



chris crawford
on interactive storytelling

SECOND EDITION

New
Riders

NRG

chris crawford

on interactive storytelling

SECOND EDITION



New
Riders

Chris Crawford on Interactive Storytelling, Second Edition

Chris Crawford

New Riders

www.newriders.com

To report errors, please send a note to errata@peachpit.com

New Riders is an imprint of Peachpit, a division of Pearson Education.

Copyright © 2013 Chris Crawford

Senior Editor: Karyn Johnson

Development Editor: Robyn Thomas

Production Editor: Cory Borman

Technical Editor: Michael F. Lynch, Ph.D.

Copyeditor: Kelly Kordes Anton

Proofreader: Scout Festa

Composition: Danielle Foster

Indexer: Jack Lewis

Cover design: Aren Howell

Notice of Rights

All rights reserved. No part of this book may be reproduced or transmitted in any form by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the publisher. For information on getting permission for reprints and excerpts, contact permissions@peachpit.com.

Notice of Liability

The information in this book is distributed on an “As Is” basis, without warranty. While every precaution has been taken in the preparation of the book, neither the author nor Peachpit shall have any liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the instructions contained in this book or by the computer software and hardware products described in it.

Trademarks

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and Peachpit was aware of a trademark claim, the designations appear as requested by the owner of the trademark. All other product names and services identified throughout this book are used in editorial fashion only and for the benefit of such companies with no intention of infringement of the trademark. No such use, or the use of any trade name, is intended to convey endorsement or other affiliation with this book.

ISBN-13: 978-0-321-86497-0

ISBN-10: 0-321-86497-2

9 8 7 6 5 4 3 2 1

*I dedicate this book to the Storytron development team:
Louis Dargin, Facundo Dominguez, and Laura Mixon,
who pushed the technology of interactive storytelling to new heights.*

Bio

Chris Crawford earned a Master of Science degree in Physics from the University of Missouri in 1975. After teaching physics for several years, he joined Atari as a game designer in 1979. There he created a number of games: *Energy Czar*, an educational simulation about the energy crisis; *Scram*, a nuclear power plant simulation; *Eastern Front (1941)*, a wargame; *Gossip*, a social interaction game; and *Excalibur*, an Arthurian game.

Following the collapse of Atari in 1984, Crawford took up the Macintosh. He created *Balance of Power*, a game about diplomacy; *Patton Versus Rommel*, a wargame; *Trust & Betrayal*, a social interaction game; *Balance of the Planet*, an environmental simulation game; and *Patton Strikes Back*, a wargame. In 1992, Crawford decided to leave game design and concentrate his energies on interactive storytelling, a field that he believed would become important. He created a major technology for interactive storytelling systems, patenting it in 1997. He is now commercializing his technology at his company website at storytron.com.

Crawford has written five published books: *The Art of Computer Game Design*, now recognized as a classic in the field, in 1982; *Balance of Power* in 1986; *The Art of Interactive Design* in 2002; *Chris Crawford on Game Design* in 2003; and *Chris Crawford on Interactive Storytelling* in 2004.

He created the first periodical on game design, the *Journal of Computer Game Design*, in 1987. He founded and served as Chairman of the Computer Game Developers' Conference, now known as the Game Developers' Conference.

Crawford has given hundreds of lectures at conferences and universities around the world, and published dozens of magazine articles and academic papers.

Crawford served as computer system designer and observer for the 1999 and 2002 NASA Leonid MAC airborne missions; he also has done some analysis of the resulting data. He lives in southern Oregon with his wife, 3 dogs, 7 cats, and 16 ducks. You can read more about him on his website at www.erasmatazz.com.

Table of Contents

	Introduction	xi
PART I	The Basics	1
Chapter 1	Storytelling	3
	The History of Storytelling	4
	Two Mechanisms of Thinking	6
	The Tyranny of the Visual	22
	Spatial Thinking	24
	Temporal Discontinuity	26
	Conclusions	26
Chapter 2	Interactivity	27
	My Definition of Interactivity	28
	Second-Person Insight	30
	A Model for Human Understanding	32
	Ego Control	35
	Degrees of Interactivity	37
	So What?	42
	Conclusions	44
Chapter 3	Interactive Storytelling	45
	Extrapolation from Games	47
	Interactivized Movies	50
	Plot vs. Interactivity	50
	Interactive Storytelling Is Unique	54

	Individuation	61
	What Can't Be Part of Interactive Storytelling?	62
	Conclusions	64
PART II	Mentalities.	65
Chapter 4	Abstraction.	67
	Justice	68
	Science	70
	Finance	71
	The Moral of These Stories	74
	Playing God	76
	Translating Principle into Instance	78
	Conclusions	80
Chapter 5	Verb Thinking	81
	Verb vs. Noun	82
	Case in Point: Multimedia	88
	Getting Started with Verb Thinking	90
	Conclusions	92
Chapter 6	The Agony of Math	93
	Embracing Mathematics.	94
	Causal Relationships	96
	Modeling as a Metaphor	98
	Conclusions	102
Chapter 7	Two Cultures, No Hits, No Runs	103
	Neurophysiological Basis	105
	History of Games and Stories	106
	Exhortations and Prognostications	108
	Conclusions	111

PART III	Evolutionary Strategies	113
Chapter 8	Branching Trees	115
	Overall Game Structure	118
	The Solution: State Variables	123
	Conclusions	126
Chapter 9	Interactive Fiction	127
	Text Adventures	128
	Graphic Adventures	128
	IF: Son of Text Adventure	129
	Puzzles	130
	A Real Example	131
	Under the Hood	133
	Is It Interactive Storytelling?	138
	The Future of IF	140
	Conclusion	140
Chapter 10	Role-Playing Games	141
	The Verbs	142
	Little Room for Creativity	143
	The Dangers of Overspecialization: Koalas vs. Goats	145
	What About MMORPGs?	147
	The Audience	149
	Conclusions	150
Chapter 11	Video Games	151
	Constipated Stories	152
	Multiple Endings	153
	Branching Trees	154
	Open-Ended Stories	155
	Fully Player-Driven Stories	156
	Cost vs. Storytelling	156
	Conclusions	158

PART IV	Component Technologies	159
Chapter 12	Modeling	161
	The Medium	162
	Basics	163
	Adding More Complexity	166
	Abstracting the Process	169
	Fine Points	172
	Conclusions	175
Chapter 13	Bounded Numbers	177
	Overruns	180
	Bounded Numbers	181
	Bell Curves and People	183
	Technical Details	184
	The Blend Operator	185
	Boolean Calculations	189
	Conclusions	192
Chapter 14	Personality Models	193
	Complete	194
	Concise	194
	Orthogonal	194
	Behaviorally Discriminating	196
	Overspecific	197
	Types of Personality Traits	198
	Expanding on the Three Core Personality Traits	203
	Physical Traits	204
	Moods	204
	Inclination Formulae	206
	Two Exercises	208
	Conclusions	208

Chapter 15	Fate Makes the Storyworld Go 'Round	209
	Listen	210
	Think	213
	Speak.	214
	Scoring	223
	Conclusion.	226
Chapter 16	Verbs and Events	227
	Verb Counts.	228
	Generalized Verb Handling	229
	Sentence Structure.	231
	Plans and Events	232
	HistoryBooks	233
	Conclusions	244
Chapter 17	Languages	245
	Using Devices to Talk	246
	The Sapir-Whorf Hypothesis	247
	Faking Real Language	248
	Creating a New Language.	258
	Conclusions	267
Chapter 18	Scripting Languages	269
	Desiderata	270
	Sample Script for Text Generation	283
	Conclusions	286
Chapter 19	Engines	287
	Event-Based Engines.	288
	Clock-Based Engines.	288
	Plot-Based Engines	295
	Conclusions	296

Chapter 20	High-Level Structures	297
	Plot Points	298
	Goals	300
	Anticipation	301
	Conclusions	332
PART V	Wrapping Up	309
Chapter 21	Academic Research	311
	Comme il Faut	315
	Prom Week	316
	Release the Kraken!	316
	The IRIS Project	318
	Conclusions	318
Chapter 22	The Future	319
	It Will Be Big	320
	Negative Predictions	321
	Positive Predictions	322
	Conclusions	325
	For Further Reading	327
	Books	327
	Conference Proceedings	330
	Internet	330
	Index	332

Introduction

Interactive storytelling is a young field; first conceived in the late 1980s and first experimented with in the 1990s, it did not attract substantial attention until the new millennium. By 2010 it had become a hot topic. The field's youth has endowed it with a diversity of ideas and a cacophony of discussions. Like the proverbial elephant, everybody perceives it from his or her own vantage point. Moviemakers see it as a form of cinema, video-game people claim it as an extension of their own field, computer scientists think of it as part of the broader field of artificial intelligence, and experts in the art of improv consider it to be the computerization of their skills.

The truth is that interactive storytelling is not an extension of or variation on any of these fields; it must be approached as something new and unique. An appreciation of all these fields (and several more) certainly helps you appreciate the challenges of interactive storytelling, but to use one of these fields as a platform from which to launch your thinking is a grievous mistake. There is no launching point; you must hurl yourself into the raging waters and kick, paddle, and swim through as best you can.

Many fields of study are based on combinations of other, more fundamental fields. A good moviemaker must understand storytelling as well as optics, the human visual system, and camera technology. A good game designer must understand programming, games, and user interface issues. Interactive storytelling is no different; indeed, it stands on the shoulders of many fields, including games, cinema, storytelling, programming, and mathematics. Because interactive storytelling is a new and poorly understood field, this book approaches it from many different angles. It also devotes space to negative viewpoints; you can better understand what interactive storytelling *is* if you appreciate what it *isn't*.

I wrote the first edition of this book in 2004; in the intervening years, everything has changed except the underlying principles. This edition of the book explains those principles with the greater clarity I have learned from eight years of work on interactive storytelling. This second edition also presents more technical details that will surely be superseded within 20 years, by which time I will have prepared a third edition.

A cautionary note: My writing style is unconventional and might set you aback. (Here I express my gratitude to my editor, Robyn Thomas, for reining in my worst excesses, straightening out my tangled prose, and saving me from my misanthropic tendencies.) My objective in writing is not to impress you with my erudition, but to help you understand a difficult subject. I will goad you to think about the material by presenting ideas in their strongest, starkest terms. I will not waste your time with qualifiers and mush-mouthing nothingburger pap. You're smart enough to fill in the exceptions for yourself. If you find yourself in agreement with everything in this book, you haven't been reading closely enough.

I'd rather serve up chili than oatmeal.

My Interactive Storytelling Journey

The starting point for my interactive-storytelling quest came way back in 1983 with the realization that when games finally matured as a medium of artistic expression, they would have to be about people, not things. All other artistic and entertainment media are fundamentally about people, but games were unique in their obsession with things. In games, you chase things, things chase you; you shoot things, and things shoot at you. You search for things, acquire things, navigate things, move things, destroy things—it's always things, things, things and never any dramatically significant interactions with people in games. That, I decided was the fundamental difference between games and other media, and that was the central problem I had to tackle.

What would a game about people look like? I came up with a simple idea for a game based on interpersonal interactions. It was called *Gossip*, and all the player did in the game was talk to people on the telephone about who likes or dislikes whom. There were eight characters on the screen; the player was the one in the upper left (**Figure 1**).

The basic concept of the game was to call up other people and gossip about who feels what about whom. By carefully calibrating your statements to the preferences of your listener, you could build up your popularity.

Sadly, *Gossip* came out not long before Atari collapsed, so it was a commercial failure.



FIGURE 1 *Gossip*, my early game about people

I made my next attempt in 1987 with the design of *Trust & Betrayal*, a strange game that I still consider to be the finest, most creative game I ever designed. By then, I had identified the central problem for games about people to be the difficulty of language. You really can't have much of an interaction with other people if you can't talk to them, so I decided that I had to come up with a language system (Figure 2).

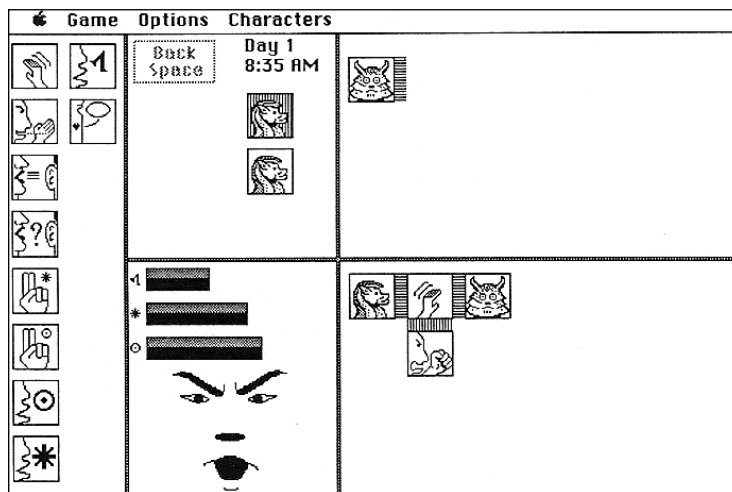


FIGURE 2 *Trust & Betrayal*, my self-professed finest and most creative game

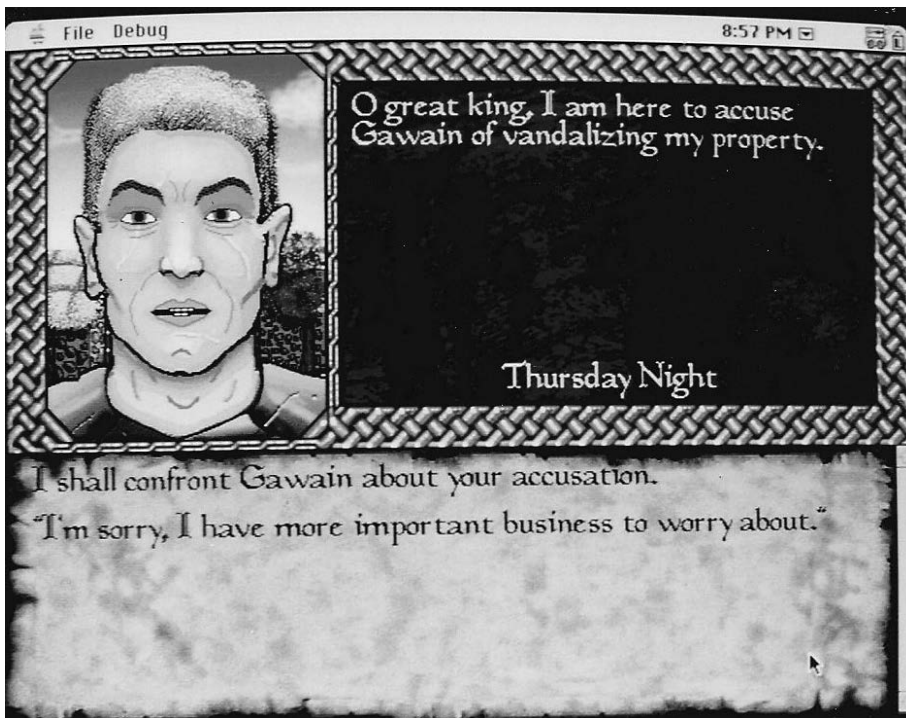


FIGURE 3 *Le Morte D'Arthur*, a game that included a graphics system for presenting individual character faces that could have different emotional expressions

This game's ploy is that it takes place on an alien world where different intelligent species communicate with each other by means of a telepathic language. I created a simple language so I could represent it with a set of 87 icons. This game also saw my first use of an inverse parser, a personality model, the emphasis on verbs, face displays with emotional expressions, and interstitial stories. It, too, was a commercial failure.

But I wasn't about to give up. In 1992, I began working on *Le Morte D'Arthur* (Figure 3), a game about the Arthurian legends. For this game, I developed my first general-purpose interactive storytelling engine. I also came up with a graphics system for presenting individual character faces that could have different emotional expressions.

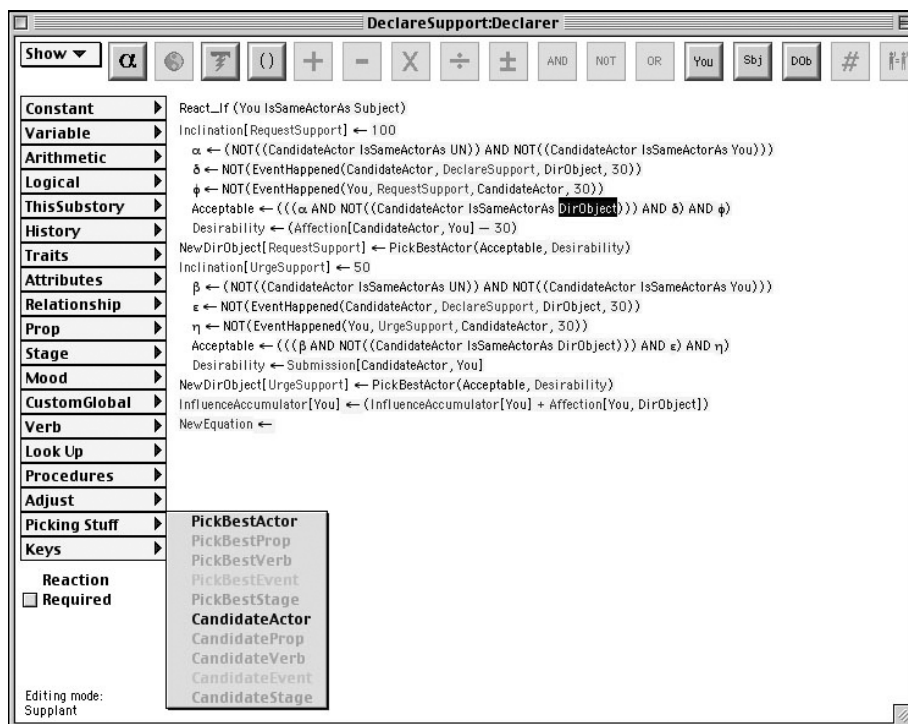


FIGURE 4 Erasmatron, a system for computer users to create their own storyworlds

Sadly, I couldn't get funding to complete the game.

A few years later, I got funding to develop an interactive storytelling system with which typical computer users could build their own storyworlds. I got carried away and built a huge system, called Erasmatron, that included a scripting language, a development editor, a powerful storytelling engine with advanced linguistic capabilities, and even an editor allowing the user to design unique faces for each character (Figure 4).

There was only one hitch: The system was far too complicated for anybody to use, much less a typical user. We built some interesting storyworlds with it, but they were incomplete and just weren't entertaining. Erasmatron flopped.

In 2007, I recruited Laura Mixon, Louis Dargin, and Facundo Dominguez to build the next generation of my technology, dubbed Storytron (Figure 5). The Storytron technology was a huge leap forward and it pioneered many of the technologies presented in this book. Indeed, the stuff we had working was more powerful than what I describe here.

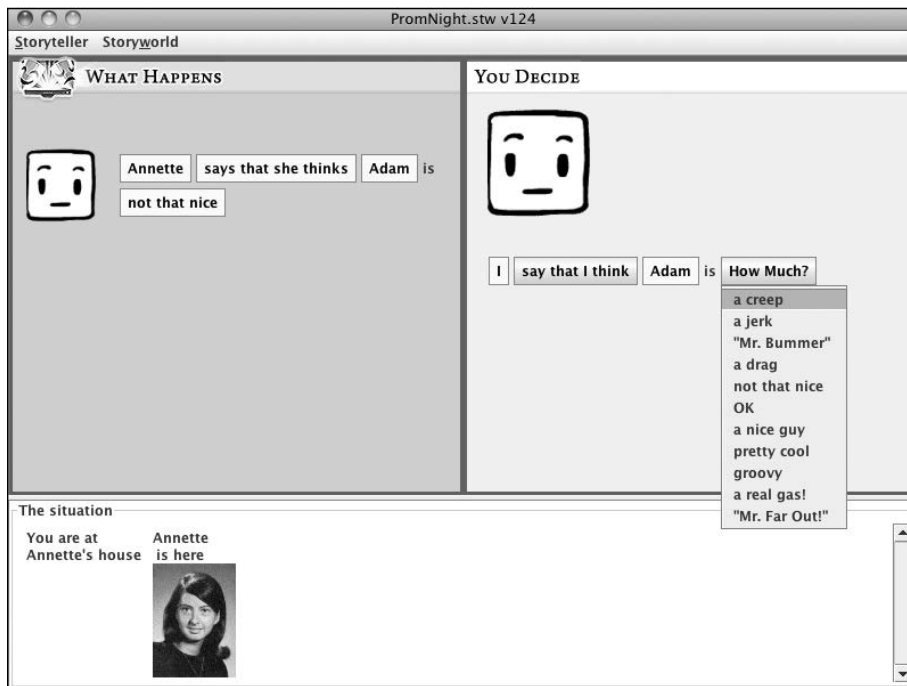


FIGURE 5 A game created in Storytron

The problem with Storytron was the same problem that killed Erasmatron—it was too complicated. Despite an avid following of determined storyworld authors, nobody was able to build a complete, working, entertaining storyworld. I built one, *Balance of Power 21st Century*. It stank. We gave up on Storytron in 2010.

Since then, I've continued thinking about interactive storytelling, and have carefully digested the many mistakes I made with previous efforts. I am resuming work on the project, with some major new design concepts. With so many failures behind me, how can I go wrong?

PART I

The Basics

BEFORE YOU CAN UNDERSTAND interactive storytelling, you need to get a good understanding of two things: *interactivity* and *storytelling*. Then you need to get a grip on what happens when you combine them.

Chapter 1: Storytelling is ancient, much written about, and still not fully understood. I can't add anything to the massive literature about storytelling as art, but in this chapter I offer a unique angle of approach: *neurophysiology*.

Chapter 2: Here we are 30 years into the personal computer revolution and people still don't understand interactivity. Think of it as a conversation between the user and the computer. Tattoo Crawford's First Law of Software Design onto your navel.

Chapter 3: When you put interactivity and storytelling together, you get interactive storytelling, a completely new medium of expression. It's not cinema with interactivity superglued on; it's not video games with story superglued on; it's a brand spanking new and unique medium.

This page intentionally left blank

CHAPTER 1

Storytelling

OUR ICONIC IMAGE OF STORYTELLING is a circle of young people around a campfire with an old man regaling them with a story. This image evokes warm and cozy feelings of pleasure and camaraderie. However, humankind did not actually develop storytelling as a leisure activity to while away the hours—storytelling evolved for serious purposes, as a necessary component in the development of human culture.

Imagine two kittens at play. One stalks the other, sneaking up from behind; he pounces and the two kittens roll over each other, batting and kicking. They're so cute that we can't help but laugh and envy the fun they must be having. It's true that the kittens are having fun, but they're not playing like this to pass the time in the months before cat adulthood. This is serious business; they are learning the skills they'll need to survive in the real world. Stalking, pouncing, grappling, clawing, and biting: These are all skills crucial in a cat's life. Do you think it's easy catching mice? I certainly couldn't. Consider this: Your best weapons (your fangs) are only an inch away from the most vulnerable part of your anatomy (your eyes). Should that mouse, in his struggles, succeed in clawing at your eye, you'd be blind in one eye and your hunting—and eating—days would be over. So how do you close in for the kill without risking your eyes?

This is a tricky problem and it requires a lot of practice to get it down pat. In fact, every part of the hunt is tricky and requires lots of practice. That’s precisely what kittens do for months on end: They stalk, they pounce, they grapple, they claw, they kick, and they bite, alternating between roles as predator and prey. What we call “kitten play” is first and foremost “cat education.” The fact that it’s obviously fun is not a coincidence; fun is the emotional driver behind true learning.

Storytelling serves people in a similar way. It’s not a waste of time; it’s serious education about the complexities of social interaction. We have other forms of play, of course, but storytelling is one of the most important forms of play. In fact, storytelling is so important that we continue to enjoy it throughout our lives. Without storytelling, humans never could have communicated complex information and, therefore, could not have developed culture. Storytelling isn’t merely characteristic of the human culture—it’s absolutely necessary to the existence of human culture.

The History of Storytelling

Storytelling was a natural, almost inevitable consequence of human evolution. The human brain developed in response to the environmental pressures facing early hominids. Each major problem triggered some sort of change in the human body. If a problem could be solved by mental effort, an existing part of the brain took over that problem, often expanding as a consequence. Most of the time, these new mental abilities were cobbled together from a combination of existing parts of the brain, so they weren’t often localized to one region of the brain. As they developed, these *mental modules* were recruited to solve new but related problems.

Because mental modules aren’t localized tightly, scientists can’t pin them down with anatomical precision. Brain function is so complex that one mental module blends into the next. This makes the job of identifying mental modules a subjective task; every scientist slices the pie a little differently. Nevertheless, some slices seem to be more popular than others. Following are four basic mental modules: visual-spatial, social reasoning, environmental knowledge, and language.

- ▶ **Visual-spatial** handles visual perception and spatial imagination. We use our visual-spatial modules to figure the distance to an object, determine its shape (from its texture), and navigate through the environment. This is the

only well-localized mental module; it's concentrated in an area at the rear of the brain, which is why you see stars when you are struck on the back of your head.

- ▶ **Social reasoning** allows us to anticipate the likely behavior of others. Here's an example of social reasoning actually observed in chimpanzees: A troop is out foraging, each one searching a different patch of ground. One chimp spies a tasty piece of fruit. He realizes that the dominant chimp nearby will take it away if he sees it, so the discoverer acts as if he's seen nothing. Five minutes later, when the others have moved on and the coast is clear, the discoverer sneaks back to the fruit and starts to eat it. At this point, the dominant chimp, who had noticed the first chimp's odd behavior and anticipated this development, leaps out from behind a tree and robs the first chimp of his fruit. Consider all the computations that went into the thinking of both chimps. Those computations are what we mean by social reasoning.
- ▶ **Environmental knowledge** contains our memories of how things work in the real world. Don't think of it as a computer database, however, as it contains no numbers. Instead, it takes the form of long lists of if-then statements.
 - ▶ If the animal's track is sharply defined in soft dirt, then the animal passed here recently.
 - ▶ If the animal's track is sharply defined in dried mud, then it passed here some time ago. Even what we think of as recalled images and sounds are actually constructs built of if-then facts.
 - ▶ If the antelope's horns are straight, smooth, and parallel, then it's a steenbok.
 - ▶ If the antelope's horns have a spiral on them, then it's an eland.

There are thousands of these relationships tucked away in our brains.

- ▶ **Language** permits complex communication with others. But language has dramatically changed the way our minds work. In the days before language, things were simple. Any problem we encountered was routed to the appropriate mental module, which then solved the problem as best it could and directed our behavioral response. But language had to send its tentacles

into every mental module; after all, you can't talk about what you're seeing if your language module doesn't have access to the visual-spatial module. Thus, language links all the modules together. That linkage triggered a quantum leap in human thinking: Instead of a simple one-to-one relationship between a problem and its appropriate mental module, now *any* mental module could put its two cents' worth into the solution. This triggered an explosion of human creativity. It also rendered us less decisive. Nowadays, we can't be certain the first solution that pops into our minds is the best; a better one might emerge from another module in a moment. So we hesitate.

Two Mechanisms of Thinking

There are two profoundly different mechanisms of thinking. The dominant mechanism is pattern based. This is how nervous systems originally developed. Pattern-based thinking is fast and powerful: A pattern of inputs enters the brain; the brain processes the new pattern with its own patterns, producing a pattern of responses. For example, a small animal looks upon a scene and sees a pattern that includes large teeth, eyes, and big claws. The small animal's brain processes that pattern and almost instantly determines an output pattern: It activates different muscles in its body to turn and run.

This pattern-based thinking has a crucial benefit: No matter how vague or indeterminate the input pattern, brains can interpret that pattern in some fashion. We look at clouds and see faces or animals; dark shapes in the night seem like monsters and ghouls. Faint noises in the wind sound like voices.

However, about one or two hundred million years ago, a new mechanism developed: sequential thinking. Pattern-based thinking is instantaneous; it cannot fathom the significance of sequences of events. For example, let's imagine two early rat-like mammals rummaging around on the late Jurassic forest floors of 150 million years ago. One of our ancient rats has a brain with only pattern-based processing. The second rat-mammal, however, is equipped with the latest neural technology: sequential processing. Now, suppose a big nasty dinosaur comes their way, looking for a nice mammal burger. They both hear its footfalls and scurry to hide. The dinosaur looks around, sees nothing, and clomps away. The old-style pattern-processing rat hears heavy footfalls and thinks, "Ack! Heavy

footfalls! Danger!” So, he stays in hiding. But the rat with the hot new sequential processing recognizes that the sequence of footfalls shows them growing progressively weaker. He thinks, “Weakening footfalls. He’s going away,” and emerges from hiding to resume his search for food. The second rat literally eats the first rat’s lunch. Guess which rat species goes extinct?

There’s a nasty catch here: Neurons in the brain aren’t designed to handle sequential reasoning. It takes quite a kludge to get them to carry out sequential tasks and demands a lot of neurons. The advantages of sequential thinking, however, are so great that many animals, most notably mammals, spent millions of years developing ever larger and more complex brains, with most of that expansion handling the expensive sequential processing mechanism. Our monstrously large brains evolved to handle the extremely complex sequential processing in which we humans specialize.

The older visual-spatial and social reasoning mental modules rely entirely on pattern-based thinking. The environmental knowledge module uses a mixture of pattern-based thinking and sequential thinking. But the language module is almost entirely sequential in structure. Think about it: A word is a sequence of sounds, and a sentence is a sequence of words, and this book is a sequence of sentences. Language uses a lot of sequential processing.

Mental Modules Interact

Once the development of language had pulled the mental modules into communication, all sorts of fascinating interactions began between the modules. For example, the environmental knowledge module and the social reasoning module interacted in a surprising fashion. The environmental knowledge module impelled humans to inquire into the causes of phenomena they observed in their environment, trying to find a simple if-then rule to store in memory. But the world is a complicated place, and all too often a clear cause was wanting. For example, an especially important question for the early farmers was, “Why does it rain?” Or, more to the point, “Why does it sometimes fail to rain?”

