input. These pauses create excellent places to switch readers, at random, or otherwise.

Students aren't always aware, though, that they enjoy a sense of creating a story, rather than just reading it. When I ask them about this advantage, they report that they do feel that they are being creative when they read IF, and that they like this sense of writing a story while they're reading it.

If you'd like to explore this sense of creating a story further, have a look at “Joys of the Parser” in Chapter 2 of the “Teaching with Interactive Fiction” part of this book.

You can also find more information about what kids can learn from interactive fiction in “Teaching With Interactive Fiction.”

Chapter 20 – Top IF Stories for Kids

The Top Fifty-One Works of Interactive Fiction
(In One Person’s Opinion)
For People Aged 10-16, and Older

If you’re having trouble finding a story on the Interactive Fiction Archive, search for it in The Interactive Fiction Database (http://ifdb.tads.org). In fact, you may be happier if you simply start at the IF Database. At the IF Database, many stories, on their main pages, have a link that allows you to try the story online. To save your progress when playing online, type “Save,” follow the prompts. If you don't get any prompts, bookmark the page after you've typed “Save.” and then bookmark the page. For information on obtaining commercial stories from the publisher called Infocom, see the chapter on Obtaining Interactive Fiction from this book.

1. *Arthur: the Quest for Excalibur* by Bob Bates, a well-plotted version of the story of King Arthur as a boy, excellent for middle schoolers and older (ages 11 and up) (Available in *Masterpieces of Infocom* and other collections of Infocom stories, often Offered at ebay and Amazon)

2. *Lost Pig* by Admiral Jota, an award-winning, comic story in which the reader plays the role of an orc. (Available at the IF Archive and IF Database)

3. *Wishbringer* by Brian Moriatry, a rather gentle fantasy

adventure, excellent for middle school students (Available in *Masterpieces of Infocom*)

4. *The Firebird* by Bonnie Montgomery, a comic retelling of the Russian folk tale, excellent for kids aged eleven and up (Available at the Interactive Fiction Archive, and at the IF Database)

5. *Winter Wonderland* by Laura Knauth, a finely-crafted winter solstice story, excellent for middle schoolers (Available at the Interactive Fiction Archive, http://www.ifarchive.org and at the Interactive Fiction Database.)

6. *A Bear's Night Out* by David Dyte, a funny story of a teddy bear who comes to life (Available at the Interactive Fiction Archive, and at the Interactive Fiction Database)

EARTH AND SKY

7. *The Earth and Sky Trilogy–Earth and Sky* by Paul O’Brian, three comic superhero stories (Available at the Interactive Fiction Archive, http://www.ifarchive.org)

8. *The Earth and Sky Trilogy–Another Earth, Another Sky* by Paul O’Brian, three comic superhero stories (Available at the Interactive Fiction Archive, http://www.ifarchive.org)

9. *The Earth and Sky Trilogy–Luminous Horizon* by Paul O’Brian, three comic superhero stories (Available at the Interactive Fiction Archive, http://www.ifarchive.org)

10. *Mother Loose* by Irene Callaci, an amusing retelling of some classic fairy tales, excellent for middle school students (Available at the Interactive Fiction Archive, http://www.ifarchive.org)

11. *Mrs. Pepper’s Nasty Secret* by Eric Eve and Jim Aiken, a clever and lighthearted puzzle-fest, good for beginners. (Available at http://ifdb.tads.org/viewgame?id=dcvk7bgbqeb0a71s)

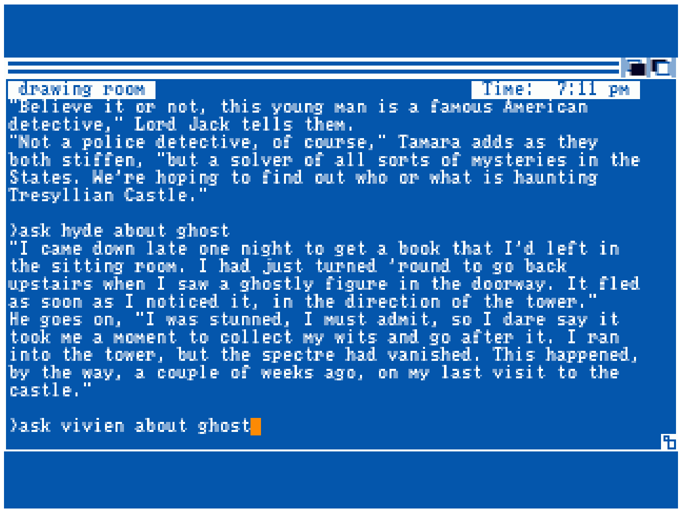
12. The Matter of the Monster by Andrew Plotkin, an inventive variation on the “Choose Your Own Adventure” type of story. Quite brief and very enjoyable.

13. Jack Toresal and the Secret Letter by Michael Gentry, an exciting interactive novel with a suspenseful plot and wonderfully interactive characters (Available for purchase athttp://textfyre.itch.io/jack-toresal-and-the-secret-letter)

14. Bonehead by Sean M. Shore, the mostly-true story of Fred Merkle, who made one of major league baseball’s most famous mistakes. (Available in the IF Archive and in the IF Database)

15. Aoteoroa by Matt Wingall, a lively tale of the modern world–with dinosaurs. (Available in the IF Archive and in the IF Database)

16. *History Repeating* by Mark and Renee Choba, a time travel story about a man who neglected an important assignment in his high school history class (Available at the Interactive Fiction Database, (http://ifdb.tads.org)



```
 drawing room                                  Time:   7:11 pm
"Believe it or not, this young man is a famous American
detective," Lord Jack tells them.
"Not a police detective, of course," Tamara adds as they
both stiffen, "but a solver of all sorts of mysteries in the
States. We're hoping to find out who or what is haunting
Tresyllian Castle."

>ask hyde about ghost
"I came down late one night to get a book that I'd left in
the sitting room. I had just turned 'round to go back
upstairs when I saw a ghostly figure in the doorway. It fled
as soon as I noticed it, in the direction of the tower."
He goes on, "I was stunned, I must admit, so I dare say it
took me a moment to collect my wits and go after it. I ran
into the tower, but the spectre had vanished. This happened,
by the way, a couple of weeks ago, on my last visit to the
castle."

>ask vivien about ghost
```

17. *Moonmist* by Jim Lawrence and Stu Gally, a mystery for kids, set in a Cornish castle, good for readers aged eleven and up (Available in *Masterpieces of Infocom*)

18. *Zork I* by Marc Blank and Dave Lebling, a fantasy treasure hunt (Available at the IF Database)

19. *The One That Got Away* by Leon Lin, a brief, funny story about fishing (Available at the Interactive Fiction Database)

20. *Small World* by Andrew Pontius, a fantasy about a

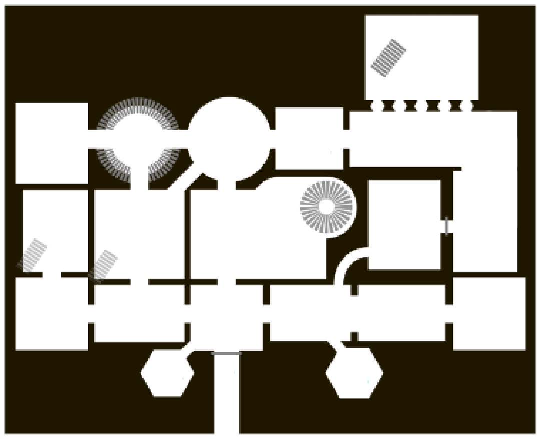
boy who sets a mixed up world right (Available at the Interactive Fiction Archive, http://www.ifarchive.org)

21. The Magic Toyshop by Gareth Rees, an imaginative puzzle fest (Available at the Interactive Fiction Database)

22. Mingsheng by Deane Saunders, a sometimes mystical story, always gentle story involving martial arts (Available at the Interactive Fiction Archive, http://www.ifarchive.org)

23. Dragon Adventure by William Stott, an adventure story for children aged nine and up (Available at the Interactive Fiction Archive, http://www.ifarchive.org)

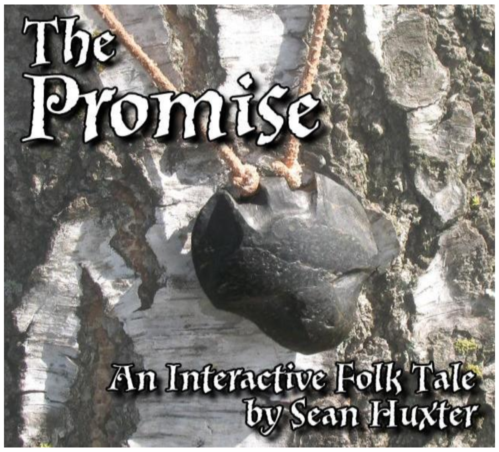
24. The Great Xavio by Reese Warner, a comic detective story (Available at the Interactive Fiction Archive, http://www.ifarchive.org)



25. Bronze by Emily Short, a Gothic retelling of Beauty and the Beast, with some relatively mild references to sexuality and, in its original version, other material that is inappropriate for younger children. (A specially-edited version for mature young people is available at http://inform7.com/teach/downloads/Bronze.z8.zip)

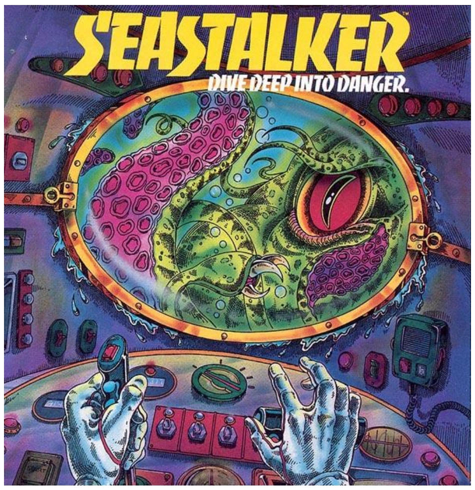
26. Dreamhold by Andrew Plotkin, a difficult fantasy story with help for beginners (Available at the Interactive Fiction Archive, http://www.ifarchive.org)

27. The Lost Islands of Alabaz by Michael Gentry, an action-packed fantasy tale (Available at the IF Archive and at the IF Database)



28. The Promise by Sean Huxter, a fantasy story that may remind you of a movie called *The Secret of Kells* (Available at the IF Archive and at the IF Database)

29. The Shadow in the Cathedral by Ian Finley and John Ingold, a fantasy puzzle fest with a compelling plot (Available for purchase at http://textfyre.itch.io/the-shadow-in-the-cathedral)



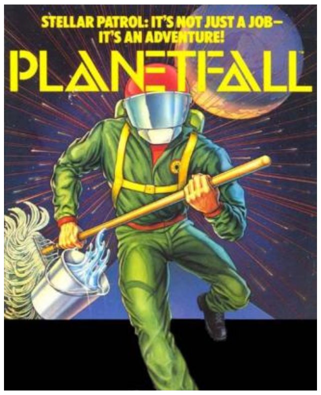
30. *Seastalker* by Stu Galley and Jim Lawrence, adventure in a futuristic submarine, good for middle school (Available in *Masterpieces of Infocom*)

31. The Orion Agenda by Ryan Weisenberger, a Star-Trek-like story of a distant planet (Available at the Interactive Fiction Archive, http://www.ifarchive.org)

32. Taco Fiction by Ryan Veeder (2011), a young man plans on making a big mistake, but thinks better of it. Recommended for older kids, sixteen and up. (Available at the Interactive Fiction Archive, http://www.ifarchive.org)

33. At the Bottom of the Garden by Adam Biltcliff, a comic tale of an invasion by miniature dragons (Available at the Interactive Fiction Archive, http://www.ifarchive.org)

34. Arrival by Stephen Granade, funny sendup of low-budget science fiction movies (Available at the Interactive Fiction Archive, http://www.ifarchive.org)



35. Plantefall by Steve Meretzsky, a comic science fiction story with some tough puzzles, good for high school (Available in *Masterpieces of Infocom*)

36. Zork III by Marc Blank and Dave Lebling, a fantasy story with interesting character interaction. (Available at http://www.csd.uwo.ca/Infocom/games.html)

37. Sherlock: the Riddle of the Crown Jewels by Bob Bates, a complex mystery with a realistic map of Victorian London (Available in *Masterpieces of Infocom*)

38. MythTale by Temari Seikaiha, a clever blending of several Greek myths (Available at the Interactive Fiction Archive, http://www.ifarchive.org)

39. *Worlds Apart* by Suzanne Britton, an extraordinarily rich fantasy story for skilled readers (Available at the Interactive Fiction Archive, http://www.ifarchive.org)

40. The Meteor, the Stone, and a Long Glass of Sherbet by Graham Nelson, a funny and cohesive fantasy tale (Available at the Interactive Fiction Archive, http://www.ifarchive.org)

41. *You've Got a Stew Going* by Ryan Veeder (2011), funny tale with a rat progagonist. (Available at the IF Archive and at the IF Database.)

42. Spider and Web by Andrew Plotkin, a spy story with an unusual plot structure and theme, ideal for skilled readers (Available at the Interactive Fiction Archive, http://www.ifarchive.org)

43. Wetlands by Clara Raubertas, a rather difficult fantasy story with lots of atmosphere (Available at the Interactive Fiction Database (http://ifdb.tads.org)

44. Moon-Shaped by Jason Ermer, a compelling, somewhat Gothic fairy tale. Though sad and intense, this story is accessible to a good many teenagers. (Available at the Interactive Fiction Archive, http://www.ifarchive.org)

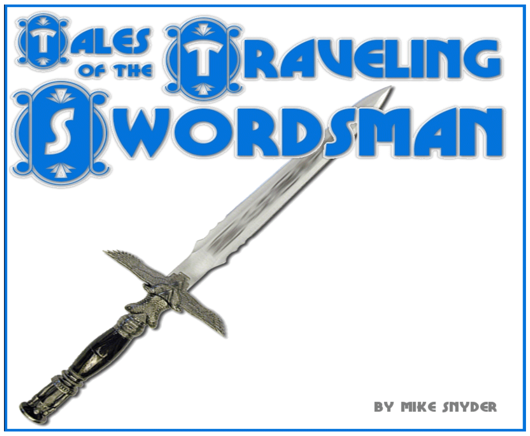
45. Zork II by Marc Blank and Dave Lebling, a challenging fantasy treasure hunt with an amusing wizard character (Available in *Masterpieces of Infocom*)

46. Hoist Sail for the Heliopause and Home by Andrew Plotkin (2010), a graceful space-exploration fantasy. Best for older students who have some literary-analysis skills. (Available at the Interactive Fiction Archive and the Interactive Fiction Database.)

47. Beyond Zork by Brian Moriaty, a detailed and difficult fantasy tale, with some options for shaping the player-character, good for high school (Available in *Masterpieces of Infocom*)

48. It by Emily Boegheim (2011) Read an appealing story and learn to play "sardines." (Available at the IF Archive and at the IF Database.)

49. The Beetmonger's Journal by Scott Sharkey, well-written, enjoyable tale of archeology (Available at the Interactive Fiction Archive, http://www.ifarchive.org)

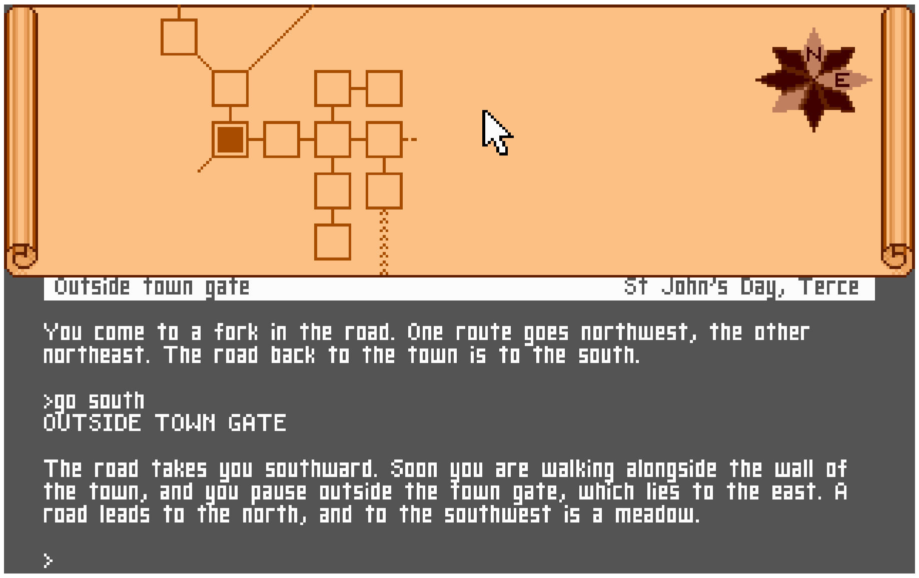


50. Tales of the Traveling Swordsman by Mike Snyder, a swashbuckling adventure with a twist at the end (Available at the Interactive Fiction Archive, http://www.ifarchive.org)

51. Six by Wade Clarke (2011), playful story with a six-year-old protagonist. Second Place in the 2011 Fall IF Competition. (Available at the IF Archive and at the IF Database)

Chapter 21 – A Classic Interactive Fiction: *Arthur: the Quest for Excalibur*

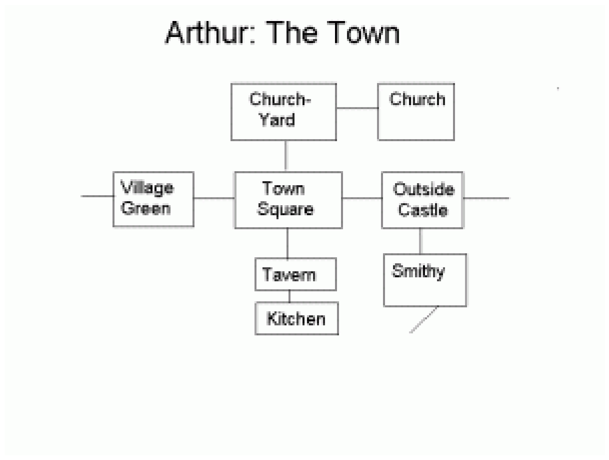
Arthur, the Quest for Excalibur is a great story for kids. It has an exciting plot, enthralling characters, clever puzzles, and lively humor. It even helps kids learn about British history and the King Arthur legends, which they often have to study in school anyway. And the story boasts on-line mapping, helpful hints, and fascinating notes, which appear in the hint menu at the end of the story.

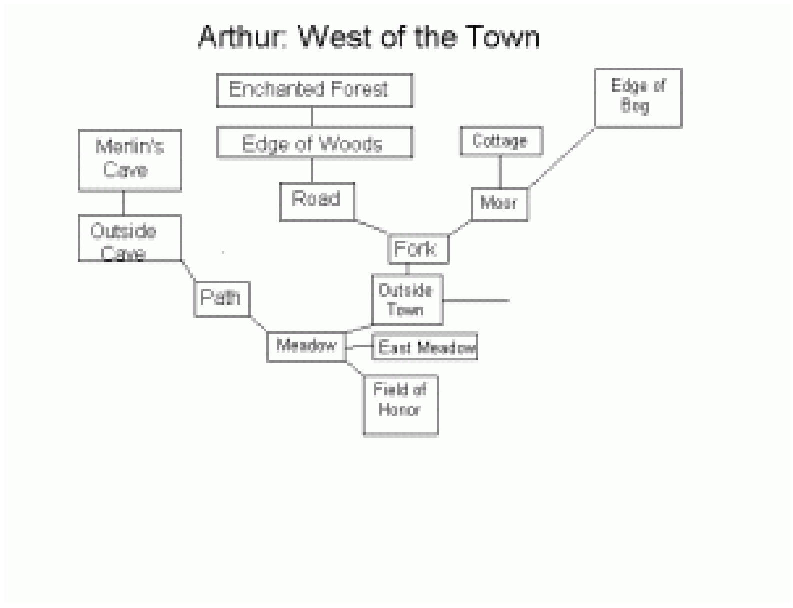


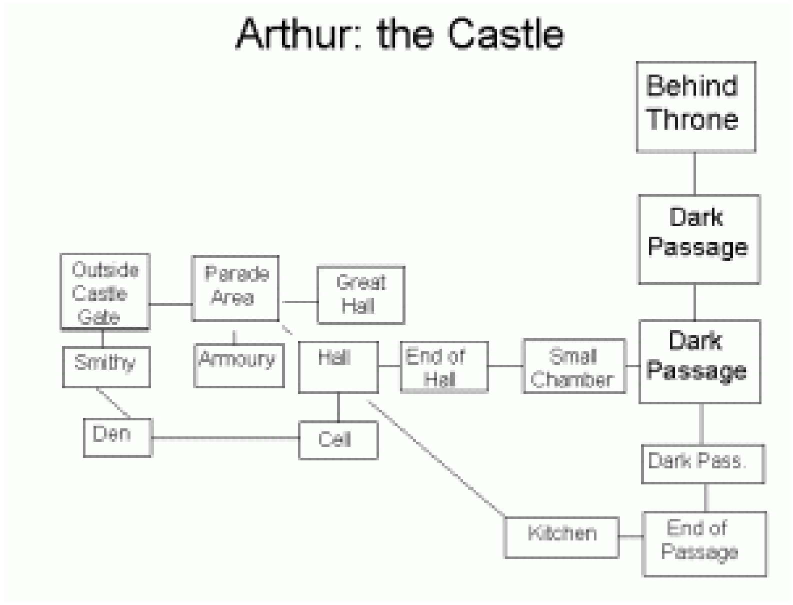
```
Outside town gate                                    St John's Day, Terce

You come to a fork in the road. One route goes northwest, the other
northeast. The road back to the town is to the south.

>go south
OUTSIDE TOWN GATE

The road takes you southward. Soon you are walking alongside the wall of
the town, and you pause outside the town gate, which lies to the east. A
road leads to the north, and to the southwest is a meadow.

>
```

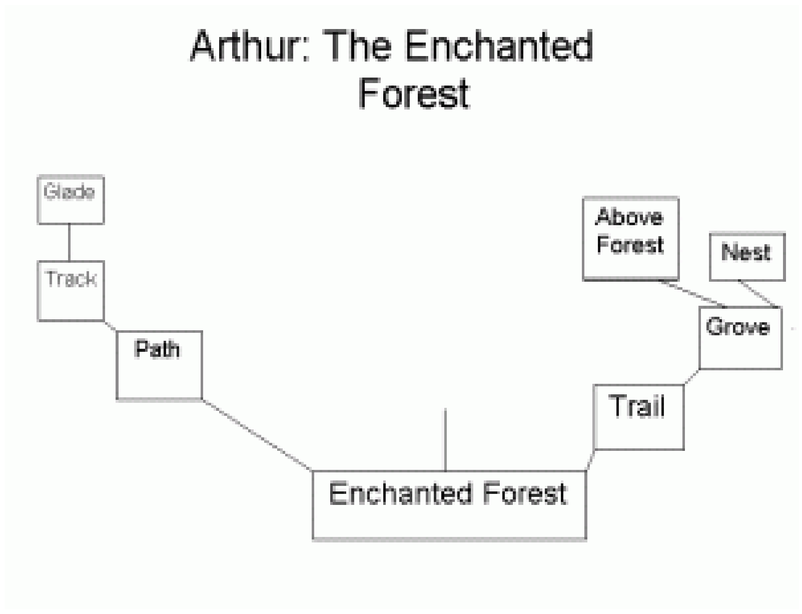
Though *Arthur* does provide on-screen mapping, as shown above, some kids may find more extensive maps of stories' settings fun to use. However, such maps give away the solutions to some problems.