When language put the environmental knowledge module in touch with the social reasoning module, these two modules interacted to devise an answer that made sense: Powerful people—gods—caused natural phenomena. Whenever a

phenomenon lacked an obvious cause, assigning the phenomenon to a god and then explaining the apparently erratic behavior with the deity's mood swings was a simple matter. Not only did the social reasoning module suggest an explanation to the problem, but it also offered the solution: propitiate the god. Every time a new problem cropped up, a priest revealed another new god who controlled that phenomenon. Thus, most ancient societies had a large pantheon of gods, each with his or her own particular physical jurisdiction. In ancient Greece, for example, you made a sacrifice to Poseidon before undertaking a sea voyage; since he ruled the waves, his goodwill was nice to have. Going to war? Better be in good graces with Ares. Problems with your love life? Aphrodite is the one to curry favor with. And so on.

Because the gods seemed so arbitrary, it behooved society to have someone on hand who could communicate with the gods. There was never any shortage of applicants for the position; whoever communicated with the gods was, essentially, in charge of the society.

But the environmental knowledge module also interacted with the language module, producing “sequential thinking environmental knowledge,” also known as *science*. Sequential thinking at this level took a long time to develop, but once the ideas of logic and mathematics took hold, science took off. The conflict between science and religion becomes clearer when viewed this way. Both arose from the environmental knowledge module; both attempt to explain the world in which we live. Religion takes the social reasoning route, and science takes the sequential logic route. The choice between them might have more to do with the relative strengths of these two modules inside a person's mind than anything else.

There are four more interactions between modules to consider. I lump two of them together: The combination of the visual-spatial module with the environmental knowledge and social reasoning modules to produce *representational art*.

When visual-spatial reasoning combined with the language module, we got writing, which triggered its own cultural revolution. And when the language module combined with the social reasoning module, it produced storytelling.

For those whose spatial reasoning module is stronger than their verbal reasoning module, **Figure 1.1** shows a summary of these results.

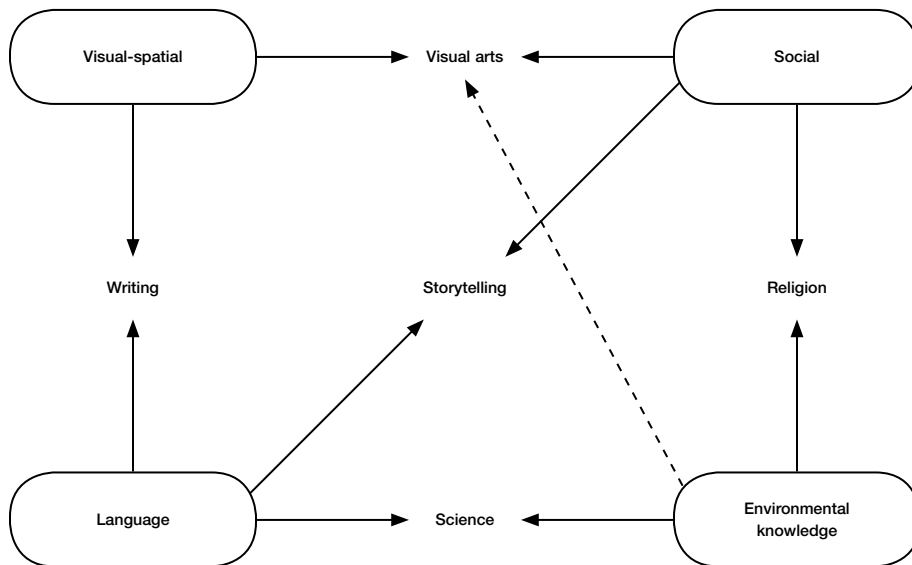


FIGURE 1.1 The mental modules interact

How Storytelling Arose

Storytelling is a linguistic action, so it should come as no surprise that storytelling arose from language. The smallest meaningful linguistic expression is a sentence; words by themselves communicate little. A sentence communicates important information, such as “The antelope is running away.” Put that same sentence into past tense and you have a tiny story. Indeed, it’s difficult to disentangle past tense from storytelling. Thus, storytelling was the inevitable result of the development of language.

Storytelling is universal: Anthropologists have never found a culture bereft of a storytelling tradition. Why is storytelling universal? The obvious answer, of course, is that stories are the vehicle by which cultural knowledge is communicated from one generation to the next. They’re not the only vehicle, of course, but they certainly play an important role in transferring cultural information. Storytelling, language, and culture developed together.

But why should information be transmitted by stories? Why couldn’t cultures simply compile their knowledge into a simple, compact list of important truths—and then impart those truths by requiring every young person to memorize the list? This approach seems quicker and more efficient than the long-winded storytelling system, so why didn’t any culture ever adapt such a superior alternative?

The answer has to do with the character of the information transmitted. Most of the informational content of these stories pertains to social reasoning. Some pertains to the environmental knowledge module, but the bulk of these stories concern interpersonal behavior: trustworthiness, faithfulness, perseverance, and so forth. The mental module for social relationships relies on pattern recognition, which raises an ugly problem: How do you communicate pattern-type information to a pattern-recognizing mental module using a sequential medium such as language? In computer terms, it's like trying to send parallel data down a serial cable. It's like speaking Chinese to a Spaniard. The data is in the wrong format!

What's needed is a reformatter—something that converts one thinking mechanism to the other. Storytelling is that reformatter. It's an ad-hoc solution to an interfacing problem that arose early in the development of language.

Consider that a story is a linear sequence of events; its architecture is referred to as a *plotline*. That linearity is the inevitable outcome of using language to relate the story: You can't tell a story all at once. A story must be told step by step, event by event. Yet you can't understand the story fully until you receive the entire content. If you're downloading your email and the transmission link breaks 90 percent of the way through, you can still read the email and figure out most of its content. But if you're watching a movie in a theater and the projector breaks 90 percent of the way through, you have every right to demand a 100 percent refund; without that last 10 percent, the story never snaps into place.

Stories are patterns broken down into a linear sequence. How does that happen? I'll use a visual metaphor to clarify what happens (well, it's clear to visual thinkers). Imagine your knowledge to consist of a meshwork of connected ideas. A young hunter's knowledge of antelope might look something like the diagram in **Figure 1.2**. The specific positions of the individual ideas are arbitrary; what matters is their pattern of connections.

Now, imagine a hunter telling this story about hunting the antelope:

We were out hunting when we saw some antelope in the distance. We were almost close enough to throw rocks and hit them. But then the wind came up, and it was blowing in the direction from us to them. The antelope smelled us and ran away.

Now, let's draw a schematic diagram of the relationships between the events of the story (**Figure 1.3**).

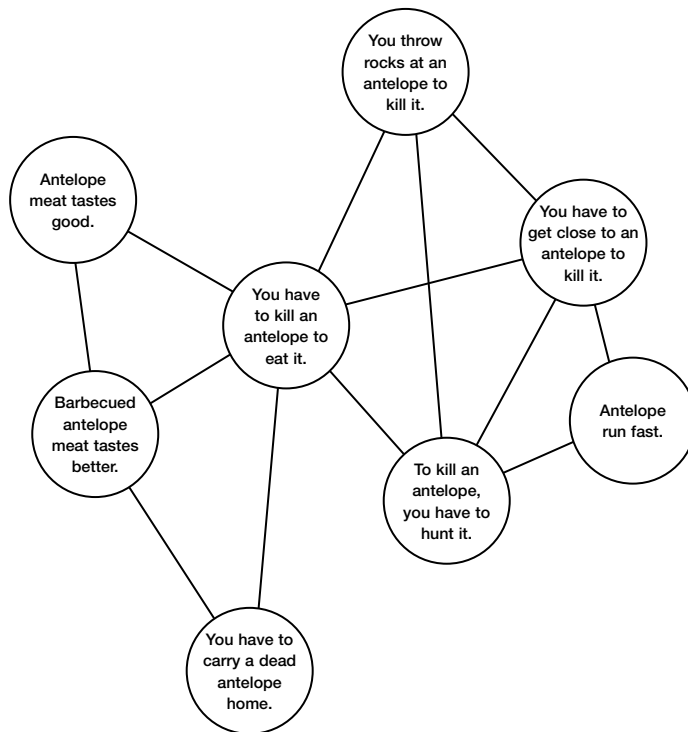


FIGURE 1.2 Schematic of knowledge in the mind

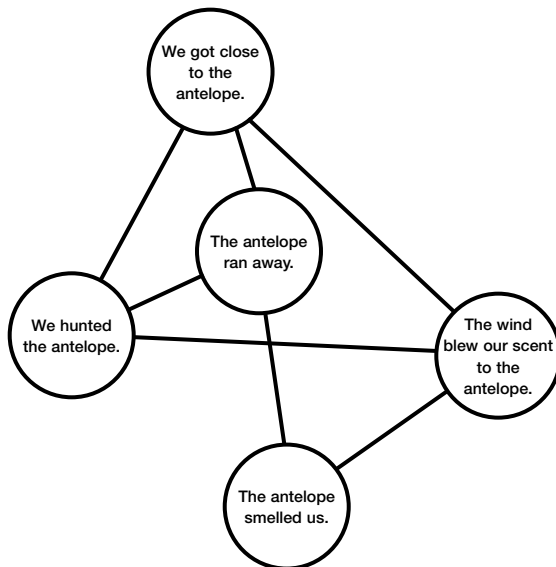


FIGURE 1.3 The story of hunting the antelope

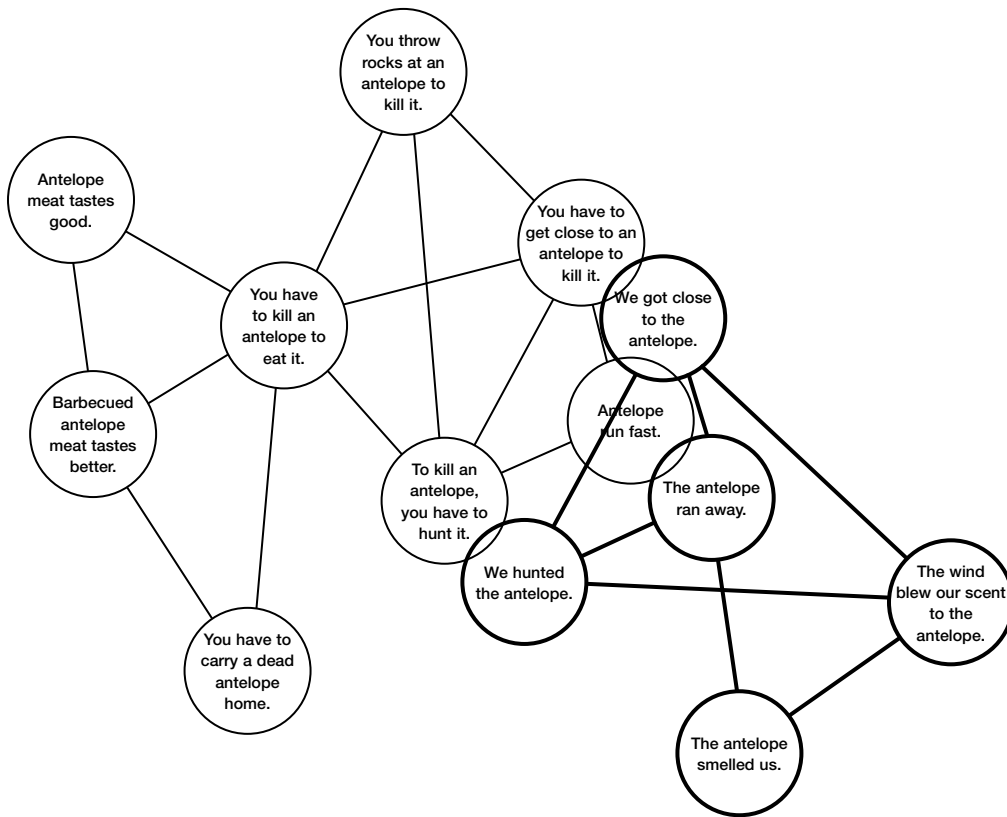


FIGURE 1.4 Merging the story with existing knowledge

The young hunter superimposes this story onto his existing knowledge of antelope (Figure 1.4).

The young hunter can readily see the parallels between his own knowledge and the story. However, he's not finished. That night, while he dreams, his brain reorganizes his associative memory to integrate the experiences of the day into the existing knowledge. In the morning, the young hunter will have this new knowledge structure (Figure 1.5).

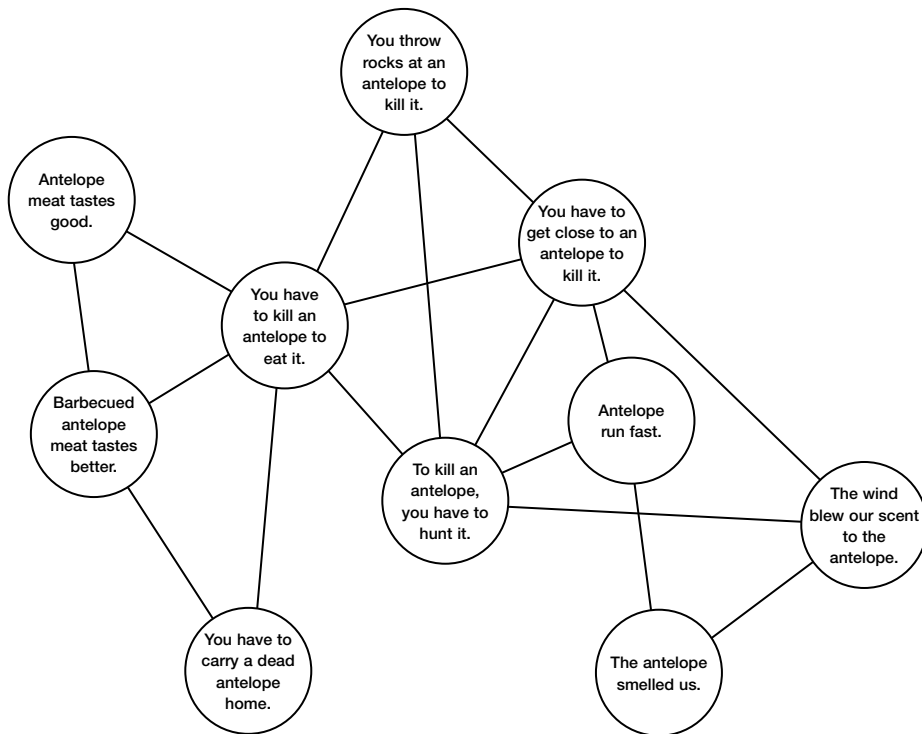


FIGURE 1.5 After integrating the story into existing knowledge

This is the simplest way we learn from storytelling. But there's another, even more important way that storytelling changes our memories: It can alter existing knowledge.

Now, suppose the young hunter listens to another story. I'll leave out most of the story to focus your attention on the crucial part:

We kept throwing rocks at the antelope and hurt it so it couldn't run as fast. We closed in on the antelope and caught it. We smashed its head with big rocks. That killed it.

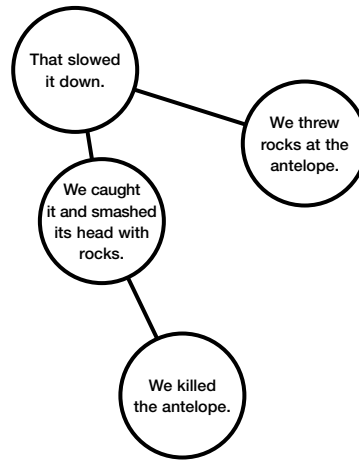


FIGURE 1.6 A fragment of a story

Figure 1.6 shows a schematic of what the story fragment might look like.

The young hunter will easily match up the new story with his existing network of knowledge about antelope (**Figure 1.7**).

Here’s the important part: That night, when the young hunter dreams and integrates the new knowledge into his memory, the brain will also tidy up the memory structure to make it neater.

Note that the previous connection between killing the antelope and throwing rocks at it has been broken: Throwing rocks doesn’t kill antelope—it slows them down. Smashing its head with a big rock is how you actually kill an antelope. Two new bits of knowledge have been added to the network. Also note that the positions of the two original bits of knowledge (“you throw rocks…” and “you have to kill…”) have shifted somewhat. Some associations have been made more distant while some are now closer. That’s what happens whenever you learn something: Everything associated with it in your memory shifts somewhat.



FIGURE 1.7 Superposing the story onto memory

The “Aha!” Insight

This learning process can sometimes produce radical changes in the way we think; sometimes those radical changes take place consciously. Martin Gardner called these *Aha! insights* because we can’t help but exclaim something like “Aha!” or “I’ll be a monkey’s uncle!” (and because *I’ll be a monkey’s uncle insights* just doesn’t sound right).

Aha! insights occur when our network of ideas suddenly transforms into a very different shape. (See Chapter 2, “Interactivity,” for more on the “Aha!” insight.)

Storytelling was invented thousands of years ago and remains the most powerful medium for communicating complex interrelated ideas.

The Nature of Stories

So much has been written about the nature of stories that it seems cheeky for me to throw in my own two cents' worth. Nevertheless, there are a few fundamental points I want to emphasize, if only to get you oriented toward the concepts that underlie this book.

Strong Structure

Stories must satisfy tight structural requirements to be acceptable. Many definitions I have seen strike me as too academically broad. Yes, any sequence of statements can be construed to be a story, but only in the most academic of senses. I can't specify the structure of stories, but I can point out that even a 4-year-old child has a solid grasp of the concept of story. Here are two examples to run past your neighborhood 4-year-old:

*The itsy-bitsy spider crawled up the water spout.
Down came the rain and washed the spider out.
Out came the sun and dried up all the rain.
And the itsy-bitsy spider crawled up the spout again.*

Here you have a clear, simple story in four lines. It has a protagonist (spider), conflict (the rain), struggle (being washed out), resolution (crawled back up), and even a moral (the value of perseverance).

Here's another story:

Once upon a time, there was a handsome young prince who lived in a shining castle. One day he mounted his beautiful white steed and rode out of the castle into the forest. In the forest, his horse fell into a hole and they both died.

This story has most of the elements of many stories: the classic "Once upon a time" opening, a handsome prince, a beautiful white horse, a castle, and a forest. These elements are quite conventional, but the ending is not at all what's expected. Tell this story to 4-year-olds and they'll cry out, "That's not a story!" They'd be right. Whatever structural elements this pseudo-story has, it can't be called a proper story because it violates some fundamental expectations about stories.

I challenge you to devise a set of rigorous criteria for defining stories, a set that would accept the itsy-bitsy spider story while rejecting the prince who died story. I certainly can't meet that challenge.

Lesson 1

Stories are complex structures that must meet many hard-to-specify requirements.

People

Stories are about people. This is such a simple, basic truth that it's often lost in the high-falutin' analysis of narrative theory. Sometimes the reference to people is indirect or symbolic—as in the case of the itsy-bitsy spider, which isn't a person but is understood by the audience to represent a person. An extreme example of this indirection is offered by the movie *Koyaanisqatsi*, which lacks a protagonist or dialogue. The movie presents an artful sequence of images of nature contrasted with the world of civilization. Yet the story is strong, clear, and beautifully summarized in the final extra-long shot of a rocket ascending higher and higher into the atmosphere and exploding. The camera centers on the largest fragment and follows it down, down, down. The symbolism is intense and powerful; this movie is about what we have become.

In addition to exalting the role of people in stories, storytellers minimize the role of things. You must concentrate your attention on the people in stories and simultaneously reject any significant role for things.



But there are plenty of stories that revolve around things. What about The Maltese Falcon, Raiders of the Lost Ark, and The Lord of the Rings?

No, the objects in the cited stories do not play central roles. *The Lord of the Rings*, for example, is not about the ring, but about Frodo's struggle. Replacing the ring with a magic sock or hat or eyeglasses would not have changed the story much. Replacing Frodo with, say, Han Solo, Don Quixote, or Huck Finn would have changed the story beyond recognition. This simple truth—that stories are about people, not things—explains the utter failure of games to incorporate storytelling in any but the most mechanical and forced manner. Games concern

themselves with things: things you acquire, things you use, things you destroy, and so on. That's why they're so emotionally crippled—dramatic power comes from people and their problems, not things. Games aren't antisocial; they're asocial. They just don't bother with people other than as walking dolls that perform mechanical functions.

Lesson 2

People, not things!

Conflict

All stories have some sort of conflict. Sometimes the conflict is direct and violent, as in *Star Wars* or *The Lord of the Rings*. The good guys wear white hats, and the bad guys wear black hats. The good guys are handsome and noble, and the bad guys are missing some teeth and have bad breath. Youngsters with their simplistic view of the universe favor such stories. But there are stories in which the conflict is more indirect as well. Sometimes the conflict is social, sometimes it's symbolic, but there's always some sort of conflict.

The second *Jurassic Park* movie illustrates indirect conflict. The primary conflict is between the mathematician and the soulless businessman; the conflict is established early in the movie when the mathematician tugs at the businessman's sleeve and the businessman warns "Careful. This suit cost more than your education." As the story progresses, dinosaurs rampage with toothy glee, chomping and stomping people in horrifically entertaining ways, but they aren't the antagonists. Indeed, the dinosaurs are presented as something like "noble savages," devoid of evil intent, merely acting out their reptilian instincts with dinosaur integrity. The central conflict of the movie is indirect: The businessman gives orders to his minions, and the mathematician opposes him with vain pleas and dark warnings. Spielberg could have moved all the dinosaurs offstage, saving millions of dollars, without compromising the integrity of the story—although it wouldn't have been nearly as much fun.



Well, here at least is one place where games shine.

Not exactly. It's true that games have plenty of conflict, but it usually takes the simplest and most direct form: violence. Games are deficient in other forms of conflict, such as social conflict. The game designer's spice shelf has lots of sugar and nothing else.

Lesson 3

Stories are about conflict, most commonly social conflict. Violent conflict is for simple stories.



What about Macbeth, Hamlet, or Romeo and Juliet? Those plays end with everybody dying. Would you dismiss them as simplistic?

The violence in those plays is not part of the conflict—it's part of the resolution. A brutally short summation of all three plays is this: "A profound personal conflict was tragically resolved in violence." In storytelling terms, the violence is the denouement, not the climax.

Puzzles

Stories are not puzzles. It's true that puzzles often form a part of the story; indeed, puzzles play a large role in mystery stories. The puzzles in a mystery story are primarily about people, however. The bulk of the story is devoted to detailing the sleuth's machinations in getting people to reveal crucial clues to the mystery.



But there's also plenty of nonsocial sleuthing going on. Take the television series CSI, for example. It's about using science rather than interpersonal skills to solve crimes, and it has been a big hit.

True, but the science is not what gives the stories their power. The gee-whiz technology is certainly fun, but the strong characters and often poignant storylines are the true strengths of the stories. Without those strong character interactions, *CSI* would be a run-of-the-mill whodunit.

A story always contains some kind of problem or challenge: How will the protagonist resolve the conflict of the story? If this problem is an intellectual one requiring a logical solution, it's called a puzzle. Nonetheless, the exercise of intellectual

legerdemain is seldom central to stories. Sometimes, the protagonist recruits other characters to assist with the solution of the puzzle or undergoes a major personality change to discover the solution to the puzzle. The greatest stories provide challenges that are personal or social in nature, not logical or intellectual.

Lesson 4

Puzzles are not a necessary component of stories.

Choices

Ultimately, stories concern the choices that characters make. Indeed, the entire point of many stories is revealed through a key choice the protagonist makes. In *Star Wars*, it's "Trust the Force, Luke." In *Macbeth*, it's the decision to murder for ambition. In the third *Matrix* movie, it's Neo's decision to sacrifice himself. In many versions of the Arthurian legends, it's Guinevere's decision to act on her love for Lancelot. In each of these examples, the entire story builds up to or revolves around a key decision.

But the key decision is only the most dramatically salient one; other choices throughout the story establish character. Neo is reluctant to accept his role as the savior of humanity; he expresses that reluctance in a hundred small actions. Slowly Neo comes to accept the truth; again, his acceptance is established by the decisions he makes, demonstrating greater confidence in his skills.

Spectacle

Novel experiences provide a certain amount of entertainment value arising solely from the fact that they're new. Sometimes you try out an exotic food merely for the simple pleasure of tasting something you've never tasted before. A great deal of music, especially popular music, relies on including odd new sounds. Rock music revels in unconventional sounds: The electric guitar, feedback effects, the fuzzbox, and other such acoustic effects play a major role in rock music.

The visual form of this pleasure from novelty is *spectacle*, providing exotic imagery as a form of entertainment. Spectacle dominates the movies these days. The very first movies were spectacles without stories—trains crashing and so forth. *Star Wars* in 1977, *Jurassic Park* in 1991, and *The Matrix* in 1999 exemplify the appeal of spectacle.

Computer games have followed a related course. The earliest computer games offered nothing more than moving squares on a television screen; the evolution of computer games has been dominated by the quest for ever more realistic graphics to provide ever better spectacle. The progress made in the past 25 years is truly astounding. Expect even greater things in the future.

All this fabulous spectacle has engendered in the minds of most younger people an unfortunate confusion between story and spectacle. Many associate spectacle with story so strongly that they believe spectacle is a necessary component of story.

A sad experience of mine forcefully demonstrates the degree to which spectacle has crowded out story in the minds of many. Once upon a time, I acquired a copy of Charlie Chaplin's classic *The Gold Rush*. After watching it, I enthused about the beauty of the story to a young friend, urging him to come watch it with me. With some reluctance, my friend consented. Barely a quarter of the way into the movie, he stood up to leave the room. Chaplin's great work bored him.

The scenes were all "plain," the story's pace was torpid, and the movie was black and white and had no sound. How could I expect him to endure such a boring film? In my friend's opinion, a movie without spectacle simply wasn't worth watching.

But Aristotle, in his classic *Poetics*, ranked spectacle as the least important of the six elements of story. (His priority list was plot, character, thought, diction, song, and spectacle.)



Yeah, well, things have changed a bit in the last two thousand years. Maybe Aristotle gave scant value to spectacle because the computer-generated imagery in those days was really crummy.

It's true that spectacle sells; the top-selling movies usually have great spectacle. But this is correlation, not causation. Some spectacular movies flopped, such as *Final Fantasy*, *John Carter*, and *Catwoman*. Meanwhile, plenty of movies lacking spectacle made huge profits: *The Sting*, *The Godfather*, and *Forrest Gump* provide good examples of films that were hugely successful because of their stories, not their spectacle.

An even better case is made by a head-to-head conflict between two movies in 2001: *Final Fantasy* and *Shrek*. The first of these was a computer-graphics film boasting state-of-the-art graphics. The producers boasted that the software

modeled 30,000 individual hairs on the heroine's head. The imagery truly was spectacular. By contrast, *Shrek* used simple computer graphics, much inferior to those of *Final Fantasy*. However, *Shrek's* story completely outclassed *Final Fantasy's*. The result was predictable: *Shrek* outsold *Final Fantasy* by ten to one.

The exaltation of spectacle is a recent and temporary infatuation. Consider that literature has never relied on spectacle, theater relies on a limited form of spectacle, and cinema underwent short bursts of spectacle arising from technological advances (initial invention in the 1890s, sound in 1930, color in 1940, and special effects starting in the 1970s and continuing through today). Each of these technological advances was initially the source of much infatuation, with movies overusing the technology. After a few years, the technology reverted to its normal position as a supporting element of the story rather than the central attraction.

Lesson 5

Spectacle does not make stories.

The Tyranny of the Visual

Many observers have noted that our culture is increasingly dominated by image. Indeed, a recent book was titled *The Rise of the Image, the Fall of the Word*. No doubt, image plays a larger role in our culture now than at any time in the past. In many ways, this is good. Glorifying war, for example, is much harder when the news gushes with images of the horrible reality of war. This applies in fictional representations as well. Old war movies showed victims clutching their chests and sanitarily falling face down. Today, a movie such as *Saving Private Ryan* shows heads being blown off bodies and people being ripped apart with such graphic realism that, after watching the movie, a friend of mine declared he could not see how anybody could ever again support a war.

Tremendous effort has gone into improving the quality of images. Computer graphics has absorbed billions in research and development money and consumed the energies of thousands of our brightest minds. Hundreds of people labor over the computer graphics in a major movie, and their efforts are usually rewarded with bounteous ticket sales. Television in 2012 is looking forward to the imminent dominance of HDTV and the accompanying huge improvement

in image quality. Electronic networks are dramatically increasing their capacity, primarily to permit the transmission of larger and more detailed images. The text you type amounts to a few kilobytes, while images gobble up megabytes.

I won't condemn the rise of the image in our culture, and I will not bemoan the decline of the word. Instead, let's consider the ways visual thinking has come to dominate our thinking to the exclusion of subtler cognition.

When I gaze upon a scene, I realize that I'm perceiving but a tiny fragment of reality—I am perceiving reality through the narrow window of the visual. When I look at a tree, I perceive so much more than the visual image. I imagine the fluids slowly creeping through its cambium, the photosynthesis taking place in its leaves, the absorption of nutrients from the soil—all these invisible processes that are central to the life of a tree. My eyes tell me only so much about the tree; there's much more going on outside my view.

Note that my perception of the tree is informed, indeed driven, by my education. Because I have read about biology, trees, and physics, I bring an understanding that allows me to see deeper inside the tree. My perception of the universe is an integration of my knowledge and my senses.

Consider this analogy: Suppose that you are watching a black-and-white movie. You see an apple presented in shades of gray, but you know that the apple must be red. Your real eyes see a gray apple, but your mind's eye fills in the color. Now extend that analogy. What if you also perceive the smooth texture of the apple's skin, the slow oxidation of the apple's flesh—everything going on in that apple. By living only in the world of the visual, are you "seeing" less of the world than you could?

The Wachowski brothers created a stunning visual analogy to this process in the first *Matrix* movie. At the climax, Neo has revived from death and can now see the matrix for what it truly is. He looks down the corridor at the three agents and sees not the corridor, but the code behind it. The image communicates the idea of seeing the processes behind reality rather than just the visual skin of reality. Ironic, isn't it: a visual representation of an idea that attempts to get around visual thinking?

Lesson 6

Visual thinking should not dominate storytelling.

Spatial Thinking

Closely related to visual thinking is spatial thinking. Spatial reasoning is one of the brain's greatest achievements. When you reach around behind the refrigerator to retrieve the fallen spoon, you're using spatial reasoning. People often metaphorically apply spatial reasoning to nonspatial problems, as evidenced by such statements as "Joe and I aren't very close friends" or "Your statement is wide of the mark." Spatial reasoning used metaphorically, as in these examples, works well. The problem comes when people apply it too literally to storytelling. Spatial reasoning is out of place in the universe of drama because drama is about people, not things.

In my designs for interactive storytelling, I have always used a simple arrangement: Space is composed of individual stages with no spatial relationships whatever between stages. In my definition, a stage is simply a location containing characters and props. Characters simply disappear from one stage and reappear on another. Inside a stage, all characters are able to interact with each other without any spatial considerations. It's a simple, robust model, and it closely approximates the way space operates in most stories. This broader definition of "stage" can be applied in any storytelling medium: theater, cinema, literature, or interactive storytelling.



You can't ignore spatial considerations in stories! Luke Skywalker and Darth Vader can't fight with their light sabers unless they're close together. Han Solo can't kiss Princess Leia unless they're face to face. They couldn't have rescued the princess unless they got on board the Death Star.

How many times do the spatial relationships of actors play a crucial role in those interactions? The important factor in the fight between Luke Skywalker and Darth Vader was not the separation between them; it was a dramatically insignificant act to close the gap between them. The crucial factor was the decision on Luke Skywalker's part to take on Darth Vader. Han Solo didn't kiss Princess Leia because they just happened to be close together; the feelings they had for each other drove that interaction. Rescuing the princess was primarily a dramatic decision on Luke's part.

Don't let your familiarity with movies dominate your thinking about stories. Consider a medium in which spatial factors are not automatically included: literature.

For example, *The Odyssey* is at heart a story of a journey around the Mediterranean. Isn't that fundamentally spatial? Indeed, the journey motif shows up repeatedly in literature, from *The Odyssey* to *Huckleberry Finn* to *Star Wars*. But are the spatial motions central to the story? Are they not merely transitions from one stage to another, without any genuine spatial content? The best evidence for the chimerical nature of spatial factors in stories is that the actual spatial relationships are never specified. Odysseus traveled for many days and came to the Land of the Lotus Eaters—where is that? How far away is it from Scylla and Charybdis? Is it closer to Troy or to Ithaca? These relationships aren't specified in *The Odyssey*—because they're not important.

The same theory applies to other journey stories. *Huckleberry Finn* takes place on a specified river with specified places that could be found on a map. Yet many of the details just aren't there. Where was it that Huck saw the body of his father? And how far away was that from the town where they tarred and feathered Huck's shyster buddies? Or the place where the feud led to the murders of his hosts? The reader doesn't know because those spatial relationships are irrelevant to the dramatic matters addressed in the story.

Consider this standard sentence from a romantic novel: "Overpowered with passion, he seized her in his arms and kissed her frantically." Now consider the sentence modified to include spatial factors: "Overpowered with passion, he walked over to her, seized her in his arms, moved his head directly in front of hers, rotated his head slightly to avoid a collision of noses, then closed the gap between their lips and kissed her frantically." Pretty silly, eh? Spatial factors just aren't that important in drama. If an actor needs to alter a spatial relationship to get something done, he simply does it, and you don't need to worry about the mechanical details of how that's carried out. Stories are about the exercise of emotion, not musculature.

Consider the *Star Trek* universe, developed over the course of more than three decades by hundreds of people. Fat encyclopedias define and list all manner of details of the *Star Trek* universe. You can read technical specifications for all kinds of starships and wondrous devices, plus detailed histories of dozens of different species and their home worlds. Yet this stupefying mountain of data lacks one of the simplest of all documents: a map. Nobody can tell you whether the planet Vulcan is closer to the Klingon home world or to Earth. Where is the

planet Andoria? How do you get to Raisa or Cardassia? There are literally hundreds of stories from the *Star Trek* universe, but there's never been a map.

Lesson 7

Stories take place on stages, not maps.

Temporal Discontinuity

Just as stories break up space into discrete chunks called stages, they play havoc with time—breaking it up, jumping backward and forward, and skipping it altogether. Actors are shown embarking on a journey and arriving at their destination; the time the journey itself consumes is skipped over. Years are disposed of with the note, “Many years later...” Simultaneity is presented with the simple phrase, “Meanwhile, back at the ranch...” Flashbacks jerk you backward in time and then return you to the present. Dramatic time just doesn't behave like physical time; it follows whatever course the story requires.

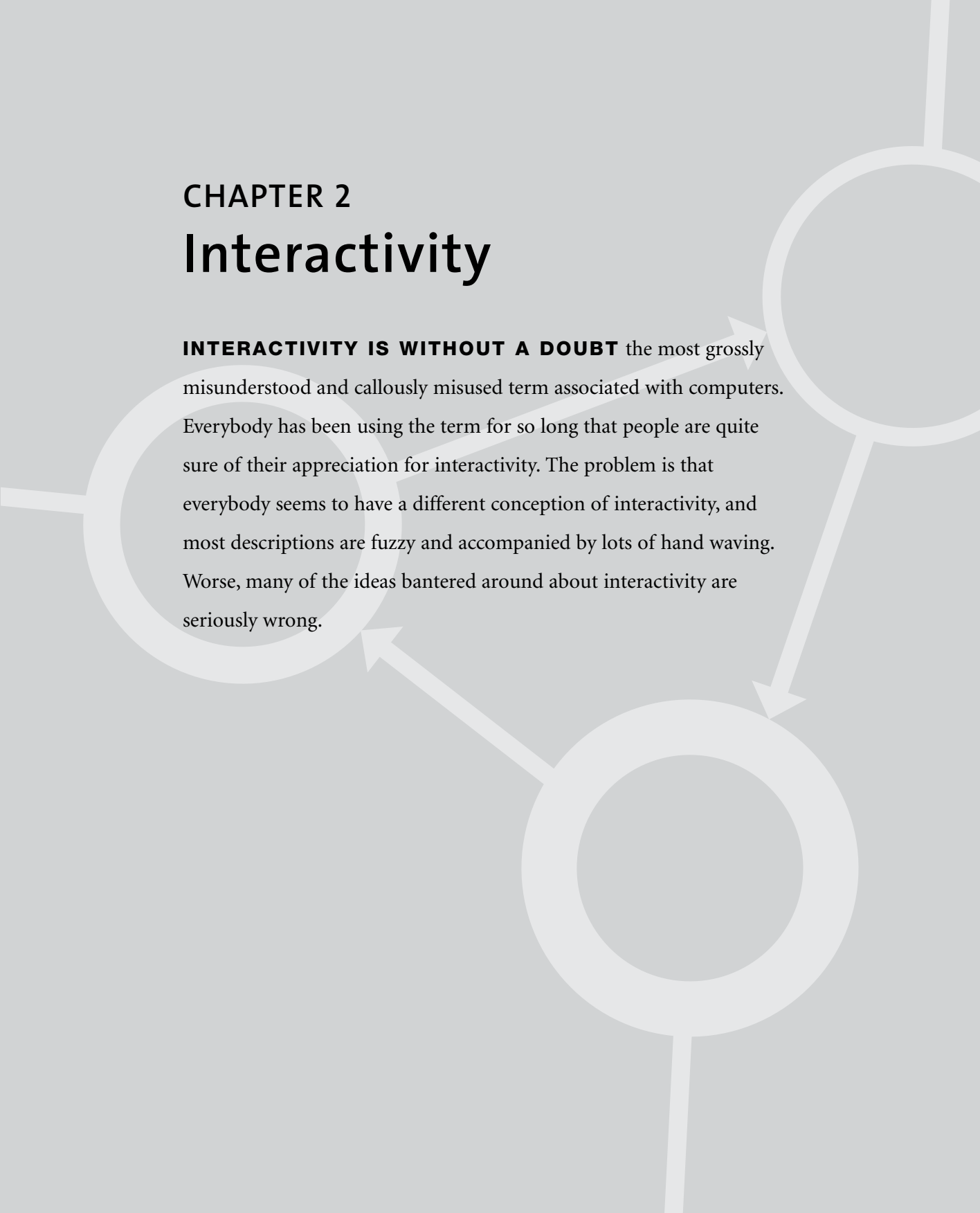
Fortunately, temporal reasoning facilities aren't as unconsciously pervasive as spatial reasoning facilities, so people seldom have difficulty appreciating the liberties storytellers take with time. Nevertheless, computer game designers contemplating the problems of interactive storytelling should disencumber themselves of the old notions of physical time that dominate games.

Conclusions

This chapter offers no grand theory of storytelling. Instead, it offers a number of separate points about story that are important considerations of interactive storytelling. Storytelling is so deeply entwined in human nature and cultures that I doubt whether the true essence of stories can ever be nailed down. If you mean to build interactive storytelling systems, however, the points in this chapter should be clear in your mind before you turn to interactivity.

CHAPTER 2

Interactivity



INTERACTIVITY IS WITHOUT A DOUBT the most grossly misunderstood and callously misused term associated with computers. Everybody has been using the term for so long that people are quite sure of their appreciation for interactivity. The problem is that everybody seems to have a different conception of interactivity, and most descriptions are fuzzy and accompanied by lots of hand waving. Worse, many of the ideas bantered around about interactivity are seriously wrong.

My Definition of Interactivity

I offer this definition of interactivity:

A cyclic process between two or more active agents in which each agent alternately listens, thinks, and speaks—a conversation of sorts.

In this definition, the terms *listen*, *think*, and *speak* must be taken metaphorically. A computer doesn't listen in the strict sense of the term, but it does listen to its mouse and keyboard. It may not speak, but it does something operationally similar when it displays output on its screen. And, of course, a computer never thinks in the true sense of the word, but it does process data or calculate. I suppose I could have used the terms *accepts input*, *processes input*, and *outputs results*, but those terms are just as narrowly computerish as the earlier description's terms are narrowly humanistic. With this proviso for the broader sense of the terms *listen*, *think*, and *speak*, the definition is a clear statement of exactly what constitutes interactivity.

The value of this definition lies in its reference to conversation, a well-understood form of interaction. Our experiences with conversation offer useful guidance in software design. Obviously, the overall quality of a conversation depends on the particular quality with which each step—listening, thinking, and speaking—occurs. Even more important is the way those three qualities combine. Many people assume that maximizing the quality of each step is all that's required to achieve the maximum quality of interactivity, but the truth is more subtle.

Consider that the overall quality of a conversation doesn't depend on the isolated qualities of each step—each step must be executed well if the conversation is to succeed. Can you recall conversations in which your participants weren't listening to your words? In such cases, no matter how refined their thinking or eloquent their speaking, the conversations fell short because without good listening, it's not an effective conversation—it's a lecture. In the same fashion, I'm sure you can recall conversations with people who were incapable of understanding your point—and these conversations were just as frustrating and pointless as the previous type. Lastly, you can likely recall conversations with a tongue-tied, inarticulate person who simply couldn't construct an informative sentence. Again, the conversations were failures because without quality in that third step—speaking—the quality of the first two steps didn't matter.

Lesson 8

The overall quality of interactivity (human-with-human or human-with-computer) depends on the product, rather than the sum, of the individual qualities of the three steps. You must have good listening and good thinking and good speaking to have good interaction.

My definition rejects a number of phenomena mistakenly held to be interactive. For example, *reaction*, no matter how intense, is not the same as *interaction*. If you're watching a great movie, and your heart is pounding with excitement and your fingers trembling with emotion, you're still not interacting with the movie. The movie is not listening to what you are saying, nor is it thinking about anything. It is only speaking. It speaks well and powerfully—that's good! But it is not interacting.



Not so! The viewer engages in active interpretation of the movie and therefore is not in a passive role.

Here is a snippet from the book *Interactive Storytelling for Video Games*, which makes the same argument.

One of the illusions of the “interactive age” is that great art is not interactive. Okay, so indeed not all art has buttons to click on or hyperlinks to follow, but all deep and long-lasting art is participatory in some fashion. Whether it simply evokes an emotion in the viewer or alternatively invites analysis to truly understand the work itself, any art that lasts is multilayered and reveals more upon each repeated examination. To me, this is the very essence of interactivity: the invitation to probe deeper... Art evokes reaction from the audience, which is in part what makes gaming a wonderful medium for artists.

This argument confuses the active/passive dichotomy with the interactive/reactive dichotomy. Reaction is an action, but that doesn't make it an interaction. The audience can actively think, but that doesn't change the fact that the movie isn't thinking. The relationship between the movie and the audience is fundamentally one-sided: The movie does all the speaking, and the audience does all the listening and thinking. The audience does not act on the movie; it merely reacts to the movie.

Interaction requires that the “action” be “inter” (between or among) the agents. If the action is one-sided, it’s not “inter”; it’s “re.”

I belabor this point because so many people balk at it. Perhaps they are influenced by the current status of interactivity as the latest buzzword and the implication that interactivity is somehow “New! Better! Hot! Cool!” My declaration that movies are not interactive becomes, by implication, an assertion that they are “Old! Worse! Tepid!”—a suggestion that any knowledgeable person would reject. Hence, people reject the notion that movies aren’t interaction.

Let me set this matter straight: I am not denigrating movies. I like movies. Some of my best friends are movies. I take a movie out to lunch every year on the American film director D.W. Griffith’s birthday. Cinema is a highly developed medium that does its job very well. But every medium has its strengths and weaknesses. Movies aren’t interactive. Because interactive storytelling is procedurally manufactured rather than handcrafted, it will never have the highly polished internal structure that movies have. They are two different media.

Second-Person Insight

All great artists have some special insight that gives their work profundity. A brilliant composer has an inner ear that can judge the feel of music. A painter has an eye for form, shape, and color. A deep insight into the language gives a top-notch writer the ability to come up with the perfect phrase. In the same way, the interactive artist needs a special kind of insight, an artistic acuity few others share. I call that acuity *second-person insight*. It’s the ability to think primarily in terms of how an expression will be perceived by the audience.



This isn’t unique to interactive storytelling; all artists worry about how their expression will be perceived by the audience.

True, but this isn’t an artist’s primary concern. An expository artist’s main task is to get the expression right. A writer prepares a first draft and then attempts to reread it from the reader’s point of view. In other words, the primary emphasis is on the expression itself, and the secondary emphasis is on the audience’s perception. An artist whose work isn’t understood by the masses doesn’t seek the nearest tree with

a noose in hand. Whether it's a play, a painting, a movie, a novel, or a poem, if people don't understand it, that's their problem, not the artist's.



This is true for fine artists, but entertainers cater to the tastes of their audiences. What difference is there between the second-person insight and the sensitivity to the audience intrinsic to any good entertainer?

Entertainers do indeed require such insight, but it differs in two ways from the second-person insight required of designers of interactive storytelling. First, it is necessarily one-size-fits-all in nature. The entertainer must create the expression that will entertain the average member of the audience. If a few members of the audience don't get the joke, realize the implications of a dramatic action, or fail to remember the crucial link from the character's past, that's an acceptable loss. The interactive storyteller is expected to address a broader range of mentalities, coming up with ways of reaching people further from the average.

The second difference lies in the depth of reaction required. The entertainer's actions are one-shot in nature: The joke, plot twist, or sudden violence evokes a reaction, and that's as far as the entertainer goes. The designer of interactive storytelling must think several layers deep: If the user does X, can I respond with Y? And what happens if the user reacts to Y with Z? The interactive designer is like a chess player, thinking many moves ahead of the user; the entertainer concentrates on a single step at a time.

Therefore, second-person insight requires a fundamental and profound shift in attitude. You must go beyond the normal consideration of your audience's perceptions and make those perceptions the entire thrust of your efforts. You must develop the mental discipline to get out of your own mind and get into your audience's mind.

There are two facets to second-person insight. The first is enhancing your empathy, but it goes much further than having an emotional appreciation of the audience's state. It's not just the ability to empathize with their likely emotions, but the ability to empathize with emotions you wouldn't feel. You might design a corner of your storyworld with pathos in mind, but your player might react to that situation with anger. You must not reject or suppress that emotional response; you must anticipate and respect it. The second facet operates on an intellectual plane; you must be able to visualize the confusion audience members

bring to the experience. What questions will be going through their heads? What assumptions will they be making? How will their minds mesh with the thinking in your work, and where will their thinking clash with yours? How can you minimize the likely clashes between your storyworld and their thinking?

We spend so much time inside our own heads (24/7 for sane people) that seeing the world through another's eyes is immensely difficult. Few people appreciate just how differently other people think. Teachers do; it's quite a jolt teaching your first class. You stand up in front of your students; reveal the truth to them in a few clean, simple sentences; and note with shock the utter incomprehension in their faces. So you repeat yourself with more elaboration, and behold the same blank stares. The act of teaching is mostly a matter of finding an infinite number of ways to communicate an idea. All good teachers have strong second-person insight, at least in the intellectual dimension.

Here's a simple test of your second-person insight. Imagine that you're having a video conversation with a friend using a handheld smartphone or tablet. If you hold the device in your hand, you're failing a simple test of second-person insight. Your hand jiggles, making the image of your face flit about your friend's screen, which can give her a headache. If you could put yourself in your friend's shoes, you'd know to prop your device in a stable position.

A Model for Human Understanding

A detailed appreciation of the nature of human understanding will clarify the role of second-person insight and reveal the importance of interactivity. The starting point is the notion of associative memory. Information in the human mind is not stacked neatly in files and folders the way it's organized inside a computer. Rather, the ideas are organized by association. For example, credit cards and bank accounts are perceptually different from money, yet most people associate credit cards and bank accounts with coins and bills, even though credit cards and bank accounts don't contain actual coins and bills. You might imagine coins and bills moving into or out of your bank account, even though nothing is moving; it's just numbers being added or subtracted. The mental associations spread out from there: Money flowing into accounts from your paychecks,

flowing out to pay rent, gaining interest, being transferred to savings or invested in stocks—all an imaginary process carried out in your mind by associations between these concepts.

Many of these associations are natural and logical, but people put different weights on them. Perhaps your view of these associations puts your checking account at the center of a web of connections. Another person might use the credit card as the focal point. A third person, blessed with lots of financial savvy, might well refuse to think in terms of a financial center of gravity and instead connect all these ideas in a more egalitarian style, concentrating on the balance of the overall distribution of assets among stocks, bonds, T-bills, cash accounts, gold, and so forth (Figure 2.1).

Financial Assets

Liquid Assets

Checking Account
Savings Account
Piggy Bank

Semi-liquid Assets

Stocks
Bonds
T-Bills

Illiquid Assets

Home Equity
Insurance Policy

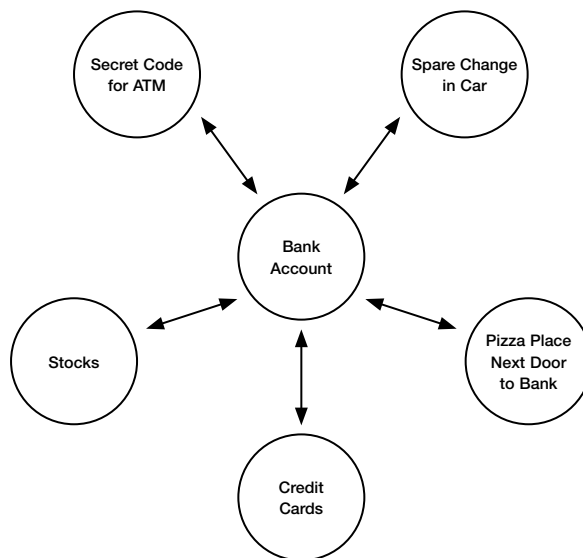


FIGURE 2.1 Structured memory versus associative memory

Even more important for interactivity considerations is the idiosyncratic nature of many of these associations. Some people have completely different connections; some people have connections in common but put different weights on those connections. Someone who invested heavily in Enron will have strong emotional associations with stocks; another person who held lots of dotcom stocks in the early 2000s will have different emotional associations. A person who recalls grandfather's

old safe with its pile of grand stock certificates will associate stock with certificates; the absence of those certificates from modern stock trading might induce a certain nervousness not shared by others. And what if grandfather had often intoned in his deep, resonant voice, “Stocks, my child—you can’t go wrong with a solid portfolio!” How would that fond memory distort the web of associations in a person’s mind? And what if grandfather had died penniless?

It is this network of associations that constitutes human memory and human understanding. We each build our own network, node by node, connection by connection, as we learn and grow. Because we all live in the same universe, our networks often share gross similarities, but the differences in our experiences ensure that each network’s particular structure is unique to its builder. This explains why we so often misunderstand each other. You and I might have a roughly similar overall understanding of the stock market, but your associations are so different from mine that we can come to stunningly different conclusions. You can explain your reasoning with determination and patience, but if my mental network doesn’t coincide closely with yours, I just won’t get your point.

These discrepancies of experience and understanding are just part of the human condition; we accept them, shrug our shoulders, and mumble, “*De gustibus non est disputandum.*” (“There’s no arguing about taste.”) If the audience rejects an artist’s work, the artist can dismiss the audience members as obtuse cretins who are simply unable to perceive his or her vision (network). Too bad for them.

The revolutionary value of interactivity lies in its ability to get past this limitation. Imagine the teaching process not as expository lecture, but as interactive conversation. The teacher probes the student’s network, noting carefully the deeper significance of the questions the student asks. The teacher uses that information to infer the structure of that network. Here is where the teacher’s second-person insight comes into play; a good teacher can quickly reconstruct the student’s network and identify the misplaced node or incorrectly weighted association. With the problem identified, the teacher can work to shift the student’s network. Initially, the student will resist, attempting to fit the teacher’s observations into the network in a manner that doesn’t stress or distort it. But the teacher presses on, and suddenly the student’s network snaps into place. “Aha!” says the student.

This is the power of interactivity: Interaction reveals the discrepancy between the artist's and the audience's networks and makes it possible for the artist to address that discrepancy. The result is an ability to reach people with tremendous impact. That's what makes interactivity so powerful and what justifies interactive storytelling's loss in narrative finesse.

To use interactivity effectively, you must understand the human truth that people see and hear what they want to see and hear, and then you must come to understand those desires. Being right is not good enough; you must somehow see your truth through other people's eyes.

Second-person insight, like so many artistic gifts, involves a certain degree of mental aberration. To be of any value, the artistic expression must be unconventional, or at least nonobvious. At the same time, you must see your truth from many points of view. You must be able to see how your truth fits into many different networks of knowledge. It's a weird way to think, perhaps within the reach of only a gifted/cursed few.

Ego Control

Another requirement for the interactive artist is an iron determination to subordinate your own desires and interests to those of your audience. You must push down the artistic egotism that glories in self-expression, replacing it with the audience's perception.



So you want me to abandon my own artistic drive and instead just "give 'em what they want." Sounds more like Barnum & Bailey than Michelangelo.

Absolutely not. I'm not suggesting that you should abandon your own artistic interests—after all, that's what you have to offer your audience. Yes, you want to speak your truth, but that's the easy part; the hard part is making them hear it. Ergo, you must silence the shouts of your ego so that you can hear the whispers of your audience's needs.

Perhaps a metaphor will help. Imagine the king of a medieval country deciding how much of the country's wealth should be dedicated to his own comfort and pleasure. "As the king, I'm the most important decision maker in this country, so it's vital that I be cushioned against the distracting vexations and tribulations of life. It's my job to concentrate on the highest-level decisions. It is better for all if I wear the finest clothes, eat the best foods, and live in the most luxurious palace in the country. Some of the peasants might starve, but it's all for the greater good."

You can instantly see through the selfishness of this argument, but how does it differ from this one: "As the artist, my ideas are the most important in the world, so it's vital that I be cushioned against the distracting desires and interests of my audience. It's my job to concentrate on the grandest artistic ideals. It is better for all if I heed my own voice, aspire to my own goals, and pursue my own interests. Some of the audience might not understand my work, but it's all for the greater good."

This argument doesn't apply to all art—just to interactive art. Artists in other fields are perfectly justified in neglecting the needs and desires of their audiences because their audiences are so large that those needs and desires average out to meaningless gray murk. The great Greek sculptor Polyclitus once proved this point with a simple exercise. He showed a work in progress to a series of critics and asked each what could be done to improve the work. Then he carried out all their suggestions. The result was an ugly monstrosity. A work of expository art must have a single unifying vision; all those audience ideas and variations only muddy the waters.

Interactive art is profoundly different because it's experienced individually by millions of people. Millions of people have seen reproductions of Michelangelo's *The Creation of Adam* on the ceiling of the Sistine Chapel, and every single one of them has seen exactly the same thing because there's just one image. But millions of people can play an interactive storyworld, and each one can experience something that nobody else has ever experienced. That's the whole idea of interactivity: It responds to each person individually. Accordingly, you cannot hide your ego behind the argument that the audience's needs and wishes are all averaged together. You must face each player individually.

Degrees of Interactivity

Interactivity is not a Boolean quantity like mortality (either you have it or you don't); it's a numeric quantity like weight (you can have more or less of it). My favorite example of a low-interactivity phenomenon is the refrigerator light. You open the door and the refrigerator light turns on; you close the door and it turns off. That's interactivity! (The refrigerator light "listens" to the door switch being opened, "thinks" with the simple-minded logic of "Switch open, turn on light!" and "speaks" by turning on the light.) But it's dumb interactivity. It might entertain a 3-year-old for a little while, but even a 4-year-old quickly outgrows the insipid interactivity of the refrigerator light.

At the other end of the scale, sex is an example of the most intense interactivity. Powerful lovemaking is the deepest interaction two people can have; is it any wonder that society adorns it with so many mores?

Lesson 9

Your designs should aspire to the ideal of metaphorically having sex with your users.

Three factors determine the degree of interactivity in storytelling: speed, depth, and choice.

Speed

Of the three factors that determine the degree of interactivity, speed is the simplest to understand. At the bottom end of the scale, slow applications destroy interactivity. At the opposite of the scale, the faster the speed, the greater the opportunity for interactivity. Three examples demonstrate this point: the VisiCalc spreadsheet, the BASIC programming language, and the Internet.

The first example of the impact speed has on interactivity is the spreadsheet. VisiCalc was the first spreadsheet for personal computers, and computer historians agree that this program did more to launch the PC revolution than any other program. But VisiCalc wasn't the first spreadsheet—not by a long shot. Plenty of spreadsheet programs were available for big mainframe computers, but they were batch-processing programs. You punched your data onto punch cards, submitted your job to the computer center, and then picked up your output the next day.

You studied the printouts, made a few changes in your data, punched up new cards, and resubmitted the job. If you were lucky, or you stayed at the computer center until 3 a.m. when nobody else was submitting jobs, you could get your turnaround time down to a few hours, in which case you could run through half a dozen scenarios in one night.

VisiCalc wasn't as powerful as mainframe spreadsheets; after all, it had to run on a tiny microcomputer. Moreover, a PC's small screen showed only a fraction of a typical spreadsheet; the "real" spreadsheets at computer centers could print results on sheets of paper so big that you could see everything at once. VisiCalc had just one advantage over conventional mainframe spreadsheets: If you changed a number, it processed the change immediately and presented the results in a flash. In terms of features, display, and overall computational power, VisiCalc was a loser. Its interactivity was thousands of times faster than mainframe spreadsheets, however, and that made all the difference in the world.

The second example of the impact speed has on interactivity is the BASIC programming language, developed at Dartmouth in the late 1960s. A number of languages that emphasized simplicity had been designed for students, but something quite unexpected made BASIC stand out: its interactivity. BASIC was an interpreted language, not a compiled language. Most computer languages are compiled. You type up your program, submit it to the compiler (a program that translates your program into machine language), and then run your program to see how it works. The compilation step could take several minutes in the old days, so after submitting your program to the compiler, you would take a coffee break before returning to see how it came out.

It probably had a few bugs, so you would fix one or two bugs, submit it again, and go have another cup of coffee. Repeat this process all day long and your eyes were bulging out of their sockets from the caffeine—and you made only minor progress on your program.

But BASIC is interpreted: It's designed to be run immediately, without compiling. You type up your program and run it; the results appear immediately. If there are bugs, which is usually the case in a first-cut program, you make a change and run the program again. This process is so quick and easy that you never quit for a cup of coffee; you just sit in front of the computer, lost in intense interaction with it.

As a result, BASIC took the programming world by storm. Within just a few years, everybody was using it to teach students. It's actually a crummy language (mostly because it's interpreted, not compiled) with all sorts of problems, and it teaches bad habits. The single factor of rapid interactivity, however, put it way ahead of everything else.

On a more modern note, the third example of the impact speed has on interactivity is the Internet, aka the “World Wide Wait.” Back in the 1990s, and well into the new century, people used modems on telephone lines to use the Internet. It was slow and frustrating to wait for web pages to load. I will never forget the sense of exhilaration the first time I sat down and worked with a broadband connection—the web came alive. That reduction in turnaround time made a huge difference.

We have an even better example of the value of interactivity with digital cameras. In the bad old days, when we used film cameras, first you took all your photographs. Once the film roll was full, you sent it off to be developed. You'd get your pictures back a while later and look them over. Many weren't quite right, and often you'd curse yourself for messing up a great photo opportunity.

With digital cameras, you take the shot and immediately see how it came out. The faster turnaround allows you to interact with the imagery, try different approaches, and choose the best.

Lesson 10

Fast turnaround is always better than slow turnaround.

Depth

Some of the activities performed on computers are mindless: Searching through a few dozen websites to find a bit of information doesn't take a lot of concentration. A video game might move at a frantic level, but it doesn't reach deep into the most important areas of your brain. Other activities require more mental exertion and hence provide deeper interaction. A game of chess, for example, moves slowly but provides a deeper interaction than a game of tic-tac-toe.