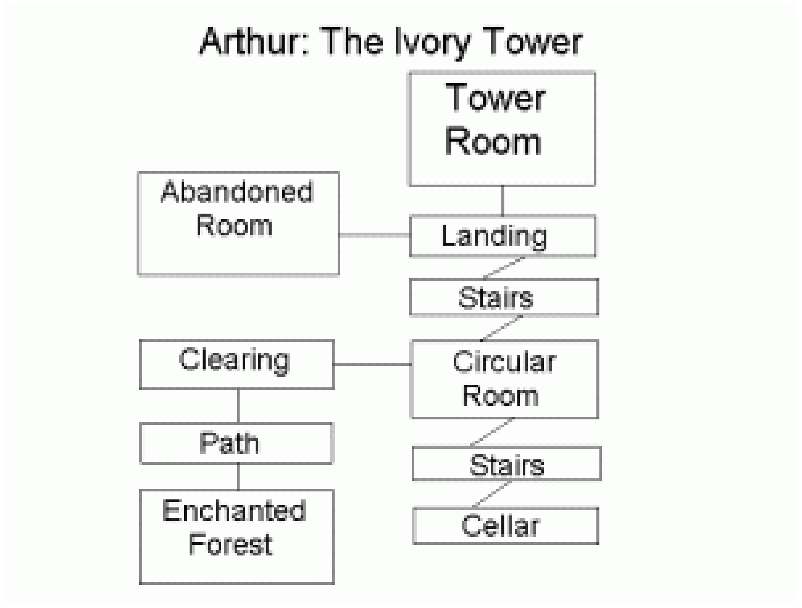
Here are the maps, covering almost all of the locations in the story. These maps contain some "spoilers." In other words, they convey some information that gives away some information that you might not want to have before you read. For this reason, you may not want to look at the maps until you need them.

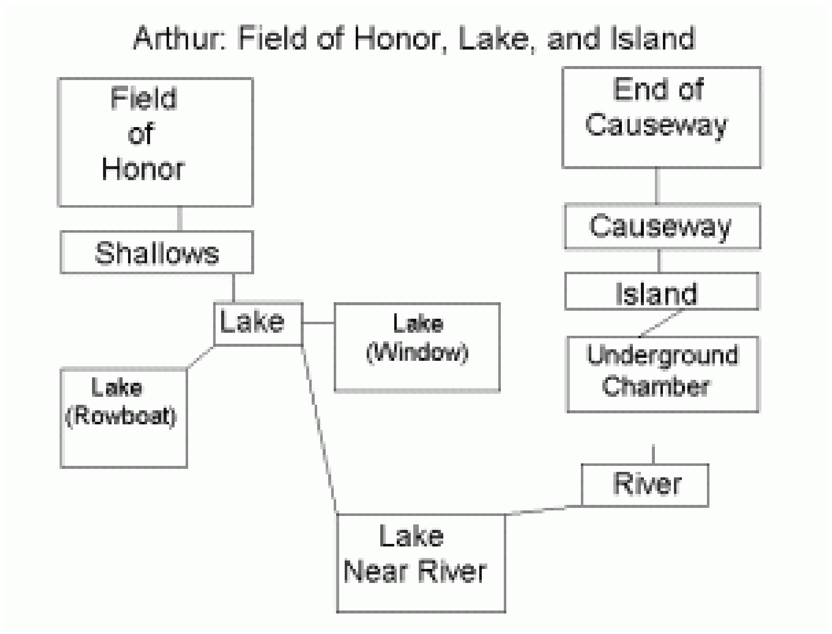

Arthur: The Town
Church-Yard
Church
Village Green
Town Square
Outside Castle
Tavern
Smithy
Kitchen

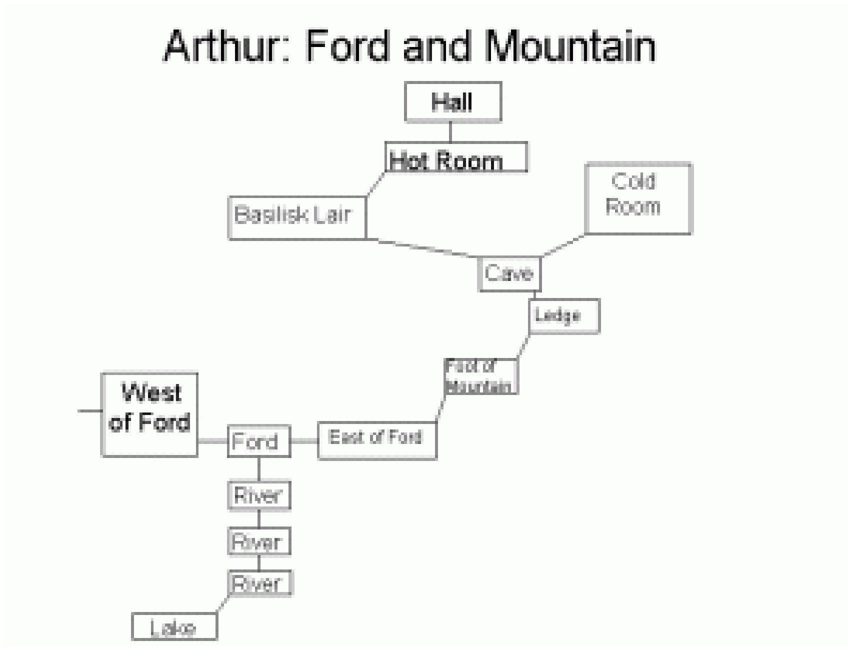

Arthur: West of the Town
Enchanted Forest
Edge of Bog
Edge of Woods
Cottage
Merlin's Cave
Road
Moor
Outside Cave
Fork
Path
Outside Town
Meadow
East Meadow
Field of Honor


Arthur: the Castle
Behind Throne
Dark Passage
Outside Castle Gate
Parade Area
Great Hall
Dark Passage
Smithy
Armoury
Hall
End of Hall
Small Chamber
Den
Cell
Dark Pass.
Kitchen
End of Passage


Arthur: The Enchanted Forest
Glade
Track
Path
Above Forest
Nest
Grove
Trail
Enchanted Forest


Arthur: The Ivory Tower
Tower Room
Abandoned Room
Landing
Stairs
Clearing
Circular Room
Path
Stairs
Enchanted Forest
Cellar





One map that you should not see is in advance is a map of the maze-like badger's den, since the principal problem in this part of the story is to map the maze. Such a map is not too hard to make, once you discover one special secret. The directions "up" and "down" are especially important in the maze.

Here’s the web address of an excellent walkthrough, which includes directions for getting through the maze: http://www.gamefaqs.com/pc/564481-arthur-the-quest-for-excalibur/faqs/1630

Chapter 22 – A Funny Tale of Adventure: *The Firebird*

1998 was a great year for interactive fiction. It produced a variety of good stories by a number of authors, including three extraordinary works for adults, *Spider and Web* by Andrew Plotkin, *Once and Future* by Kevin Wilson, and *Photopia* by Adam Cadre.

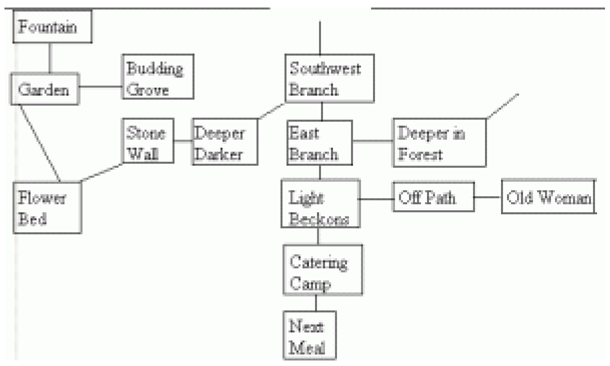
Fortunately for kids, the good news didn't stop there. Bonnie Montgomery's *The Firebird,* a great story for kids and grownups, appeared in the same year.

The Firebird retells a famous Russian folk tale in an adventurous and humorous way. It offers puzzles that are fair, enjoyable, and not too difficult; and it gives the reader several possible ways to a favorable ending. In addition, it develops some memorable characters, especially the Firebird herself; raises interesting questions about the roles of women in fairy tales; and helps us to understand the story on which one of our most famous twentieth-century ballets is based.

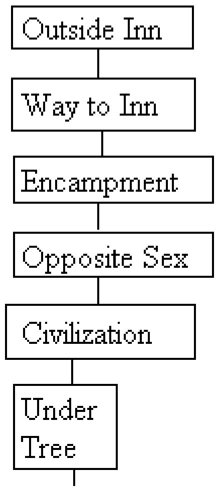
You can download the story and read a walkthrough, using links at the IF Database (http://ifdb.tads.org/viewgame?id=d9h1r3d920ap8ajf)

Below, you'll find some maps for *The Firebird*. These maps give away the solutions to some problems, so use them with caution.

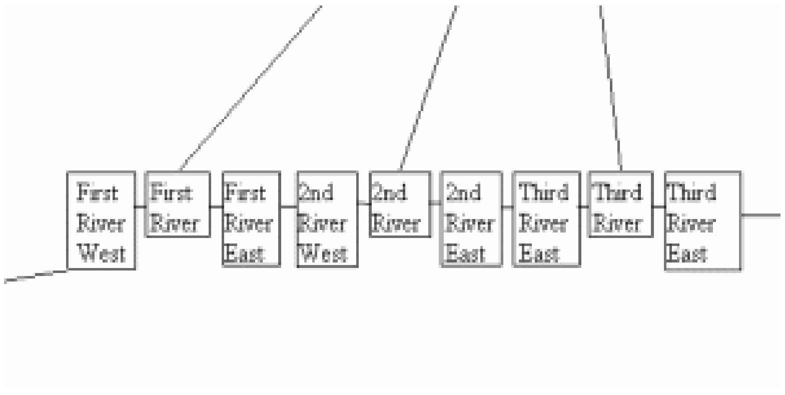
The Forest



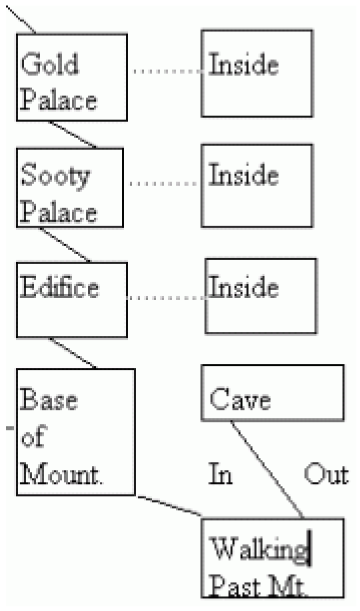
Civilization



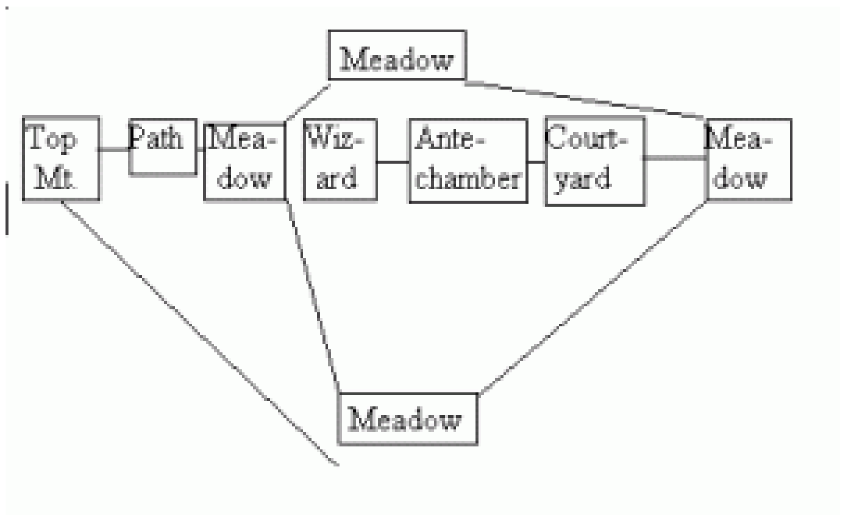
The Rivers



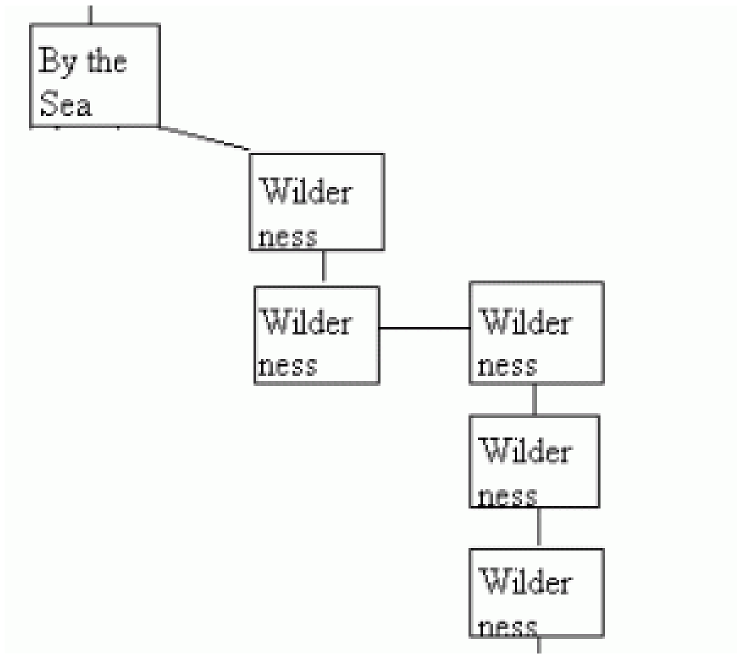
The Mountain



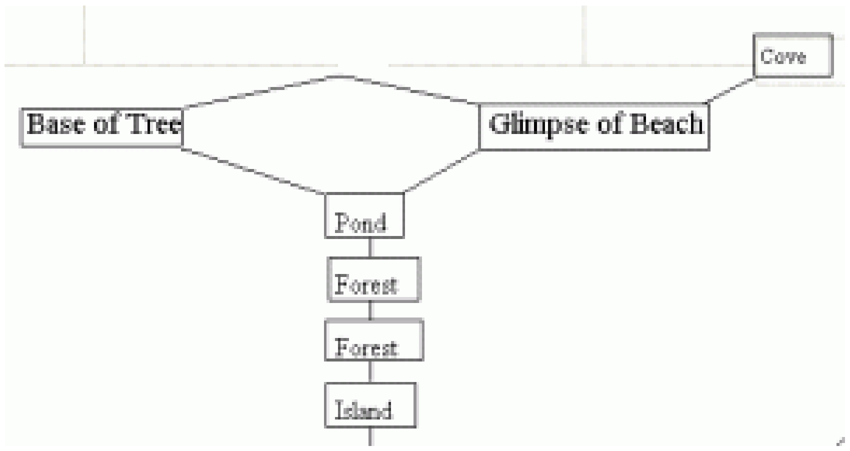
Top of the Mountain



Wilderness Maze



The Island



Chapter 23 – A Middle-School Story: *The Enterprise Incidents*

The Enterprise Incidents: a Middle School Fantasy, is a story about middle school kids and teachers. Its puzzles are fairly easy, though its reading level can be challenging at times. *The Enterprise Incidents* was written by Brendan Desilets, the author of this book, with lots of help from beta testers Sophie Fruehling, Jeff Nassiff, Matt Carey, and Sean Desilets. Hints and a walkthrough are available within the story.

You can download the z-code story file for *The Enterprise Incidents* from the IF Database (http://ifdb.tads.org/viewgame?id=ld1f3t5epeagilfz). The file is called “enter.z5.” For more information on using a story file, read the “Obtaining Interactive Fiction” chapter in this book.

Or, if you prefer, you can probably read *The Enterprise Incidents* on line, using a link that appears on the IF Database. To save your progress, type "save" and then bookmark the resulting Web page.

Another version of "The Enterprise Incidents" includes some on-screen maps and other graphical elements. The filename for this version is “enterprise.blb.” You can download the story file for this version from the IF Database, too. This variation of the story uses an IF-creation tool called Glulx and requires an interpreter that is different from the ones used with other stories featured on this website. I recommend that interpreter called Gargoyle (http://ccxvii.net/gargoyle/). It's free, and it's available for Windows, Mac, and Linux.

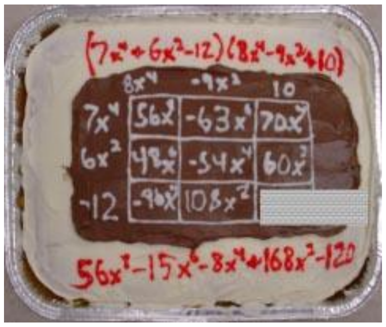
The pictures and maps that appear below may add to your enjoyment of the story, if you are reading the z-code version. The pictures appear in the Glulx version.

The mural in the cafeteria, where the story begins

Be My
Valentine!
To: Judy
From: Biff

A candy gram card

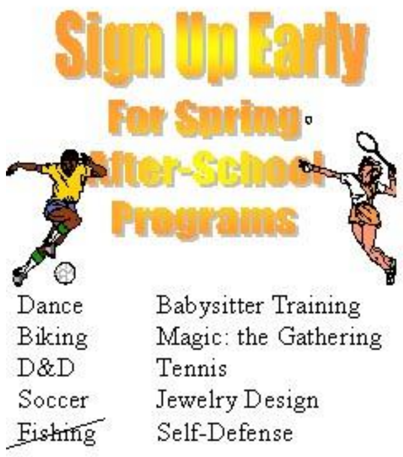
A decorated locker



A math cake from Ms. Garrulous' class

The cover of Ed Dibbles' science folder

Meghan Mascaras' science folder


Sign Up Early
For Spring
After-School
Programs

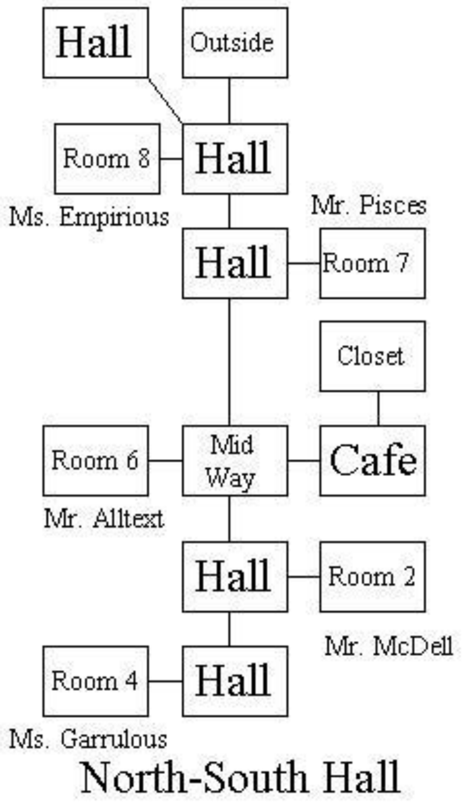
Dance	Babysitter Training
Biking	Magic: the Gathering
D&D	Tennis
Soccer	Jewelry Design
~~Fishing~~	Self-Defense

A poster from outside the office. Fishing is cancelled because of Mr. Pisces' need for a bone-marrow transplant.

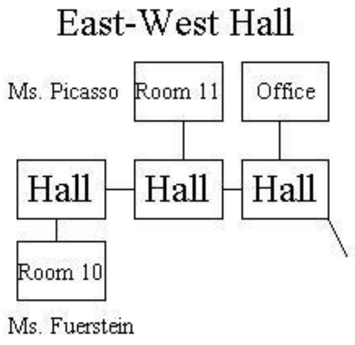
An art project by Silas Gibber

Lightning above a thistle
Sprung from the guts
of an old, grey stump

Jim Hastely's riddle poem



A Map of the North-South Hall



A Map of the East-West Hall
To reach the North-South Hall,
go southeast from the hall
near the Office.

Chapter 24 – IF at Its Comic Best: *Lost Pig, and Place Under Ground*

When you look at lists of the best IF stories ever, you often find, near the top, some long and serious works, like Andrew Plotkin's *Spider and Web*, Brian Moriarty's *Trinity*, *Blue Lacuna* by Aaron Reed, and *Savoir Faire* by Emily Short. But you also find one brief and funny story, in which you play the part of an orc who speaks broken English. This story is called *Lost Pig,* and it was written by a person whose pen-name is Admiral Jota. In 2007, when it first appeared, *Lost Pig* won the fall IF Competition, and it later took prestigious XYZZY Awards for Best Game, Best Writing, Best Individual Non-Player Character, and Best Player/Character.

Critics of all ages have praised the story in nearly every possible way. Some love the way it accommodates just about any response a player tries, even some very unlikely ones. (Try “Burn pants” or “Burp.”) Others can't get enough of Grunk himself, a person of endless good nature and bad grammar, who always makes himself perfectly clear. Many laugh out loud at Grunk's “opposite number,” a highly-cultured gnome who learns, it seems, a renewed sense of purpose from his new orc acquaintance. And even the pig has its own funny, “trickster” personality.

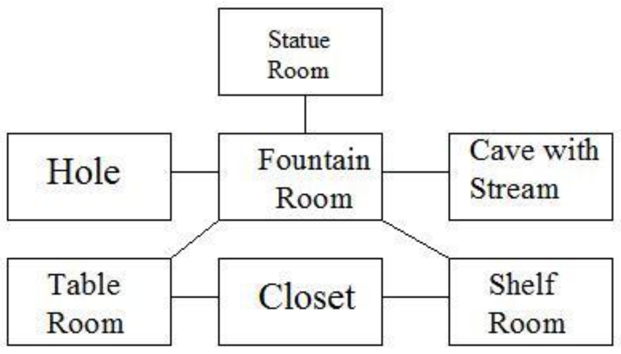
Kids a appreciate *Lost Pig* as a tale for all ages. The IF Database quotes a review by Racquel and Liza, aged ten, from Massachusetts in the USA.

“It was very fun and exciting and I liked the characters, especially Grunk. I liked the part with the bread machine. I also liked that whatever you ordered Grunk to do he did, including burping. I also liked that one thing led to another and you had to do things in order to solve the game."