In terms of depth as a factor that determines the degree of interactivity, *deeper* means “penetrating closer to what makes you human.” Computers can easily beat you at tic-tac-toe, but that wouldn't bother you because tic-tac-toe isn't that

important. But what if your girlfriend ran away with a computer? “I’m sorry, Mortimer,” she says, “but you’re just not as exciting, not as sensitive, not as satisfying as my Super iMac. Sure, he’s plain white, but in every way that counts, he’s a real man.” Now that would strike you in the gut! This is an extreme case, but it serves to illustrate what I mean by penetrating closer to what makes you human.

Many dimensions of depth are available to the artist. Games confine themselves to a few of the simplest modalities of human cognition: hand-eye coordination, puzzle solving, spatial reasoning, and resource management are the four primary challenges of games. For interactive storytelling, however, the foremost cognitive modality at play is social reasoning. The infinite complexity of the dynamics of human social relationships gives the interactive storyteller a bottomless well of material; the problem lies in getting some sort of algorithmic grasp on the problem. Reducing social machinations to mathematical form without compromising their richness, however, requires deftly combining artistic insight and mathematical fluency. This topic is addressed in the discussion on personality modeling in Chapter 14, “Personality Models.”

Lesson 11

The overall quality of an interaction depends on its depth as well as its speed.

Choice

Carl von Clausewitz, in his monumental work *On War*, noted that battle is to war as cash payment is to business. A businessperson can make deals, write contracts, design and build products, obtain loans, and arrange foreign exchange, but in the end, cash payment is the decisive point; everything else is merely a preliminary step leading to that moment. A general can obtain weapons, train troops, and maneuver around with clever strategy, but in the end, battle is the deciding moment. The same idea applies to the process of thinking: Choice is to thinking as battle is to war. You can philosophize and deliberate all day long, but the end result of all your mental gymnastics has to be a choice of some sort. Your choice might not seem like much of a choice. (For example, “Do I eat lumpy oatmeal or pickled prunes for breakfast?”) But it’s still a choice, and all your mental processes are geared toward making it, even in the absence of clear information. (For example, “When I hear footsteps behind me in the dark alley, do I run or ignore them?”)

The ability to make choices, along with speed and depth, determines the degree of interactivity. The quality of any interaction depends on the *richness* of choices available to the user. Richness breaks down into two factors:

- ▶ The functional significance of each choice
- ▶ Perceived completeness: the number of choices in relation to the number of possibilities the user can imagine

Functional significance means the degree to which a choice satisfies users' desires, needs, and interests. For example, a word processor could offer a feature that randomly changes fonts and font sizes while typing, but this choice would be useless, so providing it doesn't improve interaction. A better example comes from a hypothetical game that offers the player the opportunity to wander all over a huge region—but nothing interesting happens in the huge region. The poor player wastes hours of time exploring a dead space that offers no further opportunities for interaction. Sure, the game offers zillions of choices in terms of where the player might go, but none of those choices is functionally significant.

Feature bloat is an example of the reverse of this issue. Consider, for example, the Microsoft Word feature that allows you to add borders and shading to a document. I have never used this feature, nor do I expect to ever use it. It therefore represents a choice that has no functional significance to me. From my point of view, this choice is a liability in the program. Every time I consult the Format menu, my eye must glance at this option, and I must make a decision to ignore it. Of course, other users might love the feature, throwing in borders and shading all over the document. For them, this feature doesn't constitute a liability—it offers an additional choice that they find functionally significant. Thus, the determination of what constitutes functional significance is subjective.

For the second factor, the absolute number of choices isn't important; it's the number of choices offered compared to the number of possibilities the user can imagine. If the user has reached the climax of the story and must choose between leaving his girlfriend for the war or shirking his duty, having only two choices doesn't detract from the power of the interaction; it's difficult to imagine any other reasonable possibilities.

This brings me to the most important point in this book.

Crawford's First Law of Software Design:

Always ask, what does the user *do*? What are the verbs?

Every piece of software is defined by the set of verbs available to the user. Those verbs constitute the design skeleton of the software. If I tell you that a secret piece of software permits the user to enter text, modify its font characteristics, and set tabs and margins, you can be pretty certain it's some sort of word processor or publishing program. If I tell you that a secret piece of software permits the user to turn right or left, move forward or backward, run, jump, duck, and fire, you know that it's a first-person shooter game. The verbs define the software and answer the question, "What does the user do?"

So What?

All my ranting and raving about interactivity fails to address an important question raised by my Alter Ego:



Who really cares about interactivity? Why bother with it?

I can offer three reasons for getting on the interactivity bandwagon: It's the medium's basis of competitive advantage, it's revolutionary, and it's powerful.

Incentive 1: Basis of Competitive Advantage

One of the great rules of competitive behavior is to place the competition in the context most advantageous to you. In military science, this rule is expressed as, "Fight on the ground of your own choosing," which means that a general should choose a battlefield best suited to the advantages and disadvantages of his own army. Political pundits always advise their candidates to "fight the campaign on your own issues, not your opponent's." Every MBA quickly learns to identify and exploit a company's "basis of competitive advantage." It's the product or service that the company can supply better than anybody else. Concentrating your efforts on that basis of competitive advantage is the only way to profit.

The computer is a medium of expression, and the artist using this medium must understand its fundamental basis of competitive advantage: interactivity. Computers can do a lot of things well: graphics, animation, music, sound effects, and text. However, other media can do these things better than computers. Sure, a computer can present beautiful images, but a printing press can still deliver better images for less money. A \$10 calendar or a \$20 poster delivers better imagery than a \$1000 computer. If you want animation, you can rent a DVD for a few bucks. Plus, it doesn't take a computer to play that DVD—just a DVD player costing perhaps a tenth as much as the computer. And the cost and quality of the text in a paperback book trumps what you get on a computer. In all these areas, the computer is second best. The computer might be a great development system for creating your masterpiece, but it's never the delivery system of choice—not if you want to get the best possible presentation of your work.

Interactivity is another matter entirely. No other medium can deliver true red-blooded interactivity—not movies, not audio CDs, not DVDs, and certainly not books. When it comes to interactivity, computers are the only game in town. It's the basis of competitive advantage of this medium.

Incentive 2: Revolutionary

Hey, who wants to work in a tired old field like cinema, music, or literature? Interactive storytelling is so new that nobody has any idea of what it is or how it works. If you're the adventurous type, it's the field for you. Besides, opportunities to get in on the ground floor of a new medium don't come along often. Literature was a hot new field about 2500 years ago, and the printing press opened up a lot of opportunities 500 years ago. Movies were young and wild a century ago, radio was young in the 1930s, and television had its heyday in the 1950s. I was in on the ground floor of computer games in the early 1980s, and that was fun, but nowadays it's just another case of Big Media. If you've got the creative itchies, interactive storytelling is the place to be.

Incentive 3: Power

Perhaps you're the kind of artist who lusts for the power to influence people. You have something you want to say to the world, and you don't want to whisper—you want to shout. You want your message to hit people in the gut, to knock their socks off, to take their breath away. Harken back to the earlier section

in this chapter “A Model for Human Understanding.” Remember how I talked about the “Aha!” experience that people get when their network suddenly snaps into a new position? That’s what any great work of art does. The audience for expository art stands at arm’s length from it; people look at paintings, watch movies, read literature, and listen to music, but that’s as close as they can get. Only with additional mental exertion can they reach out to the art. Interactivity draws them right into the middle of the art; they don’t just witness the art, they make it happen themselves. This deeper involvement amplifies the power of the experience. Isn’t that what you want?



Why not simply use the computer to enhance conventional storytelling?

The computer has been used to enhance storytelling for a long time; the movie *Jurassic Park*, for example, couldn’t have been made without computers. Indeed, there’s an entire field of effort known as *digital storytelling* that attends to the task of using computers to present conventional stories. Sure, the computer makes it possible to do the same old stuff faster and cheaper, and that’s great for accountants and beginners. So if you want to use the computer as a tool rather than a medium, be my guest. But if your purpose is primarily artistic, you want to run with the artistic strength of the computer—its interactivity.

Conclusions

Now that I’ve pumped you up with revolutionary fervor, I shall cut you off at the knees with the warning that this revolutionary stuff is tough, sweaty, bloody business. The road to creating interactive storytelling in games is no cakewalk; those who attempt to travel it will face innumerable difficulties. This book doesn’t walk you down that road; it can give you only general guidelines as you move forward. Remember, it’s the choices you offer your player that determine the quality of the interactivity. If those choices permit players to fully engage their personal network of ideas with your own, then you can bestow an “Aha!” experience on them.

CHAPTER 3

Interactive Storytelling

NOW THAT WE HAVE ESTABLISHED some basic concepts of stories and interactivity in Chapters 1 and 2, our task now is to combine the two and ask, “What is interactive storytelling? What lies at the intersection of interactivity and stories?”

The plethora of terms used in discussing interactive storytelling indicates the confusion that surrounds the subject. Over the years, people have used “interactive story,” “interactive storytelling,” “interactive drama,” “interactive narrative,” “interactive fiction,” and “interactive movies” to describe this field. I use “interactive storytelling” because it seems to be the most commonly used term.

Moreover, there is a subtle but important source of confusion. Terms such as “interactive story” are oxymoronic, because a story is data, not process. You can’t interact with data; interaction centers on process. Images, sounds, text, and numbers are all forms of data. You can’t interact with any of these. They don’t *do* anything. You *can* interact with a process because it’s alive—it’s taking place, and you can intervene in that process to change the way it operates, creating an interactive loop with it. A story is data; it’s fixed, permanent, and unchanging, so you can’t interact with it. But *storytelling* is a process, a dynamic process that you can intervene in, alter, play with, and thereby interact with. Therefore, the appropriate term is “interactive storytelling.”



*So what do you call the product that delivers interactive storytelling?
An “interactive storytelling thingamabob”?*

No, I call it an “interactive storyworld.” It’s a universe that contains all the dramatic elements the author wants the player to be able to interact with. A single traversal of this storyworld yields a story. A good storyworld is rich enough to permit many different and interesting traversals.



*Can’t the same thing be said of games? A good game has “replayability.”
The player can play it many times, each time producing an interesting
play experience.*

Yep, that’s true. Let’s look at the kind of story generated in playing a game. It might read like this:

I entered the new level and saw a monster waiting for me. I ducked to the left and then blasted the monster. Another monster saw me and started approaching; I ran behind an obstacle, popped around the other side, and shot the monster. I ran to another obstacle, peeked around it, and saw five monsters. I threw a grenade into them. When it exploded, I dashed forward and mowed down the survivors. But there was another monster waiting just around the corner, and it caught me unaware, wiping out half my hit points. I ran for cover, but he was right on my tail. I couldn’t shake him, and my gun wasn’t big enough to take him out, even though I shot him at least four times. He killed me.

This is unquestionably a story—but is it an interesting story? It’s certainly an interesting experience, but imagine reading a novel that consisted of page after page of this kind of stuff. How many pages would you read before you got bored? Games provide great experiences but lousy stories.

Extrapolation from Games

When people are at a loss to understand a new phenomenon, they fall back on what they already know and describe the mysterious phenomenon in familiar terms. For example, the American Indians of the 19th century saw the railroad as an “iron horse.” Americans of the mid-20th century perceived the computer as a “giant mechanical brain.” In much the same way, people who try to grasp interactive storytelling fall back on games as the closest experience they can imagine. After all, like games, interactive storytelling is played on a computer, is interactive, and is entertaining. But a locomotive is not a horse made of iron, a computer is not a brain made from a machine, and interactive storytelling is not a story made from a game.

I define a game as “a goal-oriented form of interactive entertainment in which one or more active opponents attempt to hinder the player’s attainment of his or her goal.” This definition could apply just as well to almost any interactive storyworld. The game’s opponent is the same as the story’s antagonist. The player in the game is the protagonist in the story. The player has goals, and the opponent acts to hinder those goals. Clearly, in this definition, there’s no difference between a game and an interactive storytelling system.

The real world is a messy, complicated place, and all too often definitions run afoul of ugly reality. It’s easy to accept a mouse, a deer, or a lion as a mammal, but what about platypuses or whales? They’re mammals, too, but they certainly stretch the common notion of a mammal. If you define food as a form of nutrition, does a diet soda with no nutrients constitute food?

The game world is a messy, complicated place, too. The broad definition I offered of a game being “a goal-oriented form of interactive entertainment” is academically correct, but the games that people actually play on computers can

be defined in much narrower terms. Here's a definition for video games as they are actually played:

A form of interactive computer entertainment involving simple and/or violent themes, relying heavily on cosmetic factors, in which players must exercise precise hand-eye coordination, puzzle solving, and resource management skills.

Although this definition doesn't cover all video games, it does a good job of covering most. Perhaps it would be more accurately termed a characterization than a definition.

Now let's compare these basic components of video games with stories, piece by piece.

Simple and/or Violent Themes

The simple or violent themes in our working characterization of games reveals the huge chasm between games and stories. Stories for young children are certainly simple, but the full range of stories is characterized by the complexity of themes and story structure. Even simple movies, such as *The Terminator* or *Godzilla*, have more complex characters and more subtle story structures than the most intricate video game. The level of character development and plot intricacy demanded of stories goes far beyond what games offer.

Relying Heavily on Cosmetic Factors

Like games, cosmetic factors certainly play a role in movies. Lately, computer graphics advances have made wondrous imagery possible. Yet movies that rely exclusively on cosmetic factors always fail at the box office. The 1990s brought some fantastic dinosaur movies, but can you recall any of them other than the *Jurassic Park* series? Movies take advantage of cosmetic factors wherever possible, but they don't rely on cosmetics to sell the product. Cosmetics are a supporting element; the story comes first.

Precise Hand-Eye Coordination

It's true that hand-eye coordination plays a role for some characters in movies; the swordfights of the Jedi knights in the *Star Wars* movies or the feats of any action hero come to mind. Physical prowess, however, never plays a major role

in any genre but action movies, and even those movies often portray physical prowess falling prey to cleverness. The delicate damsel in *The Terminator* eventually crushes the robot, after spending the entire movie running and hiding from him. Luke Skywalker might have used physical prowess to destroy the Death Star in *Star Wars: A New Hope*, but his ultimate triumph came from begging his father, Darth Vader, to help him when the Emperor was torturing him to death. As far as stories are concerned, hand-eye coordination is a supporting element.

Puzzle Solution

Plenty of stories focus on puzzles. All mystery stories are elaborate puzzles, and puzzles often play a role in action movies (again, however, their role is always subordinate to the story). Most games, on the other hand, do include puzzles—and some games revolve around their puzzles. If you forbid game designers to include puzzles in games, the games industry would be crippled. If you forbid Hollywood from including puzzles in movies, it would lose a genre (mysteries), but little more. Moreover, the puzzles in stories are more subtle and personal than those in games. A game puzzle might require you to figure out how to combine a jump with a twist and a rocket launch; a puzzle in a story, meanwhile, is often based on the intricacies of human behavior. For example, Agatha Christie's Poirot mysteries are widely considered to be classics of that genre. In one of these, *The Hollow*, Poirot discovers a clay sculpture of a horse head in the studio of a suspect. Recalling that the suspect had expressed disdain for horses, Poirot realized that the sculpture was not made for artistic reasons. He dug inside and found the murder weapon.

Resource Management Skills

Stories have always played fast and loose with resource management issues. Six-shooters always have more than six shots (except maybe Dirty Harry's—"Did he fire six shots or only five?"). In *The Lord of the Rings*, Aragorn, Legolas, and Gimli run all day long for days on end without food or water. And people never have to stop and go to the bathroom in movies! Yet games impose all manner of resource management problems on players. Stories don't worry themselves with accounting problems (except as a secondary factor accentuating the stress on the protagonist), but game players revel in maintaining exactly the right amount of ammunition, food, health, energy, and so forth as they play.

These observations point to the conclusion that stories and games, as they actually exist, are distant cousins at best. Visualizing interactive storytelling in terms of games is rather like describing a whale by using a camel as a reference. Sure, they're both mammals, but they are so different that the effort is a waste of time and ultimately misleading.

Lesson 12

Interactive storytelling systems are not “games with stories.”

Interactivized Movies

Some game designers have pursued the notion that interactive storytelling is just like the movies, only the player gets to make all the dramatically interesting decisions. Wouldn't it be wonderful to find yourself playing the role of Luke Skywalker in *Star Wars*, facing evil, becoming a Jedi knight, and swashbuckling across the galaxy?

This possibility has inspired a generation of would-be moviemakers, shut out of Hollywood but desperate to prove their talents. The powerful tools the computer makes available to low-budget operations encourage these moviemakers to experiment, and while they're at it, they figure they might as well toss in some interactivity. The inevitable result is what I call an *interactivized movie*: a product that is for all intents and purposes a movie with some interactivity tacked on.

Starting with an idea for a movie and then asking, “How can I add interactivity to this?” is every bit as silly as designing a new kind of computer mouse and then asking, “How can I add video to this?” You can't meld two very different ideas by simply appending one to the other. The challenge is to integrate them, not to bind them together with duct tape.

Plot vs. Interactivity

Comparing plot with interactivity leads many to conclude that there's a fundamental conflict between the two. Academics prefer to use the term “agency” to refer to what most people call “interactivity” and “narrativity” to refer to

what most people would call “storiness” (or “plot”). Andy Cameron of the Hypermedia Research Center at the University of Westminster wrote:

I will argue that there is a central contradiction within the idea of interactive narrative—that narrative form is fundamentally linear and non-interactive. The interactive story implies a form which is not that of narrative.

Three researchers from Drexel University and the University of Central Florida recently wrote:

A key challenge in computer-based interactive narrative is the conflict between user agency and authorial control.

Two researchers from the University of Portugal designed an architecture “to increase the likelihood of the users experiencing the story as intended by the author.”

Some authorities assert that the intersection of interactivity and story is empty. Dr. Andrew Glassner, a noted computer scientist, wrote:

The popularity of the concept of “interactive fiction” for computer-based stories and games is surprising. Is there anything compelling in our cultural history that suggests people want to participate in received stories? Are there stunning examples of successful interactive fictive experiences that have turned doubting Thomases into true believers? No.

It’s the Myth of Interactivity again. As you recall, this myth tells us: Interactivity makes games better, and a game designer should try to make the experience as richly interactive as possible. And what goes for regular games goes for story games. This belief in the universal power of interactivity is what leads people to try to marry interaction and storytelling.

Dr. Glassner is correct in observing that interactive storytelling has never been done before. But then, lots of things had never been done before, such as interactive word processing, interactive database management, interactive spreadsheets, and, of course, the interactive Internet. Until the advent of the computer, rich interactivity could be accomplished only by committing a human being to the process—and, most of the time, the human was too slow or too expensive to make the interaction worthwhile. To dismiss interactive storytelling on the grounds that it hasn’t been done before is to reject the entire basis of the human intellectual adventure.

Interactivity vs. Plot

The difficulty in building interactive storytelling technology doesn't mean that interactive storytelling is impossible. There are theoretically sound reasons for the apparent conflict between interactivity and plot. The problems are best seen from the plot faction's point of view. Plot creation is an enormously difficult task, demanding talent and creative energy. Permitting the grubby-fingered audience to interact with the carefully crafted plot will surely ruin its delicate balance. Knowing how difficult it is to get a plot to work well, writers insist that any audience intrusion into the process yields only garbage. If interactivity requires the audience to involve itself in the direction of the plot, clearly interactivity and plot are incompatible.

Adding to this apparent incompatibility is the attitude of the other side. The protagonists of interactivity tend to take a dim view of plot. The strongest example of this attitude is the possibly apocryphal story about the id Software company and the creation of the game *Doom*. There was some dispute within the organization about the proper role of the story in the game. One faction argued that there should be some story element to tie everything together. The other faction argued that *Doom* was an action game, pure and simple, and that they didn't "need no steinking story." Eventually, the anti-story faction won out, and the losers left the company. So the tale goes. *Doom* went on to become one of the most successful games in history, which confirmed (in the eyes of the gamers) the uselessness of storytelling. As John Carmack, the creator of *Doom*, said "Story in a game is like a story in a porn movie. It's expected to be there, but it's not that important."

So you have an apparent incompatibility between plot and interactivity. It would seem, from both theoretical considerations and many failed attempts, that plot and interaction cannot be reconciled. This, in turn, implies that the dream of interactive storytelling is a chimera.

The belief that narrative and interactivity are mutually exclusive, or at least in fundamental conflict, is well-nigh universal. As I have shown, it's certainly easy to believe; a superficial examination of the issue leads to this conclusion. But the ease of reaching it and the popularity it enjoys do not constitute justification.

Free Will vs. Determinism

The central issue here isn't new. In slightly different terms, some of the brightest minds in human history have struggled with a closely related problem: the classic theological problem of free will versus determinism.

I'll explain the connection using the terminology of Christian theologians who wrestled with the problem for centuries: God is omniscient and omnipotent. Every event that takes place in the universe happens according to His benevolent design. There are apparent evils in the universe, but they are all part of God's greater intentions. But these God-willed events include the actions of people as well as natural phenomena. Thus, a terrible disaster is an "act of God," but so is a murder. So how can human beings have any free will? They are puppets in the hands of an omnipotent God. Our lives and actions are all predetermined in the "plot" that God has written for us. But then the very notion of sin evaporates. Clearly, this is unacceptable.

If, on the other hand, humans did have free will, God would be neither omnipotent nor omniscient. In that case, He wouldn't control or know what we would do. But if He is neither omnipotent nor omniscient, how can He fit any definition of God? Clearly, this is unacceptable.

Thus, free will appears to clash with determinism.

The connection with interactive storytelling should be obvious. Determinism in theology is analogous to plot in storytelling. The plot is the storyteller's predestined plan for the story's outcome. Free will is analogous to interaction, for how else can players interact without exercising their free will? To make the theological analogy more explicit, view the story author as the creator of a miniature universe. The author, for example, creates an imaginary universe populated by characters. Like some omnipotent god, the author decides their actions and predestines their fates. Conversely, the history of the universe is nothing more than a huge story written by God that we act out.



Hold it a second! The game designer is also a god of sorts. He too creates a tiny universe and exercises godlike control over that universe. Yet free will surely exists in the game universe.

Exactimosso! Therein lies our answer: The game designer never specifies the events that the player experiences; instead, the game designer specifies the processes that enable and constrain the player. The game designer doesn't specify what path the player takes to get to the victory condition, only the rules by which the player can attempt to do so. The gameworld is an orderly place with free will for the player. That's because the designer's control is exercised through the processes of the gameworld rather than the events of the gameworld. That's the resolution of the dilemma of plot versus interactivity. A plot specifies events, not processes. Interactivity requires that authorial control be exercised through processes, not events. Yes, plot and interactivity are mutually exclusive; but process-driven narrative is perfectly compatible with interactivity.

By the way, Christian theologians settled on this solution 750 years ago. Someday interactive storytelling experts will catch up, I'm sure.

Lesson 13

There is no conflict between process-driven narrative and interactivity!

Interactive Storytelling Is Unique

The conclusion you must draw from these considerations is that interactive storytelling is a whole 'nuther ball game, a horse of a different color, a rare bird, a queer duck, an eccentric emu. You cannot apply tried-and-true expertise (from games or from stories) to this unprecedented medium. In fact, the tried-and-true expertise that storytellers and game designers bring to interactive storytelling only obscures their vision. You're navigating through uncharted waters; the rules that work so well in familiar waters simply don't work here.

So it's back to basics—all the way back to the absolute fundamentals. Then you'll work your way forward from there. The most fundamental rule of interactivity is Crawford's First Law of Software Design: Ask, "What does the user *do*? What are the verbs?" So ask yourself, "What should the user of an interactive storyworld be able to do?"

The answer is obvious: The user should be able to make *dramatically* significant decisions. Some designers stuff in decisions involving spatial reasoning ("Should

I stay at this corner of the building or run to that truck?”). Some designers toss in decisions involving resource allocation (“Should I use up some grenades here or just wade in with my sword?”). Some designers throw in a puzzle or two. None of these decisions are dramatically significant; they add nothing to the storyworld. Yes, they’re interactive. No, they’re not interactive storytelling. Adding green to a greenhouse doesn’t make it better; adding irrelevant interactivity to interactive storytelling doesn’t make it better. The interactivity is not independent and separable from the storytelling; the two must be in harmony to be effective.

Second, the decisions presented to the player must be balanced. If you present the player with a choice between eating an onion and saving the princess from the dragon, the choice is too obvious to be meaningful. The choices you offer the player need not be balanced on a razor’s edge; they could be somewhat lopsided without detracting from their dramatic value. But seriously lopsided decisions are either boring or stupid puzzles. In the onion-versus-princess decision, a blockheaded author might rig the storyworld so that the only way to defeat the dragon is to overcome him with onion breath—in which case, eating the onion is the correct choice. That choice, however, is not a dramatically realistic choice; it’s just a ridiculous puzzle.

This realization creates a conflict for traditional storytellers, for whom a fundamental rule is to contrive circumstances to make each character’s decisions seem obvious and natural. Luke Skywalker meets Obi-Wan Kenobi only because R2D2 runs off; he races back only because, on their way to Mos Eisley spaceport, they just happen to run across the Jawas who were killed by Imperial Storm Troopers. Luke agrees to accompany Obi-Wan to Alderaan only because he discovered that his aunt and uncle were dead.

Contrivances like these are necessary and universal in conventional storytelling; great storytellers make them seem perfectly natural. The interactive storyteller, however, must not concoct lopsided decisions like these; instead, they must be closer to evenly balanced.

Third, the storyworld must present the player with many decisions to make, most of which, necessarily, will be small. If the player careens from one life-or-death decision to the next every 60 seconds, the storyworld will seem more like a comic book than a graphic novel. Therefore, the frequent decisions must be smaller and less impactful. Conversations provide us with the perfect arena for

such decisions. A rich conversational system (not the primitive trees now common in games) permits the player to develop his relationships with other characters and gently push dramatic progress.

The player should be faced with decisions that are dramatically significant, balanced, and frequent (**Figure 3.1**).

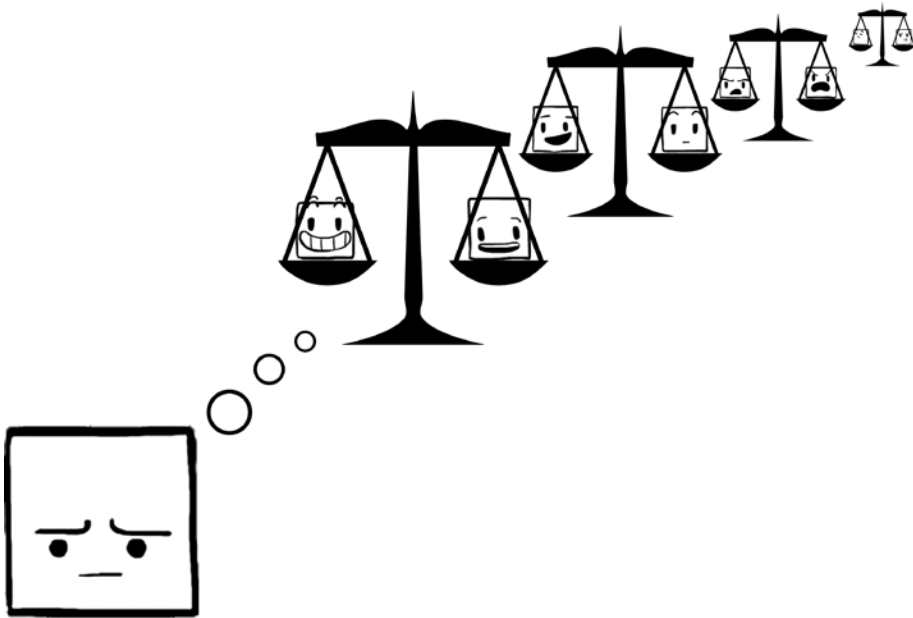


FIGURE 3.1 Balancing player decisions

The creator of a story sweats the dramatically important moments; everything else in the story serves only to prepare the ground for those big moments. But interactive storytelling is completely different: It is the player, not the author, who prepares the ground for the big dramatic developments that flow naturally from the dramatic context the player has developed. We don't give the player a choice between betraying their lover by taking the money or leaving the money (thereby cementing the relationship with the lover). That's too simple and too brief to be interesting. Instead, we offer the player small verbs that cumulatively create the circumstances for a dramatically satisfying ending.

Remember mental modules from Chapter 1? One of those mental modules is social reasoning—the ability to figure out how people will respond to one another’s actions. It’s a difficult skill to develop, and females tend to be better at it than males. After all, little girls play with dolls while little boys play with trucks and guns. Which gender is doing more to learn social reasoning?

The Role of Social Reasoning

Anyway, social reasoning never concentrates on a single decisive action; personal relationships are built on a lengthy sequence of small steps. One way to learn about social reasoning is to read one of Jane Austen’s books. These are lengthy explanations of the intricate processes by which a female determines and achieves the monogamous relationship best suited to her needs and desires. Compile a list of the verbs Austen uses. You’ll find that individually the verbs seem inconsequential, but cumulatively they add up to big results. In *Pride and Prejudice*, Elizabeth hits the jackpot by snaring Darcy, the rich, handsome, and virtuous aristocrat. How she achieves this is the content of the novel.

The verbs used in interactive storytelling will be the stuff of everyday interpersonal interactions, including elements of conversation:

- ▶ Declaring one’s feelings to another (“I think Marcie is cool.”)
- ▶ Describing a third person’s feelings toward somebody (“Well, Darla doesn’t like Marcie.”)
- ▶ Relating events that reveal personality traits or feelings (“The other day Darla took Marcie’s name off the invitation list.”)
- ▶ Lying about such events (“Who me? No I wasn’t the one who took your name off the invitation list!”)
- ▶ Revealing secrets about such events (“That’s what she said, but I saw her taking your name off the invitation list.”)
- ▶ Contradicting or questioning statements made by the interlocutor (“I don’t think Darla would stoop that low.”)

At a higher level will be verbs involving the following:

- ▶ Deals (“If I agree to put Marcie back on the invitation list, will you tell her to mind her behavior?”)
- ▶ Promises (“I promise to be nice to Darla.”)
- ▶ Threats (“If you insult me at the party, I’ll throw you out right then and there!”)
- ▶ Demands (“You just stay away from me!”)

The verbs help drive the story.

Lesson 14

A storyworld does not offer the player a few momentous decisions; instead, it offers many small decisions.

A fourth element of interactive storytelling is that the decisions must permit unsatisfying outcomes. Once again, all the instincts of conventional storytellers rebel at this; they believe that every story must conclude in a satisfying manner. This is true of conventional stories, but quite false when applied to interactive storytelling. For example, suppose we have a romantic storyworld in which Romeo and Juliet are intended to marry. Suppose that our player Romeo decides that he wants to kill Juliet. We cannot have this; he can’t marry her if she’s dead! He pulls out his six-gun and blasts away; miraculously, every bullet misses! He grabs his knife and aims to stab her in the heart. Miraculously, the silver pendant he gave her deflects the blow and causes the knife to bounce away, where it falls into a crack and is irretrievable. He seizes her by the neck, intending to strangle her, but a sharp edge on her necklace catches and cuts his hand, causing him to recoil in pain...

This is ridiculous. If the player wants to deviate from the plot, let him deviate! The mistake lies in setting up a plot in the first place. A romantic storyworld is not about a happy ending—it’s about *romance*! Romance includes all the variations: happy endings, tragic endings, angry endings, funny endings. Therefore, a storyworld about romance must address all these possibilities.



That's absurd! How can any human possibly include so much material in a single storyworld? Even the greatest stories address only some small fraction of the human condition.

That's why you shouldn't attempt a storyworld about anything so large as romance. Instead, how about a storyworld about two lovers caught in a rivalry between their families? Think of all the different turns Shakespeare's play, *Romeo and Juliet*, could have taken. Perhaps Romeo tries to forge a peace with the Capulets; perhaps he succeeds and happily marries Juliet. Perhaps he is killed, and Juliet kills herself. Perhaps Romeo successfully elopes with Juliet, but the happy couple is then caught by her father's minions, who kill Romeo. Perhaps, perhaps...there are so many possibilities. If traditional storytellers can concoct so many contrivances to make a character's decisions appear necessary, why can't you concoct variations on a narrow central theme?



How can such a storyworld have any consistent message? If Romeo can be a cad in one story, a greedy jerk in another, a coward in a third, and so on, how can any story from that storyworld have any meaning?

If some of the pixels in an image are black and some are white, how can an image be coherent? Must all the pixels be white? Of course not! The shadow reveals as much as the light. To truly understand something, we must learn both what is true and what is false. If the player as Romeo behaves as a cad, a jerk, or a coward, then the storyworld should show the consequences of such choices.



But it would be too frustrating to fail repeatedly!

A Look from Different Angles

If a writer using a word processor is dissatisfied with the book she's writing, does she blame the word processor? If a spreadsheet shows the entrepreneur that the start-up will fail, does that mean that the spreadsheet is a bad program? The whole point and purpose of software is to enable you to try different things safely, to see what happens. Interactivity allows you to turn an idea over and over in your hands, to see it from different angles. **Figure 3.2** illustrates this concept.

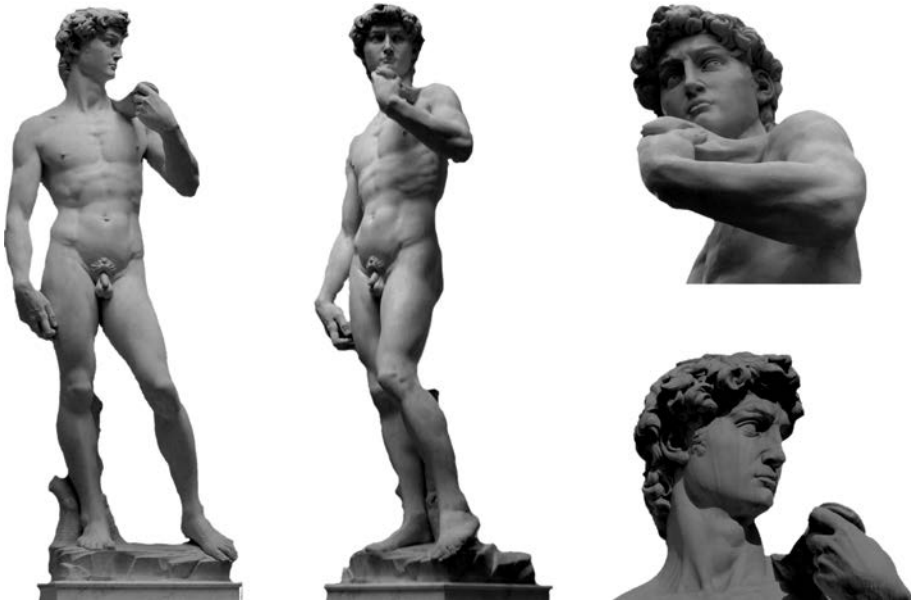


FIGURE 3.2 Statue of David from different angles

This is the statue of David by Michelangelo; it is one of the most photographed statues in the world. The view on the left is what you see as you walk down the hall toward the statue; it is arguably the best angle from which to view it, and it is certainly the most commonly shown angle. But the rotunda permits you to walk around David, viewing him from different angles. Why would the conservators permit that if the best view is to be had at a single angle? The view of his behind doesn't inspire us, so why do the conservators permit it?

Lesson 15

A story presents a single, picture-perfect view of the truth; a storyworld presents truth in three dimensions, including the less elegant angles.

Individuation

Consider that you are one of seven billion human beings, a tiny ant in a huge colony of clambering creatures. What makes you worthy as an individual? You certainly can't base your sense of self-worth on being the best at any single activity; there's nothing you can do that can't be done better by somebody else. Do you like to design games? Think about how many game designers are out there and how low you rank compared to some of them. Do you play tennis? Good for you. Now imagine how many tennis players could beat you with one hand tied behind their backs. The same thing goes for cooking, photography, programming, gardening, or anything else you choose.

Have I made you feel like worm snot? Do you realize just how miserably mediocre you are? There isn't anything special about you; you're just one more petty creature with an overblown sense of your own importance. Right?



Don't you think that calling your readers "worm snot" is a little over the top?

I'm just getting to the punch line. Consider that when you play tennis, you do so with your own special style. That style is unique to you; nobody plays tennis the way you play tennis. It may not be the best way to play tennis, but it's your own personal style. You may have some fast moves or some powerful strokes or a certain elegance in the way you play. When you play tennis, you show the whole world that you're a unique individual.

The same thing goes for anything you take pride in. Your linguini is like nothing else in the world; your writing has a tone that nobody can match; when you belt out a song, the key may be wrong, but nobody can match the feelings you put into your singing. The things we do for enjoyment are always things that permit us to individuate ourselves.

This law of human behavior applies to interactive storytelling. A common mistake among storyworld authors is the intent to entertain the player with a wonderful story. That's great if you're writing a novel or making a movie—but it's wrong for interactive storytelling. The spotlight in interactive storytelling is on the player, not the author. If you try to hog the limelight as the author, you may produce a great non-interactive story, but it'll be lousy interactive storytelling.

Good interactive storytelling exalts the player's ability to individuate himself or herself by playing with a particular style. This means that the interactions must be numerous. Players need plenty of opportunities to establish their own identity.

What Can't Be Part of Interactive Storytelling?

I have emphasized that interactive storytelling is quite different from conventional storytelling. In the previous pages, I have described some of the factors that characterize interactive storytelling that don't appear in traditional storytelling. This sword cuts both ways: Some elements of traditional storytelling—including alternative options, creative options, and real-time play—cannot be used in interactive storytelling.

No Third Option

For example, in interactive storytelling, you can't have what I call the "Third Option"—the protagonist's stroke of genius that gives him a third option for escaping the sharp horns of a dilemma.

Here's a somewhat obscure example requiring some explanation. In the movie *Bless the Child*, a devil-worshiping cultist accosts a God-fearing little girl on the roof of a high building. He taunts her faith in God and forces a test upon her. Placing her on the ledge of the roof, he demands that she prove her faith. She must leap off the roof and trust God to save her, or she must reject God and declare her faith in the devil by taking the hand of the cultist. A dark shroud of menace hangs over the scene; neither the audience nor the little girl can guess whether the cultist will throw her off the roof if she refuses to make the impossible choice he is forcing upon her. After a nerve-racking pause, the little girl turns to the cultist, smiles, and says invitingly "After you!"

This scene actually contains two important lessons about interactive storytelling. First, consider the long logical train her response triggers. She is suggesting that the cultist should prove his own faith in his own god (the devil) by making exactly the same leap he's trying to force on her. If the cultist is unwilling to leap off the building, how can he ask the girl to do so? He is revealed as a hypocrite, and his either-or choice is ruined. It's a complex logical twist that comes as a

complete surprise to the audience. I think we can discount the possibility of a computer coming up with a similar option. It's simply too hairy and requires too deep an understanding of human nature. Perhaps next century.

Second, even if an interactive storytelling system could concoct such a stroke of genius, it couldn't offer it to the player. If the system were to present the girl's choices in that form, the list would read:

1. Leap off the building and hope God saves you.
2. Reject God and embrace the devil.
3. Say "After you!"

This list doesn't present a true choice; only the third option is viable. This scene had great dramatic power in the movie, but it would fall flat on its face in an interactive storyworld. The audience is surprised and entertained by that third option because they couldn't have anticipated it. Suppose you restaged the scene, replacing the vulnerable little girl with, say, Arnold Schwarzenegger. Sure, the cultist might give Arnold only the two choices, but any audience could anticipate that Arnold would simply shout, "Hasta la vista!" and mow down the cultist. No surprise there, and no drama in the scene.

Creative Option

A variation on the Third Option is the "Creative Option." One early tester of my interactive storytelling technology complained that it didn't offer creative options. When I asked what that meant, she could say only that she wanted to be able to do things I might not have thought of. The problem is, how can I provide software to handle situations I haven't thought of? There's something of a catch-22 here.

Real-Time Play

An interesting design question concerns the flow of time in the storyworld. The conventional gamer expects real-time play. Time flows at a uniform rate; if the player is slow to react to the situation, he might let an important opportunity slip past. The alternative to this is turn-sequenced play—the player considers the options and play resumes only after the player has made a choice.

There have been some attempts at using unbroken time flow in interactive storytelling. The first I am aware of was *Mixed Emotions*, by Rosa Freitag, produced in 1995. This product uses continuous video to tell the story of a woman facing problems in her marriage. During the course of the story, she occasionally pauses to think, at which time two icons appear over her head, one blue and the other red. The blue icon always indicates a morally conservative course of action; the red icon represents an adventurous or risky course of action. The player can click one of these icons to guide the protagonist through the story. If the player doesn't select an icon within a few seconds, the story proceeds using the blue icon as the default choice. The system did succeed in presenting real-time interactive storytelling, but only because the choices were so simple.

A more ambitious effort was *Façade*, by Michael Mateas and Andrew Stern. This groundbreaking storyworld was the first genuine, honest-to-gum working storyworld. It also used real-time play.

My preference is for turn-sequenced play; I believe that it is necessary to permit the player the opportunity to assess the situation and weigh the options. With real-time play, the player never knows how much time is available to make a decision; consequently, real-time play has a rushed, frantic feel that I find inappropriate for dramatic interaction. This is another difference between video games and interactive storytelling.

Conclusions

Here are some of the important lessons to remember from this chapter:

- ▶ Interactive storyworlds are not “games with stories added.”
- ▶ Interactive storyworlds are not “movies with interaction added.”
- ▶ There's a fundamental conflict between plot and interactivity
- ▶ There's no conflict between process-driven narrative and interactivity.
- ▶ Interactive storyworlds rely on lots of small decisions rather than a few big decisions.

PART II

Mentalities

THE WAY WE THINK controls our ideas about interactive storytelling; our mentalities bias and limit our ability to comprehend this new and alien concept.

Chapter 4: We must learn to think about storytelling in a more abstract manner.

Chapter 5: We must learn to think in terms of action, as expressed through verbs.

Chapter 6: We must learn to use mathematics to design our storyworlds.

Chapter 7: We must overcome the Two Cultures prejudice that divides science and math from art.

This page intentionally left blank

CHAPTER 4

Abstraction

THE PROBLEM OF PLOT VERSUS interactivity, discussed in Chapter 3, sometimes takes another form: control versus interactivity. In its simplest form, the problem is phrased as follows:

If the story is to be truly interactive, then the player must be able to change the story. But, if the player changes the story, then the artist cannot control its development and the player will likely ruin the story.

The solution to the problem is, in a word, abstraction. Let's look at some examples of how abstraction has solved some analogous problems.

Justice

The problem of exercising control over a complex system is an old one, and in every case, we find that the solution has always been to resort to higher levels of abstraction. For example, let's consider the problem of providing justice to society. In small societies, justice can be provided by a single chief. The disputants present their case to the chief, who hears each side and then pronounces judgment. All very simple. But as societies grow, the number of cases grows and the chief—now known as king—finds himself overwhelmed with an impossible caseload. The solution is to delegate his powers of adjudication to lower officials called judges. This solution, however, creates a new problem: How is the chief to maintain control over the judges to ensure justice?

The solution is law. The chief declares the rules under which the judges will operate, and those rules are then applied to all concerned. The solution is not perfect. Judges can still apply the laws unevenly (if they are biased), misinterpret the laws, or even apply the wrong law to a given situation. But disputants who believe they have been shortchanged can appeal to the king in hope of a correct application of the law.

In this manner, the king can continue to exercise control over the society. The king gives up direct control of justice for indirect control. The king's power is exerted through the laws that are applied by the judges. The ultimate power still resides with the king, whose laws control every aspect of daily life. The king may not be looking over the shoulder of each of his subjects, but he still retains indirect control.

Conflicting Laws

The problem with the use of law is not so much the application of law as the formulation of law. The king must bring great insight and wisdom to bear in creating his laws. Here's an example from the first recorded set of laws. Nearly 4000 years ago, Hammurabi of Babylonia promulgated a set of 281 laws. Here are two of them:

- ▶ If anyone steals the property of a temple or of the court, he shall be put to death, and also the one who receives the thing from him shall be put to death.

- ▶ If anyone steals cattle or sheep, or ass, or pig, or goat, if it belong to a god or to the court, the thief shall pay thirtyfold for it; if they belonged to a freed man, he shall pay tenfold; if the thief has nothing with which to pay, he shall be put to death.

These two laws contradict each other; one says that the punishment is death, the other that it is 30 times the value of the stolen item. During the process of compiling 281 laws, Hammurabi forgot about the first law when he wrote the second law. It's important that laws not contradict each other. The larger the body of law, however, the harder it is to be certain that a new law doesn't conflict with an existing law.

All-Encompassing Laws

Another problem in formulating laws is that they must properly address every possible situation in which they might be applied. There's an old, apocryphal story about the early days of the Environmental Protection Act. A provision of the law was that if a company wanted to build a factory in an air pollution administrative district, it had to compensate for any pollution the factory would make by paying another polluting company to install equipment that would reduce its pollution by the same amount that the first business was adding. In other words, if you're going to emit pollution, you've got to do something to cancel it out.

Sounds like a good idea, doesn't it? The law doesn't punish owners of old polluting factories; it requires builders of new factories to install the equipment to reduce pollution in the older factories. No new pollution is added to the air district.

Then one day a businessman in Louisiana decided that he wanted to build a furniture factory. Unfortunately, his factory would emit a small amount of heavy hydrocarbons from the varnish and other coatings necessary to manufacture the furniture. So he searched for another factory in the same air district that he could upgrade. There were no other factories because it was a poor region with no industry—but the law must be obeyed. So when some bright fellow pointed out that trees emit heavy hydrocarbons just like varnish, the businessman had his solution. He bought a large tract of forestland and cut down all the trees. Problem solved.

Obviously, it was not the intent of the U.S. Congress that forests would be cut down in the name of the Environmental Protection Act, but the wording of the law wasn't careful enough to provide for this situation.

This problem of abstracting reality to terms that can be addressed in a law haunts all lawmaking processes. Courts will sometimes annul laws that are so poorly worded that they permit entirely unintended results. It's a tough problem.

The same thing applies to interactive storytelling: If your algorithm doesn't consider all the possibilities, you'll get a nasty screwup.

Science

Let us now turn to the biggest, most complex system of all: reality. Our efforts to understand reality have yielded an increasingly complex intellectual system: science. At first, science was a great mass of disconnected tidbits of information—essentially, random fragments of truth that people had noticed over the generations. The big step forward was the realization that different tidbits of information could be connected together to form a more coherent, albeit more abstract, whole.

For example, people had long been aware that some stars in the sky moved in relation to the other stars. Some moved faster and some moved slower, and they all seemed to move along the same path. Ptolemy assembled these disparate facts into a coherent whole and suggested that these objects (the planets) circled around the earth. This was an abstraction; nobody could actually see the evidence directly. But this more abstract view of the solar system tied together a great many observations and, therefore, provided a better explanation.

That trick—finding an abstraction that ties together simpler truths—has been the basic strategy of science ever since. Chemistry stumbled forward, building up a mass of knowledge about how various chemicals reacted with each other, until the idea of atoms forming molecules took hold and explained it all in a more powerful, more unifying, and more abstract fashion. With the development of the theory of quantum mechanics, scientists were able to explain the mechanics of chemical reactions, thereby obtaining a more broadly encompassing view of chemistry—at the expense of abstractions that are more difficult for mere humans to understand. With the development of biochemistry, chemistry and biology were merged at a

fundamental level that gave us insights into the genetic processes that govern all living systems—but again, the price we paid was greater abstraction.

Meanwhile the physicists continued their search for the fundamental laws of the universe. Isaac Newton unified all motion under three simple laws. James Clerk Maxwell unified electric theory with magnetic theory and light to demonstrate that all light is an electromagnetic wave. Einstein unified electricity with magnetism by showing that magnetism is a relativistic effect of electricity. Chemistry was further unified when physicists showed that all atoms are combinations of electrons, protons, and neutrons. In probing the nature of these three particles, physicists discovered a whole zoo of new particles—and then reduced them further by showing them to be composed of yet more fundamental (and even weirder and more abstract) particles called quarks. Meanwhile, theoreticians struggled to reduce the universe to its most fundamental constants and equations, making slow and jerky progress. The end result of all these labors is a highly abstract system of ideas that explain the workings of the universe—but this system is so abstract that it lies beyond the reach of all but a few specialists.

Mathematics followed a similar course. Starting with simple counting combined with simple addition and subtraction, people developed the ideas of multiplication and division, and arithmetic was born. Algebra, a more abstract approach to mathematics, came next. Then, in the 17th century, mathematics exploded with analytic geometry, calculus, probability, and so forth. Nowadays, of course, mathematics has reached levels of abstraction utterly beyond everybody but the specialists.

In science, we see the same process that we saw in justice: As our knowledge expanded, we had to resort to ever more abstract ideas to stay on top of all that knowledge.

Finance

The earliest economic interactions were direct barter: I'll give you my goat for ten bushels of your wheat. Because such transactions are as explicit and direct as possible, they were easy to evaluate and police. One party examines the goat and the other party examines the wheat. They each see what they're getting and there is no further cause for interaction after the exchange is made. Thus, the earliest forms of financial interaction were direct, explicit, and utterly devoid of abstraction.

This was sometimes clumsy: What if you were willing to sell only five bushels of wheat? I couldn't sell you half a goat. This problem was solved by stepping up to a higher level of abstraction: the use of precious metals. I'll sell you my goat for three shekels of silver. A shekel, by the way, was originally a unit of weight, not a coin.

Problems then arose when people cheated by trying to pass off these precursors to coins that really didn't contain as much precious metal as they were supposed to contain. This required a new jump to a higher level of abstraction: the standardized coin. This was a chunk of precious metal that was manufactured by the state and guaranteed to be of pure metal and the correct weight. The proof of the value of the coin was the face of the king who issued it: "King So_And_So the Great guarantees that this coin is trustworthy."

Note how coinage moved the economy to a higher level of abstraction. At that point, all fiscal accounts were kept in the otherwise arbitrary units of coinage. A rich man might be worth 10,000 sesterces. What does that mean? It's an even more abstract concept.

Coinage worked well for a long time, until another new problem arose in the Middle Ages. Traders were traveling over longer distances, but the roads weren't particularly safe. Traveling across Europe carrying a few pounds of gold was an open invitation to highway robbery. Merchants needed a safe way to handle their transactions. Thus was born a new and even higher level of abstraction: the bill of exchange.

This was nothing more than a note saying "Joe the Merchant authorizes Fred the Banker to transfer 50 ducats from Joe's account to the account of Charlie the Merchant." In other words, the bill of exchange was basically the same thing as an endorsed check. This meant that merchants could carry around pieces of paper rather than coins, making them robber-proof.

This was an immensely important leap in abstraction because, for the first time, the concept of value was divorced from a tangible object. Instead of transferring wealth through tangible intermediaries such as precious metals, wealth could now be transferred through a piece of paper. This represented a higher level of abstraction of the concept of wealth.

These bills of exchange stimulated the European economy and pretty soon they were flying all over Europe. Keeping track of them was becoming quite a hassle.

If a bill of exchange bounced (due to insufficient funds), it was a huge hassle to track down the deadbeat and get your money.

Banks solved this problem by issuing generic bills of exchange. These were notes from a bank promising that the bank itself would pay the bearer a standard amount of coin. There were notes for one ducat, five ducats, ten ducats, and so forth. You gave your coins to the bank, and they gave you bank notes equivalent in value (after bank charges) to your coins. Then you paid other merchants with the bank notes. The merchants didn't have to trust you; they had only to trust the bank that issued the note.

Of course, there was even more abstraction involved in this solution: Now the value existed in a bank that didn't actually have the money. The prudent banker loaned out most of its money to get interest, and kept only a small amount in reserve to handle payments. So now you had to think in terms of value that resided in a bank and value that consisted of all the bank's assets, most of which weren't even coins.

This worked well, but there was still a problem: Sometimes banks would go bankrupt and the people holding all the banknotes from that bank lost their money. So then governments stepped in and established national banks, which were guaranteed not to go bankrupt. The banknotes issued by those national banks are what we now call paper currency. If you read your dollar bill carefully, you'll see that it's really a "Federal Reserve Note"—a banknote from the Federal Reserve Bank.

This idea of a national bank pushed the abstraction up another notch. As you might guess, a new problem arose: Those paper notes could be stolen just like the coins. So people learned to keep most of their money in banks and use banknotes only for "walking around money." This pushed the abstraction even further: Now your wealth wasn't measured by the banknotes in your possession, but by a number in the bank's records for your account. You could have a million dollars in the bank but only a hundred dollars in your pocket. So now money had become *really* abstract—just a number stored in a book somewhere.

The next step in the process was the credit card. This is nothing more than a piece of plastic with your account number on it. You charge something to your credit card, and your signature constitutes a promise to pay that amount. The bank that issued your credit card pays the merchant and deducts that amount from your credit card account.

Let's trace the levels of abstraction at work here: Your credit card points to your credit card account, which you pay off with money from your regular bank account. Your regular bank account contains a number that represents the amount of money that you have deposited with the bank. The money represents some amount of precious metal—well, it used to do that—which in turn represents a concept called wealth. Whew! That's a lot of indirection!

Thus, as economies grew larger and more complex, the financial system underlying them grew more and more abstract.

The Moral of These Stories

Over and over we see the same idea: To grow intellectually, and to understand and cope with more complex problems, we always move to higher levels of abstraction. This can be summarized in a simple lesson.

Lesson 16

When you can't work through a problem, go over its head.

Recall that in Chapter 3, "Interactive Storytelling," Lesson 13 resolves the dilemma of control versus interactivity: "There is no conflict between process-driven narrative and interactivity." The solution to the dilemma is to exert control at a higher level of abstraction. As with all the examples, that abstraction will be more difficult to understand, but it will extend our intellectual reach. Many storytellers, locked in the traditions of conventional storytelling, will be unable (and perhaps unwilling) to grasp such abstractions and will reject the whole concept. No matter—there will always be plenty of room in this world for traditional stories. But interactive storytelling demands that we move to a higher level of abstraction.

To understand the abstractions presented in this book, you must first let go of the very notion of plot. A plot is a fixed sequence of events that communicates some larger message about the human condition. In interactive storytelling, we replace the plot with a web of possibilities that comprise the same truth. Since this concept confuses most people, let's look at a number of examples at different levels.

Let's use the classic movie *Star Wars: A New Hope* as our starting point. Here is the direct representation (sequence) of the story:

1. Luke Skywalker leaves home, meets Obi-Wan, travels with him to Mos Eisley spaceport, and flies away in a spaceship. But their ship is captured by the bad guys; they fool the bad guys, rescue Princess Leia, and escape from the Death Star. The bad guys attack the last bastion of the rebels, and Luke helps to attack the Death Star. Luke destroys the Death Star and gets rewarded by the princess.

Now let's look at the same story in a more abstract fashion:

2. A young man ventures out into the world, makes new friends, and experiences many adventures. He learns much and triumphs over adversity, winning the admiration of a pretty girl.

Now let's make it even more abstract:

3. A boy confronts the challenge of growing up to become a man. He faces many difficulties, but ultimately triumphs over adversity and establishes his manhood.

It is at these higher levels of abstraction that we design storyworlds. Instead of thinking about Luke Skywalker, we think about a young man—any young man. Instead of thinking about space combat, we think about adventures—many different kinds of adventures. Instead of thinking about blowing up the Death Star, we think about triumphing over adversity.

We don't assemble storyworlds event by event. Instead, our high-level design requires us to construct storyworlds concept by concept. There aren't many variations of "rescue Princess Leia," but there are millions of versions of "faces many difficulties." Put another way, there is only one version of the first story, but there are thousands of versions of the second story and millions of versions of the third story. For now, the third level of the story is too abstract. But, if you think in terms of the second level, then you can design a storyworld that can generate thousands of different stories. Remember, there's no such thing as an interactive story; if it's already a story, then it has been nailed down and you can't interact with it. Instead, we're pursuing the concept of interactive *storytelling*, which springs from a storyworld—and a storyworld is what you create, not a story.

As long as you think in terms of a strict sequence of events (a plot), then you'll never understand the concept of interactive storytelling. Instead, you must think in terms of an entire dramatic universe of potential stories: a storyworld.



Can you be more specific?

Sure. Consider these sentences from different stories:

“You look magnificent, darling!”

“I don’t think I’ve ever seen anybody as beautiful as you.”

“You’re such a dashing fellow!”

“Ooh! You’re so strong and brave!”

I’m sure that you’ve read thousands of variations on these sentences, all of them cleverer than these. They all boil down to this single statement:

“I compliment you.”



Yuck! The abstract version is cold, lifeless, and utterly mechanical. You can’t seriously propose that we present something like this to our audience!

No, I’m not proposing that we present “something like this” to our audience. Instead, we have to use one of several strategies for presenting the idea behind it. Internally, our interactive storytelling engine will still be thinking in terms of “I compliment you,” but the presentation of that idea can be dressed up into a more palatable form. (I explain how to do this in Chapter 18, “Scripting Languages.”)

Playing God

The same concept of resorting to a higher level of abstraction provides a partial solution to the knotty problem of free will versus determinism discussed in Chapter 3. This solution embraces physics and rationalizes faith. It says that God is omnipotent with respect to process, not data. That is, God controls the universe through its laws, but not through the details. God does not dictate the position and velocity of every electron and proton in the universe; instead, he

merely declares, “Let there be physics” and then allows the clockwork of the universe to run according to his laws. In an indirect way, we could say that he does control everything that happens in the universe, but it is abstract control. God determines the principles under which the universe operates, but grants us free will to choose as we wish within that universe. He even works a little randomness—in the form of quantum mechanics—into the system to ensure that we aren’t automatons responding robot-like to our environments. The important point is this: God is an abstract designer!

The king who formulates laws is controlling his kingdom in a similar way. He doesn’t wander through the kingdom, ordering people to reap this crop, milk those cows, or build that house. Instead, he creates a set of rules constraining their behavior. His rule can be benevolent or harsh, but it is always abstract and indirect.

And the same resolution works with the apparent conflict between plot and interactivity. If you are a data-intensive designer, then you are a deterministic one. Like some Bible-thumping fundamentalist, you insist that every single word you write be obeyed literally by the characters in the story. The fundamentalist focuses all his or her beliefs in the explicit data of the Bible rather than the abstract processes behind it.

But if you are a process-intensive designer like God, then the characters in your universe can have free will within the confines of your laws of “dramatic physics.” You must abandon the self-indulgence of direct control and instead rely on indirect, abstract control. That is, instead of specifying the data of the plotline, you must specify the processes of the dramatic conflict. Instead of defining who does what to whom, you must define how people can do different things to each other.



This is too esoteric, too indirect to allow the richness of tone that a good story requires.

True, but consider what a story really communicates. A story is an instance that communicates a principle. Moby Dick is not about a whale; it is about obsession. Luke Skywalker never really existed; the truths about growing up and facing the challenges of manhood are the movie’s real messages. Stories are false at the direct level yet true at the abstract level. The instances they relate never happened, but the abstract principles they embody are the truth that we appreciate. They are false in their data but true in their process.

Given this, consider the nature of the communication between storyteller and audience. The storyteller seeks to communicate some truth, some principle of the human condition. Rather than communicate the truth itself, he creates a particular set of circumstances that instantiate the truth he seeks to communicate. This instantiation is what he communicates to his audience. The audience then interprets the story; it figures out the higher, more abstract principles from the story's details. Note, however, the circumlocution of this process. The storyteller seeks to communicate some truth of the human condition; the audience seeks to learn the same. Instead of just telling the principle, the storyteller translates the principle into an instantiation, then communicates the instantiation; the audience then translates the instantiation back into a principle. This is truly a roundabout way to get the job done.

Interactive storytelling alters this process in two important ways. First, the process of translating principle into instance is delegated to the computer. The storyteller retains full artistic control, but now she must exercise that control at a more abstract level. The basic process of translating principle into instance is retained, but is now performed by the computer in partnership with the player. This, of course, entails considerable effort in algorithm creation. The second difference is that, because the story is generated in direct response to the player's actions, the resultant story is customized to the needs and interests of the player. Thereby, the story more than makes up for any loss in polish with its greater emotional involvement.