"I enjoyed how the game felt realistic, like it really was happening. The gnome dude was really cool and nice. It's not like you can talk to a gnome every day! I also enjoyed the fact that you had to be precise on your commands. I thought it was cool that you had to say exact directions... not like right,and left. More of N,NW,NS,E,SW,SE,S,AND WEST. It was an awesome game. I hope there are other games with Grunk included!!!!!!”

Lost Pig takes place within a fairly small area, but the map that appears below may be helpful, anyway. Some very important clues appear in drawings that appear on the walls in the “Place Under Ground.” Grunk's descriptions of these drawings appear below, too.

Map for *Lost Pig*



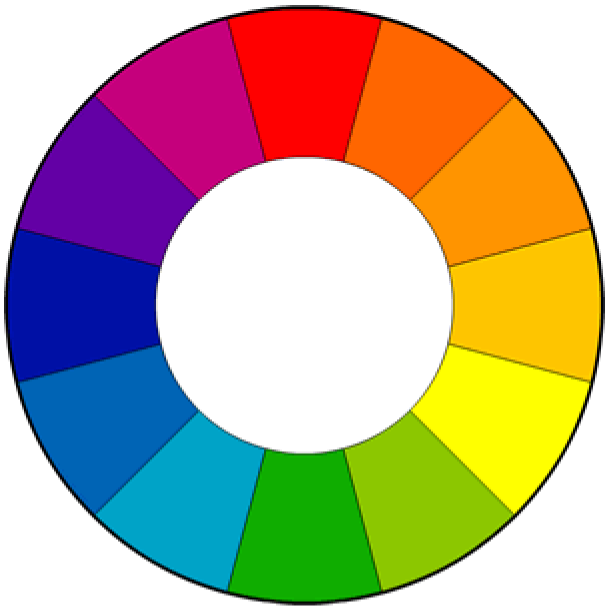
The Pictures on the Walls

Curtain in Fountain Room: Look like big rug, but it on wall instead of floor, so that make it curtain instead. Curtain have big picture of little man on it. Grunk think little man maybe called "gnome." Him holding burning torch in dark cave. Point at way out of cave.

West Wall in Statue Room: Picture have big pile of black powder. Powder all on fire. Picture have bucket of water, too. Bucket pouring water onto fire. Grunk guess that water for making fire go out.

East Wall in Statue Room: Picture have long purple pole that go from side to side. All around pole, there different yellow thing that float in air, like honey and bottle of beer and pretty flower and lots of other thing too. Under that, there picture of pie. Mmm, pie.

Pie cut up into more than seven piece, and every piece have different color. Purple piece at top. To right of purple piece there orange piece, then there white piece at right side, then there red piece, and then yellow piece at bottom. Next piece blue, then there black at left side, and then green piece, and then it back to purple piece again at top. Grunk think red piece look most tasty.



A Color Wheel, Similar to the One Described in *Lost Pig*

Lost Pig: a Walkthrough for Beginners

This walkthrough may help beginners get started with interactive fiction, using the story *Lost Pig and Place Under Ground*, by Admiral Jota. In this story the reader plays the part of an ogre-like creature called an orc. The orc's name is Grunk, and he speaks broken English. He communicates very well, though.

When the story begins, the reader sees the following text:

```
Pig lost! Boss say that it Grunk fault.
Say Grunk forget about closing gate.
Maybe boss right. Grunk not remember
forgetting, but maybe Grunk just forget.
Boss say Grunk go find pig, bring it
back. Him say, if Grunk not bring back
pig, not bring back Grunk either. Grunk
like working at pig farm, so now Grunk
need find pig.

Lost Pig
And Place Under Ground
Release 1 / Serial number 070917 / Inform
v6.30 Library 6/11 S(For help, use
"HELP".)

Outside
```

```
Grunk think that pig probably go this
way. It hard to tell at night time,
because moon not bright as sun. There
forest to east and north. It even darker
there, and Grunk hear lots of strange
animal. West of Grunk, there big field
with little stone wall. Farm back to
south.

>
```

That “greater than” sign, which looks like “>,” is very important in interactive fiction. It signals the reader that it time for him or her to tell the main character what he or she should try to do. You could think of the “>” symbol as meaning “I would like the main character in the story to try to...”

Mostly, an interactive fiction story understands only a few kinds of sentences. Some of the sentences that would usually work are:
LOOK AT ME
LOOK AT THE FOREST
GO NORTH
GO SOUTH
INVENTORY (that is, list what the character is carrying)
WAIT

Let's try some of these sentences in *Lost Pig*.

```
>GO SOUTH
>GO EAST
>GO WEST
>GO NORTH
```

It seems that Grunk is not going to go very far into the darkness, even with his torch, unless he can get a better idea of where the pig might be.

Let's try some of the other ideas we've listed above.

```
>LOOK AT FOREST
>LOOK AT FARM
>LOOK AT ME
>INVENTORY
```

“Inventory” (abbreviated “i”) gives us a list of what Grunk is carrying. By chance, in this case, it gives us an indication of how we might get a better idea of where the pig is. Grunk can't see the pig, but he can hear a sound that the pig may have made.

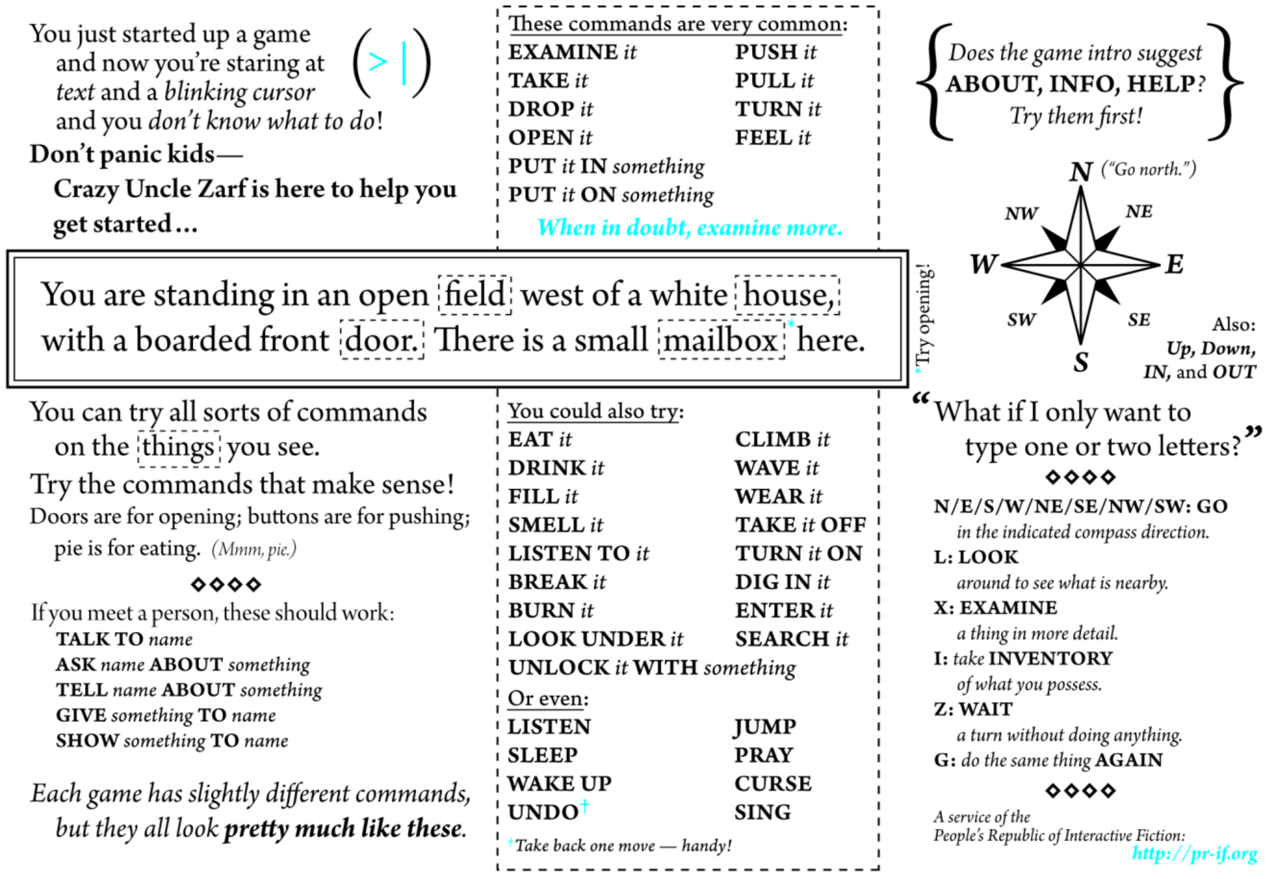
```
>LISTEN
```

It seems that the pig may be off to the northeast.
>NE (An abbreviation for “GO NORTHEAST”)

Well, it seems that Grunk's unwillingness to wander around in the dark made some sense. Even so, Grunk seems uninjured after his fall into the hole, so let's grab our torch and look around. We'll use the abbreviation "X" for "EXAMINE."

```
>TAKE TORCH
>X STAIRS
>TAKE THING
>X THING
```

Experiment a bit to see if you can find out what the "thing" might be used for. Then we'll look explore some more. But, first, courtesy of the People's Republic of Interactive Fiction and noted author Andrew Plotkin, here's a postcard that you may find helpful.



You just started up a game and now you're staring at *text* and a *blinking cursor* and you *don't know what to do*! (>|)

Don't panic kids—

Crazy Uncle Zarf is here to help you get started…

These commands are very common:

EXAMINE *it*
TAKE *it*
DROP *it*
OPEN *it*
PUT *it* **IN** *something*
PUT *it* **ON** *something*
PUSH *it*
PULL *it*
TURN *it*
FEEL *it*

When in doubt, examine more.

Does the game intro suggest **ABOUT, INFO, HELP**? *Try them first!*

N ("Go north.")
NW NE
W E
SW SE
S

Also: ***Up, Down, IN,*** and ***OUT***

You are standing in an open field west of a white house, with a boarded front door. There is a small mailbox* here.

*Try opening!

You can try all sorts of commands on the things you see.

Try the commands that make sense!

Doors are for opening; buttons are for pushing; pie is for eating. *(Mmm, pie.)*

◊◊◊◊

If you meet a person, these should work:

TALK TO *name*
ASK *name* **ABOUT** *something*
TELL *name* **ABOUT** *something*
GIVE *something* **TO** *name*
SHOW *something* **TO** *name*

Each game has slightly different commands, but they all look ***pretty much like these.***

You could also try:

EAT *it*
DRINK *it*
FILL *it*
SMELL *it*
LISTEN TO *it*
BREAK *it*
BURN *it*
LOOK UNDER *it*
UNLOCK *it* **WITH** *something*
CLIMB *it*
WAVE *it*
WEAR *it*
TAKE *it* **OFF**
TURN *it* **ON**
DIG IN *it*
ENTER *it*
SEARCH *it*

Or even:

LISTEN
SLEEP
WAKE UP
UNDO†
JUMP
PRAY
CURSE
SING

†*Take back one move — handy!*

"What if I only want to type one or two letters?"

◊◊◊◊

N/E/S/W/NE/SE/NW/SW: GO
in the indicated compass direction.
L: LOOK
around to see what is nearby.
X: EXAMINE
a thing in more detail.
I: *take* **INVENTORY**
of what you possess.
Z: WAIT
a turn without doing anything.
G: *do the same thing* **AGAIN**

◊◊◊◊

A service of the People's Republic of Interactive Fiction: ***http://pr-if.org***

And now, back to our walkthrough.

```
>E (Abbreviation for "GO EAST")
```

You may find the lost pig here. And, if you don't, you'll find it soon. When you find the pig, try to catch it.

```
>TAKE PIG
>CHASE PIG
```

It looks as though we can't catch the pig just yet, but this might be a good time to think of a strategy for grabbing it. Perhaps some sort of distraction...

```
>X FOUNTAIN
```

By now, you may have noticed that, in interactive fiction, it's a good idea to examine practically everything. But sometimes mere examining isn't enough. Sometimes, we'll need a more exact kind of looking.

```
>LOOK IN FOUNTAIN
```

We now have a coin, so we should be looking for opportunities to use it. Also, the story is apparently continuing to keep a score, as we've just earned our second point. Not all IF stories keep a score, though.

It's time for a bit more exploring.

```
>SW
>LOOK AT BOX
```

What could this box be? Try to figure it out. It's important.

```
>TAKE CHAIR
```

In interactive fiction, it's often good to take anything that's not nailed down, unless there's a clear reason not to.

But what is that box?

```
>PUT COIN IN SLOT
>PULL LEVER
>TAKE BRICK
>X BRICK
>EAT BRICK
```

Maybe this edible “brick” is a way to distract the pig. Let's try it.

```
>NE
>DROP BRICK
>WAIT
>Z (An abbreviation for “WAIT”)
```

Well, the brick did seem to distract the pig a bit, but it seems that one brick is not enough. Can we get more? Remember the dents on the vending machine? How could those have happened?

```
>SW
>HIT BOX
>TAKE COIN
>PUT COIN IN SLOT
>PULL LEVER
>TAKE BRICK
>HIT MACHINE
>TAKE COIN
```

```
>PUT COIN IN SLOT
>TAKE BRICK AND COIN
>PUT COIN IN SLOT
>TAKE BRICK AND COIN
```

Let's see if we can use our supply of bricks to catch the pig.

```
>NE (You may have try some of the other
locations to find the pig.)
>DROP BRICK
>WAIT
>Z
>Z
>DROP BRICK
>Z
>TAKE PIG
```

With enough bricks, Grunk can feed the pig until it becomes distracted enough to catch, and so, in our walkthrough, we have solved one of the main puzzles in the story. Other readers might not solve this problem until much later.

Often, in interactive fiction, we find ourselves working on more than one puzzle at a time. In that way, IF is a little like real life. As a result, it's important to keep track of the problems we're working on at any one time. Some readers even write down the puzzles, so as not to forget any of them.

In our walkthrough, Grunk still has to get out of the hole, and solving this puzzle requires solving several others, first.. The hole itself, which is really more of an underground shrine, has some hints to offer, and one of them is in this room.

```
>LOOK AT CURTAIN
```

Let's look around some more. The order in which we check out the remaining rooms doesn't really matter much, but we'll have to explore them thoroughly.

```
>SE
>OPEN CHEST
>TAKE POLE
```

The pole tries to push Grunk away when he tries to pick it up. He succeeds anyway, but this sort of odd detail is often important in interactive fiction, so we should keep it in mind.

```
>NW
>N
>X STATUE
```

This statue has some details that might be worth a closer look.

```
>X SHOES
>TAKE SHOES
>X HAT
>TAKE HAT
```

We should also have a look at the pictures on the wall in this room.

```
>X EAST WALL
>X PIE
>X WEST WALL
```

Now, we've seen three wall pictures, all of which look important. Reviewing quickly, we find these descriptions of the pictures:

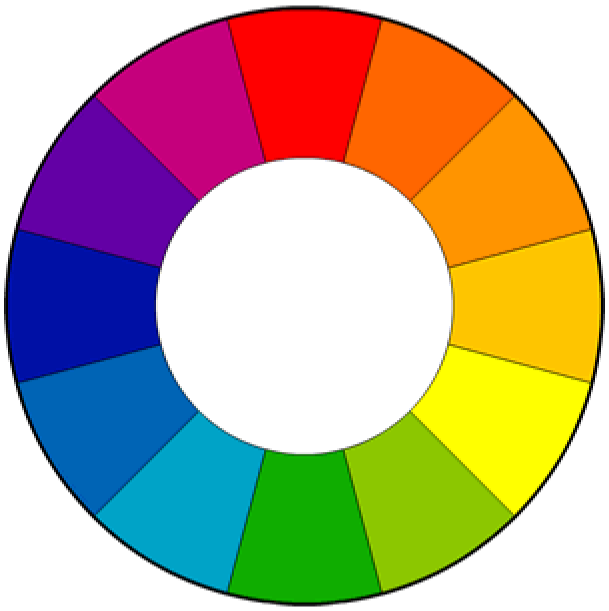
Curtain in Fountain Room: Look like big rug, but it on wall instead of floor, so that make it curtain instead. Curtain have big picture of little man on it. Grunk think little man maybe called "gnome." Him holding burning torch in dark cave. Point at way out of cave.

West Wall in Statue Room: Picture have big pile of black powder. Powder all on fire. Picture have bucket of water, too. Bucket pouring water onto fire. Grunk guess that water for making fire go out.

East Wall in Statue Room: Picture have long purple pole that go from side to side. All around pole, there different yellow thing that float in air, like honey and bottle of beer and pretty flower and lots of other thing too. Under that, there picture of pie. Mmm, pie.

Pie cut up into more than seven piece, and every piece have different color. Purple piece at top. To right of purple piece there orange piece, then there white piece at right side, then there red piece, and then yellow piece at bottom. Next piece blue, then there black at left side, and then green piece, and then it back to purple piece again at top. Grunk think red piece look most tasty.

The “pie” sounds like a standard color wheel, similar to the one below, except that Grunk's has the purple piece at the top.



```
>S
>E
>LOOK AT THING
```

This might be a chance to use one of the items we're carrying to solve a puzzle.

```
>TAKE THING WITH POLE
```

Now, Grunk has solved the puzzle of getting the red thing, a key, which might be the solution to another problem, the locked chest in the shelf room. We also have another example of the odd behavior of the green pole, which repels Grunk but attracts the red key.

```
>W
>SE
>OPEN CHEST
```

Now we have a chest full of powder, which looks a lot like what we saw on the west wall in the statue room. Could it be that the water in the picture was not really putting out the fire?

This would be a fine time to put aside this walkthrough and try to solve some problems on your own for a while. You might start with re-lighting the torch.

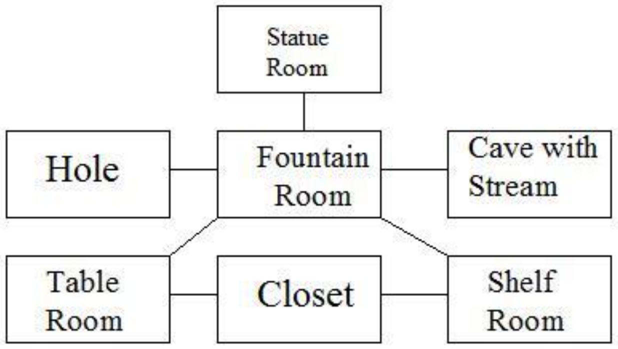
```
>NW
>E
>FILL HAT WITH WATER
>W
>SE
>POUR WATER IN CHEST
>LIGHT TORCH
```

Grunk has now explored almost all of the important locations in the story. With a lighted torch, he might be able to get a better look at one or two of them.

```
>W
>TALK TO GNOME
>TELL GNOME ABOUT GRUNK
>ASK GNOME ABOUT THIS PLACE
>ASK GNOME ABOUT ALCHEMIST
>ASK GNOME ABOUT INVENTION
>ASK GNOME ABOUT GNOME
>ASK GNOME HOW LONG GNOME HERE
```

We've now visited all the major locations in the

Map for *Lost Pig*



story. Even in a story like this one, which has only a few rooms, some readers like to make a map.

We've also encountered a gnome has quite a lot to say, in perfect, standard English. In fact, the depth of his ability to converse is one of the many strong points of *Lost Pig*.

Notice that we started the conversation off with “TALK TO GNOME.” “TALK TO” works in many, though not all, interactive fiction stories. Once we get the conversation started, the story gives us hints about what we might have Grunk say next. Even in stories that don't use “TALK TO,” a command like “ASK GNOME ABOUT ALCHEMIST” will often work.

It looks as though the gnome is looking for a book. Maybe we can help him out.

```
>E
>X SHELF
>DROP CHAIR
>STAND ON CHAIR
>LOOK AT TOP SHELF
>TAKE BOOK
>GO DOWN
>W
>GIVE BOOK TO GNOME
>TALK TO GNOME
```

Apparently, Grunk has returned the right book but there's a page missing. By poking around a bit with a lighted torch, we can find that page.

```
>W
>NE
>W
>X CRACK
>TAKE PAPER WITH POLE
Grunk stick pole into crack and poke
paper with end of it. But nothing special
happen.
```

Grunk’s pole, in its present state, won't help to retrieve the missing page. How might we change the pole to make it attract the white paper? Think about colors and magnetism. Put away this walkthrough for a while and try to figure it out.

SPOILER SPACE INTENTIONALLY LEFT BLANK HERE

```
>BURN POLE
>TAKE PAPER WITH POLE
>E
>SW
>E
>GIVE PAPER TO GNOME
>TALK TO GNOME
```

The gnome seems appropriately grateful, but there's no obvious way he can help us right now. Let's work on another problem, the problem of finding a way out of the hole. Remember the picture on the curtain in the Fountain Room?

```
>W
>NE
>N
>PUT TORCH IN HAND
```

This certainly looks promising. Let's try to get out through the formerly-secret door.

```
>TAKE TORCH
>N
>N
>N
```

Grunk has found a cave that may be the way out of the hole/shrine, but the tunnel is so windy that he can't use his torch there. Where can he find a source of light that won't blow out? Maybe he should re-light the torch first.

```
>S
>S
>E
>FILL HAT WITH WATER
>W
>SE
```

```
>POUR WATER ON POWDER
>LIGHT TORCH
>W
>ASK GNOME FOR BALL
>GIVE TORCH TO GNOME
```

With his new light source, maybe Grunk can escape from the underground shrine.

```
>W
>NE
>N
>N
>N
```

It seems that Grunk is hopelessly lost in a maze of tunnels! However, Admiral Jota is not going to make the reader map a huge web of rooms here. Actually, almost anything we try to call for help will work.

```
>BLOW IN TUBE
>SE
>D
>S

Forest
Hey, Grunk outside again! Yay! It still
night time, but Grunk have light, so that
OK. This look like it near place Grunk at
```

before. Farm not far at all from here. Just little way to southwest.

Gnome here, waiting for Grunk.

Gnome take trunk back from Grunk. "I think this is where we part ways," him say. "You can probably find your way back to the farm from here." Gnome shake Grunk hand. Then gnome say, "Remember the name Zugilbor Galrogginpots sem Endali dec Frebensalbibit. This won't be the last time you hear it."

Then gnome go one way and Grunk go other way. It not long before Grunk back at farm. Boss already sleeping, so Grunk put pig back quiet and go to bed. Tomorrow, maybe boss proud of Grunk.

    *** Grunk bring pig back to farm
***

Grunk have 6 out of 7 that time.

Time for Grunk to RESTART or RESTORE a saved story or UNDO what Grunk just do or

```
tell FULL score or look at MENU (with
different silly thing Grunk can try
doing) or just QUIT?
```

Now, see if you can figure out how to get that “last lousy point”!

Chapter 25 – Writing Interactive Fiction With Adrift

Writing your own interactive fiction can be an enjoyable challenge. Three of the best authoring systems for IF are called Inform 7, Quest, and Adrift. All are free of charge.

Below, you’ll find an introduction to creating interactive fiction with Adrift. You can find introductions to Inform 7 and Quest later in this book.

Would you like to try writing some interactive fiction without leaning to use a programming language? A Windows-based software tool called Adrift Developer makes it possible for you to do so with very little fuss. Adrift Developer costs nothing and is available on the Web at:
http://www.adrift.org.uk/

Adrift works best for stories that are fairly simple and straightforward. For such stories, Adrift is quite easy to use and requires no computer-programming skills. To make more complicated stories, a writer may need to learn some programming.

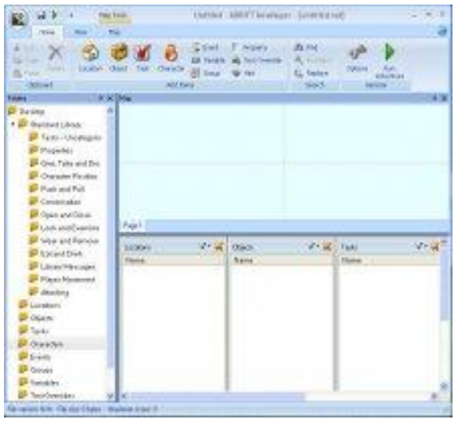
For the purposes of this tutorial, we'll write an extremely simple story. Here's a transcript of one user's experience with this story.

```
“Lost Chicken” Script

You've made it to your home, as usual,
but it seems that you've forgotten your
key.

Lost Chicken
An Interactive Fiction by Brendan
Desilets

Outside Front Door
You're outside your front door.  The door
is to the west, and your front yard is to
the south.

You can see a doormat and an oak door
here.

>take doormat
You find a note under the doormat and
pick it up. Then you drop the doormat.
```

```
>read note
The note reads, "The chicken hides the
spare."

>s

South Yard
This is the lovely south yard of your
home.

You can see a plaster chicken (closed)
here.

>open chicken
You open the plaster chicken, revealing a
key.

>take key
Taken.

>n

Outside Front Door
You're outside your front door.  The door
is to the west, and your front yard is to
the south.

You can see a doormat and an oak door
```

```
here.

>w
You unlock the door and enter your house.

Front Hall
You have made it inside. Congratulations!


    *** The End ***
```

An Adrift Tutorial



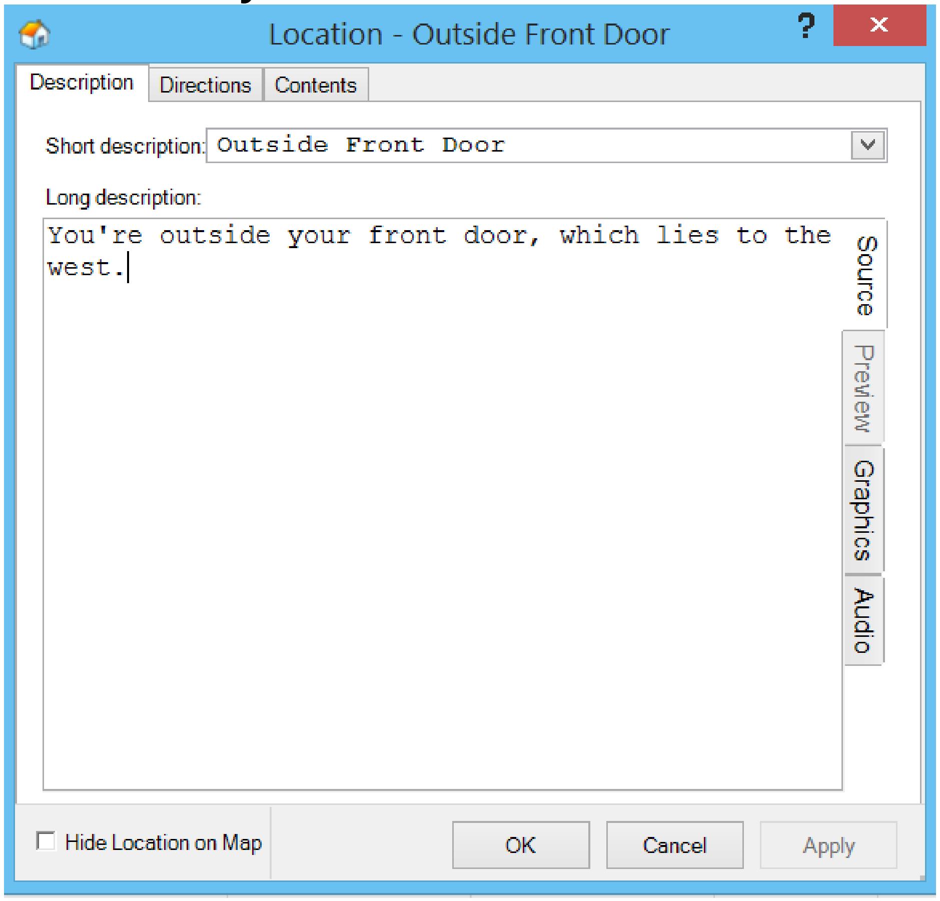
When Adrift Developer starts, it usually displays five windows. The windows are labeled “Rooms,” “Objects,” “Tasks,” “Events,” and “Characters.”

Use these steps to get started with Adrift Developer:

1. Start the computer program called Adrift Developer. Click on the “Home” tab at the top of the screen, if that tab is not already selected.
2. Click on “Options” and then on “Bibliography.” Type in the title of your piece of writing and your name, in the right spaces. Then, click on OK.
3. In the “Characters” section of the Adrift Developer screen, you'll notice that one character, the player/character, has already been created. Double-click on the word “Player,” and type in the name of your player/character.

Then type a brief description of your player/character in the appropriately-labeled space. Click on the right gender for your player/character, male or female. Then click on OK.

Create your first location



Short description: Outside Front Door

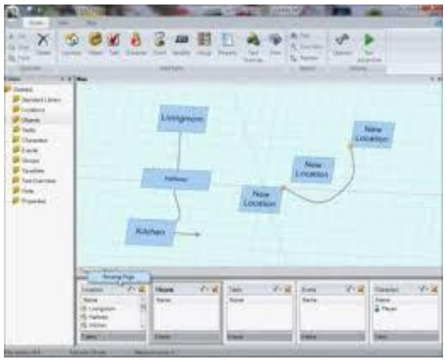
Long description: You're outside your front door, which lies to the west.

1. On the Adrift Developer screen, find the section labeled “Add Items.” In this section, click on “Location.”
2. In the form that opens, fill in the “short name” of the room. This name should be three words or fewer in length. For our “Lost Chicken” story, we'll use “Outside Front Door” as the short name of our

first location.

3. Then, in the appropriate space, fill in the “long room description.” You can copy and paste the long description from your word processor, if you like. For our long description of “Outside Front Door,” we'll use “You're outside your front door. The door is to the west, and your front yard is to the south.”
4. Click on “OK” when you have finished with this form.
5. You can use the same procedure to make more rooms. For “Lost Chicken,” we'll create two more locations, “Front Hall” and “South Yard.” We won't need a long description for “Front Hall.” Our long description of “South Yard” will be “This is the lovely south yard of your home.”
6. Double-click on “Outside Front Door,” from your list of locations. In the form that opens, click on the “Directions” tab. Using dropdown menus that appear, indicate that going west from “Outside Front Door” will take the player/character to “Front Hall” and that going south from “Outside Front Door” will take the player/character to “South Yard.” When you've indicated these directions, Adrift may ask if you want these directions to work both ways. In other words, Adrift may ask whether going east from Front Hall should lead to “Outside Front Door” and whether going north from “South Yard” should bring the player/character to “Outside

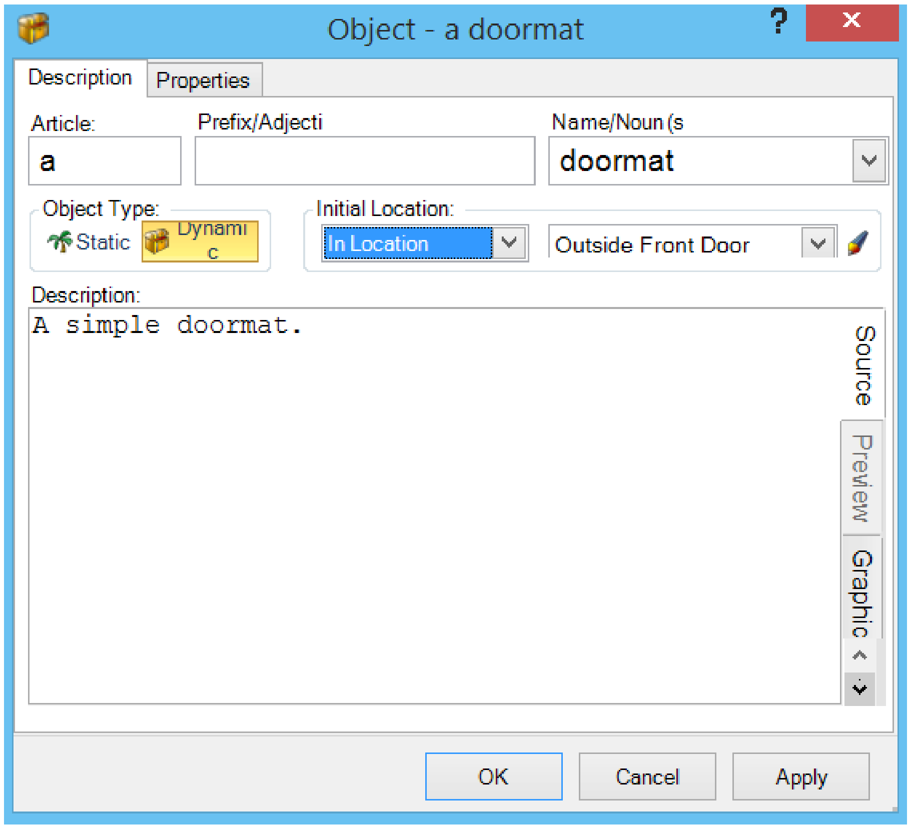
Front Door." For our story, we can allow both of these two-way pairs.
7. Adrift will show a map of the rooms you've created.



Save and Test Your Story, So Far

1. Though Adrift is a solid and mature program, it is important so save your work frequently. It's a good practice to save several versions of your work, in case something goes wrong with your latest saved version.
2. To save your story, click on the circular icon in the top left of the screen, and choose "Save As."
3. To test your story, click on the green triangle in the top center of the screen. This triangle is labeled, "Run Adventure." Adrift Runner will open and display your story. Your player/character should be able to move between the rooms you've created.

Create your first “object.”


Object - a doormat

Description | Properties

Article: a

Prefix/Adjecti:

Name/Noun (s: doormat

Object Type: Static [✓ Dynamic]

Initial Location: In Location — Outside Front Door

Description:

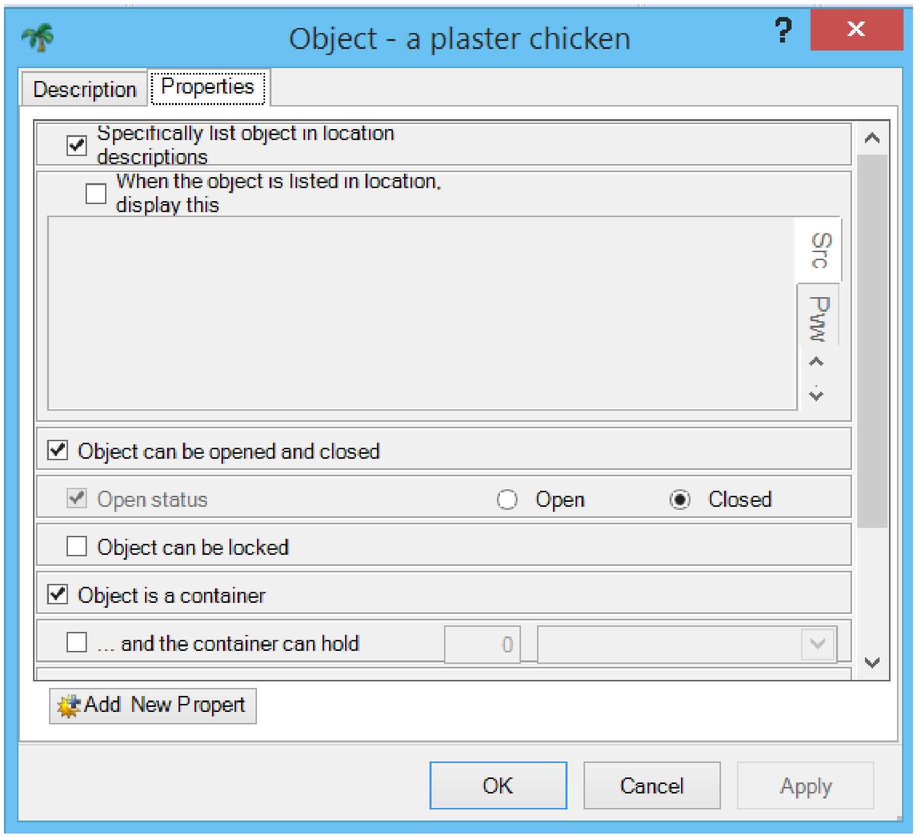
A simple doormat.

1. From the “Add Items” list, choose Object.
2. In the form that opens, type in the name of the object. In our example, we'll create the doormat as our first item. Using the buttons and dropdown menus, indicate that this object is “dynamic” and that its initial location is “Outside Front Door.” A dynamic object is an item that the player/character can pick up.
3. Then, in the “name” box, type “doormat” and then press the Enter key. Next, type “mat.” Now, we've signaled that we want to create an object named “doormat,” which the user can also refer to as “mat.”
4. Next, type in the description of the object.

5. Click on the Properties tab, and indicate that the object should be mentioned in room descriptions.
6. Click on “OK.
7. You can use the same procedure to make more objects.

Create more objects.

1. Next, let's create the object called the “note.” This object will be similar to the doormat, except that its location will be “hidden.”
2. Now, let's create, or “implement,” the plaster chicken, which will contain the
key. This object will be similar to the others, but with several differences.
    1. It will be static, rather than dynamic.
    2. It will be in the South Yard.
    3. Its properties, accessed through the “properties” tab, will specify that
        1. it is a container,
        2. it can be opened and closed,
        3. it is closed, and
        4. it should be included in room descriptions.
3. Finally, let's implement the key. It will be a dynamic object, and its initial location will be inside the plaster chicken.


Object - a plaster chicken

Description | Properties

- [✓] Specifically list object in location descriptions
- [ ] When the object is listed in location, display this
- [✓] Object can be opened and closed
- [✓] Open status ○ Open [● Closed]
- [ ] Object can be locked
- [✓] Object is a container
- [ ] ... and the container can hold 0

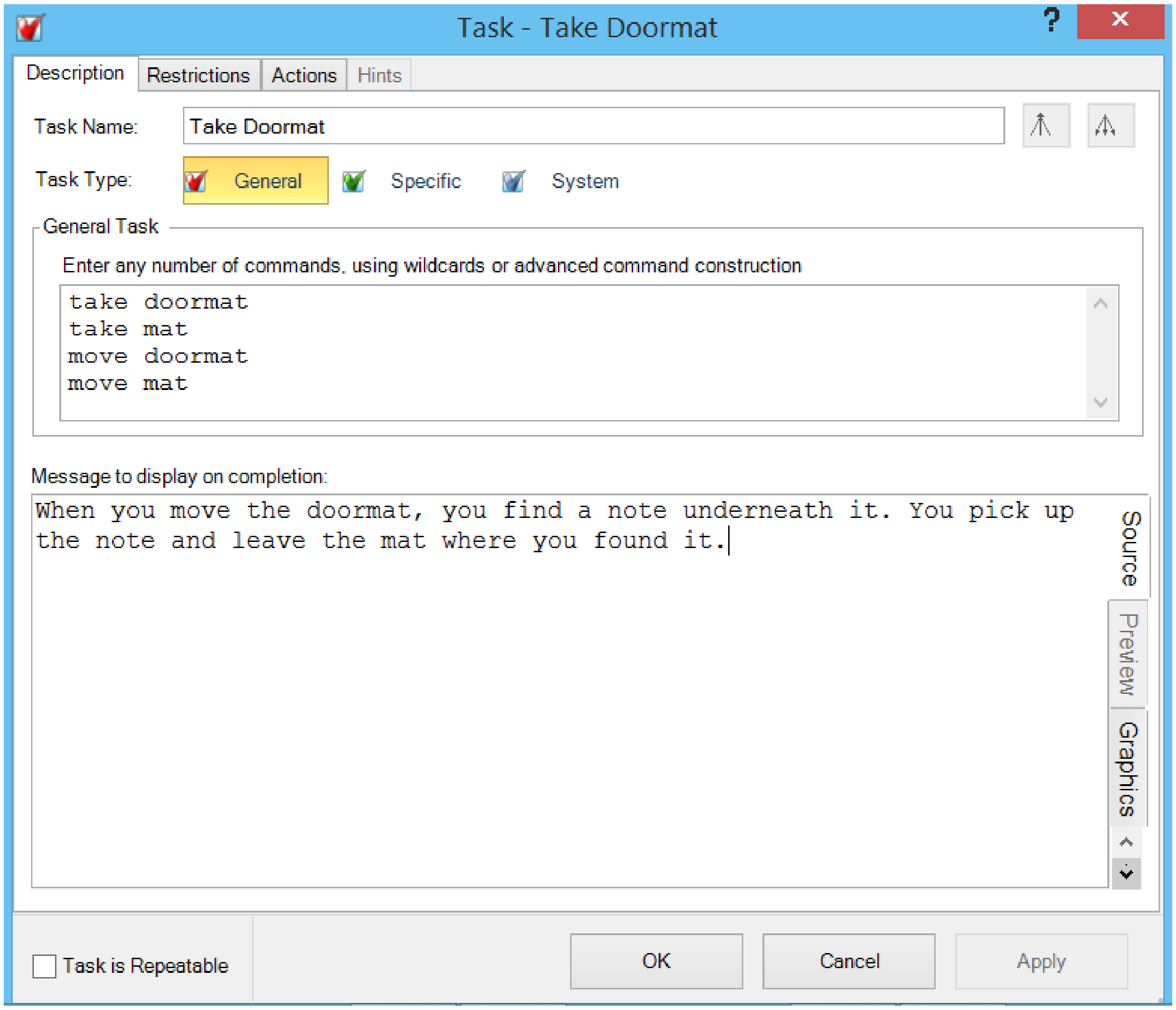
Set the Opening and Closing Text

1. Click on the circular icon that appears in the top left corner of the Adrift Developer screen. From the menu that opens, choose “Introduction & End of Game.”
2. The page that opens will have two tabs, one for the opening of the game and one for its ending.
3. The “Introduction” tab allows for the creation of opening text, for specifying the story's first location, and for displaying, or not displaying, the description of the opening location.
4. Fill in the opening text that you'd like to show at the start of your story, something like, "You've arrived at your home as usual, but you've forgotten

your key."

5. This would be a good time to save and test your story, so far.

Create your first "task."



Task Name: Take Doormat

Task Type: General [✓] Specific System

General Task

Enter any number of commands, using wildcards or advanced command construction

```
take doormat
take mat
move doormat
move mat
```

Message to display on completion:

```
When you move the doormat, you find a note underneath it. You pick up
the note and leave the mat where you found it.
```

Task is Repeatable

1. A task is an action that the player/character must, or may, take to produce a certain result. Adrift has lots of built-in tasks, such as picking up an object, but you'll want to create your own, too. Tasks that you create generally override Adrift's built-in tasks.

Tasks can be a bit complicated, but they're necessary.

2. Our first example of a task will implement what happens when the player/character first takes the doormat, thus revealing the note. Recall that, when we created the note, we placed it in the "location" called "hidden." In other words, it's not anywhere in the story's world until we bring it in.

3. In the "Add Items" section at the top of the screen, click on "Task." A fairly complex window will open.

4. With the Description tab selected, type in a name for the task. We'll use "Take Doormat."

Our "Task Type" will be "General." (We could also implement this task as "Specific," but, if we did, we would not be able to introduce the action called "move," which Adrift does not normally understand.)

5. In the box labeled "Enter any number of commands," type in all of the commands that the reader will be able to use to activate the task. Put one command on each line. In our case, we'll list these commands:

```
take mat
take doormat
move mat
move doormat
```

6. In the box labeled "Message to display on completion," type something like "When you move the doormat, you find a note underneath it. You pick

up the note and leave the mat where you found it."

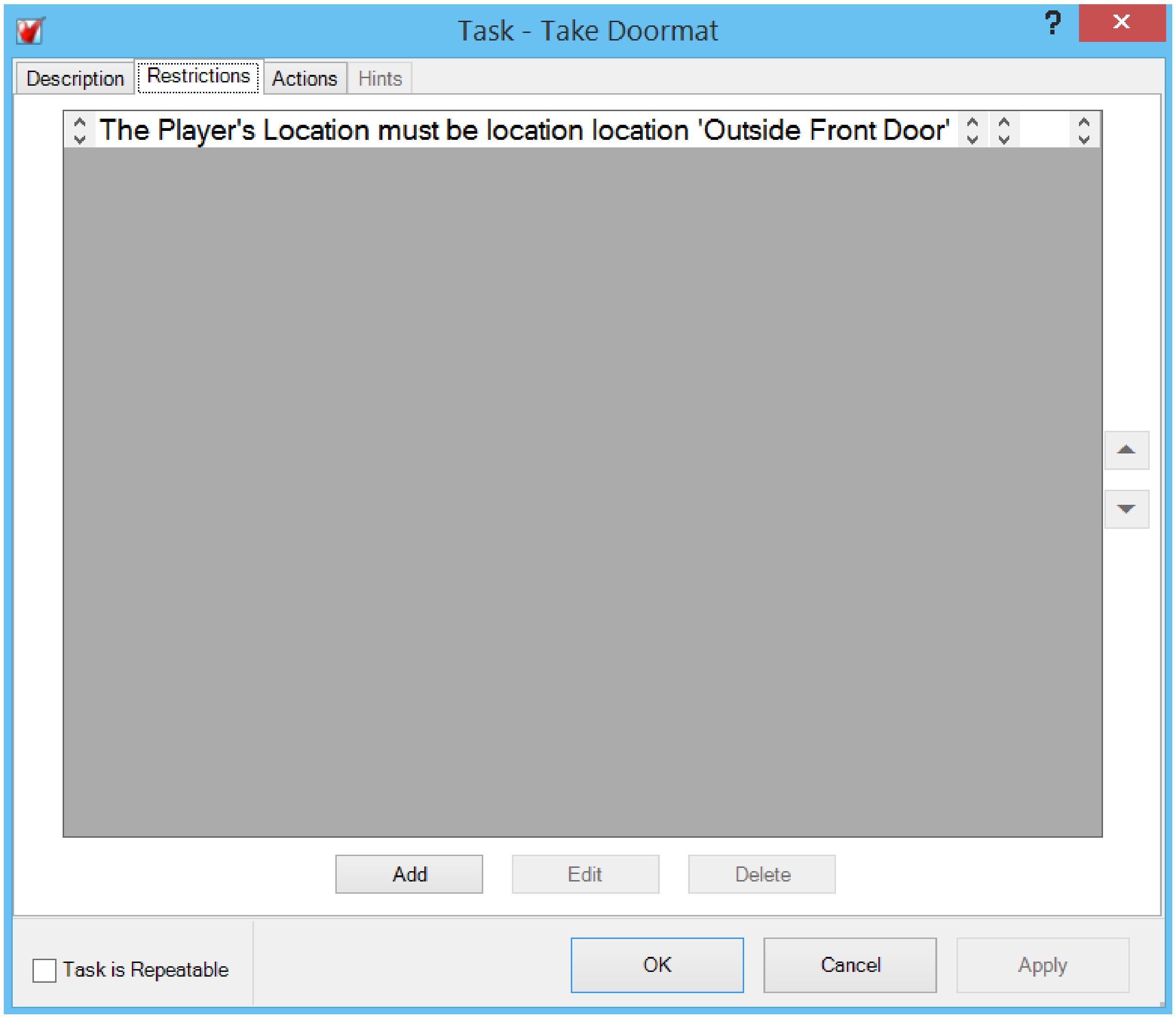
7. Since we want the note to be revealed only once, uncheck the box that's labeled "Task is repeatable." Then click on "Apply."