Translating Principle into Instance

The key to applying abstraction is the process of converting principle into instance. The author must be able to express a principle in a form the computer can process, then provide algorithms that translate that principle into instances that are specific to the context. Here's a very simple example:

Principle: Women dislike unwanted attentions.

First draft articulation: "Unwanted attentions" are actions of a romantic nature by a man to a woman that do not naturally flow from previous events. Women react to unwanted attentions by rejecting the man.

Second draft: "Unwanted attentions" comprise any form of touch with sexual or romantic connotations, or overt expressions of sexual desire. A woman's

rejection of unwanted attentions is proportional to the degree of inappropriateness of the unwanted attentions.

Third draft: The verbs that can constitute unwanted attentions are *kiss*, any form of the verb *touch*, and any instance of *declare*, *inquire*, or *demand* taking as a secondary verb any verb with sexual content. The verbs that can be used to express rejection are *declare* (dislike), *declare* (shock), *state imperative* (desist), and *strike* (with hand).

This example serves only to show, in a general way, how to go about solving the problem. Details of how to implement this kind of approach appear in later chapters.



This is great theory, but in practice, the act of reducing storytelling to grand principles is beyond human intellectual ability. Nobody could ever handle so deeply intellectual a process.

This process-intensive style of storytelling is done all the time—and by amateurs, no less. Here’s Grandpa taking little Annie up to bed:

“Tell me a story, Grandpa!” she asks.

“OK,” he replies. “Once upon a time there was a pretty little girl who had a pony...”

“Was it a white pony?” Annie interrupts.

“Oh, my, yes, it was as white as snow. It was so white that the sunlight reflected off its coat dazzled the eye. And the little girl and the pony would go riding along the beach...”

“Did they go riding in the mountains, too?”

“Why yes, as a matter of fact, they did. After riding along the beach, they would ride up the green canyons, jumping over the brush and ducking under tree branches, until they came to the very top of the mountains. And there they would play at jumping over boulders...”

“I don’t like to jump.”

“Well then, instead of jumping, she would let her pony graze in the rich deep grass on the mountain’s summit while she sat in the sun...”

And so the story goes on. Note that Grandpa does not respond to Annie’s interruptions with, “Shuddup, kid, you’re messing up my carefully prepared plot!” He wants those interruptions; his storytelling thrives on them. Grandpa does not enter the room with a carefully planned and polished plot, all set to dazzle Annie. Grandpa knows basic principles of storytelling, and then he makes up the story as he goes along—in response to Annie’s needs and interests. The story is the joint creation of Grandpa and Annie. It is their very special story, just for Annie and Grandpa, and no other story will ever be the same. Because it is their very special story, it means more and has more emotional power than any high-tech Hollywood extravaganza. Yes, it lacks the careful plotting, the intricate development, and the glorious special effects of the Hollywood product. But its roughness is more than compensated for by its customization. Sure, Annie likes *The Lion King*—but she treasures *Annie and the White Pony*.

Everybody understands the basic principles of storytelling. Everybody tells stories many times a day. Everybody knows how to translate the basic principles of storytelling into specific stories. If an amateur storytelling Grandpa can pull that off, why can’t we big-shot professionals do the same?

Conclusions


Abstraction is the means by which we can transcend distracting arguments over plot versus interactivity. Instead of thinking about a specific instance (a plot), we must learn to think in terms of something more abstract. I refuse to resort to the pseudointellectual term “metaplot.” Instead, “storyworld” describes the same concept more clearly. A storyworld is a complete, closed dramatic universe of ideas about a specific theme. “The Human Condition” is the largest storyworld of all. “Romance” is a smaller storyworld; “Tragic Romance” is an even smaller storyworld. For now, we shall have to confine our efforts to tiny storyworlds that address narrow themes.

Each storyworld contains within it myriad possible plotlines. Just as the variable x can hold an infinity of possible numbers, or a law can address a passel of possible actions, or a scientific law can describe zillions of possible physical actions, or a bank account can hold many different amounts of money, a storyworld contains implicit within it the possibility of many different plots.

CHAPTER 5

Verb Thinking

The greatest obstacle to the advancement of interactive storytelling is the difficulty of *verb thinking*. This is an unconventional and almost unnatural style of thinking that is, nevertheless, central to understanding interactivity. Verb thinking can be appreciated only in the context of its yin-yang relationship with *noun thinking*.



Verb vs. Noun

What is the universe? Most people answer this question by describing a collection of objects existing in space. The universe is so many stars, galaxies, planets, and so forth. Our corner of the universe is composed of so many rocks, trees, animals, plants, houses, cars, and so forth. To completely describe the universe, we need to list all the objects within it. It's an impossibly huge task, but conceptually it's simple enough.

But there's another way to think of the universe: We can think of it in terms of the processes that shape it—the dynamic forces at work. Instead of thinking in terms of objects embedded in space, we could just as well think in terms of events embedded in time. Is a tree a collection of parts (roots, trunk, bark, branches, leaves) or a system of processes (photosynthesis, nutrient absorption, cell division, water transport)?

These two ways of looking at the universe comprise the yin and yang of reality. They show up in every field of thought. In linguistics, we note that the two most fundamental components of all languages are nouns and verbs. Nouns specify things, while verbs describe events. Nouns are about existence; verbs are about action. Together, they allow us to talk about anything under the sun. Yin and yang shows up in other fields as well:

- ▶ In economics, goods are objects, things that exist. Services are actions, desirable processes. Goods are (mostly) permanent, while services are transitory.
- ▶ In physics, we talk about particles and waves. Particles are things; waves are transitory processes.
- ▶ Military strategists worry about assets and operations. Assets are the troops, weapons, and ammunition that can be used in a war; operations are the military actions they perform. Together, assets and operations define military science.
- ▶ Programmers see the dichotomy even more clearly: Memory stores data, and the CPU processes data. Data is the “noun” of computers: It is what is chewed up, sorted out, and worked over. Process is the “verb” of computers: It is what does the chewing, sorting, and working over.

- ▶ Even a factory manager unconsciously thinks in these terms. The factory starts with parts, the nouns of the operation. It then applies labor, the verb of the operation, to build its products.

The list of terms in **Table 5.1** demonstrates just how fundamental the concept of nouns versus verbs is to understanding our world.

TABLE 5.1 Comparing nouns (things) to verbs (processes)

Noun Concept	Verb Concept
goods	services
particles	waves
assets	operations
data	processing
number	algorithm
fact	principle
truth	logic
knowledge	understanding
parts	labor
character	plot

The Gray Between the Black and the White

Noun-stuff and verb-stuff do not constitute a black-and-white dichotomy; it's a polarity with lots of gray where the two sides blend together.

- ▶ **Language:** Nouns can be verbified (“*Trash* that memo!”), and verbs can be nounified (“That was a good *play*.”). In the process, the definition of the word itself can blur. Is “trash” intrinsically a noun or a verb? How about “play”?
- ▶ **Economics:** You purchase a hamburger at a fast food restaurant. Are you purchasing goods (the burger itself) or services (the preparation of the burger)? When you hire a tax preparer, are you purchasing a good (the completed tax return) or a service (the efforts of the tax preparer)?

- ▶ **Physics:** The wave-particle duality drives everybody nuts. If you shoot electrons through a double slit, they'll diffract, just like a wave. Yet you can detect each electron individually, just like a particle. So, are electrons waves or particles? It depends. Massive objects behave more like particles. Tiny objects behave more like waves—but you can never isolate a single aspect of the behavior. Sometimes they act like particles, and sometimes they act like waves.
- ▶ **Military science:** A small number of fast ships aggressively patrolling a sea zone can accomplish just as much as a large number of sluggish ships hanging around port. And one of the cardinal rules of battle is to bring more of your assets to bear on the enemy than he can bring to bear on you. In other words, strong operations can compensate for weak assets. And obviously, a large army with plenty of assets can still accomplish a great deal if handled properly.
- ▶ **Computers:** From the earliest days of computers, programmers have realized that any computation can be carried out with almost any combination of data and process. If you can throw more memory at the problem, you will require less processing and the program will run faster. If you are short on memory, you can always rewrite the program to use more processing and less memory.
- ▶ **Factory:** The tradeoff between raw materials and labor consumption is well known. You can work “up the supply chain” by taking in more of the job yourself. Instead of purchasing finished printed circuit boards for your computer, you can purchase the parts for printed circuit boards and add employees to make the printed circuit board yourself. More labor means fewer parts to buy. Conversely, you can subcontract out the manufacture of some parts to reduce your labor costs. It all depends on the precise balance of parts costs and labor costs.

Thus, the yin of nouns and the yang of verbs complement each other neatly; either one can be readily substituted for the other. There are plenty of special circumstances, of course, that would cause us to lean more toward one side or the other. It's a lot easier to think of Jupiter as a particle than to think of it as a wave with a very tiny wavelength. In your factory, if labor is cheap, then you want to use less finished parts and more labor. On the other hand, if labor is expensive, you want to use more finished parts.

Noun Prejudice

Another factor heavily biases the noun-verb balance in everything we do as well. For some reason, it's easier for us to think in terms of nouns rather than verbs. This bias shows up in each topic mentioned in the previous section, "The Gray Between the Black and the White."

- ▶ **Language:** I leafed through my dictionary, picked pages at random, and counted the number of nouns and verbs on each page. I counted a total of 101 nouns and 26 verbs, a four-to-one ratio. I repeated the experiment with a Latin dictionary and counted 57 nouns to 20 verbs, a 3:1 ratio. We seem to have a strong preference in our language for nouns. One language, Hopi, is famous for its preference for verbs—but that is what makes it such a weird language.
- ▶ **Economics:** Adam Smith's book *The Wealth of Nations* laid the foundations of modern economics and established all the basic principles: the relationship between price, demand, and supply; the utility of division of labor; the advantages of free trade; and so forth. Yet the concept of services as an economic entity was poorly developed in Smith's work. He was well aware of labor in economic production, but only as a means of creating goods. Thus, the father of modern economics never saw the yin and yang of goods and services; he thought almost exclusively in terms of goods. And indeed, the notion of services as an economic output did not establish itself firmly in economic thought until the early 20th century. We figured out the "goods" side of the dichotomy quickly; the "services" side took a century longer.
- ▶ **Physics:** Isaac Newton laid down the physics of particle motion in 1664; it took another 200 years before the physics of wave behavior was worked out in as much detail. Even for the brilliant men who created the science of physics, particles were much easier to understand than waves.
- ▶ **Military science:** People recognized the concept of "assets" from the earliest history of military efforts. The actions of generals in ancient times clearly demonstrate that they were quite cognizant of the importance of how many soldiers and weapons they had. But the notion that *the way* those assets were applied was just as important as the assets themselves dawned only dimly on early generals. Alexander the Great seemed to appreciate the idea, but his use of the operations concept seems spotty. Napoleon was the first general

to demonstrate a consistent application of the idea that armies are only as useful as the extent to which they are maneuvered. Between World Wars I and II, two military theorists (Liddell-Hart and Guderian) first put down on paper the ideas behind the operational approach to military strategy. And it took the Blitzkrieg to wake up the rest of the world to the value of these ideas. Thus, it took more than two millennia for military thinkers to recognize the importance of the yang of operations compared to the yin of assets.

- ▶ **Computers:** Historically, the relative importance of memory and processing were recognized from the start, largely because memory was in such short supply that computer scientists were desperate to find any way to get the work done. Indeed, the history of the relationship of memory to processing has been almost the reverse of every other field: Early programmers were forced by hardware limitations to put great emphasis on processing. As computer memories became larger, however, programmers put less and less emphasis on the processing side and more and more emphasis on the data side of the dichotomy. A second factor contributing to this process was the explosion of demand for programmers. In the 1950s through the 1970s, there wasn't much demand for programmers, so only the best and brightest became programmers. But when the demand for programmers exploded in the 1980s, businesses could no longer afford to limit the ranks of programmers to the cream of the crop; large numbers of programmers flooded the field, and these people were not as comfortable working with algorithms as they were slinging bytes around.

This shift is clearly seen in the programming microcosm of games. One of the early video game systems, the Atari 2600, was equipped with a powerful processor: an 8-bit 6502 running at just under 1 MHz. However, it initially allowed only 2 kilobytes of ROM and just 128 bytes of RAM. Games designed for this system were heavy on processing and light on data. Over the years, games have grown vastly in data size. A typical game these days can easily include gigabytes of data. We can see the nature of the change most clearly by comparing the shift in hardware capabilities in video game machines (**Table 5.2**).

TABLE 5.2 Shifting hardware capabilities in video games

Machine	Atari 2600	Microsoft Xbox	Ratio
RAM	128 bytes	64 MB	512 thousand
ROM	2 K	8 GB	4 million
Clock speed	1 MHz	733 MHz	733
Bus width	8 bits	64 bits	8

Note how the data capacity has increased far more than the processing capacity. The Xbox is millions of times better than the Atari 2600 when it comes to data, but only hundreds of times better when it comes to processing. Game programming has shifted away from processing and toward data. Much of this change is due to technological limitations; it's easier to expand storage than increase processing. Nevertheless, this change also represents (to a lesser extent) the data intensity that people seem to demand in their games as well as the proclivities of the programmers. And there remains the major question: If game machines are millions of times better, why aren't the games themselves millions of times better?

These examples demonstrate just how frozen our minds are in noun thinking. We just can't seem to "get" verb thinking—and when we do, it's always late and second best.

Recall Chapter 2 on interactivity; an important point of that chapter is that the degree or quality of the interactivity increases with the richness of the choices available to the user. Let's pursue that idea a little further. Choice takes action as its direct object. You don't choose between the éclair and the cotton candy; you choose between *eating* the éclair and *eating* the cotton candy. You choose between bowing your head to your boss and saying "Yes, sir" and telling him what you really think. You choose between verbs. Verbs lie at the heart of choice, and choice lies at the heart of interactivity.

Lesson 17

Interactivity requires verb thinking.

This emphasis on nouns is the core cause of ill-designed software. Designing good interactivity requires clear verb thinking, yet most people have trouble with verb thinking. So we struggle forward with our noun thinking, building software that's noun-heavy and verb-light (or, to use my preferred nomenclature, data-intensive instead of process-intensive).

Case in Point: Multimedia

Does anybody remember the multimedia craze of the early 1990s? It all started when CD-ROM drives became standard equipment on personal computers, and software developers could take advantage of the storage capacity of these babies. It may be hard to believe, but the average personal computer back then had only about 4 MB of RAM and 100 MB of hard disk storage. The floppy disk was the standard means of distributing software, and its capacity was only 1.4 MB. By contrast, a CD-ROM boasted 650 MB of storage. The transition from floppy disk to CD-ROM increased delivery capacity by a factor of more than 400! To make a similar-size jump from CD-ROM to something bigger, that bigger medium would need to hold about 260 GB!

Freed from the constraints of floppy disks, software developers began stuffing huge quantities of data onto their CD-ROMs. That data took the form of graphics. Because most personal computers in those days had 640h x 480v by 8-bit displays, requiring only 307 kilobytes of data per screen, developers could stuff about 2000 screen images onto a single CD-ROM. Overnight, the demand for artists to prepare all this artwork exploded.

Joy reigned throughout software development land. The old constraints had been smashed; the sky was the limit. Artists previously discouraged by computers due to harsh programming requirements could now make lots of dumb software with great graphics. And that's exactly what they did. The floodgates opened and out gushed a tidal wave of software called "multimedia." In design terms, it was pretty much all the same: a big pile of images and sounds presided over by a tiny little program that did little more than shuffle bytes from the CD-ROM to the screen and speakers.

Amid the cheering masses at the grand parade stood one scowling curmudgeon: me. I rained all over their parade, dismissing CD-ROM and multimedia as a waste of time. After all, I pointed out, CD-ROM is a data technology, not a processing technology. It gives a huge boost to the noun side of the interactivity,

but what matters is the verb side. Therefore, CD-ROMs and multimedia weren't going to do much to advance the software revolution.

Nobody believed me; they thought I was crazy. And certainly the sales figures made me look like an idiot. People made millions and millions of dollars making multimedia. The proof of the pudding is in the eating, and multimedia people were eating quite well.

I had seen it all before. Back around 1983, a similar fad swept the world of arcade games: *Dragon's Lair*. This was the first arcade game to use an optical disk, a precursor to the CD-ROM that was bigger, clumsier, and more expensive. The makers of this game stuffed actual animated video (from Don Bluth, no less!) onto the optical disk, with a simple branching system that permitted the internal computer to shunt different video snippets to the screen. The structure of the game was simple (**Figure 5.1**).

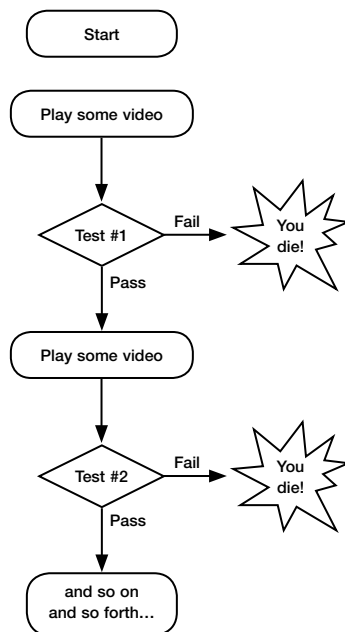


FIGURE 5.1 Game architecture of *Dragon's Lair*

The game was a huge hit—because nobody had ever seen such glorious graphics in an arcade game. Naturally, every other arcade game maker rushed to deliver its own optical disk game. Within a year, the arcades were full of optical disk

games that promptly flopped. Optical disk games were a fad, a flash in the pan, because they really weren't fun to play. Sure, they looked great, but they didn't *do* anything interesting. And play, as well as any form of interactivity, succeeds or fails on what the user *does*. Remember Crawford's First Rule of Software Design: Ask, what does the user *do*? What are the verbs?

Getting back to the 1990s and multimedia, I recognized that the whole thing was a fad, just like *Dragon's Lair*. It would climb a little higher and last a bit longer, but it had no staying power. And that is exactly what happened. The craze lasted three, maybe five years. By the late 1990s, multimedia was dead. The CD-ROM lived on. After all, it was a better distribution medium than the floppy disk. But people stopped marketing their software based on the size of its multimedia content.

It is instructive that, in the games industry, multimedia was killed by 3D graphics. The distinction is lost on many observers, who figure that one graphics medium is pretty much the same as any other. But there is a huge difference between CD-ROM-based multimedia and software-based 3D graphics rendering: the former is data-intensive while the latter is process-intensive. Multimedia merely shovels bytes from the CD-ROM onto the screen; multimedia designers sweat data capacity. But 3D graphics software calculates its imagery using the CPU; its designers sweat machine cycles. Because 3D-graphics software is processed rather than shoveled, it can interact with the user. The interaction is tighter and more intimate because it is process-intensive.

Getting Started with Verb Thinking

OK, so I've convinced you. You have seen the light, and you are a true believer in the gospel of verb thinking. How can you learn verb thinking?

First, you lay the foundations by shifting your outlook on the world. **Table 5.3** shows that list of dichotomous words again.

Every time you think about any of these topics, contrast the nouns on the left to the verbs on the right. Are you worrying about the facts of an issue when you could be thinking about the ideas behind it? Go down the street and look at the shops and stores. Are they selling goods or services? Can you imagine their output solely in terms of the services that went into the merchandise?

TABLE 5.3 Comparing nouns (things) to verbs (processes)

Noun Concept	Verb Concept
goods	services
particles	waves
assets	operations
data	processing
number	algorithm
fact	principle
truth	logic
knowledge	understanding
parts	labor
character	plot

Above all, try to think about things in terms of what things *do*, not what they *are*. A window is not glass; it's something that blocks air movement while permitting light to pass freely. A car is not an engine, a body, seats, and so forth; it's something that carries people and things from place to place; everything else is subsidiary. A computer is not a box with a whirring fan; it's a processing machine. A medicinal pill is not a bundle of exotic chemicals; it's something that alters the biochemistry of your living processes.

Even more effective is the uninhibited exercise of the most important mental faculty: your curiosity. It is likely that your schooling has left your curiosity a wisp of its former self, but with effort you can nurture it back to health. Everywhere you go, everything you encounter, ask yourself, "Why is it that way?"

Perhaps you've noticed that some coins have smooth edges while others have ridges—have you ever wondered why? Investigate it; there's a fascinating story behind it. Do you know why the sky is blue? If not, why haven't you found out the answer? As a child, you probably asked this question; have you lost that curiosity? Did you know that there's a perfectly good reason why so much of our legal language uses pairs of words, such as breaking and entering, assault and

battery, aid and abet? Why do our keyboards use the screwy QWERTY pattern? Do you still look up at the sky at night and wonder about the stars?

The important issue is not whether you know the answers to these questions; it can take a lifetime of asking questions to get these answers. What's important is whether you continue to ask such questions. Our educational system concentrates so hard on giving us answers that it stifles our talent for asking questions.

The world is full of fascinating mysteries. A person who does not wonder about those mysteries has lost his intellectual innocence. There are far too many mysteries for one person to answer in a lifetime. At 62 years of age, I'm still struggling to find answers to thousands of questions. But if you have given up asking such questions, you have already lost your youth.

Recall my reference to the image taken from the end of the movie *The Matrix* in Chapter 1. The protagonist, Neo, finally realizes his potential to see behind the artificial world of the matrix, to see that it is really just numbers and algorithms. The walls dissolve into numbers and his antagonists are stacks of numbers; everything is just numbers that he can control at will. This imagery provides the metaphor for how you must see the world. Those aren't walls in the hallway; they're zillions of atoms holding together with atomic forces, transmitting weight from roof to floor, interacting with photons of light to bounce those photons in different directions. Things don't just sit there—they happen. If you can come to understand how and why they happen, you can understand the universe at a deeper level, like Neo. And like Neo, just maybe you'll be able to stop bullets in midair.

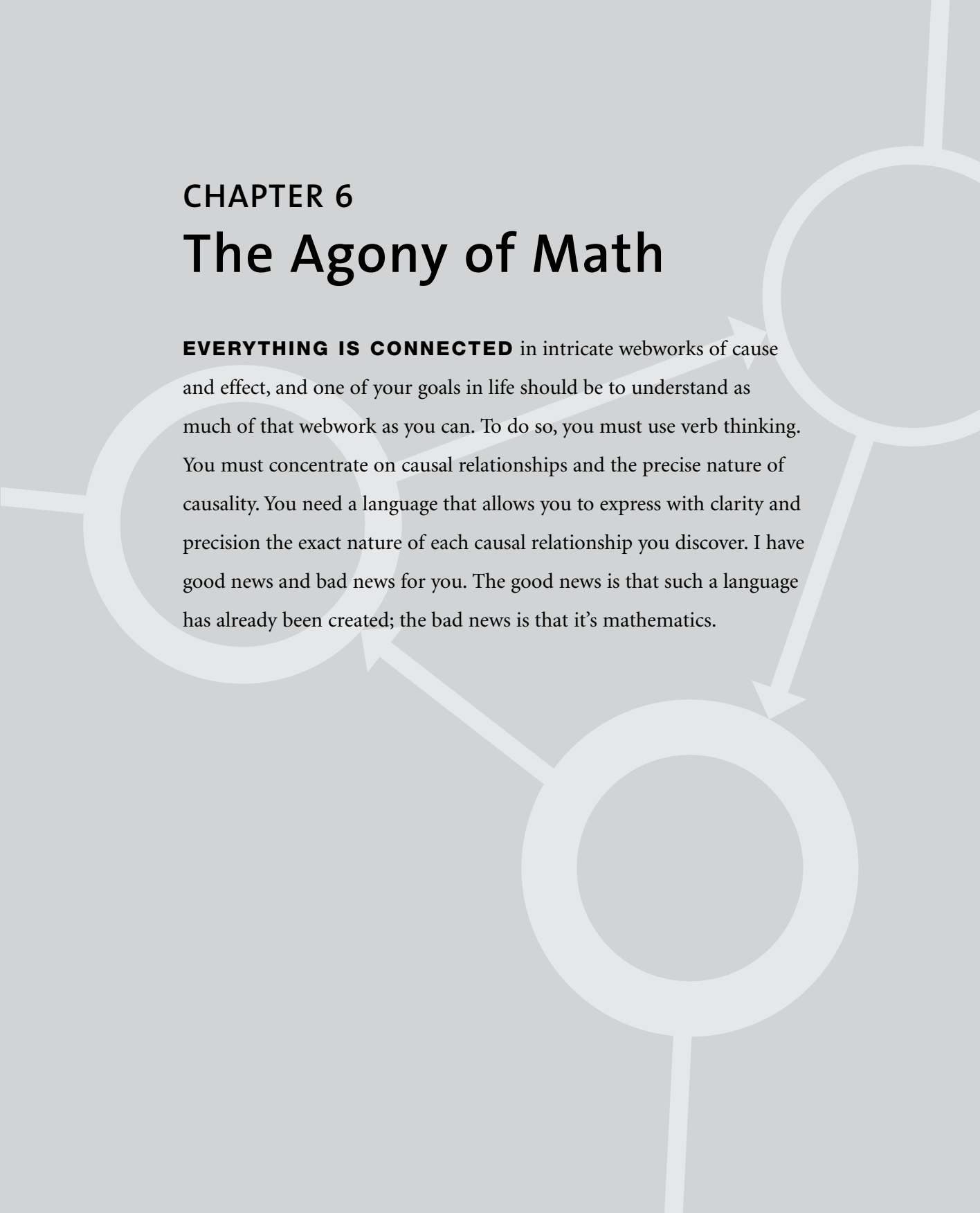
That's the power of verb thinking.

Conclusions

The first big mental shift you must push yourself through is the transition from a noun-oriented way of thinking about the world to a verb-oriented mental stance. This profound shift will challenge your open-mindedness and test your determination. Once you make the transformation, software design will become clearer to you. For the umpteenth time, I enjoin you to remember Crawford's First Law: Always ask, "What does the user *do*? What are the verbs?"

CHAPTER 6

The Agony of Math



EVERYTHING IS CONNECTED in intricate webworks of cause and effect, and one of your goals in life should be to understand as much of that webwork as you can. To do so, you must use verb thinking. You must concentrate on causal relationships and the precise nature of causality. You need a language that allows you to express with clarity and precision the exact nature of each causal relationship you discover. I have good news and bad news for you. The good news is that such a language has already been created; the bad news is that it's mathematics.

I can hear the hissing sound of your deflating balloon. Your lips are curling in anger and disgust; the very thought of using mathematics repels and terrifies you. Mathematics is cold and heartless. You are an artist seeking to create interactive storytelling, and you will have no truck with the narrow-minded world of mathematics. But I must somehow gently convince you to confront your fears and set aside your disgust so that you may harness this beast—because it is my experience that a willingness to embrace mathematics is the single most significant factor in the success or failure of interactive storytelling. Many are those whom I have tried to teach mathematics for use in software. Most refused to tackle it, like debutantes unwilling to seize a greased pig. A few girded their courage and dove in, getting filthy but winning the prize. I fear that many readers of this book will be unwilling to grab the pig, and will remain ineffective in their efforts to create interactive storytelling.

Embracing Mathematics

The refusal to employ mathematics to your understanding of causal relationships is not a matter of intelligence, innate ability, mental blocks, or anything like that. Blaming the problem on stupidity is a lame excuse for what is really an emotional block. You have every reason to have a gut-level aversion to mathematical thinking: It's unnatural! Our brains are wired to think in associations and patterns. Mathematical thinking occurs in a sequential, step-by-step style—something our brains are not wired to do. Hence, mathematical thinking requires you to twist your mind around in knots and make it operate in a style it just isn't built to do. You are right to balk at my suggestion that you subject yourself to this kind of mental torture.

But what, I ask you, is wrong with the idea that you have to work to accomplish wonderful things? Great artists have never shied away from unpleasant tasks in pursuit of their artistic goals. Leonardo da Vinci, Michelangelo, and many other Renaissance artists wanted to paint the human body in all its glory, but they realized that, to paint the human body, they needed an intimate knowledge of its structure. There were no books of human anatomy in those days; the only way to learn human anatomy was to attend human dissections carried out for medical

students. This was before the days of formaldehyde, alcohol, and freezers. In the warm Italian summers, those bodies quickly began to stink. After several days of this, you can imagine how disgusting it had to be to step up close to examine the muscles, tendons, and bones that shape the human body. Yet Leonardo, Michelangelo, and many other Italian artists compiled large volumes on human anatomy (Figure 6.1).

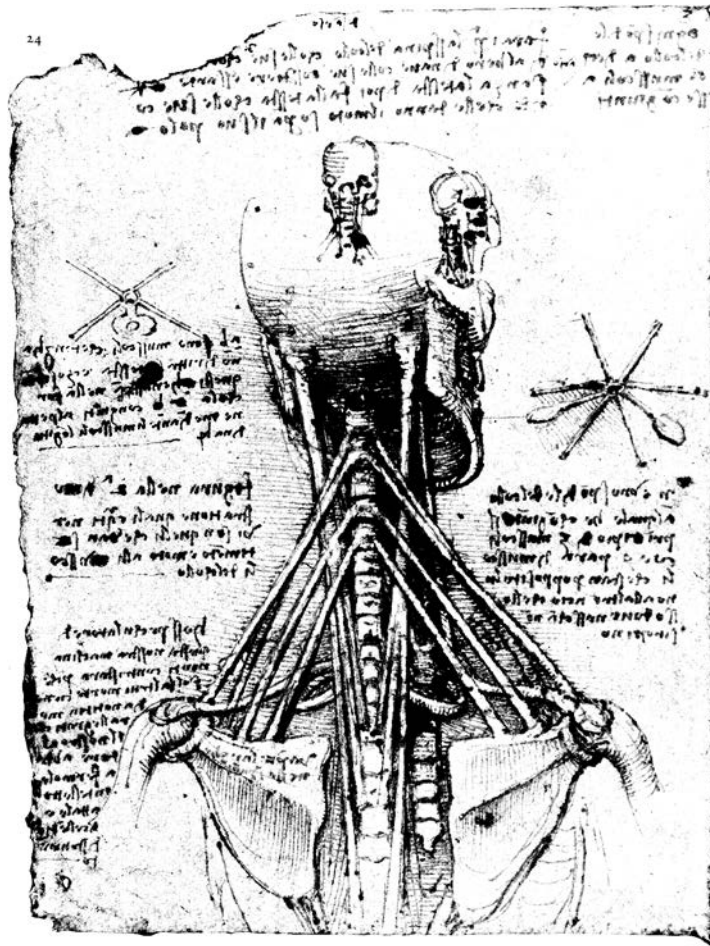


FIGURE 6.1 Anatomy drawing by Leonardo da Vinci

The artists suffered through these ghastly experiences because they wanted to improve their art. If they could put up with the stench of rotting corpses, who are you to balk at relearning high-school math?