8. Click on the "Restrictions" tab. Here we'll indicate what conditions must be met for the task to do whatever it does. In this case, our only restriction will be that our character must be in the location called "Outside Front Door."

9. However, as of this writing, the screen that you're looking at right now is a little buggy. It displays a large box, currently blank. When you create restrictions, the box is supposed to display them. It should also allow you to change the order in which the restrictions are applied and to edit each restriction by clicking on it and then on an "Edit" button. However, when you've created a restriction, even if you make no mistakes in doing so, the restriction usually does not appear in the large box. To find out if the restriction is really in effect, you usually have to close the whole task-creation window by clicking on the "OK" button at the bottom of that window. When you do, you'll see that the name of your task appears on the "Task" list on the main screen of Adrift Developer. If you double-click on the name of your new task on the task list, and then on the "Restrictions" tag, the large box will list all the restrictions that you've created, just as it was supposed to all along. A bit later in this

tutorial, when we create actions for a task, you'll work with a very similar big box that lists the actions that you've created. This box exhibits the same bug.

9. With the “Restrictions” tab open, click on “Add.” Using the dropdown menus that appear, create the restriction that the player/character must be in the “Outside Front Door” location. Click on “OK” and then on “Apply.” You should now see your newly-made restriction in the previously-blank list on the “Restrictions” tab. Your restriction should read, somewhat awkwardly, “The Player's Location must be location location 'Outside Front Door'"

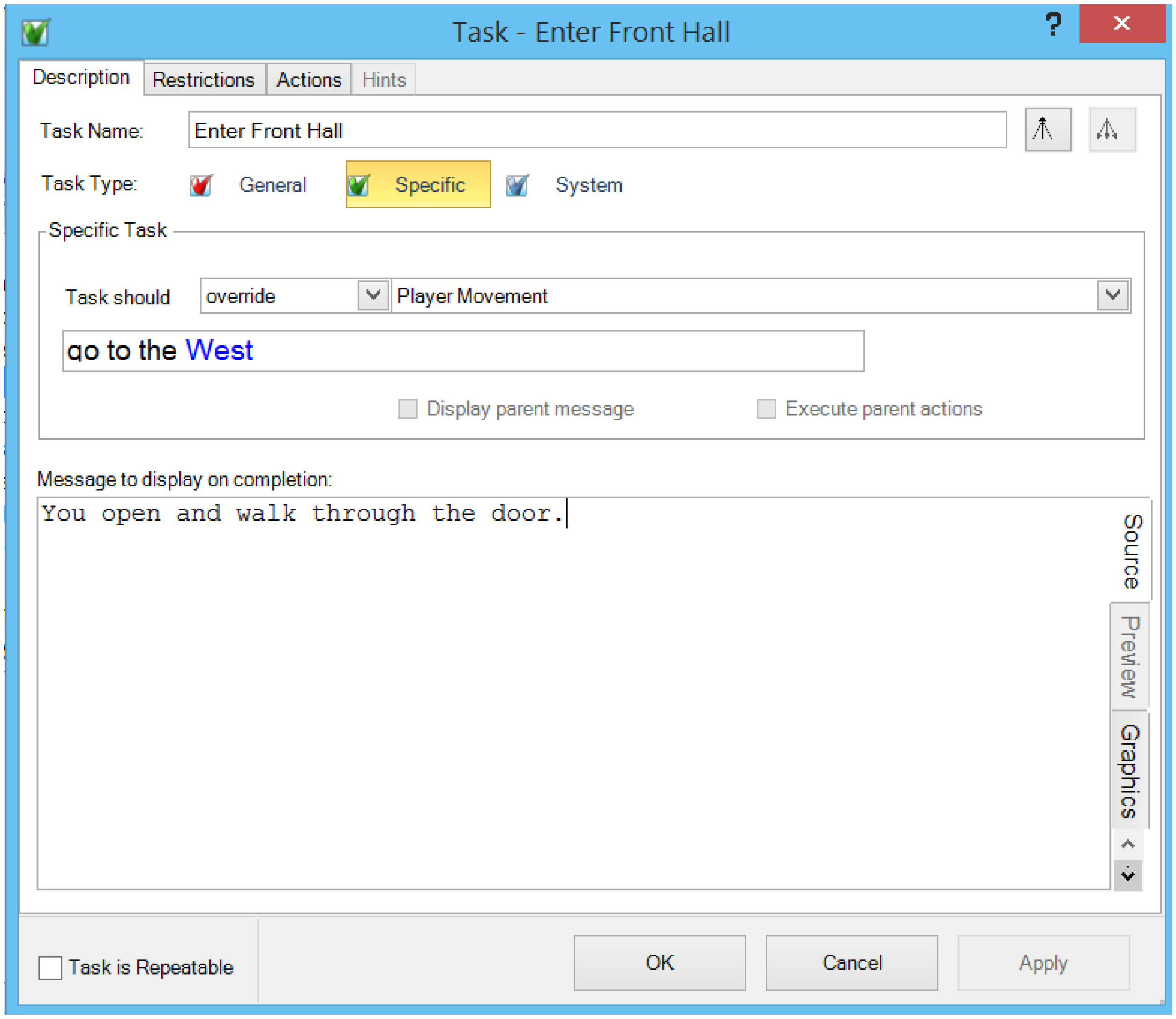


The Player's Location must be location location 'Outside Front Door'

10. Click on the “Actions” tab and then on “Add.”
11. Use dropdown menus to construct an action that moves the note from “Hidden” to “Held by the player.” Click on “OK” and then on “Apply.”
12. Your action should now appear on the previously blank list of actions, in the odd but useful form, “Move object 'a note' to held by character 'Player'"
12. Click on “OK” until the “Task – Take Doormat” window closes.

Create a task that ends the story

1. From the "Add Items" Section, click on "Task." We'll call this task "Enter Front Hall."
2. This task will not require that we introduce any new verbs, and so our task type will be "Specific."
3. Using dropdown menus, indicate that the new task should "override" "player movement" "go west."
4. As a "Message to display on completion, use something like "You open the door and walk through it."



Task Name: Enter Front Hall

Task Type: General [✓ Specific] System

Specific Task

Task should override Player Movement

go to the West

Display parent message Execute parent actions

Message to display on completion:

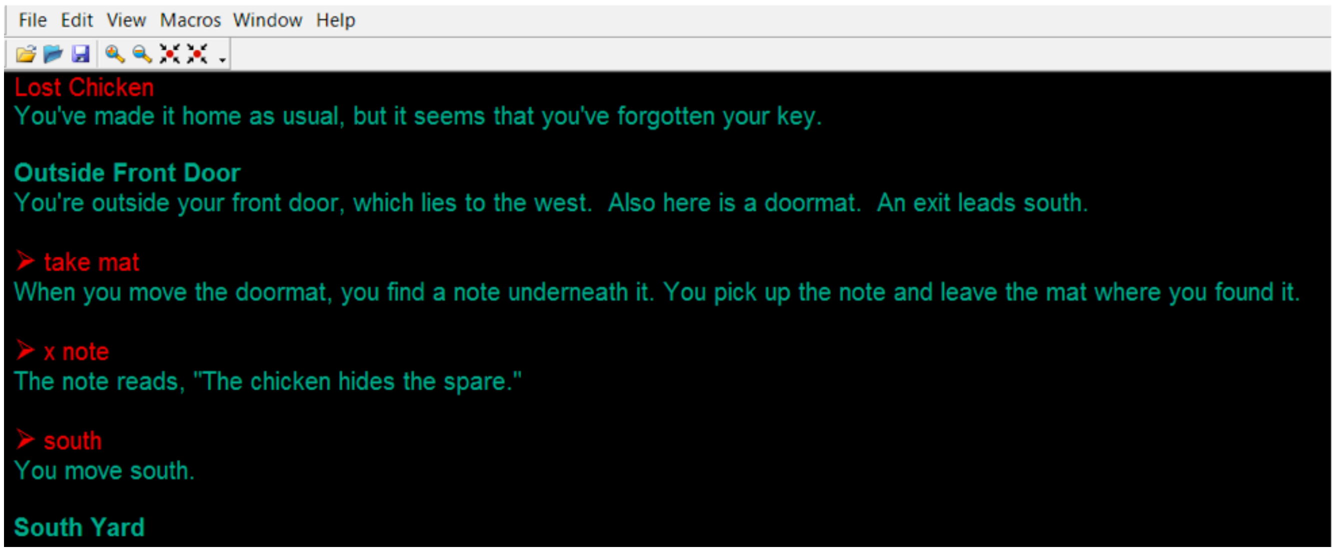
You open and walk through the door.

Task is Repeatable

5. Click on the “Restrictions” tab. Click on “Add.”
6. Using the dropdown menus create the restriction, “The Player's Location must be location location 'Outside Front Door'"
7. Click on “OK,” and “Apply.”
8. Again, click on “Add” in the “Restrictions” tab. Using the dropdown menus, create the restriction, “Object, 'a key' must be held by a character 'player'”
9. Click on “OK” and “Apply.”
10. Click on the “Actions” tab.
11. Click on the “End Game” tab, and choose “In Victory.”

Save and Test Your Story

1. To save your story, click on the circular icon in the top left of the screen, and choose “Save As.”
2. To test your story, click on the green triangle in the top center of the screen. This triangle is labeled, “Run Adventure.” Adrift Runner will open and display your story. Your player/character should be able to work through the entire story.



```
Lost Chicken
You've made it home as usual, but it seems that you've forgotten your key.

Outside Front Door
You're outside your front door, which lies to the west.  Also here is a doormat.  An exit leads south.

> take mat
When you move the doormat, you find a note underneath it. You pick up the note and leave the mat where you found it.

> x note
The note reads, "The chicken hides the spare."

> south
You move south.

South Yard
```

A challenge to try on your own

1. Implement the door, as mentioned in the transcript.
2. The Adrift Manual Wiki (http://wiki.adrift.co) explains how to create a fully-featured door, but you could get away with something much simpler in this story.

Chapter 26 -- Writing Interactive Fiction With Quest

Quest offers a way to create interactive stories with very little programming. Like Adrift, Quest works mainly though a series of windows and drop-down menus. Originally, Quest was, like Adrift, a Windows-only application, and, as of February 2015, its Windows version remains its most complete version. However, Quest comes in a Web version, too, and we'll use the Web variation for this tutorial, since it's more widely available. Still, if you have a Windows computer, you should use the Windows version of Quest. It's more mature, less buggy, and more complete, but it works very much like what you'll see in this tutorial. In our Quest tutorial, we'll implement the super-simple story "Lost Chicken," which we also used in our chapter on writing IF with Adrift. Here's a transcript of the story:

```
“Lost Chicken” Script

You've made it to your home, as usual,
but it seems that you've forgotten your
```

```
key.

Lost Chicken
An Interactive Fiction by Brendan
Desilets

Outside Front Door
You're outside your front door.  The door
is to the west, and your front yard is to
the south.

You can see a doormat and an oak door
here.

>take doormat
You find a note under the doormat and
pick it up.

>read note
The note reads, "The chicken hides the
spare."

>s

South Yard
This is the lovely south yard of your
home.

You can see a plaster chicken (closed)
```

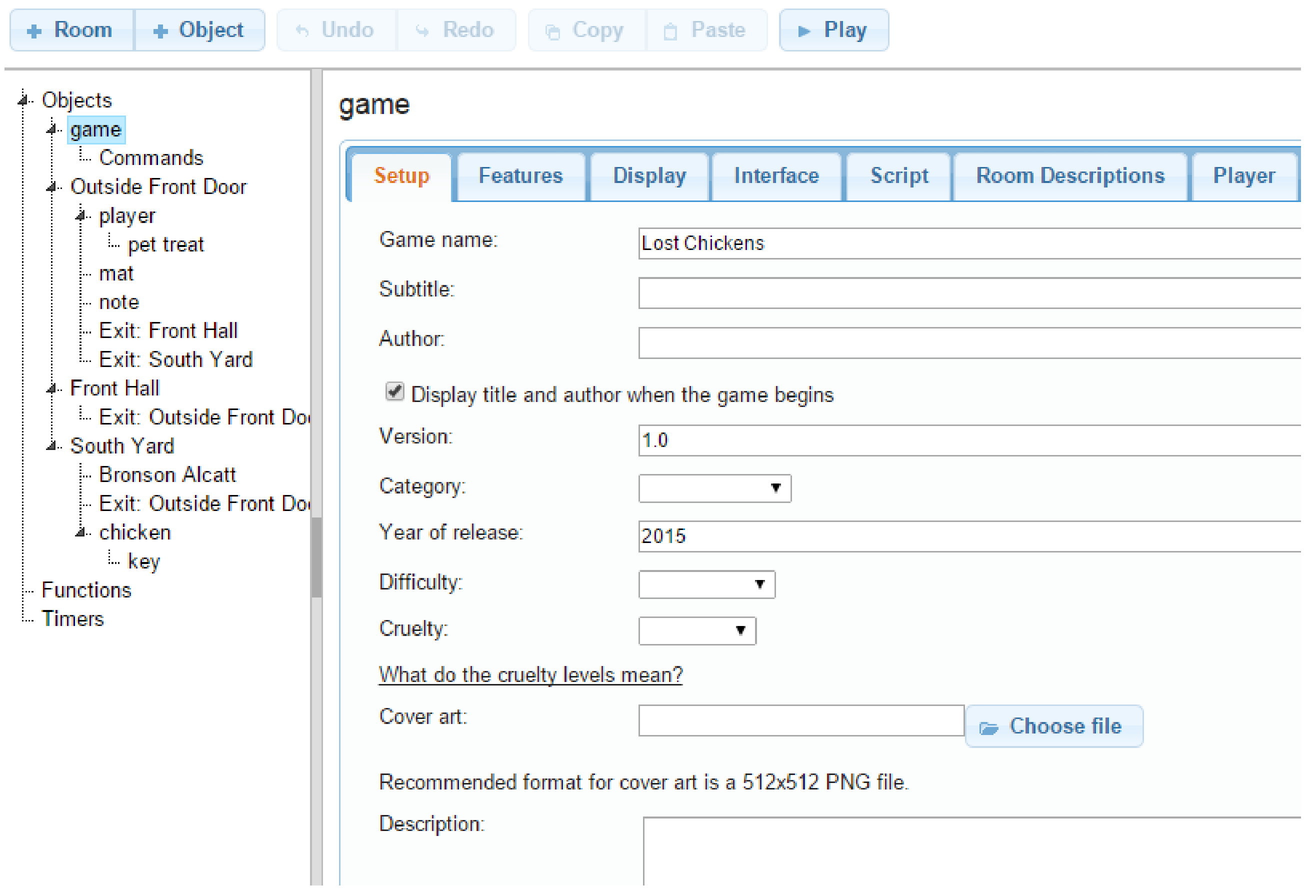
```
here.

>open chicken
You open the plaster chicken, revealing a
key.

>take key
Taken.

>n

Outside Front Door
You're outside your front door.  The door
is to the west, and your front yard is to
the south.

You can see a doormat and an oak door
here.

>w
You unlock the door and enter your house.

Front Hall
You have made it inside. Congratulations!


    *** The End ***
```

Quest’s Web version is at

http://textadventures.co.uk/quest. In order to create a story with the Web version of Quest, you first have to set up an account at the Quest website.

Set Up Your Story

1. Log in to your Quest account and click on the “Create” button at the top of your screen.
2. Click on “Create a New Game.”
3. In the “Game name:” box, type in the title of your story. Your “Game Type” is “Text Adventure.”
4. Click on “Create.”
5. Click on “Start editing!”
6. Wait around for a minute or two. Like many Web applications, Quest can be a bit slow to execute.
7. Quest will open with a screen that looks like this:



Game name: Lost Chickens

Subtitle:

Author:

✓ Display title and author when the game begins

Version: 1.0

Category:

Year of release: 2015

Difficulty:

Cruelty:

Cover art:

Recommended format for cover art is a 512x512 PNG file.

Description:

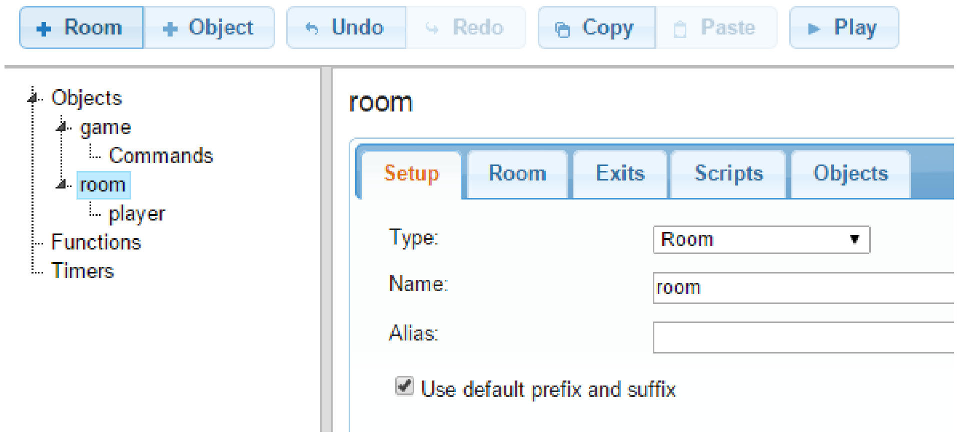
8. Notice that the word "game" is highlighted on the diagram that appears on the left side of the screen. That diagram is important, and we'll have to monitor it carefully as we go along.
9. Fill in the information that's asked for on the "Setup" tab. Feel free to examine and even experiment with the other tabs if you wish, with the exception of the "Player" tab. It's best to leave that one as it is, in order to head off possible buggy behavior.

Improve Your First Room

1. You may have noticed, in the chart on the left of the screen, that Quest has already created a

room for you and placed the player/character in that room.

2. Click on the word “room” in the chart at the left. Do not click on the button labeled “+ Room” at the top of the screen. We'll use that one later.
3. A screen that looks like this will open:



Type: Room

Name: room

Alias:

☑ Use default prefix and suffix

In the screen that opens, fill in the name of the opening room, in our case “Outside Front Door.”

4. As an “alias” use something like “outside your front door, which lies to the west.”
5. Click on the “Room” tab and fill in a description, such as “This is a grassy area outside your front door, which lies to the west.”
6. We'll use the other tabs later. For now, click on the “Save” icon at the upper right corner of the screen, unless Quest has already greyed it out, indicating that the story is already saved.

7. In addition to saving your story, you'll want to download it occasionally as your project gets larger. Downloaded versions of you story are excellent insurance, in case something goes wrong with your on-line copy.
8. The downloaded stories will require the full Quest program, a Windows-only tool, for editing them. However, in the case of a complex story, most users will be better off with the Windows program anyway, as it's faster, more complete, and more reliable.

Add Two More Rooms

1. Click on the word “game” in the chart at the left of the screen. Make sure that the word “game is highlighted, as we want our additional rooms to be inside the game, but not inside anything else.
2. Click on the “+ Room” button at the top of the screen.
3. Fill in the “Name” as “South Yard” and the “Alias” as something like “a lovely yard, south of your home.”
4. Click on the “Room” tab and fill in the description text, “This is the lovely South Yard of your home.”
5. Click on the “Exits” tab. Since we want the player to go north from here to Outside Front Door, click on the circle beside the word “North.”

When you've done so you'll see a dropdown menu labeled “Create an exit to:” Use this dropdown menu to select “Outside Front Door.”

6. We want this exit to work in both directions, so check the box that is labeled “Also create exit in other direction.”
7. Once again, click on the word “game” in the chart that appears at the left of the screen. Click on the “+ Room” button at the top of the screen.
8. This time, we'll name the room “Front Hall,” and we'll give it the alias “cozy front hall of your home.”
9. We'll not use the “default prefix” for this room. Instead we'll use the prefix “the.”
10. We won't use the “Room” tab here, but we will click on the “Exits” tab.
11. With the “Exits” tab open, we'll create a two-way exit that leads east to “Outside Front Door.” Later, we'll create a restriction on when the player can go west from Outside Front Door to Front Hall.

Try Out Your Story, So Far

1. Click on the “Save” button in the upper right part of the screen, unless the button is already greyed out.
2. Click on the “Play” button at the top of the screen. After a few seconds, you should see you story running, with the player in the

room called “Outside Front Door.”
3. You should be able to move around through the three rooms you've created.
4. Occasionally, the Quest game-running program seems to spit out odd error messages, in response to common IF commands. Often, simply trying your command a second time will solve the problem.

Create Two Objects, the Doormat and the Note

1. New writers of interactive fiction often ask about how a writer can cause a concealed object to appear, the way the note appears in “Lost Chicken,” when the player/character takes the doormat. Here's one of the several possible ways to get this effect.
2. First, let's create the note.
3. Click on “Outside Front Door” on the chart that appears at the left of your screen. You will see the familiar screen which you used to provide information about the “Outside Front Door” room.
4. On this screen, click on the “Objects” tab. A screen will open, showing that one object, the player, is already in “Outside Front Door.”
5. In the “Objects” tab, click on “Add.” A screen will open, allowing you to provide information about your new object.

6. Give your new object the name “note.” Make its description, “The note reads, 'The chicken hides the spare.'"
7. Click on the “Inventory” tab, and indicate that the new object can be taken.
8. Make sure that the box beside “Visible” is not checked. The note, in its initial form, will not be visible to the player, but we'll make it visible soon.
9. Once again, click on “Outside Front Door” on the chart at the left of the screen. Then click on the “Objects” tab.
10. Click on Add.
11. This time, call your object “mat,” and make its alias “doormat.” Its description should be something like “an ordinary doormat.”
12. Click on the “Object” tab and then click on the “+ Add” button that appears beside “Other Names.” Add the new name “mat.”
13. Click on the “Inventory” tab and indicate that the mat can be taken.
14. Click on the “Save” button if it's not already greyed out.

Create Your First Script

Some Background Concepts

1. Scripts are very important in Quest.
2. Scripts can be quite complex.
3. Scripts allow an author to change the usual

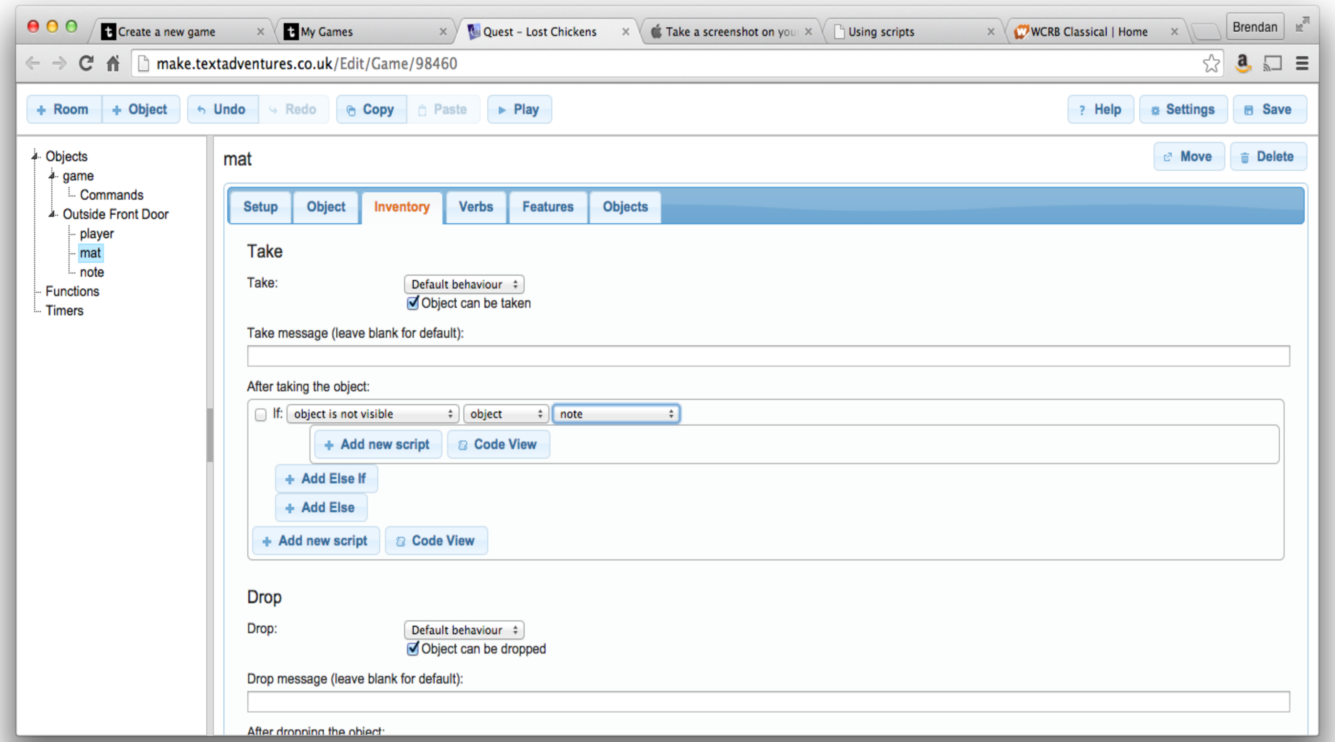
behavior of the game world. For example, in our "Lost Chicken" story, we can change what happens when the player/character takes the mat, causing the note to appear.

4. As we build a script, we will often add other scripts to it. It is extremely common for an author to start creating a script and then to add more and more scripts to it.
5. In order to make the note appear when we want it to, we'll build a script. These are the characteristics of that script.
   1. It takes place only after the player/character takes the mat when the note has not yet been revealed. We don't want the note to mysteriously reappear every time the player/character drops mat and picks it up again.
   2. It causes the note to appear.
   3. It tells the reader what the previous elements of the script caused to happen.
   4. We'll build our script, using the "Inventory" tab for the object called "mat."

Our First Script – the Building Proces

1. On the chart that appears at the left of your screen, click on "mat."
2. Click on the "Inventory" tab.
3. Under "After taking the object," click on the button labeled "+ Add new script." A popup

window will open.
4. Click on the word “If,” which is in the upper right part of the popup window.
5. The word “IF” will appear, followed by a series of dropdown menus.
6. Using the dropdown menus, create an “if” clause that reads, “IF: object is not visible object note.”



Take

Take: Default behaviour
☑ Object can be taken

Take message (leave blank for default):

After taking the object:

☐ If: object is not visible | object | note

Drop

Drop: Default behaviour
☑ Object can be dropped

Drop message (leave blank for default):

After dropping the object:

7. By now, you've probably noticed that the screen you're looking at has become pretty complicated. For one thing, it contains at least three identically-labeled buttons that read “+ Add new script.”
8. One of these “+ Add new script” buttons is directly under “If: object is not visible object

note." And this particular button is indented a bit.

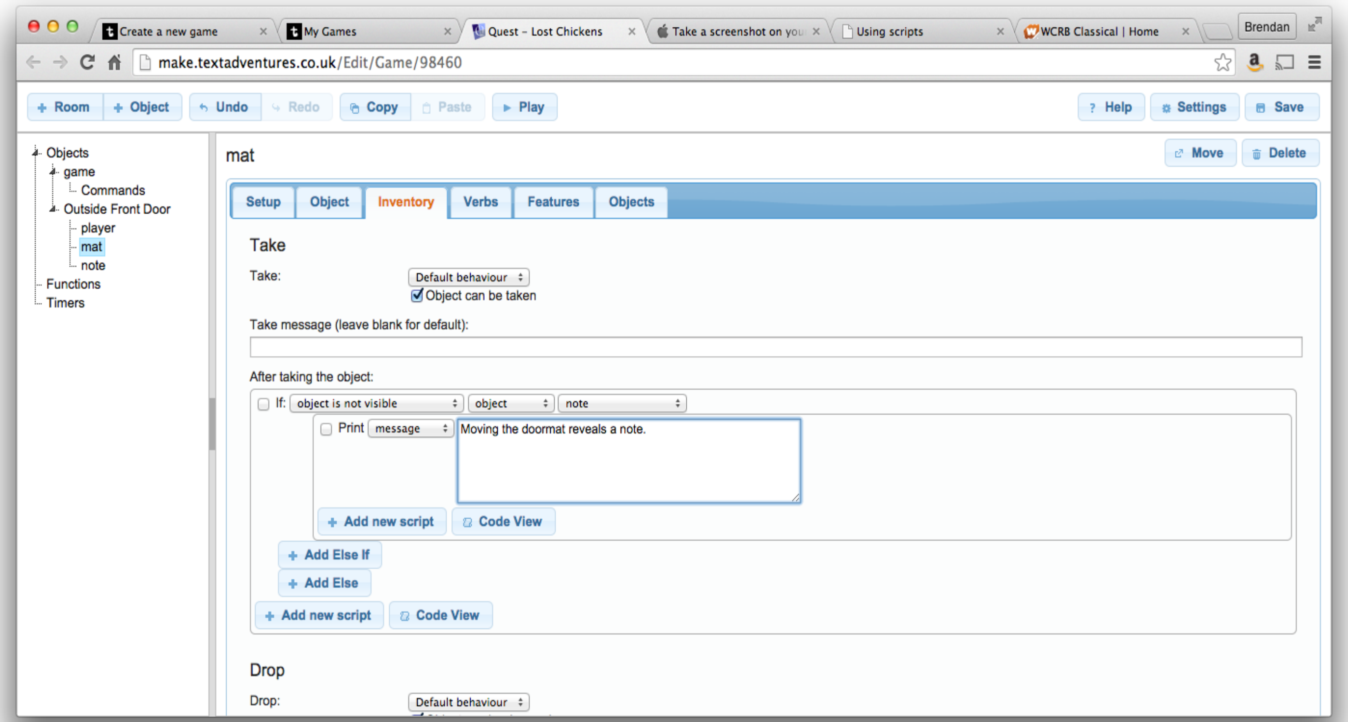
9. The indenting is intended to show that whatever command (or "script") you add using the indented button will be carried out only if the note is not visible.

10. This sort of indenting is very important in creating good scripts. Such scripts often require placing groups of commands within other groups of commands, and the indenting is helpful in keeping all the groups sorted out.

11. Click on the indented button that's labeled "+ Add new script." A popup window will open."

12. The defaults that are already selected in the popup window indicate that you want to print a message. As a matter of fact, that's a good idea here, so just click on "OK."

13. Indicate that the message you want to print is something like "Moving the doormat reveals a



mat

Take

Take: Default behaviour
☑ Object can be taken

Take message (leave blank for default):

After taking the object:

If: object is not visible | object | note

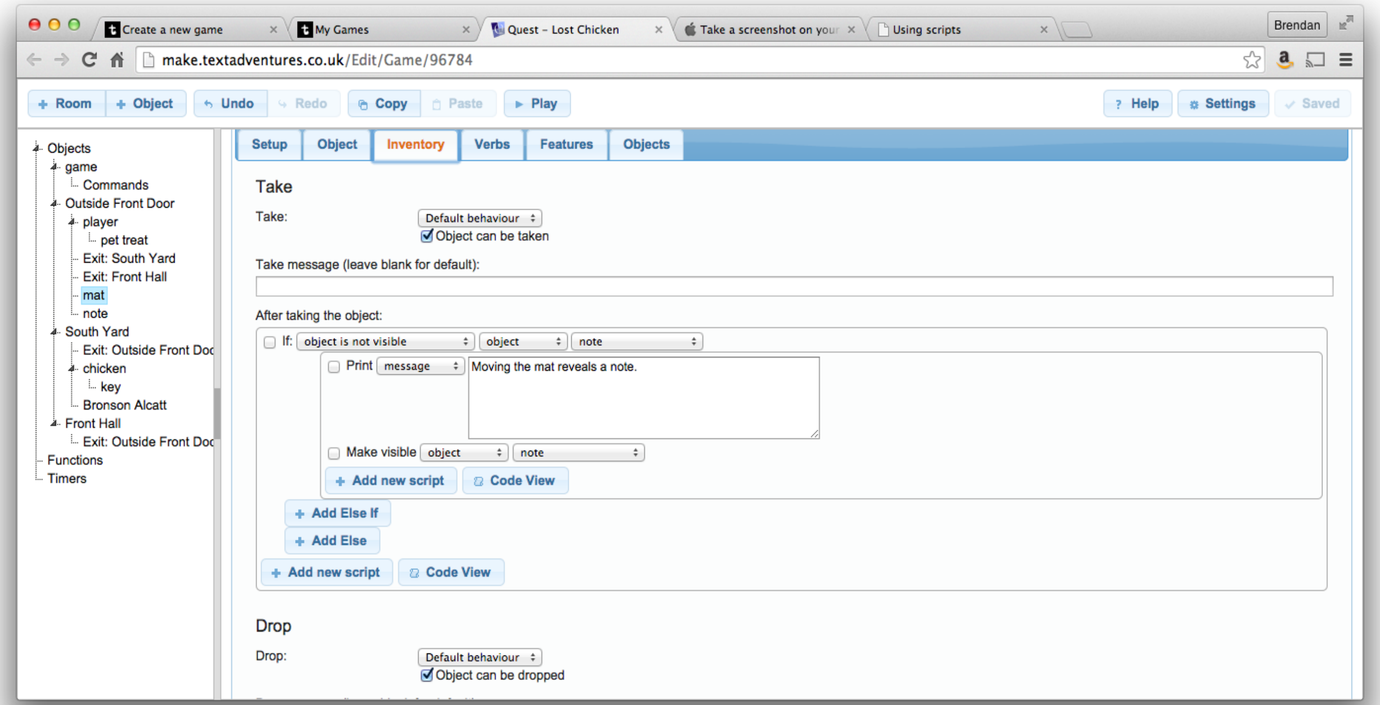
Print message | Moving the doormat reveals a note.

Drop

Drop: Default behaviour

note.”

14. Notice that there's now a “+ Add new script” button directly under the word “Print.” Click on that “+ Add new script” button.
15. This time, on the popup window, select the “Objects” button and choose “Make object visible.” Click on “OK.”
16. Now using the dropdown menus, assemble this command, “Make visible object note.”
17. Your screen should now look like this:



Take

Take: Default behaviour
☑ Object can be taken

Take message (leave blank for default):

After taking the object:

If: object is not visible | object | note

Print message: Moving the mat reveals a note.

Make visible object | note

Drop

Drop: Default behaviour
☑ Object can be dropped

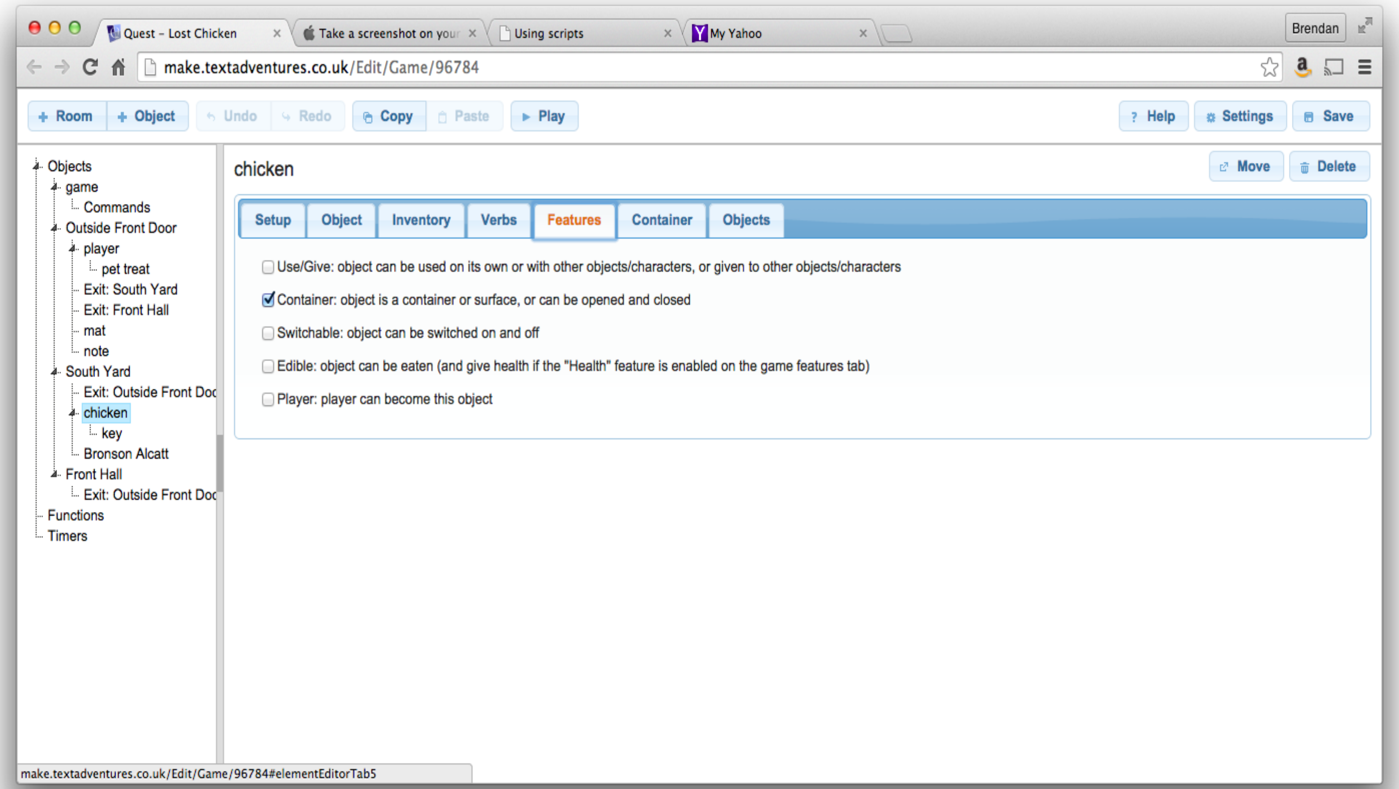
18. Click on “Save” if it's not already greyed out.

Then try out your story so far, using the “Play” button.

Create Your First Container

1. IF authoring systems usually have some built-in object types. One of these is the container, a type of object that can hold other objects. In “Lost Chicken,” the plaster chicken is a container that has a key in it.
2. To create the plaster chicken, begin by clicking on “South Yard” in the chart on the left side of the screen. Then click on the “Objects” tab.
3. Click on “+ Add.”
4. A familiar screen for creating an object will appear, with the “Setup” tab selected. Fill in the name “chicken” and the alias “plaster chicken.” For a description, type something like “An old plaster chicken, used as a container.
5. With the “Object” tab depressed, add “chicken” as an “other name.”

6. Click on the “Inventory” tab. Make sure that “Object can be taken” is not checked.



☐ Use/Give: object can be used on its own or with other objects/characters, or given to other objects/characters

☑ Container: object is a container or surface, or can be opened and closed

☐ Switchable: object can be switched on and off

☐ Edible: object can be eaten (and give health if the "Health" feature is enabled on the game features tab)

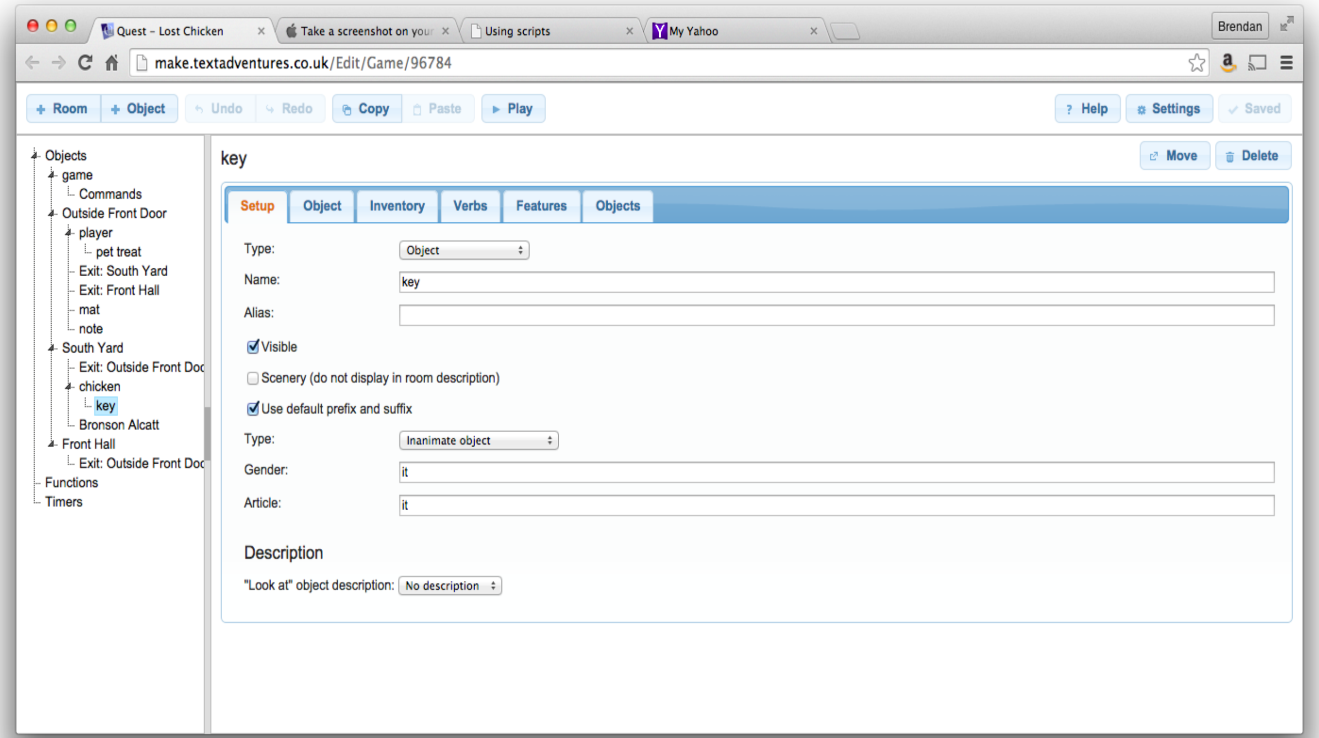
☐ Player: player can become this object

7. Click on the “Features” tab and select “Container: object is a container or surface, or can be opened and closed.”

8. A new tab, labeled “Container,” will appear. Click on this tab, and chose, as the “Container type,” “Openable/Closable."

9. Then, check “Can be opened,” and “Can be closed.”

10. Click on the “Objects” tab and then on “+ Add.” Add the object “key,” and fill in its details



Type: Object

Name: key

Alias:

☑ Visible

☐ Scenery (do not display in room description)

☑ Use default prefix and suffix

Type: Inanimate object

Gender: it

Article: it

Description

"Look at" object description: No description

in our now-familiar fashion.

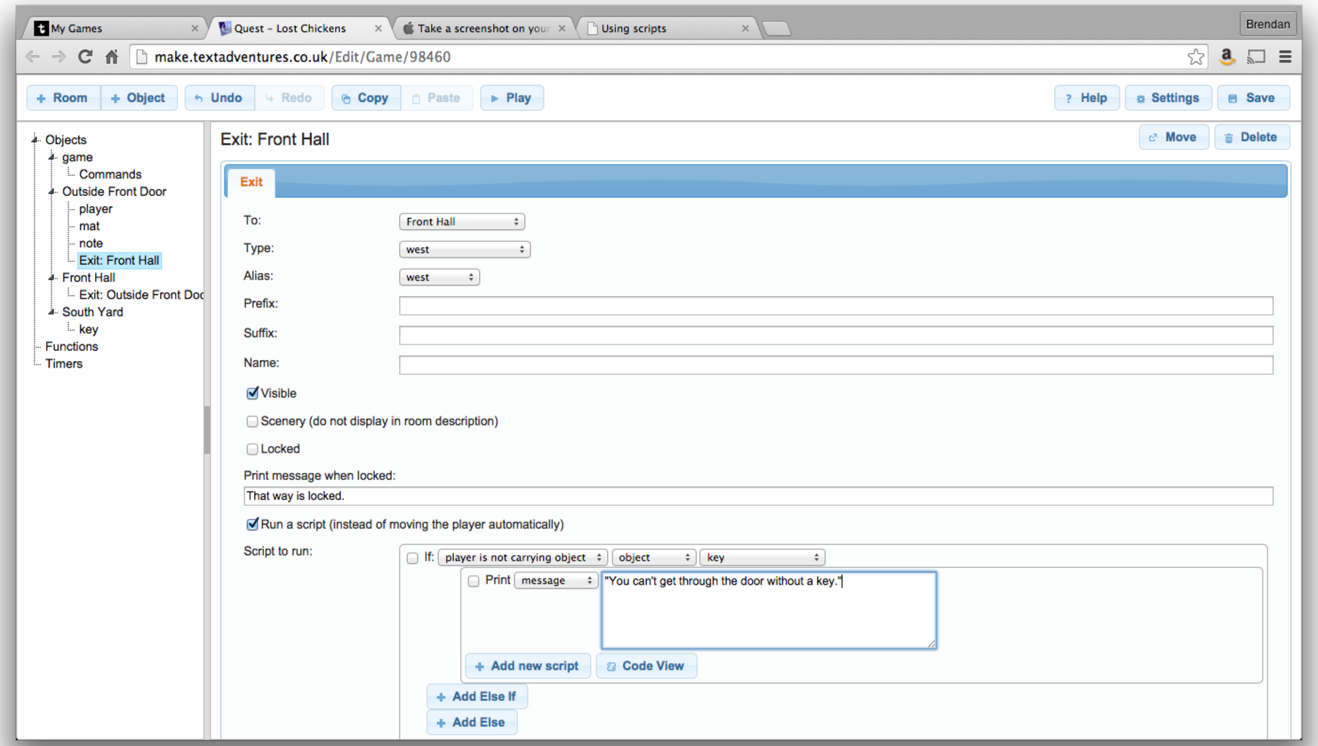
11. Try out your story. It should be possible for the player/character to go south to the South Garden, open the plaster chicken, and take the key.