And don't kid yourself that there just has to be a way to do interactive storytelling without mathematics. The fundamental concepts of interactivity lead inevitably to mathematics. There's no getting around this one. Lord knows, I've tried in my own work on interactive storytelling. I have struggled to cushion the spikiness of mathematics in layers of soft verbal down. I have tried to concoct other forms of expression that skirt mathematics completely. Despite the exertion of all my creative powers, I have managed to accomplish little more than putting lipstick on the face of this ugly pig called mathematics.

Lesson 18

You must embrace the drooling monster of mathematics if you are to succeed in interactive storytelling.

Causal Relationships

We normally think of causality in merely binary terms: Socrates is either mortal or not mortal. So we have our basic syllogism:

All men are mortal.

Socrates is a man.

Therefore, Socrates is mortal.

The syllogism is trivially easy to understand, because it uses simple black-and-white logic. But why must the syllogism be limited to yes-or-no terms? Why can't it be numeric, like this:

All men's weights are equal to 4 times their waist measurements.

Socrates has a 48-inch waist.

Therefore, Socrates weighs 192 pounds.

The difference between the two syllogisms is that the first syllogism gives mere binary results (yes or no to the question “Is Socrates mortal?”). The second syllogism, meanwhile, can have all sorts of different results, depending on the waist size of the man. The first divides up truth into black and white, yes or no, while the second permits shades of gray. The first syllogism can address only the most simple-minded questions, but the second can address far more—in fact, anything that can be quantified.

Having established the basic principle that mathematical formulas are simply a kind of syllogism, we can then make those formulas more complex. For example, a more accurate formula for a man’s weight might be something like this:

All men's weight is equal to 3 times their waist measurement plus their height.

or, to put it in mathematics terminology:

$$\text{Weight} = 3 \times \text{Waist} + \text{Height}$$

This equation is a little more accurate. It’s still not really true, but only in the absolute sense of getting exactly the right number for every man. This equation might get within 20 pounds of correctly predicting the weight of 70 percent of all men. That’s pretty good. We could probably make it better if we made the formula even more complex:

$$\text{Weight} = 2 \times \text{Waist} \times 1.5 + 0.75 \times \text{Height} - 12.1 \times \text{Shoe Size}$$

Perhaps this equation would get within ten pounds of correctly predicting the weight of 80 percent of all men. The perfect equation—the one that correctly predicts with zero error the weight of all men—is impossible to figure out. Perhaps it could be found with enough effort, but it would probably involve thousands of variables put together in the most intricate fashion. The equation would probably be dozens of pages long. For all practical purposes, perfection is impossible.

But we seldom need perfection. A chair designer needs to know the approximate relationship between the size and the weight of most people. There’s no need for exact numbers when designing chairs; getting within 20 pounds is good enough. An interactive storyteller need not achieve perfection of mathematical description; getting close enough to make the point is good enough.

Modeling as a Metaphor

One of the oldest figurative metaphors is Homer’s “rosy-fingered dawn.” It’s a powerful metaphor because it makes a comparison between the rays of predawn sunlight and the fingers of the hand, suggesting an animation of a natural process. Poets are in the business of creating metaphors. So are most artists.

Mathematical modeling is also fundamentally metaphorical. Indeed, the mathematical modeling used in interactive storytelling is the essence of its artistic content. *Not* the images, *not* the sounds, *not* the dialogue, *not* even the plot development. Sure, you can have artistic images in an interactive storytelling product, and you can stuff in a symphony or two, mix in a dash of poetry, and so forth—but that’s not interactive storytelling. You could create a movie consisting of little more than pictures of great paintings—but would it be a good movie? You could create a movie of a great poet reading poems—but would that be a good movie? You could create a movie showing an orchestra playing a Beethoven symphony, but would that be a good movie? Taking good art in one medium and simply transferring it to another medium doesn’t make the result good art. Indeed, such efforts are pathetic; a true artist strives to express content most syntonic with the soul of the medium.

This is an immensely important point and, sadly, one that many readers will find difficult to accept. Part of their resistance arises from the passionate attachment that any artist feels for his or her art and its medium. Moviemakers love the cinema; poets adore poetry; writers worship literature. My claim that these magnificent forms have no primary place in interactive storytelling sounds like a principal telling parents that their child is not good enough to attend the school. The same outraged rejection is the inevitable reaction to my claim.

But hold on, friend. I write these words not to belittle you but to help you find the ideal place for your efforts. In past times, sons were forced to take up their father’s trade regardless of their own proclivities. This interactive storytelling medium may not be the best place for your talents or inclinations. If so, don’t fight it—follow your muse.

And don’t try to prostitute your muse into a role for which she is ill-suited. In the earliest years of cinema, some gifted thespians attempted to force Melpomene and Thalia, the muses of tragedy and comedy, into this new and alien medium. They simply set up the camera in the middle of the audience and acted out a play on the

stage. The results were inferior versions of the real thing, but they were adequate for distribution to remote areas lacking access to the genuine article. They were also financial flops. As a consequence, thespians came to disdain cinema as a simian medium.

It took D.W. Griffith, a failed actor with no emotional attachments to the theater, to approach the cinema without preconceptions. No worshipper of Melpomene and Thalia, he conjured up his own muse and succeeded where so many talented thespians had failed.

Therefore, friend, if you have already set your path on the well-trod way up the mountain peaks of other media, then I wish you well but urge you not to drag the medium of interactive storytelling along with you. That is another mountain peak, along a very different path. Worse, the path to that mountain peak is unexplored and unmarked; you'll need a machete as much as good hiking boots. Progress up this path will be slower than what you can accomplish in other media. You'll not get there faster by taking the easier path.

If you can jettison your emotional attachments to other media (including games!) and approach interactive storytelling with the naïveté of a child, then you can accomplish something. If you try to force it to fit into your preconceptions, you will fail.

Lesson 19

Don't view interactive storytelling through old-media-colored glasses.



Hold on. You're proposing to apply mathematical thinking to storytelling, but human beings are infinitely complex creatures. Reducing a person to a set of numbers dehumanizes us all. The function of art is to explore and glorify the wonder of our existence, not deny our richness.

Remember, this is drama, not reality. Yes, real human beings are infinitely complex, but in drama we strip away much of that complexity. Imagine the nastiest real person you've ever known; can that nastiness hold a candle to some of the bad guys of fiction? Have you ever known any real person remotely similar to Darth Vader? Anybody as dashing as Han Solo? The characters in movies are not real, complex persons; they are simplifications. A movie that's 100 minutes long can't delve into the infinite complexities of the human condition, so it discards

most of that complexity to show a few glimpses of it in shining clarity. Thus, the mere fact that mathematical representation simplifies is not in itself a significant argument against it.



But reducing human characteristics to numbers is still demeaning.

Let's use love as an example. Is not love a trait that someone can have more or less of? Are there not degrees of love? If we can agree that love does exist in varying degrees, then what is wrong with assigning numbers to those varying degrees? If we can say that Romeo's love for Juliet is greater than his love for his family, is there any fundamental shift in thinking if you assign numbers to reflect that?



But love is a multidimensional concept; it cannot be measured by a single number. There are differing kinds of love: the love we hold for our parents, the love we feel for our children, the love we have for our spouse. These are all different kinds of love that cannot be represented by a single number.

Are these differences significant to the storyworld you intend to create? Will actors in that storyworld behave differently because they have these different dimensions of love? If so, then we must break down the single variable Love into separate flavors of love and quantify them separately. But if such differences in love are not significant to the artistic content of our storyworld, then we can dispense with them and use the single factor Love.



There's a huge difference between talking about "greater" or "lesser" love and 0.57 Love or 0.23 Love. The verbal description leaves plenty of room for the approximate nature of our appreciation of human qualities, while the quantitative description creates an inappropriate impression of accuracy.

Consider the following list:

- Greatest love
- Greater love
- Lesser love
- Least love
- No love

Clearly, this list constitutes a verbal description of varying degrees of love. Hence, we could easily assign numbers to it:

Greatest love	4
Greater love	3
Lesser love	2
Least love	1
No love	0

We could readily extend the resolution of this list by adding more terms:

Greatest love	7
Strong love	6
Greater love	5
Moderate love	4
Lesser love	3
Insignificant love	2
Least love	1
No love	0

We could continue to extend this list, and obtain greater resolution, if we didn't stumble over imprecise and therefore overlapping terms such as “insignificant” and “least” or “strong” and “great.” But these are limitations of vocabulary; they are not intrinsic to the idea of love. I agree that specifying somebody's love as 0.2364 is ridiculous; we could not possibly insist that 0.2364 is correct and 0.2365 is wrong. The *principle* of quantification is not at issue—only the degree of resolution is debatable.

Quantification is not dehumanizing nor does it trivialize the complexity of the human condition. Finding the right value for a variable is analogous to finding the right word for a sentence. It's just a matter of zeroing in on the precise content we want. The only difference is that it's numerical rather than verbal. We're using a different vocabulary, but we're saying the same things. The benefit is that this vocabulary can be used with a computer.

Despite all these arguments, some readers will still choke at the very notion of quantifying human qualities. Of such readers, I ask: If all my reasoning fails to move you, is it perhaps due to a profound emotional aversion to all things mathematical? The world is full of people who struggled with math in school; perhaps they were made to feel stupid by the experience. It would be no surprise if such people nursed a combination of fear, disgust, and anger toward math throughout their lives. If you are such a person, I ask you to confront your feelings and determine whether you can set them aside in pursuit of the grand and glorious goal of realizing interactive storytelling.

Conclusions

Pretty much everybody hates mathematics; it twists our minds around like pretzels. But mathematics is the language we must use to express our artistic metaphors in terms that the computer can understand. Some artists use the intestines from cows to make great music. Violin strings are made of catgut. Some artists use rocks to make great sculptures. Some artists attended dissections to make great paintings. With the computer, mathematics is the means you use to make great art.

CHAPTER 7

Two Cultures, No Hits, No Runs

IT'S BEEN NEARLY 50 YEARS since C.P. Snow, a British scientist and novelist, warned us of the emerging problem of the *Two Cultures*. The scientific community and the arts/humanities community were drawing apart in mutual incomprehension and antagonism. Snow believed strongly in the necessity for the two cultures to work together in solving the problems that bedevil Western civilization.

Although his thesis caused a sensation and triggered much discussion, the problem has grown worse. Scientists, engineers, and mathematicians are not just unschooled in the humanities—they frequently dismiss the arts and humanities as wastes of time. The arts and humanities people have gone just as far in the other direction: They simply refuse to have anything to do with the sciences, and they disparage science as linear thinking.

The fundamental difference in style between the rigorous sequential thinking of the sciences and the pattern-recognition thinking of the arts and humanities divides the two camps even further (see the “Two Mechanisms of Thinking” section in Chapter 1, “Storytelling”). There’s even a bit of a gender factor here, with the sciences attracting more than its share of men and the arts and humanities grabbing more of the women. Whatever the cause, the basic trend is undeniable: The two sides are drawing ever further apart.

We’re paying the price for this divisiveness in the field of interactive storytelling. A physicist doesn’t really need to know the Louvre from the loo, and an artist can get along quite well without knowing the difference between energy and entropy. But interactive storytelling requires creative people who straddle the divide. The verb thinking that Chapter 5, “Verb Thinking,” is devoted to is really a generalization of the algorithmic thinking that scientists routinely engage in—and who can deny that a solid appreciation of the nature of storytelling is necessary to success in interactive storytelling? But getting the two sides of the divide to work together smoothly can be difficult.

All of Me, a delightful comedy starring Steve Martin and Lily Tomlin, demonstrates this divisiveness. Released in 1984, the movie told the story of how the mind of Edwina Cutwaters (Lily Tomlin’s character) gets stuck inside the brain of Roger Cobb (Steve Martin’s character). Edwina gets control of Roger’s left side while Roger retains control of his right side. The consequences are demonstrated in a hilarious scene in which Roger attempts to walk down the sidewalk. Edwina, wanting to go one direction, uses her right leg and hand to march one way, while Roger, using the other side, desperately tries to do the opposite. Steve Martin’s physical comedy is magnificent; you can see the battle between the two sides of his body as they struggle against each other. And, of course, they get nowhere. (You can see a snippet of this on YouTube at www.youtube.com/watch?v=pe2JWivXyN4.)

That’s pretty much where we stand now with interactive storytelling. The programmers and games people on one side struggle against the new media people on the other, and with all the spastic staggering, nobody gets anywhere. There has definitely been progress this century as the two sides have spent more time rubbing shoulders. In the movie, the two antagonists eventually learn to work together, and they fall in love in the end. Let us hope that interactive storytelling has such a happy ending.

Neurophysiological Basis

To some degree, the Two Cultures problem can be traced to events during the evolution of the human brain. As humans faced new challenges, *mental modules* developed to address those challenges. Two broad classes of mental modules exist: pattern recognizing and sequential. The pattern-recognizing modules were finely tuned to process complex patterns. For example, our visual processing systems include lots of pattern-recognizing elements for recognizing textures, shapes, and so forth. Another form of pattern recognition is used in social reasoning, including figuring out who your friends and enemies are, who is likely to help you, and who might betray you. Sequential processing is used for auditory processing and language processing; it is also important in understanding and appreciating certain natural phenomena commonly encountered by hunters, such as tracking the trail of wounded prey.

These two basic styles of thinking (pattern recognition and sequential processing) exist in every brain, but some minor genetic predispositions affect it. Gender specialization in hunter-gatherers (men did more of the hunting, women did more of the gathering) endowed men with slightly stronger skills in the sequential processing, and women with a slight advantage in the pattern-recognition. These small tendencies are magnified by the cultural milieu in which children are raised. The net effect is that people concentrate on one of the two forms of cognition according to a slight gender bias. Men have traditionally dominated the sequential logic cultural functions, while women have played a larger role in the pattern recognition cultural functions.

Thus, there's a bit of correlation between logic versus pattern and science versus art. It's not an ironclad law, just a vague correlation. People who are stronger in sequential logic gravitate toward the science and engineering side of the Two Cultures. People who are stronger in the pattern-recognizing side are more comfortable in the arts and humanities.

Thus, the Two Cultures war is, to some extent, based on styles of thinking that stretch back millions of years.

History of Games and Stories

The disappointing history of the relationship between games and stories demonstrates why we're in such a deep hole now. Game designers have linked stories to their games from the beginning, using the best available technology. This forced them to tack the stories onto the games rather than integrate the stories into the gameplay. That basic approach has now become the convention, a self-imposed box that game designers can't mentally break out of.

The Age of Print Stories

Early video game designers who appreciated the Two Cultures sensed the importance of story and attempted to give it a role in their designs. Of course, they couldn't put the stories into the program itself—computers were too weak to handle so much computational load. Instead, they wrote stories for the manual that came with the game. I remember one such effort from 1979: The manual for a game from Automated Simulations began with “The derelict was old, old...” Putting the story in a manual dominated the game efforts in the 1980s. A video game for the Atari Video Computer System, *Yar's Revenge*, included a comic book presenting the backstory for the game. I used the technique three times in my games *Excalibur* (1983), *Trust & Betrayal* (1987), and *Guns & Butter* (1989). For each of these I wrote a novella that demonstrated the concepts behind the game; the insightful reader could discover effective play strategies by reading the novella.

The Age of Cut Scenes

The first big change came in the late 1980s with the development of the *cut scene*: an interstitial episode between portions of the game. *Wing Commander* (1989) used this technique with great success. It was clumsy as you played the game for a while, and then took a break to read a page or two of a comic book–style story. Then it was back to the game.

Everybody started using cut scenes; the basic technique remains with us. With the passage of time, cut scenes became more elaborate. Once we had CD-ROMs for lots of storage space, the cut scenes proliferated and were upgraded with snazzier graphics.

The Age of Recorded Video

The next step in the process was the inclusion of video in cut scenes. *The Seventh Guest* (1993) was the first game to use video, although its use of video was more tightly integrated into the game. Most games simply replaced still-image cut scenes with video cut scenes.

The Age of Rendered 3D

The next step took place almost immediately after the advent of recorded video. It was implemented by id Software, first with its 3D game *Wolfenstein 3D* (1992) and then with its huge hit, *Doom* (1993). These games did not use recorded video; they calculated the animated 3D imagery directly with the CPU. In one stroke, they destroyed recorded video in games. This was a clear case of a process-intensive technology crushing a data-intensive technology.

Sadly, id Software took a dim view of storytelling. This set back the coevolution of story and game, but it didn't take long for other people to leap into the gap that id Software left unaddressed.

The Age of Fractured Stories

Around the turn of the 21st century, game developers began to soup up the stories they added to their games. They began to hire people who knew something about storytelling. But their plans fell afoul of the Two Cultures divide. The game designers treated storytelling the same way they treated the animation, music, and sound effects—as a supporting element, an embellishment to the game. They didn't bring the storytellers into the project until after they had already sketched out the basic game design; the storytellers were required to come up with a story that fit into the game.

Interactive storytelling lives and breathes as a symbiotic combination of the technical and the artistic; making one subordinate to the other prevents any such symbiosis and ensures that the storytelling will not be interactive. And that is exactly what happened. The storytellers were confined by straitjackets forced on them by the game designers. The results were pathetic.

The Two Cultures divide exacerbated the problem by isolating the storytellers from the designers who were creating the interactivity. Interactivity was the responsibility of the game design department, and narrative was assigned to the story design department. The storytellers never employed the principles of interactivity because they were never required to work with interactivity.

Three Cultures?

Ironically enough, when it comes to interactive storytelling, we have three cultures that eschew each other, not just two. The first of these three is the video-games culture. These people consider interactive storytelling to be something that's tacked onto existing video-game designs as an entertaining embellishment, another bullet point to add to the box copy. The second group consists of academics, who write lots of papers about interactive storytelling but seem less inclined to actually build it. Lastly come the traditional story people, who are eager to interactivize their stories, but haven't mastered genuine interactivity.

Exhortations and Prognostications

What can you do to prevent the Two Cultures divide from sundering your mind?

Listen

Take the time to really listen to people.

Let me tell you a story about a friend who changed my life. I call her Vero, short for Veronique, and she is the most intuitive person I know. Since I worship at the altar of Reason, you might think that Vero and I were fated for mutual disdain. But when we first met, there was some sort of emotional connection that transcended our differences. Perhaps it was the attraction of yin for yang; perhaps it was Platonic romance. I don't know. Whatever the attraction, we always enjoy our time together, talking endlessly about all manner of things.

One day we began discussing reincarnation. Being a rationalist, I reject such a notion as idle superstition. Vero thinks differently. So I decided to walk her down the primrose path of logic, right into a trap. Step by step I led her toward my inexorable conclusion, making each step absolutely bulletproof. However, one

step in my logical sequence was unavoidably soft. While every other logical step was solid, this one was not quite ironclad. I rushed through it and moved on to the next step. Vero listened politely as I went a couple of steps further, and then she pounced on the vulnerable spot. With unerring accuracy she stripped bare the flaw in my logic and demolished my argument. I was flabbergasted; how could she have seen the flaw? I interrogated her, trying to determine what logic had led to her discovery. There wasn't any; she just knew.

From that day forward I have held Vero and other intuitive thinkers in much higher respect. I don't understand how they do it, and it bothers me that they cannot justify their conclusions. There's no question, however, that intuitive thinkers are capable of intellectual feats that defy my comprehension. I have started cultivating my own intuitive faculties. Maybe I can learn how to do it, too! So far, though, I feel like a child trying to ride a bicycle.

Fraternize with the Enemy

Take the time to cultivate the other style of thinking. If you're an intuitive, pattern-recognizing artist, tackle some icy-cold logic. Remember, it doesn't come easily to anybody—we all have to learn it. If you're a sequential-thinking person, take some time to understand art. Don't force yourself to endure art that you don't appreciate; just delve deeper into something that does appeal to you. There really is something important going on there, a strand of human genius that you need to respect and appreciate.

Note that there is an important social trap you must be aware of. We humans are innately tribal. We like to hang out with people who share our tribal values, and we tend to view those outside our tribe as vaguely threatening if not outright enemies. The relationship between conservatives and liberals in the United States exemplifies this problem, preventing the government from getting much done. In exactly the same way, the social isolation of techies from artsies makes it difficult to accomplish interactive storytelling. It is imperative that people from both tribes take the time to get to know each other. You'll start to appreciate their point of view if you are open-minded.



I've worked with those artsies and they just don't get it! I try to explain interactivity to them, but they just can't understand it.

Are you confident that you grok interactivity? Do you know it deep down, inside and out, from stem to stern, in your bones? Could you teach a semester course on interactivity? This concept is new to our culture, and it's understandable that people are having a rough time comprehending it. So take some time to explain the concepts, and be patient. They'll get it if you teach them well enough.



I've worked with techies and they simply can't get outside of their own little techie universe. They live and breathe programming and computers. How can I discuss complex narrative matters with people whose idea of a classic story is Star Wars?

Most techies are smart people; they're just smart in ways that are alien to you. If you think that you understand complex narrative issues, then shouldn't you be able to teach those matters to techies? Sure, it'll take a while; did you become an expert overnight? How many years has it taken you to develop your understanding of narrative? Give techies a break; they'll figure it out if you take the trouble to teach them and you are patient.

Learn the Language

We need to standardize our terminology. Techies and artsies have completely different vocabularies, which only serves to widen the gulf between them.

I think the artsies have a better-developed vocabulary than the techies. Consider some of these rough parallels in **Table 7.1**.

TABLE 7.1 Artsie versus techie terms

Artsie Language	Techie Language
Character	Non-player character (NPC)
Prop	Object, thing
Stage	Location
Fate	Drama manager

Consider further that the artsies have a great many more terms that will surely serve us better in the future: plot twist, climax, denouement, resolution, protagonist, antagonist, and many more. By contrast, most of the terminology used by techies is derived from computer science.

For Techies: Experience Some Art

Get out of your shell! When was the last time you went to an art museum? Listened to classical music? Watched a Shakespeare play? Read anything by Twain? Are you sure you know what you're missing? Sure, some of it won't work for you; I hate Dickens, opera bores me, and I've never gotten a charge out of paintings. Just as you don't need to learn every computer language, you don't need to pursue every artistic medium. But I guarantee that something will click for you, and you'll be glad you tried it.

For Artsies: Do Some Programming

You want to avoid C++ and the professional languages, but there are plenty of cute little programming languages that are easy to learn and fun to play with. Try Python; write a program that does something cute. Don't ask a programmer for help; just get in there and play around. That's how I learned programming.

Conclusions

The Two Cultures war is most intense in the United States. It is nowhere near as ferocious in Europe, where the culture exalts art. Americans have shopping malls; Europeans have architecture. Where Americans have Walmarts, Europeans have museums. Europeans install public sculpture where Americans build parking lots. Because they are not crippled by Two Cultures nonsense, the Europeans are way ahead of the Americans in their efforts at interactive storytelling. I estimate that Europe will lead the way in this new form of entertainment.

Scientists generally admit that the most powerful force for change in science is not rational analysis but the mortality of older scientists. A similar pessimism permeates my appreciation of our current problem. The prejudices that underlie

the Two Cultures problem are immune to reason. This chapter will not rehabilitate any of the combatants in the Two Cultures wars. Ultimately, however, you can't beat Mother Nature; those who continue to man the barricades will surely fail. With the passage of time, we will see individual acts of treason, brave souls who dare to cross no man's land and consort with the enemy. Those few brave souls will enjoy some measure of success, and that success will encourage more to follow. Eventually the divide between the Two Cultures will be bridged, at least in the field of interactive storytelling. But I fear that such a deep cultural shift will not be completed in my lifetime.

PART III

Evolutionary Strategies

MOST ATTEMPTS AT INTERACTIVE storytelling are evolutionary in style; they are attempts to extend existing technologies to cover interactive storytelling. Such strategies work about as well as airplanes that used bird technology to make flapping-wing contraptions.

Chapter 8: The first thing interactive storytelling beginners try is a branching tree. It never works. Here's why it never works.

Chapter 9: A time-honored and highly developed genre, interactive fiction faces fundamental obstacles that prevent it from having much chance at generating true interactive storytelling.

Chapter 10: Role-playing games are a popular and rich genre, but they're stuck in a rut that they just can't seem to climb out of.

Chapter 11: Because video games are such a lucrative genre, developers have high hopes that the money lavished on them can be used to make powerful interactive storytelling, but alas, all that money is merely lipstick on a pig.

This page intentionally left blank

CHAPTER 8

Branching Trees

EVERYBODY SEEMS TO MAKE the same mistake when they first try their hand at interactive storytelling: They build a branching tree structure. They start off with a beginning, “Once upon a time, there was a handsome young man...” Then, they attach a choice to it, “...and he either (a) went off in search of adventure or (b) stayed home to care for his sick mother.” Next, they attach a choice to each of the first choices, and then another set of choices to those, and then more choices. Pretty soon, they’ve got a huge branching tree that consumed many hours of labor. They’re proud of their work, but when they show it off, nobody likes it.

The reason for the failure of the resulting game is easy to understand with a little mathematics. Imagine the smallest, simplest possible structure. Each decision is a choice between just two options, and leads to a new situation with a new binary choice. Thus, the first level of choice will have exactly 2 states; the second level will have 4 states, the third will have 8, the fourth 16, the fifth 32, and so on. **Figure 8.1** shows what this branching tree would look like with 6 levels:

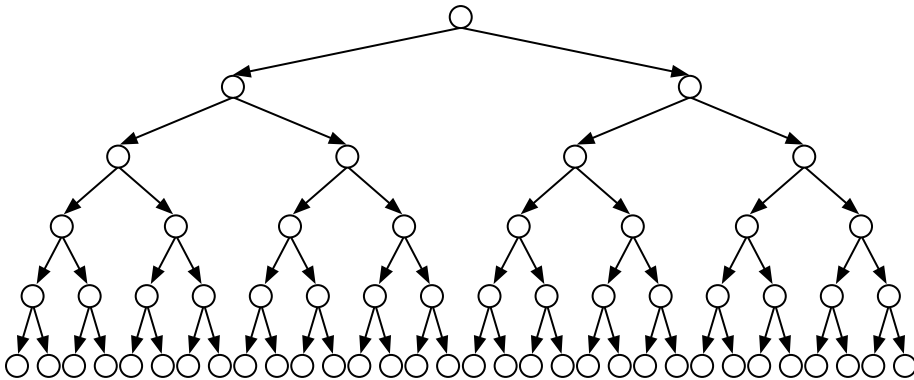


FIGURE 8.1 A six-layer branching tree

That looks pretty big and unwieldy, doesn't it? Yet it's only six layers deep, which means that it can make a story only six sentences long.

Let's say that a productive designer should be able to create a thousand such states. That seems like plenty of work. If you walk through the doubling process, you get to a thousand after just ten steps. From the designer's point of view, this represents a heavy workload. But from the user's point of view, all this work yields a story that is just ten steps long. It might read like this:

1. Once upon a time, there was a handsome young man. One day, he set off in search of adventure, and in the forest heard a cry and a growl. (Investigate or ignore?)
2. He investigated the sounds and found a maiden being held prisoner by an ogre. (Intervene or ignore?)
3. He told the ogre to release the maiden, but the ogre challenged him to a quest. (Accept or reject?)

4. He accepted the quest and entered a nearby cave. There were two paths: down one path he could hear water trickling; down the other he smelled smoke. (Which path?)
5. He went down the path that smelled of smoke. There he saw a dragon sleeping; it was clutching the key the ogre wanted. But there was also a sword lying nearby. (Take key or sword?)
6. He picked up the sword. Should he use it to attack the dragon or take it back and attack the ogre?
7. He decided to use it to attack the ogre. He went back and confronted the ogre. The ogre laughed at him and threatened to kill him. (Attack or retreat?)
8. He attacked and wounded the ogre, who fell to the ground. The maiden begged him not to kill the ogre. (Kill ogre or relent?)
9. He went ahead and killed the ogre anyway. When a drop of blood from the ogre splashed onto the maiden, she turned into an ogre and was very angry with him. (Apologize or kill?)
10. So, he killed her.

Not much of a story, is it? Even if it were fleshed out with pretty verbal descriptions and embellishments, it's still skeletal. That's because real stories have a wide array of decisions. Only a few decisions are dramatically crucial, but myriad tiny decisions help define character, including the way one character addresses another, the degree of impatience or alacrity with which a character operates, and so on. A real story will contain hundreds or thousands of such tiny decisions; this kind of fine detail just isn't possible in a branching tree. Thus, branching tree designs demand too much work from the designer and offer too little meat to the user

Lesson 20

Branching trees suffer from geometric growth that always requires more work than designers anticipate.

Overall Game Structure

The branching structure described in the previous section shows up all the time in games. Sometimes, the player makes decisions that select particular branches during the entire course of the game. The designers of such games boast that their game has, say, eight different endings. That sounds great, doesn't it? But look again at the branching tree in Figure 8.1. You get eight different endings with just three layers of branching tree. If they add just one more decision point to their tree, they get 16 different endings! But here's the question: Do you measure the quality of the game by how many endings you can experience or by how many choices you get to make? If you have any doubt about the answer, go back to Chapter 2, "Interactivity," and reread the most important point in this book: Always ask, what does the user do? What are the verbs?