Create a Script That Ends Your Story

1. Our story ends when the player/character gains entry to the Front Hall. It should be impossible for the player/character to get to the

Front Hall unless he or she is carrying the
key.
2. We'll use a script to implement this behavior.
3. In the chart at the left of the screen, locate
“Outside Front Door,” but don't click on it.
4. Under “Outside Front Door,” locate “Exit: Front
Hall,” and click on it.
5. Put a checkmark beside “Run a script (instead
of moving the player automatically).”
6. As we did on our previous script, start by
choosing “If'” from the upper right part of the
popup menu, and then build a statement, using
the dropdown menus that will appear. The
statement should be, “If player is not carrying
object object key.”
7.Once again, you'll see a fairly complex screen
with a number of buttons labeled “+ Add
new script.” Pick out the “+ Add new script”
button that is indented under your “If” clause,
and click on that button.
8. Again, a popup window will open, and, as it
happens, the default behavior, printing a
message, is what we want. Click on “OK,” and fill
in the message, “You can't get through
the door without the key.”

9. Now, we'll add some unfamiliar features to our script. First, click on the “+ Add Else” button, which appears below your “If” clause.



Exit: Front Hall

To: Front Hall
Type: west
Alias: west
Prefix:
Suffix:
Name:
[✓] Visible
[ ] Scenery (do not display in room description)
[ ] Locked
Print message when locked: That way is locked.
[✓] Run a script (instead of moving the player automatically)
Script to run:
If: player is not carrying object | object | key
Print message "You can't get through the door without a key."

10. Identify the “+ Add new script” button that is immediately under the word “Else” and slightly indented. Click on that button.

11. A familiar popup window will open. Click on “Move” at the top of the window.

12. Using dropdown menus, build the command “Move object object player to object “Front Hall.”

13. Identify the “+Add new script” button that appears immediately under the command that

you just created. Click on that button.

14. Click on “OK.” Then fill in the “Print” box that appears with a message like, “Congratulations! You've finished the story.
15. Press the “+Add new script” button that appears below your “Print” statement. A familiar popup window will open.
16. Click on the “Game State” button that appears along the left side of the popup window. Then choose “Finish the game” and click on OK.
17. Save your story and try it out. You should be able to reach the end of the story.

A Further Challenge

Implement the front door. There's a way to do this sort of implementation in the Quest documentation at http://docs.textadventures.co.uk/quest/tutorial/using_lockable_exits.html. However, you probably don't need a fully-implemented door for our story.

Chapter 27 – Writing Interactive Fiction With Inform 7

Inform 7 is an extremely powerful tool for writing interactive stories. Unlike similar tools, Inform 7 uses a form of "natural language." In other words, it allows a writer to create a work of interactive fiction by writing somewhat normal English sentences. However, the writer must follow a rather strict set of rules in creating sentences that will really lead to an interactive story.

I. Set Up Your Project

A. Start Inform 7.

B. From the list of choices that appears, choose "Start a New Project."

C. Choose the directory for your project by clicking on the button with three dots on it. This button appears on the right side of your screen.

D. From the list of drives and directories that appears, choose the directory you want to use to store your project, and then click on OK.

E. In the appropriate box, type the name of your project, which should be the title of your interactive story.

F. In the appropriate box, type the name of the author. This will be your name. Do not include any punctuation in your name.

G. Click on "Create."

H. A screen with two windows will appear. In the left window, you'll see the title and author of your story. You'll be typing your source code in this window. Your source code will describe your story in a way that Inform 7 can understand. In the right window, for now, you'll see a list of chapters that are included in the documentation for Inform 7.

I. Near the top of your screen, you'll see two boxes with magnifying classes in them. These are search boxes, one for searching through your source code and the other for searching through Inform's documentation. Use the documentation search box to find answers to your questions about Inform.

II. Create your First Room

A. In interactive fiction, a "room" can be any location, outdoors or indoors.

B. Under the title and author, in the left window, leave a blank line, so that you can see where your story really begins. Then type the name of your first room, followed by this phrase:

is a room.

C. In quotation marks, right after the sentence that names your room, type a good description for your room. The quotation marks tell Inform to display text on the computer's screen, for the reader to see. Your source code should now look something like this:

```
"A-221" by Brendan Desilets

A-221 is a room. "A-221 is a fairly drab
classroom, with twenty-four student desks
and a like number of computers. It sports
at least ten teacher-made signs about
grammar and literature and one long,
commercial poster. A filing cabinet is
the room's most prominent storage unit."
```

III. Compile Your Story

A. Compiling is changing your source code into a working interactive story. If you have completed the steps outlined above, your story can now be compiled, though the player/character won't be able to do much, as yet.

B. To compile your story, press the "Go" button at the top of your screen. After a few seconds, if the story compiles correctly, you will see the description of your first room in the window on the right side of your screen. If the story does not compile correctly, Inform 7 will try to tell you why it did not compile.

C. If your story does not compile, Inform will use a little curved arrow to show where the mistake in your code seems to be. Click on the arrow to see what you need to fix.

IV. Create a Second Room

A. Now, you can create a second room that connects to your first room. Begin by naming your second room with a sentence like this one:

The Hallway is a room.

B. Now, add a sentence that shows how the two rooms connect, such as:

It is east of A-221.

C. In quotation marks, add the description of your new room. The source code for your story should now look something like this:

```
"A-221" by Brendan Desilets

A-221 is a room. "A-221 is a fairly drab
classroom, with twenty-four student desks
and a like number of computers. It sports
at least ten teacher-made signs about
grammar and literature and one long,
commercial poster. A filing cabinet is
the room's most prominent storage unit."

The Hallway is a room. It is east of A-
221. "This is an ordinary school hallway.
Room A-221 is to the west. You can't
think of any reason you would want to go
```

```
in any other direction right now."
```

V. Compile Your Story a Second Time

A. Press the "Go" button to compile your story again.

B. Once again, after a few seconds, if the story compiles successfully, you will see the story running in the window on the right side of your screen. If the story does not compile, Inform will try to tell you why it did not compile, so that you can fix the source code.

C. Once your story with two rooms compiles, the player character will be able to move between the rooms.

VI. Create Your First Object

A. Actually, the rooms you have already created are objects of a sort. Now, you'll create an object that can be picked up and carried.

B. After leaving a line space for easier reading, type the name of your object, followed by

> is in A-221.

C. Then, if, for example, the object is a key, type:

> The description of the key is

D. In quotation marks, type the description of your object. The source code for your object should now look something like this:

The key is in A-221. The description of the key is "An ordinary brass key."

E. Compile your story once again. The player/character should now be able to pick up the object you created.

VII. Create Your First Container

A. Inform makes it easy to create objects that can contain other objects.

B. Inform also makes it easy to create scenery. The player-character cannot take a piece of scenery.

C. In this example, we will create a closed, locked container called the filing cabinet. Here is the source code for the filing cabinet:

```
   The filing cabinet is scenery in A-
221. It is a closed openable
container. It is locked and lockable.
"This filing cabinet is designed to
store and organize all sorts of
papers, but it could hold lots of
other things, too."
```

D. In order to make the key unlock the filing cabinet, we add a sentence to the source code for the key.

This new sentence is:

The key unlocks the filing cabinet.

E. Our source code for the key now looks like this:

```
      The key is in A-221. The key
unlocks the filing cabinet. The
description of the key is "An ordinary
brass key."
```

VIII. Create Your First Rule

A. A rule is a way of telling Inform about something that you want to happen, under certain conditions. The statement of the conditions ends with a colon.

B. Here is an example rule for our story. This rule uses the word "say," which tells the computer to show some text to the player. This rule applies only when the story begins. Notice how this rule uses double quotation marks (the usual kind) and single quotation marks.

```
When play begins: say "Oh, no! You've
lost your red English binder. But here
comes your teacher. Perhaps he's seen it.
    'Maybe,' he says. 'I just locked a
binder in the filing cabinet in Room A-
221. See if it's yours. You'll have to
find the key first, though. I'm not quite
```

```
sure where I left it.'
      You find your way to A-221 to begin
the search."
```

IX. Create Another Object, the Binder

A. If the player-character is going to be able to find the missing binder, we must "implement" the binder. In other words, we have to create it as an object.

B. Here's some source code that creates the binder and places it inside the locked filing cabinet:

```
The binder is in the filing cabinet. The
description is "The red English binder
that you lost recently. You've been
looking for it everywhere."
```

C. Remember to compile your story frequently, to check for any problems.

X. Create a Rule to End the Story

A. Now, let's create a rule that ends the story in victory when the player/character gets the binder. Every turn, this rule will check to see if the player-character has the binder.

B. Here's the source code for this rule:

```
An every turn rule:
if the player is carrying the
binder, end the story saying
```

```
“Congratulations! You've   won.”
```

Here is the complete source code of our story so far:

```
"A-221" by Brendan Desilets

When play begins: say "Oh, no! You've
lost your red English binder. But here
comes your teacher. Perhaps he's seen it.
'Maybe,' he says. 'I just locked a binder
in the filing cabinet in Room A-221. See
if it's yours. You'll have to find the
key first, though. I'm not quite sure
where I left it.'
You find your way to A-221 to begin the
search."

A-221 is a room. "A-221 is a fairly drab
classroom, with twenty-four student desks
and a like number of computers. It sports
at least ten teacher-made signs about
grammar and literature and one long,
commercial poster. A filing cabinet is
the room's most prominent storage unit."

The Hallway is a room. It is east of A-
221. "This is an ordinary school hallway.
Room A-221 is to the west. You can't
```

```
think of any reason you would want to go
in any other direction right now."

The key is in A-221. The key unlocks the
filing cabinet. The description of the
key is "An ordinary brass key."

The filing cabinet is scenery in A-221.
It is a closed openable container. It is
locked and lockable. "This filing cabinet
is designed to store and organize all
sorts of papers, but it could hold lots
of other things, too."

The binder is in the filing cabinet. The
description is "The red English binder
that you lost recently. You've been
looking for it everywhere."

An every turn rule:
if the player is carrying the binder:
end the story saying "Congratulations!
You've won!"
```

It is now possible to read this story to its easy end.

XI. Create Your First Character (Other Than the Player/Character)

A. Inform allows for the easy creation of characters.

A male character is called a man and a female character is called a woman, regardless of the character's age or gender identification.

B. Here is the source code for a character:

```
Jeff is man in A-221. The description of
Jeff is "A sixth grader, wearing a
baseball shirt.”
```

C. Using brackets, we can add something special to Jeff’s description by creating a condition, or an "if," like this:

```
Jeff is a man in A-221. The description
of Jeff is "A sixth grader, wearing a
baseball shirt. [if the key is carried by
Jeff] He is carrying a key." [end if]
```

XII. Create Your First "Instead" Rule

A. The real creativity in writing interactive fiction happens when we change what Inform ordinarily does. For example, if the player-character attacks someone, Inform usually responds, "Violence isn't the answer to this one."

B. If, instead, we want the player-character to lose if he or she attacks someone, we can make an "Instead" rule like this one:

```
Instead of attacking Jeff:
```

```
say "You have been suspended from school
for violent behavior."; end the story
saying "And you have failed to recover
your binder."
```

C. Notice how we use the semicolon (;) to separate instructions we are giving to Inform.

XIII. Create More "Instead" Rules

A. "Instead" rules can be used for many purposes. For example, we can use them for conversation with characters.

B. To enable our character Jeff to talk about the key, we could use this code:

```
Instead of asking Jeff about "the key":
say "It's the key to the filing cabinet."
```

C. In case the player wants to "ask Jeff about key," (leaving out the word "the") we can add this code:

```
Instead of asking Jeff about "key": say
"It's the key to the filing cabinet."
```

D. We can use an "instead" rule to get Jeff to take the key if we offer it to him, using this code:

```
      Instead of giving the key to Jeff:
      say "Now Jeff has the key.";
```

```
    move the key to Jeff.
```

E. To get Jeff to give the key to the player, if asked, we could add:

```
    Instead of asking Jeff for the key:
    say "Now you have the key.";
    move the key to the player.
```

XIV. Scoring

A. You may have noticed that Inform seems to be trying to keep score in our sample story.

B. Since scoring is probably not appropriate for our brief tale, let's add this to our source code:

```
    Use no scoring.
```

XV. Adding a Surprise

A. Suppose that we want to add a surprise to the end of the game. When the player takes the binder, he or she will find a homework
pass under it. The player will not see the pass until he or she takes the binder.

B. First, let's implement the pass. Notice that, for now, the pass will not
be in either of the story's rooms.

```
The homework pass is a thing. The
description is "A special pass, signed by
```

```
your teacher, that allows you to skip a
homework assignment."
```

C. Now, we’ll make an "instead" rule to describe what happens when the player takes the binder.

```
Instead of taking the binder: say "As you
take the binder, you find that, under it,
is a homework pass, made out to you.";
move the pass to the cabinet; move the
binder to the player.
```

D. Now, let's change our "every turn" rule, so that the game ends when the player takes the pass.

```
An every turn rule:
if the player is carrying the pass:
end the story saying "Congratulations!
You've won!"
```

XVI. Implementing

A. There's lots more to implement, even in this very brief example.

B. On your own, try implementing the posters and desks in Room-A221.

Here's the source code of our example story, so far.

```
"A-221" by Brendan Desilets

Use no scoring.

When play begins: say "Oh, no! You've
lost your red English binder. But here
comes your teacher. Perhaps he's seen it.
'Maybe,' he says. 'I just locked a binder
in the filing cabinet in Room A-221. See
if it's yours. You'll have to find the
key first, though. I'm not quite sure
where I left it.'
You find your way to A-221 to begin the
search."

A-221 is a room. "A-221 is a fairly drab
classroom, with twenty-four student desks
and a like number of computers. It sports
at least ten teacher-made signs about
grammar and literature and one long,
commercial poster. A filing cabinet is
the room's most prominent storage unit."

The Hallway is a room. It is east of A-
221. "This is an ordinary school hallway.
Room A-221 is to the west. You can't
think of any reason you would want to go
in any other direction right now."
```

```
The key is in A-221. The key unlocks the
filing cabinet. The description of the
key is "An ordinary brass key."

The filing cabinet is scenery in A-221.
It is a closed openable container. It is
locked and lockable. "This filing cabinet
is designed to store and organize all
sorts of papers, but it could hold lots
of other things, too."

The binder is in the filing cabinet. The
description is "The red English binder
that you lost recently. You've been
looking for it everywhere."

Jeff is man in A-221. The description of
Jeff is "A sixth grader, wearing a
baseball shirt. [if the key is carried by
Jeff]
He is carrying a key.” [end if]

Instead of attacking Jeff: say "You have
been suspended from school for violent
behavior."; end the game saying "And you
have failed to recover your binder."

Instead of asking Jeff about “the key”:
say “It's the key to the filing cabinet.”
```

```
Instead of asking Jeff about “key”: say
“It's the key to the filing cabinet.”

Instead of giving the key to Jeff: say
"Now Jeff has the key."; move the key to
Jeff.

Instead of asking Jeff for the key: say
"Now you have the key."; move the key to
the player.

The homework pass is a thing. The
description is "A special pass, signed by
your teacher, that allows you to skip a
homework assignment."

Instead of taking the binder: say "As you
take the binder, you find that, under it,
is a homework pass, made out to you.";
move the pass to the cabinet; move the
binder to the player.

An every turn rule: if the player is
carrying the pass, end the story saying
"Congratulations! You've won!"
```

XVII. Create Your First Value

A. A value is a quality, or "property," that changes.

Actually we have already used values in our simple story. For example, at the start of the story, our filing cabinet is closed and locked, but, later, it becomes unlocked and open. We didn't have to do anything special to set up these values because Inform already knew about closed and locked containers.

B. However, we can set up our own values or variables. For instance, we can set up a value called "mood." We can add that mood applies to people. And we can set up as many moods as we want. For now, let's settle for "unhappy" and "pleased." Let's set Jeff's opening mood as "unhappy."

C. Here's the source code we should add:

```
Mood is a kind of value. The moods are
unhappy and pleased. People have mood.
The mood of Jeff is unhappy.
```

XVIII. Use the "Mood" Value in Your Story

A. What can we do with values? Actually, values are extremely powerful, and we can do a great deal with them. Let's start by including Jeff's mood in his description, changing the description to read as follows:

```
Jeff is man in A-221. The description of
Jeff is "A sixth grader, wearing a
baseball shirt. [if the key is carried by
Jeff] He is carrying a key. [end if] Jeff
looks [the mood of Jeff]."
```

B. Now, let's invent a way to change Jeff's mood. Suppose that we want the player to bribe Jeff in order to get the key from him. We can use this source code to create a coin that the player can use as a bribe:

```
The coin is a thing. The player carries
the coin. The description of the coin is
"A typical piece of currency -- worth
something to most people."
```

C. Now, let's create an instead rule that allows the player to change Jeff's mood by giving him the coin. Here's the source code:

```
Instead of giving the coin to Jeff: move
the coin to Jeff; now Jeff is pleased;
say "Jeff looks very pleased."
```

D. Next, let's create two instead rules that force the player to change Jeff's mood before the key can change hands. The source code follows, but it's a little complicated:

```
Instead of asking Jeff for the key when
Jeff is unhappy: say "Jeff refuses to
```

```
give the key, but points to the coin
you're carrying.";

Instead of asking Jeff for the key when
Jeff is pleased: say "Now you have the
key."; move the key to the player.
```

Here's the source code for our example story.

```
"A-221" by Brendan Desilets

Use no scoring.

When play begins: say "Oh, no! You've
lost your red English binder. But here
comes your teacher. Perhaps he's seen it.
'Maybe,' he says. 'I just locked a binder
in the filing cabinet in Room A-221. See
if it's yours. You'll have to find the
key first, though. I'm not quite sure
where I left it.'
You find your way to A-221 to begin the
search."

A-221 is a room. "A-221 is a fairly drab
classroom, with twenty-four student desks
and a like number of computers. It sports
```

```
at least ten teacher-made signs about
grammar and literature and one long,
commercial poster. A filing cabinet is
the room's most prominent storage unit."

The Hallway is a room. It is east of A-
221. "This is an ordinary school hallway.
Room A-221 is to the west. You can't
think of any reason you would want to go
in any other direction right now."

The key is in A-221. The key unlocks the
filing cabinet. The description of the
key is "An ordinary brass key." Jeff
carries the key.

The filing cabinet is scenery in A-221.
It is a closed openable container. It is
locked and lockable. "This filing cabinet
is designed to store and organize all
sorts of papers, but it could hold lots
of other things, too."

The binder is in the filing cabinet. The
description is "The red English binder
that you lost recently. You've been
looking for it everywhere."

The coin is a thing. The player carries
```

the coin. The description of the coin is "A typical piece of currency-- worth something to most people."

Mood is a kind of value. The moods are unhappy and pleased. People have mood. The mood of Jeff is unhappy.

Jeff is man in A-221. The description of Jeff is "A sixth grader, wearing a baseball shirt. [if the key is carried by Jeff] He is carrying a key [end if]. Jeff looks [the mood of Jeff]."

Instead of attacking Jeff: say "You have been suspended from school for violent behavior."; end the story saying "And you have failed to recover your binder."

Instead of asking Jeff about “the key”: say “It's the key to the filing cabinet.”

Instead of asking Jeff about “key”: say “It's the key to the filing cabinet.”

Instead of giving the key to Jeff: say "Now Jeff has the key."; move the key to Jeff.

```
Instead of asking Jeff for the key when
Jeff is unhappy: say "Jeff refuses to
give the key, but points to the coin
you're carrying.";

Instead of asking Jeff for the key when
Jeff is pleased: say "Now you have the
key."; move the key to the player.

Instead of giving the coin to Jeff: move
the coin to Jeff; now Jeff is pleased;
say "Jeff looks very pleased."

The homework pass is a thing. The
description is "A special pass, signed by
your teacher, that allows you to skip a
homework assignment."

Instead of taking the binder: say "As you
take the binder, you find that, under it,
is a homework pass, made out to you.";
move the pass to the cabinet; move the
binder to the player.

An every turn rule: if the player is
carrying the pass,
end the story saying "Congratulations!
You've won!"
```

XIX. Creating Synonyms

A. In any kind of writing, it's important for the writer to be considerate of the reader. Since interactive fiction is a challenging form of writing and reading, it's especially vital for the writer to think about the reader's needs.

B. One way to help a reader is to make sure that he or she can use synonyms for the objects that we implement. Right now, as our story works, the reader can type "Give the coin to Jeff," and all will go well. However, if the reader types, "Give the coin to the boy," the story fails to recognize the word "boy."

C. To create a synonym for Jeff, we would add the following line to Jeff's description:

```
    Understand "boy" as Jeff.
```

Here's the final version of the story. Notice that we've add a couple of synonyms for the binder.

```
"A-221" by Brendan Desilets

Use no scoring.

When play begins: say "Oh, no! You've
lost your red English binder. But here
```

comes your teacher. Perhaps he's seen it. 'Maybe,' he says. 'I just locked a binder in the filing cabinet in Room A-221. See if it's yours. You'll have to find the key first, though. I'm not quite sure where I left it.'
You find your way to A-221 to begin the search."

A-221 is a room. "A-221 is a fairly drab classroom, with twenty-four student desks and a like number of computers. It sports at least ten teacher-made signs about grammar and literature and one long, commercial poster. A filing cabinet is the room's most prominent storage unit."

The Hallway is a room. It is east of A-221. "This is an ordinary school hallway. Room A-221 is to the west. You can't think of any reason you would want to go in any other direction right now."

Jeff carries the key. The key unlocks the filing cabinet. The description of the key is "An ordinary brass key."

The filing cabinet is scenery in A-221. It is a closed openable container. It is

```
locked and lockable. "This filing cabinet
is designed to store and organize all
sorts of papers, but it could hold lots
of other things, too."

The binder is in the filing cabinet. The
description is "The red English binder
that you lost recently. You've been
looking for it everywhere." Understand
"notebook" as the binder. Understand
"book" as the binder.

The coin is a thing. The player carries
the coin. The description of the coin is
"A typical piece of currency-- worth
something to most people."

Mood is a kind of value. The moods are
unhappy and pleased. People have mood.
The mood of Jeff is unhappy.

Jeff is man in A-221. The description of
Jeff is "A sixth grader, wearing a
baseball shirt. [if the key is carried by
Jeff] He is carrying a key. [end if] Jeff
looks [the mood of Jeff]." Understand
"boy" as Jeff.

Instead of attacking Jeff:
```

```
say "You have been suspended from school
for violent behavior.";
end the game saying "And you have failed
to recover your binder."

Instead of asking Jeff about “the key”:
say “It's the key to the filing cabinet.”

Instead of asking Jeff about “key”:
say “It's the key to the filing cabinet.”

Instead of giving the key to Jeff:
say "Now Jeff has the key.";
move the key to Jeff.

Instead of asking Jeff for the key when
Jeff is unhappy:
say "Jeff refuses to give the key, but
points to the coin you're
carrying.";

Instead of asking Jeff for the key when
Jeff is pleased:
say "Now you have the key.";
move the key to the player.

Instead of giving the coin to Jeff:
move the coin to Jeff;
now Jeff is pleased;
```

```
say "Jeff looks very pleased."

The homework pass is a thing. The
description is "A special pass, signed by
your teacher, that allows you to skip a
homework assignment."

Instead of taking the binder:
say "As you take the binder, you find
that, under it, is a homework pass, made
out to you.";
move the pass to the cabinet;
move the binder to the player.

An every turn rule:
if the player is carrying the pass,
end the story saying "Congratulations!
You've won.”
```

Chapter 28 – Getting Interactive Fiction

Fortunately, interactive fiction is inexpensive and fairly easy to find. Most IF stories that have been written since 1990 are free of charge and available via the Web, especially through the Interactive Fiction Database (http://ifdb.tads.org). The IF database also offers links for running many stories in a Web browser, and it presents a good set of instructions for getting the same stories running on your local computer.

Running stories on a local computer often works more smoothly than reading the same stories online, especially when it comes to saving your progress. Sometimes, saving where you've left off goes very well in an online reading, but often it doesn't. Also, reading stories online doesn't work at all for some stories, and it can leave out some elements, like pictures and sound.

So, getting the stories to run on local PC's and Macs and Linux computers is usually a good idea, but it's also a multi-step process.

To run an interactive fiction on a particular computer, you usually need two pieces of software. One piece of software is called an interpreter. The interpreter you need for a particular story depends mainly on two factors, the kind of computer you have and the tool the author used to create the story.

Interpreters

Each Internet-distributed story recommended on this site was created with one of five tools. These tools are called Inform, TADS, Hugo, Quest, and Adrift. Authors who write with Inform sometimes create stories in a format called z-code. Other Inform writers use an extension called Glulx, if they want to create very large stories or stories with multimedia elements. Interpreters for stories made with all of these tools are available for just about any kind of computer, even older, more unusual ones. If you have one Inform interpreter, one Glulx interpreter, one TADS interpreter, one Hugo interpreter, and one Adrift interpreter on your computer, you have enough interpreters to run almost all the stories recommended on this site.

Quest stories are a little different. They're often read online. Their only offline interpreter is the same Quest program that's used to write the stories. This program runs only on Windows. Beyond these five authoring systems, there are other IF-development systems, with their own interpreters, but, once you get the hang of using one interpreter, you'll probably be able to handle them all.

There's an excellent list of interpreters at http://www.ifwiki.org/index.php/Interpreter

You can simplify the interpreter-gathering process by using a tool that combines a number of interpreters into one program. The most popular of these is Gargoyle, which runs on Windows, Mac, and Linux. The Linux version is sometimes called “gargoyle-free.” Many Mac users like two other multi-interpreter programs, Zoom and Spatterlight. These multi-interpreter systems are fine for most users, though they make it a bit tricky to change font sizes and styles.

In addition, they will not easily run one of the highly-recommended stories in this book, *Arthur: the Quest for Excalibur.* To read *Arthur,* on Windows, you'll need an easily-installed interpreter called Windows Frotz. Frotz is also available for Linux and Mac, though the Mac version is very difficult to install successfully.

Fortunately, Mac users have another option. They can read Arthur with the Zoom interpreter. First, though, they have to change the name of the file called “ARTHUR.ZIP” to “ARTHUR.Z8.”

You can download interpreters from the IF Archive. The IF Database links to them, too.

Google Play and iTunes offer a few interpreters for IOS and Android.

Story Files

To run a story, you need, in addition to an interpreter, a story file. The story file you need depends only on which story you want to run – it doesn’t matter what kind of computer you have.

So, if, for example you want to read the TADS game *The One That Got Away,* using a Macintosh, you need the right story file and the right interpreter, which, in this case would be a TADS interpreter for the Macintosh, or a multi-interpreter program for the MAC. Gargoyle, Zoom, and Spatterlight will all work fine.

Where Can I Get These Things, Again?

You can find links to plenty of interpreters and story files at the Interactive Fiction Archive and the Interactive Diction Database. Downloading, at present, is usually just a matter of left-clicking or right-clicking on a link. Downloaded files are often compressed and/or archived.

Newer computer operating systems make decompressing files quite easy, by showing compressed files as folders that can be opened like other folders or extracted to show their contents.

If For Sale

A few contemporary works of IF are commercial products, always reasonably priced. Textfyre.com sells *Jack Toresal and the Secret Letter* and *The Shadow in the Cathedral.* Several works of IF are for sale at iTunes and Google Play.

Things get a bit trickier when you're looking for classic works from the 1980's. Most of the best works from this era were published by a company called Infocom. You can buy most of Infocom stories, (thirty-three, to be exact) for both PC and Macintosh computers, in one magnificent collection called *The Masterpieces of Infocom.* However, *Masterpieces* has become quite difficult to find in the last few years, and since it dates from the days of DOS, it doesn't always play well with modern computers. Fortunately, most of the story files from Infocom, the ones that have the extension .dat, will run on multi-format interpreters like Gargoyle and Spatterlight. The Infocom stories that have the .zip extension (*Arthur*, for example) will run on the z-code interpreter called “Frotz.”

The Lost
TREASURES
of Infocom
NEW CD ROM Version
20 CAPTIVATING GAMES
A Fortune in Interactive Mystery, Fantasy and Science Fiction Adventures!

Masterpieces, along with less comprehensive (and less pricey) collections like *The Lost Treasures of Infocom,* is often available at ebay and at Amazon.com. Just search ebay or Amazon, using “Infocom” as your search term. In early 2015, the lowest available price for *Masterpieces* seems to be $70.00 US. This sounds pretty expensive, but the collection includes more than thirty full-length stories. And some of them really are masterpieces!

Abandonware

It is also possible, though not really legal, to download all the Infocom stories from the Web, under the concept of abondonware. Those who offer the stories in this way often include disclaimers like this one, from Achim J. Latz:

“Copyright by Infocom, Inc. Provided for non-commercial use only, with the sole intent of making information available that would otherwise be lost.
To whomever presently holds the copyright to the information contained in this page: if you think the existence of this page violates your copyright, please complain to achim@latz.org and this page will be removed.”

One site that offers the Infocom stories in this way is called My Abandonware. This site urges its users to try to purchase the software it offers, before downloading it for free. Why would one do this? In addition to staying within the law, a person who downloads from My Abandonware will probably be able to run the story on a modern computer more easily.

Most abandonware sites are oriented toward Windows and DOS, but, in the case of Infocom stories, the story files will work on all kinds of computers. DOS programs, in general, can run on practically all modern computers, using the software emulator called DOSBox.

Raiders of the Lost Software

People who don't like obtaining software illegally have often mused, “Wouldn't it be great if a publisher like Activision, which now owns the Infocom stories, could just sell them for five dollars apiece? Acitivision, which now makes no money at all on the Infocom titles, would earn a few dollars, and readers would be happy, too.”

Yes, it would be great if Activision had a such a way to act on its own self-interest, but, in the current era of app stores and repositories, the software publishers aleady have an easy way to sell their back titles for a few dollars each.

Why don't they? Perhaps they're just a bit slow to react. By the time you read this, maybe you'll be able to buy *Zork I*, tricked out for your great new phone or laptop, for a sum of money that you'll never miss.

Notes and References

Chapter 1 – What is Interactive Fiction?

For discussions of the precise nature of interactive fiction, see *IF Theory Reader*, (2011) edited by Kevin Jackson-Mead and Robinson Wheeler (http://pdf.textfiles.com/books/iftheorybook.pdf) , which includes “Characterizing, If Not Defining Interactive Fiction,” by Andrew Plotkin. Of equal merit is *Twisty Little Passages* by Nick Montfort (2005, http://ebook.stepor.com/book/twisty-little-passages-an-approach-to-interactive-fiction-40473-pdf.html.)

In recent years, choice-based, hypertext stories, which do not include parsers, have been welcome in the most prominent interactive fiction competition, an event that ends in mid November. Some of the non-parser stories have done well, finishing as high as second. However, parser-based stories and hypertext narratives are really very different from one another. In 2014, a prominent IF writer, Carolyn VanEseltine, initiated “ParserComp,” for stories with parsers. This new “comp” was a great success, drawing some fine new games.

Chapter 2 – The Pain and Process of the Parser

In the 1980's, a company called Infocom published more than thirty commercial works of interactive fiction, some of them classics of the medium. The history of the company, in varying degrees of depth, is at http://web.mit.edu/6.933/www/Fall2000/infocom/. A supplement to the documentary film *Get Lamp* (2011) tells the Infocom story in an especially dramatic way. This supplement is not available on YouTube, but it can be purchased, as part of the DVD version of the documentary, at http://www.getlamp.com.

Emily Short has commented on the "false promise" of the IF parser, and on many other issues, theoretical and otherwise, in her remarkably insightful blog, "Emily Short's Interactive Storytelling" (https://emshort.wordpress.com/).

Jason MacIntosh's blog, which includes his videos, is "The Gameshelf" (http://gameshelf.jmac.org/essays/jmac-on-games/).

Andrew Plotkin makes his comments on the parser in the documentary film *Get Lamp,* directed by Jason Scott (2010, https://www.youtube.com/watch?v=LRhbcDzbGSU).

Chapter 3 – Interactive Fiction and Critical Thinking

As noted in the chapter's text, Robert Sternberg's *Intelligence Applied* informs much of this chapter. However, the chapter owes just as much to the work of

Robert Swartz of the National Center for Teaching Thinking (http://www.nctt.net/).

Chapter 4 – Interactive Fiction and the Reading Process

David F. Lancy and Bernard L. Hayes offer further thoughts on the motivational aspects of IF in "Interactive Fiction and the Reluctant Reader." The article appeared in the *English Journal*, in November of 1988.

Mark Engleberg writes about how IF can motivate students in “IF for Home-Schooled Students” at http://inform7.com/news/2009/03/16/mark-engelberg-on-if-for-homeschooled-students/.

Chapter 5 – Building Fluency With Interactive Fiction

Laura Robb offers a balanced discussion of the advantages of reading fluency in *Teaching Reading in Middle School: a Strategic Approach to Teaching Reading That Improves Comprehension and Thinking* (2000).

The *Report of the National Reading Panel* (2000) proved controversial in some of its claims and results, including the widespread use of time-consuming testing for fluency. Still, the report does offer worthwhile suggestions for building fluency.

Timothy Rasinski and Nancy Padak stress the role of public performance in building fluency in “Fluency Beyond the Primary.” The article appeared in *Voices from the Middle*, September 2005. Rasinski writes about coaching readers for fluency in his book, *The Fluent Reader: Oral Reading Strategies for Building Word Recognition, Fluency and Comprehension* (2003).

Nick Montfort describes the way readers of interactive fiction feel that they are, to some extent, crafting IF stories in his seminal book *Twisty Little Passages* (2003).

Chapter 6 – Creating Interactive Fiction With Adrift

Lively debates about the merits of the various IF authoring systems continue on the IF Forum and elsewhere. It’s clear, though, that, since 1990 or so, the best parser-based IF stories have come from the TADS and Inform communities. These languages are plainly the most mature and powerful tools we have. Adrift and Quest authors have produced some fine stories, too, however. *The PK Girl* by Robert Goodwin, Helen Travillion, Nanami Nekono, and Oya-G took sixth place in the fall 2002 Interactive Fiction Competition, thus announcing the Adrift had truly arrived. In 2014, a

Quest story did even better, as *Jacqueline, Jungle Queen!* by Steph Cherrywell finished third.

Since its arrival in 2006, Inform 7 has emerged as the choice of a great many, if not a majority of, serious parser jockeys. Inform 7 stories have won the fall IF competition in every year but one since it appeared on the scene. The exception, *Lost Pig,* was written in Inform 6. From 1995 to 2014, all but two of the winners of the fall IF Competition were written with Inform 6 or 7. The other two were TADS stories.

Adrift, the principal topic of this chapter is a truly admirable piece of programming. It's solidly coded (though not without bugs), and, in my experience, it really is the fastest way to get a decent IF story up and running. However, Adrift Developer is Windows-only, and that's a severe limitation for some of us. Adrift interpreters, on the other hand, are available for all the widely-used full-scale operating systems, so you can read Adrift stories on most computers.

The most popular authoring system for choice-based IF is called "Twine." Twine was developed by Chris Klimas in 2009, and it's available, free of charge, at http://twinery.org/.

Chapter 7 – Creating Interactive Fiction With Quest on the Web

In some ways, Quest fulfills the dreams of many teachers of interactive fiction. It’s easy to use, it boasts a good library of completed games, it can produce both choice-base and parser-based stories, it has a Web version, and it boasts a very active presence among educators, especially in the United Kingdom. Its developers and supporters, especially British educator Christian Still, deserve great credit for the leadership that they’ve provided.

Still, in my own experience, Quest does not quite measure up to Adrift in its maturity and stability.

Chapter 8 – Inform 7 and the Writing Process

Seymour Papert’s *Mindstorms* is probably the most important book ever on computers in education. Though its advocacy of the Logo programming language may seem naïve in the twenty-first century, its argument for a computer-rich environment and what to do with it are entirely contemporary. Consider, for instance, the endless debate about whether or not children should be taught to program computers. Papert answers this question in the most rational way possible, by looking at the precise benefits of programming, not by advocating or rejecting some particular style of coding.

My own article, “Logo and Extended Definition” (*Journal of Teaching Writing,* Volume 5, Number 1, 1986) applies Papert’s ideas to a typical priority of secondary-school English teachers.

Graham Nelson’s announcement of the Inform 7 public beta changed the world, or at least the small part of the world that teaches children and young adults to write IF. See “Inform 7: Interactive Fiction from Natural Language" (http://groups.google.com/forum/#!topic/rec.arts.int-fiction/9ZGc8bSbraw).

Aaron Reed stands as one of the most important of all IF writers, known especially for his massive and masterful story *Blue Lacuna*. He’s also the author of *Creating Interactive Fiction With Inform 7* (2011), which offers a clear and thorough tutorial on the Inform language. Reed’s book is a nearly perfect complement to the hundreds of pages of clear and literate document that come with Inform 7 (http://inform7.com) itself.

Gareth Rees is another important writer of IF stories, who has also produced “must read” essays about writing interactive fiction. He’s the author of the IF classic *Christminister* (1995), and of "Game Design and Game Analysis” (1995, http://www.doggysoft.co.uk/inform/write/desgn.html).

Chapter 9 – Why Inform 7?

Among the most prominent computer languages for children is Scratch, which, like Logo, has its roots at the Massachusetts Institute of Technology. Linda Sandvik describes her work with Scratch and other tools on the Ubuntu UK Podcast in an “Interview With Linda Sandvik of Code Club” (June 27, 2013, http://podcast.ubuntu-uk.org/2013/06/27/s06e18-a-midsummer-nights-ubuntu/).

Chapter 10 – The Writer’s Self in Interactive Fiction

Philippa Foot (innocently enough, I suspect) invented the “Trolley Problem” franchise in an essay that isn’t mostly about trolley problems at all. The article’s called "The Problem of Abortion and the Doctrine of the Double Effect," and you can find it in *Ethical Theory: An Anthology,* (2007) edited by Russ Shafer-Landau. Judith Jarvis-Thompson focused on and elaborated the trolley theme in several insightful essays, including "The Trolley Problem" in *The Yale Law Journal* 94.6 (1985). Numerous subsequent essays have tried to develop the trolley trope further, often twisting it almost beyond recognition.

Roger Giner-Sorolla identfies the problem of confusing the reader with the player character in his seminal essay, “Crimes Against Mimesis,” (1997) which originally appeared, in installments, in the USENET newsgroup rec.arts.int-fiction. The essay also appears in the *IF Theory Reader,* edited by Kevin Jackson-Mead and J. Robinson Wheeler.

Graham Nelson’s “Bill of Players’ Rights” is part of “The Craft of Adventure” (1997). It’s available at http://www.ifarchive.org/if-archive/programming/general-discussion/Craft.Of.Adventure.txt.

Andrew Plotkin comments on common misunderstandings of the IF parser in the documentary film, *Get Lamp*, directed by Jason Scott. Jason’s interview with Andrew is at http://archive.org/details/getlamp-aplotkin.

Chapter 11 – Recommended Works of Interactive Fiction

The “Top Seventy” that appears in this chapter includes some stories that are not good choices for younger students to try on their own. See the comments on each story for details.

Most of the stories recommended here can be easily obtained, free of charge, from the Interactive Fiction Database, at http://ifdb.tads.org. Just use the site's search bar to find a story and the database will offer a link for downloading, along with instructions for getting the story to run on your computer. In some cases, the database will also offer a link for trying the story online. See Chapter 17 of this book for information on acquiring stories that are harder to get.

Chapter 12 – An Interactive Classic from the Commercial Era: *Arthur, the Quest for Excalibur*

Arthur, (1989) is the last of the classic Infocom stories. It was published after Infocom's ill-fated acquisition by Activision in 1986. The author of the story, Bob Bates, along with Mike Verdu, founded Legend Entertainment in 1989. Legend issued several interactive fictions in the Infocom tradition, some written by Infocom veteran Steve Meretsky and others by Bates himself.

Chapter 13 – An Interactive Classic from the Modern Era: *The Firebird*

The fall Interactive Fiction Competition plays such an important role in the fostering of IF writing that a casual observer might think that the IF community must be extraordinarily competitive. In truth, however, the

“Comp” is less about winning than about finding an audience for new interactive fictions. Still, lots of IF classics have won the competition. Among the best-remembered and most-appreciated winners are *A Change in the Weather* (Andrew Plotkin, 1995), *Photopia* (Adam Cadre, 1998), two episodes of the “Earth and Sky” trilogy (Paul O’Brian, 2002, 2004), *Floatpoint* (Emily Short, 2006), *Lost Pig* (Admiral Jota, 2007), *Violet* (Jeremy Freese, 2008), and *Coloratura* (Lynnea Glasser, 2013).

But plenty of outstanding stories have entered the Comp and not quite won. These include David Dyte’s *A Bear’s Night Out*, the original *Earth and Sky* story, Ryan Veeder and Emily Boegheim’s *Robin & Orchid*, Mark and Renee Choba’s *History Repeating*, Jason MacIntosh’s *The Warbler’s Nest*, Leon Lin’s *The One That Got Away*, and Nate Cull’s *Glowgrass.*

The “Fall Comp” and the other IF competitions are so important that members of the IF community sometimes seem surprised when a story like *The Firebird* comes along, a really fine piece of work that might well have won a competition but wasn’t entered. That surprise can often be a pleasant one, though, when the story in question turns out to be as good as *The Firebird*, or *Spider and Web*, or *Blue Lacuna*.

Chapter 14 – An Interactive Fiction Competition Winner:

Winter Wonderland

Winter Wonderland by Laura Knauth won the fall IF Comp the year after its most revolutionary winner, *Photopia,* dominated the standings. Knauth's work was a more quiet winner, but its reputation has developed over the years. *Wonderland*'s gently playful spirit and its gracious characterizations grow on many readers, and its technically flawless programming makes possible a wide variety of puzzles. Laura even manages two very enjoyable mazes, at a time when mazes were very much out of style. In my experience, students always love the story.

Chapter 15 – An Interactive Tragedy: *Photopia*

What, exactly, is appropriate for kids and classrooms? It's hard to say. For a middle school teacher with my own particular style, it was always important not to underestimate what students could handle.

Photopia was obviously not written for an audience of children. In its original version, it began with a scene laced with rather strong profanity, and the story is extremely sad. Still, with sensitive teaching and a warning about upcoming episodes from time to time, *Photopia* works very well with twelve-year olds.

Emily Short's *Bronze* presents an equally tough case. This retelling of "Beauty and the Beast" plays directly to adult interests. It refers to sex in a direct, though always tasteful, way, and one of its climactic moments reveals the suicide of one of the Beast's victims, a young woman named Yvette who was pregnant with the Beast's child.

Still, the story offers a dark fairy tale that students can appreciate, though some specific sexual references and Yvette's horrific suicide would not work with children. As a result, I asked the author for permission to craft a slightly bowdlerized version of the story. She agreed, and the PG version of *Bronze* has delighted and angered many middle school and high school students. The edited version of the story is at http://bdesilets.com/if/Bronze.z8.

Chapter 16 – An Interactive Fiction about Middle School Students: *The Enterprise Incidents*

The Enterprise Incidents, though explicitly a "fantasy" has its roots in a real middle school and in a real middle school program for undermotivated eighth graders. The program was called "Venture," and, in it, a dozen or so students ran a real business, and went home with paychecks once per quarter. Venture ran from 1985 to 2004, and its website is still available at http://venture.home.comcast.net.

The bizarre fashion choices of the story’s “Megan” character are also real, though they’re harder to believe.

Chapter 17 – Acquiring Interactive Fiction

I should probably stress, more than I already have, that interactive fiction is inexpensive, to a degree that is practically ridiculous. Most of the best stories are flat-out free, and others, like *The Shadow in the Cathedral* cost $5.00 (U.S.) or less. Even the Infocom stories are free, through several, not-quite-legal abandonware sites. A perfectly-legal purchase of *Masterpieces of Infocom*, via Amazon.com, costs around $70.00, which amounts to about $2.13 per story.

About the Author

Brendan Desilets has taught in Massachusetts schools since 1968, at the middle school, high school, and university levels. He currently works as an adjunct professor of English at the University of Massachusetts at Lowell. Brendan’s resume is available at http://jgms.home.comcast.net/~jgms/resume.htm.