Dialogue Trees

More often, designers confine the tree to a single interaction between the player and another actor; they call these *dialogue trees*. You walk into the clichéd medieval inn with the clichéd innkeeper serving the clichéd ale to the clichéd cast of nasty-looking characters sitting at their clichéd wooden tables. You need to find out about the clichéd treasure, so you ask the clichéd innkeeper. But you can't just ask him; you have to say the right thing to wring an answer out of him. How do you figure out the right thing to say? That's impossible to know in advance. After all, you've never met the innkeeper before, and you have no idea what will influence him positively. **Figure 8.2** shows the dialogue tree.

To get the innkeeper to tell you about the treasure, first you must purchase an ale, then you must ask him how his business is going, then you must commiserate with him—and only then can you ask him about the treasure and get a useful answer. Of course, you don't know this in advance, so you simply wander through the dialogue tree. With trial and error, you'll get there eventually. Isn't this fun?

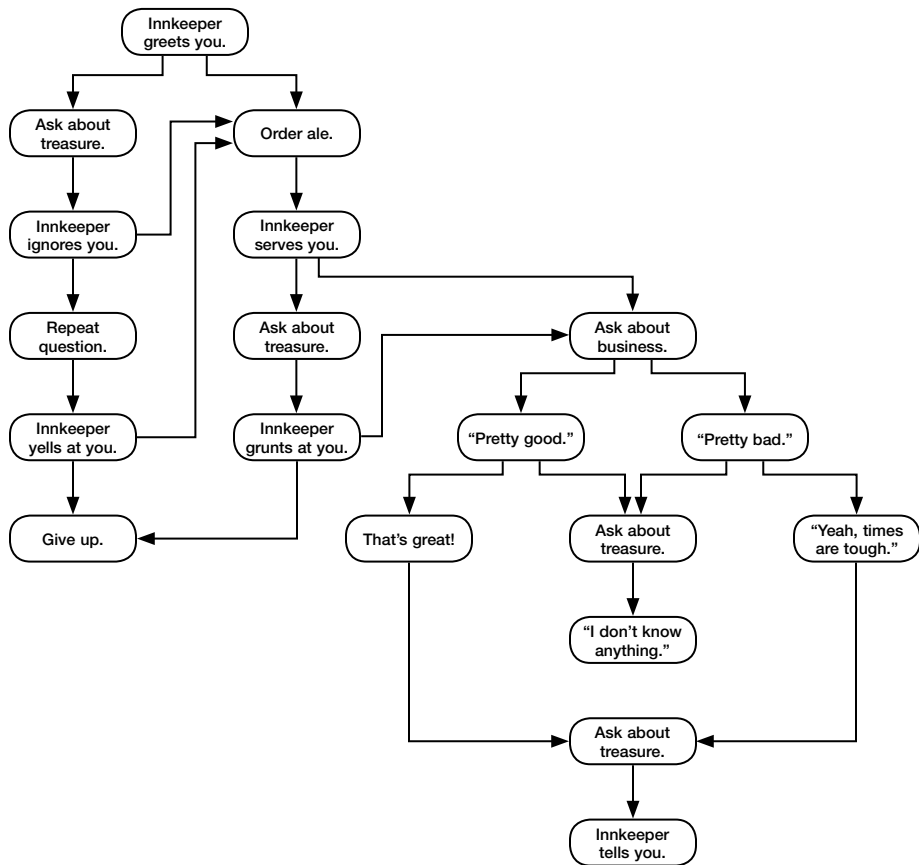


FIGURE 8.2 A typical dialogue tree

Dialogue trees do offer puzzles that entertain some people, but they certainly don't constitute anything that could be called interactive storytelling. In Figure 8.2, you're not interacting with a character in any dramatically significant fashion. Instead, you're testing a vending machine to see what buttons must be pushed for it to deliver its product.

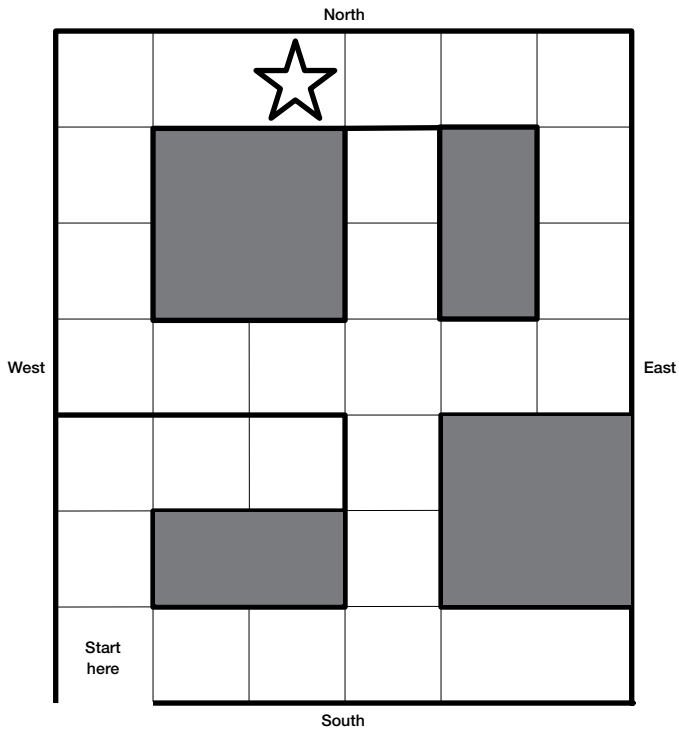


FIGURE 8.3 A very simple rat's maze

A tree is really just a different way of representing a maze; or, perhaps a maze is just a different way of representing a tree. **Figure 8.3** presents a ridiculously simple maze.

Figure 8.4 presents an analysis of how to traverse the rat's maze.

Look familiar?

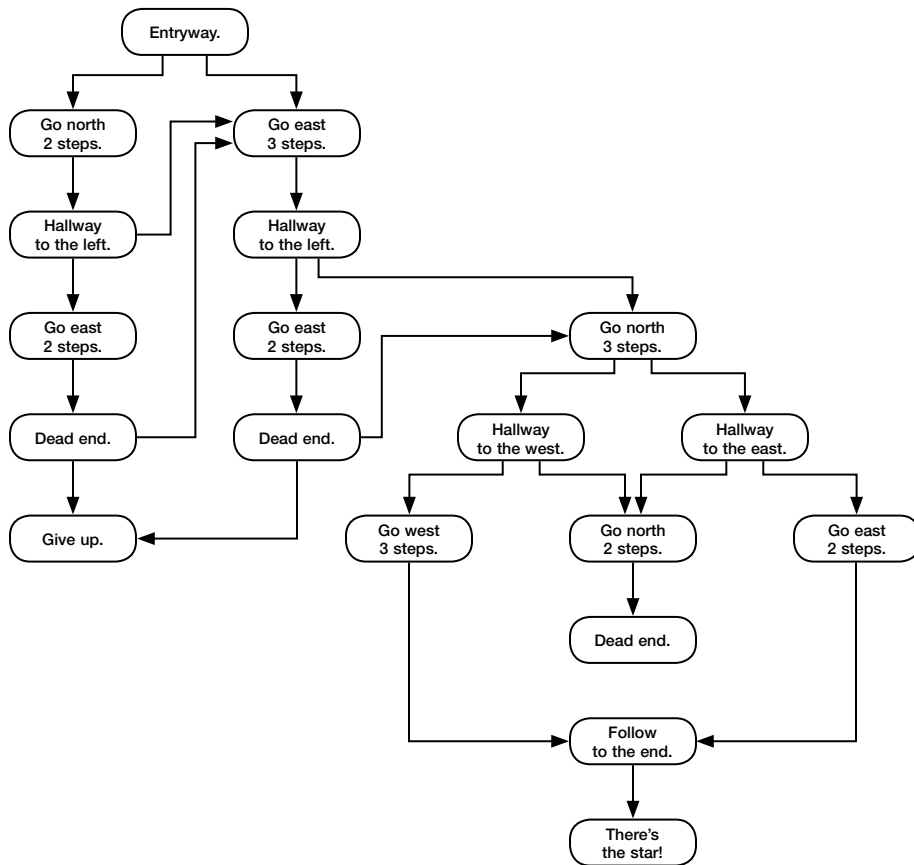


FIGURE 8.4 Maze tree

Foldback Schemes

Some clever people have tried to solve the geometric explosion problem of branching trees by rerouting the consequences of decisions so that they converge. In the example of the cave and the dragon, a clever designer might have set up the cave decision in the following way: If the user chooses to go toward the sound of trickling water, he or she encounters some minor obstacle that either deposits the player at the mouth of the cave or in the dragon's lair. I call this stunt *foldback* because it simply folds the storyline back to some predetermined path. **Figure 8.5** shows what foldback might look like:

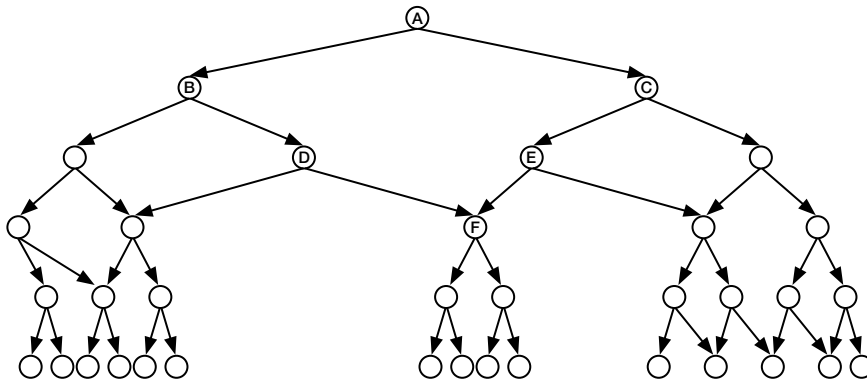


FIGURE 8.5 Six-layer branching tree with foldback

In this example, states D and E fold back to state F. You can get to F by choosing A-B-D or by choosing A-C-E. Several other cases of foldback are shown in this figure. You can see that the net effect of foldback is to reduce the designer's workload while still giving the player the impression of a broad range of choices.

Foldback does permit more decisions for the user without requiring a great deal of additional effort on the part of the author. It achieves this, alas, by fakery.

Consider closely the decision presented to the user at the mouth of the cave: Follow the sound of trickling water or the smell of smoke. That decision has no significance. Both choices ultimately lead to the same result. In other words, foldback merely tricks the user into thinking that he or she is making a choice. This is fraudulent interactivity; when users discover the fraud (as they inevitably do when they play the storyworld several times), they feel cheated.

People keep coming up with variations on the foldback system and combinations of foldback with trees, hoping that there's a magic combination that will work. There isn't. Every case of foldback renders some decision meaningless. If you have only one instance of foldback in your design, then your design is only a tiny bit fraudulent. As you add more foldback, you create many more new (but meaningless) pathways and add to the overall fraudulence of your design.

The Solution: State Variables

There is one situation in which foldback works: when internal variables alter as a result of the difference in pathways. In the example story, suppose the user took the path toward the trickling water and found a sleeping potion that could be used on the dragon, enabling the player to wrest the key away from the dragon without being cooked. Thus, the user would take the path toward the trickling water, obtain the sleeping potion, return to the main entrance, take the path toward the smell of smoke, disable the dragon, and so on. This is foldback with a boost that makes all the difference: By changing a state variable, we have rendered the player's decision meaningful.

We can examine this idea by considering the classic video game *Pac-Man* (1980), shown in **Figure 8.6**.

In **Figure 8.6**, the Pac-Man (the circle with the slice taken out of it in the lower left) has reached a junction; he must make a decision to go up, down, or back to the right. In terms of design structure, this is the same situation that a user has in a storyworld when confronted with a choice. In this situation, it appears that Pac-Man's best choice would be to go down; going up would probably lead to an encounter with a ghost, and going to the right would not collect any dots.

Now consider the situation in **Figure 8.7**.

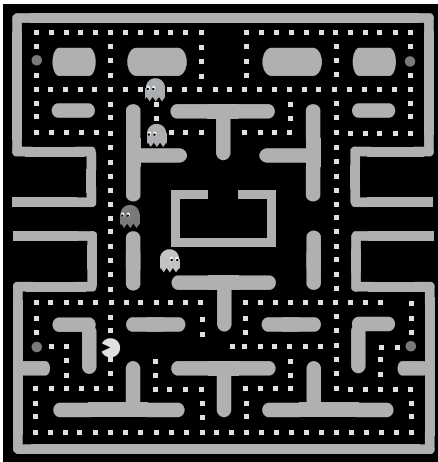


FIGURE 8.6
Pac-Man, shown early in a game

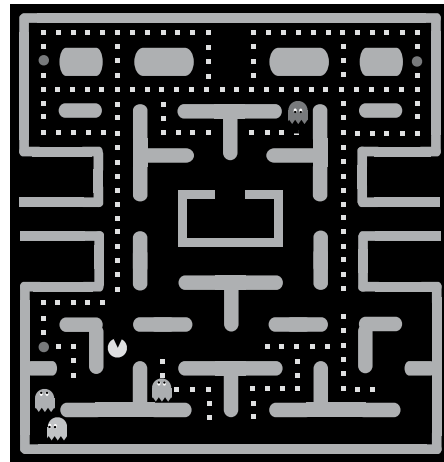


FIGURE 8.7
Pac-Man, shown later in the same game

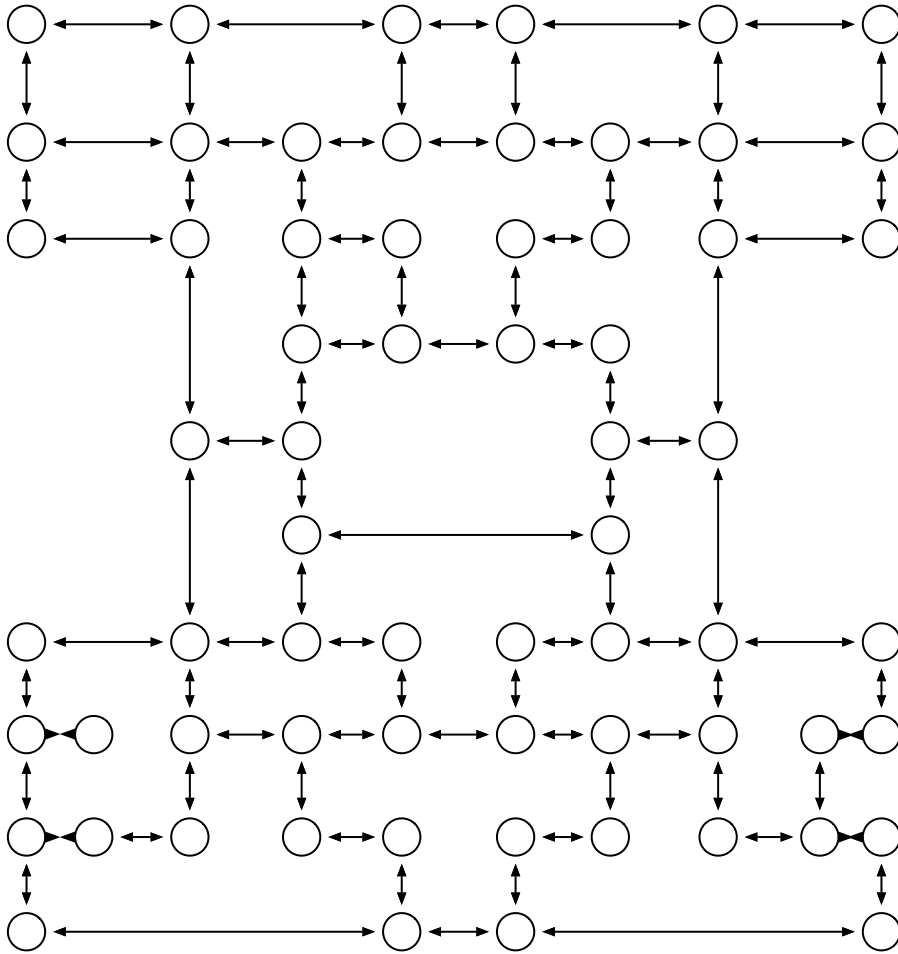


FIGURE 8.8 *Pac-Man* directed graph

Now Pac-Man finds himself right back at the same intersection, with exactly the same three choices: up, down, or right. But this time, the context is different; the positions of the ghosts indicate that Pac-Man should go up. Thus, it is possible to design a network of choices so that the player faces the same choices over and over, but they mean something different each time. **Figure 8.8** shows the network diagram for *Pac-Man*.

Computer scientists call this formal representation of the *Pac-Man* maze a *directed graph*, which is a network with arrows connecting the nodes.

If we reject turning around in the middle of a pathway and reversing course, *Pac-Man* offers its player exactly 23 choices. Of those choices, 19 are binary; 4 intersections offer 3 choices. That's all there is. Instead of creating a thousand different branch points, the designer of *Pac-Man* created only 23. It's the context that makes those choices variable in meaning. We can quantify that context. There are 4 power pills and 270 edible dots. Each of these has just 2 states—eaten or uneaten—which requires just 1 bit of information to store. That's 274 bits. Next come the 4 ghosts and Pac-Man himself; each of these has a position that could be stored as just 9 bits of information. That's a total of 45 bits for the 5 movable characters. Add up everything and we get a grand total of 319 bits of information. That's the data context of *Pac-Man*. It may not sound like much, but it amounts to 10^{96} differentiable states. Yes, 10^{96} is a large number—ridiculously large. The designer of *Pac-Man* built a system with a tiny number of branch points—but made it rich by giving it 319 bits of context, which transformed a simple maze into something much more interesting.

Interactive storytelling is not so mechanical a subject that it can be handled with simple geometric mazes and up/down, left/right decisions. The systems we build for interactive storytelling cannot directly use a system as elementary as that used in *Pac-Man*. But we can use the example of *Pac-Man* to guide our efforts. We cannot return our player to the same decision point a hundred times, but we can return him to similar decision points several times. For example, consider the layout of a romantic storyworld; surely kissing plays a large role in such a storyworld. What's wrong with presenting the player with an incipient-kissing situation several times during a storyworld? The first time, it's a clumsy oaf who wants to kiss her; the next time, it's a smooth operator; the third time, it's the clumsy oaf who has since turned out to be quite sweet. It's the same situation in three completely different contexts.

In general, branching stories don't work, even with foldback. But if you twist a branching system around enough, you can transform it from a tree into a directed graph. If you then set up appropriately differentiating context, you can have your players move through the network of dramatic possibilities, revisiting each point with a different context each time.

Conclusions

There's an old tale, almost certainly apocryphal, about an Indian mathematician who invented the game of chess. He presented it to the king, who was so impressed with the game that he invited the mathematician to name any gift he desired.

The mathematician humbly asked for a little rice: one grain of rice on the first square of the chessboard, two grains on the second square, four grains on the third square, eight on the fourth, and so on until the entire chessboard was full. The king was taken aback by this modest request; he asked if the mathematician didn't want gold, or jewels, or land. No, the mathematician told him, just the rice. So the king ordered his steward to dole out the rice. Shortly thereafter, the steward returned and explained that he had already doled out three tons of rice and was only on the 27th square. Fulfilling the mathematician's request would require more rice than had been grown in human history. The king was astounded.


That's the problem with geometric growth: People can't grasp how quickly it grows to astronomical sizes. Branching trees also grow exponentially; to build a decent branching tree requires hundreds of thousands of nodes.

Branching trees are the most obvious way to build interactive storyworlds, which is why everybody tries them. In fact, I recommend that you try to build one, because that's the only way you'll really learn how futile the effort is. But perhaps you could build something like a branching-tree haiku: something tiny yet elegant. Good luck!

CHAPTER 9

Interactive Fiction

INTERACTIVE FICTION is something of a parallel universe to video games. The two genres are so different that interactive fiction thrives in complete isolation from the video games universe. Few video game aficionados play interactive fiction, and few interactive fiction enthusiasts play video games. Interactive fiction has matured on its own path and has come a long way since its earliest days.



Text Adventures

Interactive fiction all started with a game called *Adventure*, which was built for mainframe computers in 1976. Because it was played on conventional computer terminals, which had no graphics capability, it was implemented entirely in text. The user typed commands, and the software responded with text. That basic model of interaction remains unchanged today.

As the player, you wandered through a labyrinth of caves, encountering oddments and struggling with various obstacles. It was all cleverly designed so that if you carried out the correct sequence of actions, you could penetrate the deepest levels of the caves and win the game.

The concept was much improved with another mainframe game called *Zork* in 1979. At about the same time, personal computers were just beginning their ascent, and versions of this class of software quickly appeared. Because they were based on the original *Adventure* game, and were implemented entirely in text, they were called *text adventures*. The 1980s saw rapid improvements in this genre, both in the technology of the software and its content.

The biggest problem with text adventures was the parser puzzle. Although the software seemed very smart in responding to your typed entries, it used a small dictionary of words, and if you used a word not in that dictionary, it responded with “I don’t understand that word”—an occurrence common enough to drive some players to apoplexy. The dictionary was hidden, so you had to guess what word might solve your problem. This could be especially infuriating when obvious synonyms were mishandled. I recall one case in which the text referred to a rock. I typed, “Pick up rock,” and it responded, “I see no rock here.” After much experimentation, I discovered that the correct entry was “Pick up stone.” I’m still gunning for the author of that text adventure.

Graphic Adventures

In the late 1980s, text adventures were supplanted by *graphic adventures*. These were structurally identical to text adventures, but they replaced the text with imagery. Instead of typing “Go west,” you moved your avatar to the left. Instead of reading lushly worded descriptions of places and things, you saw low-resolution images of them augmented with some textual description. These were considered superior to text adventures, which (mostly) shriveled up and died.

With graphic adventures, the puzzles became visual. Instead of reading and rereading the text, you scanned the image pixel by pixel, looking for something odd that might be active. When you found a likely candidate, you could “examine” it to see a close view, which would usually reveal the hidden key, magic potion, or other important item.

Initially, the graphic adventures were limited by the poor resolution of computer monitors. In the early 1990s, however, the graphics capabilities of personal computers improved dramatically. So when *Myst* appeared in 1993, boasting truly lavish images generated on big Silicon Graphics workstations, the world went gaga over the game. *Myst* featured a large and complex world and, for once, a less puerile story. But it was still essentially a large, beautiful collection of devious puzzles. It was, for a time, the most successful game in history. But then the craze over beautiful graphic adventures such as *Myst* died out and people moved on to the next fad.

IF: Son of Text Adventure

The text adventure, however, wasn’t completely dead: It had shivered in the cold for a few years before two new developments changed everything. The first of these was the release of easy-to-use development environments (vaguely like “programming languages”) for making text adventures. These programs had been around for many years. I remember using one for the Atari home computer in 1979. But this new generation of development environments was more powerful and easier to use, encouraging authors to push the envelope. The second, and more important, development was the entry of a completely new and different class of authors. These were literary types interested in exploring the narrative potential of text adventures. They saw it as a new literary medium crying out for investigation, and they rose to the challenge. They heralded this fundamental shift by adopting a new name for their baby: *interactive fiction*, or IF, as everybody now calls it. A new community of aficionados crystallized around the idea, and it quickly grew into a vibrant (if rather small) school of thought.

Especially striking is that IF is the first—and so far, the only—computer medium of expression that is dominated by artsy people rather than techies. We can learn a great deal by comparing the artistic development of IF with that of video games. Of course, video games are big business and, therefore, are driven by the need to give people what they want. (This commercial aspect only aggravates the vaulting

crassness of video games.) By contrast, IF has never had any commercial success; it is, therefore, the work of *amateurs*—people who do it for love, not for money.

The nature of the IF writers and the lack of commercialism has given discussions of IF a scholarly tone. Literature about IF is intellectually deeper than the corresponding work on video games. The vocabulary used in IF is several grades higher than that of video games. Should we ever reach the day when obscenely wealthy people, seeking to advertise their good taste, pay ridiculously high prices for impressive works of interactive art, they'll be buying IF, not video games.

IF cannot be read like a book; it has to be played on a computer. The software for reading IF is free to download; in fact, most IF products are free to download.

Puzzles

Puzzles have always comprised the essence of the text adventure genre. At first, the puzzles were not too difficult to figure out, but over the years they grew increasingly devious. The craziest example of this tendency was the infamous “babel fish puzzle” in the 1984 game *The Hitchhiker's Guide to the Galaxy* by Infocom. In this game, it was necessary to obtain something called the “babel fish.” This could only be accomplished by the following sequence of actions carried out in the correct location:

1. Take off the gown.
2. Hang the gown on the hook.
3. Get the towel.
4. Put the towel over the drain.
5. Wait until Ford is asleep.
6. Take Ford's satchel.
7. Put Ford's satchel in front of the panel.
8. Put the junk mail on the satchel.
9. Press the dispenser button.

While this puzzle was generally considered to have gone too far in the direction of weirdness, this was the kind of thing that text adventure players reveled in: puzzles with totally crazy solutions. Eventually, a backlash arose; nowadays, puzzles are expected to have rational solutions that are in some fashion suggested by a careful reading of the text.

A Real Example

Here's an example from a highly regarded work, *Make It Good* (2009), by Jon Ingold. This is the printout of the unedited text generated by the system; my own entries are prefixed with the greater than (>) sign. You can readily see that the writing is more mature than that of the typical video game; there's obvious literary quality here.

Broken Top Boulevard, Outside No. 15 (in the black chevy)

The boulevard through the windscreen is lined with ash trees, thick trunks casting shadows and gnarled roots mangling up the sidewalk. You're sitting in your car, parked too high up the kerb; just outside the gate to No. 15. Just an ordinary house. With a body inside.

"Homicide. One Jack Draginam, accountant. Married, no kids. Stabbed. Yadda yadda, blah blah. We got the call from the maid - geez, who has a maid? Apparently she wanted to stress there's a lot of blood."

"Oh, Inspector. Word is, if you don't crack this one, you're out of a job."

The glove compartment is closed. Sat on the passenger seat is a whiskey bottle.

>open glove compartment

(unlocking it first - with the car key)

You pull open the compartment, and it catches after only opening a tiny crack. Stupid thing.

>go inside

(opening the black chevy first)

You open the car door.

You clamber out of the driving seat.

Broken Top Boulevard, Outside No. 15

The boulevard is lined with ash trees, thick trunks casting shadows and gnarled roots mangling up the sidewalk. Your car is parked too high up the kerb; just outside the gate to No. 15. Just an ordinary house. With a body inside.

There'd better be some faces too: you need the slam. What did the peep at the station say? *Inspector, the only case you cracked open last year was the Michaelmas liquor.* He has a point. If you come up blank (or facedown in a gutter) you'll be off the payroll. And then you'd really have a problem.

Down the street there's a dog, running about.

(opening the front gate first)

The gate swings creakily.

You step into the porch.

Front Porch

A tasteful little porch; white painted wood strapped onto red-brick building. The front door is north, and through a small diamond of frosted glass you can make out nothing of the wood-panelled hallway inside. Your neck is sweating; sunlight hides behind the trees to the south.

A hanging basket with a couple of petunias in is attached just to the left of the door.

>go inside

You knock a smart rap on the door.

After a short wait the door opens, Angela's face appears, then the rest of her. She looks just like she should considering, no time to do her face.

"Yes? What do you want?" she says uncertainly.

"Can I help you?"

>show badge

(to Angela Draginam)

"Oh, Inspector! I'm sorry.. do come in, I'm glad you came so quickly." You follow her into the house. It has a smell you recognise, all right.

In its current form, IF is much superior to the old text adventures. The parser puzzle problem has been almost banished. In the bad old days of text adventures, you wasted hours figuring out by trial and error (mostly error) the contents of the dictionary; nowadays, IF software is kind enough to reveal the dictionary.

Moreover, IF software can handle several ways of saying the same thing, and can understand a goal as a sequence of steps. In the sample text on page 131, I entered the command `go inside`, which it understood as a sequence of commands: `open car door`, `open gate`, and `enter porch`. We have every reason to expect that the intelligence in IF software will continue to advance, permitting better understanding of text entries.

Even so, the syntax of your entries is stilted; you must enter your command in a standard format with little latitude. It can figure out a few simple variations on its format, but the intelligence of the IF software (indeed, of all natural language processing software) falls well short of the syntactic freedom we normally expect of our human interlocutors.

IF no longer challenges the player with weird, stupid puzzles; nowadays, it is expected that the puzzles will make sense and that they should be solvable by a careful reading of the text. In many cases, the puzzles can be solved only with the exercise of deeper insight into human nature.

Under the Hood

A *development environment* is a big program that allows authors to create software. Programmers have huge development environments that provide lots of power. You could call a word processor a “development environment for documents,” but that would be stretching the definition, because the term normally applies to programs that allow authors to build interactive software. Because IF is interactive, it must be programmed and therefore requires a development environment. One of the easiest ways to figure out the capabilities of a technology is to look at the development environment that it is built with. If you can see the artist’s tools, you get a clear idea of the range of possibilities.

Inform 7

Of the number of development environments for IF, the most popular is Inform 7. The software is now (obviously) in its seventh generation and has been in use since 1993. It is a sophisticated program that packs a lot of power. Programs written in Inform 7 read almost like normal English:

"Feline Behavior"

The Kitchen is a room. The cat is an animal in the Kitchen. In the Kitchen is a bowl, a ball of wool, a newspaper. The bowl contains a quantity of cream.

The cat is wearing a silver collar. The description of the cat is "It is wearing [a list of things worn by the cat]."

The player carries a closed openable container called a bag. The bag contains catnip.

The cat behavior rules is a rulebook producing an object.

A cat behavior rule when the cat can touch the catnip:

say "The cat frolics with the catnip until nothing remains of it.";
rule succeeds with result catnip.

A cat behavior rule when the cat can touch the cream:

say "The cat laps up the cream.";
rule succeeds with result cream.

A cat behavior rule when the cat can touch the ball of wool:

say "The cat makes the ball of wool into a useless tangle which must be discarded.";
rule succeeds with result ball.

A cat behavior rule when the cat can touch the newspaper:

say "The cat bats playfully at the newspaper until all the nasty boring articles are destroyed.";
rule succeeds with result newspaper.

A cat behavior rule:

say "The cat looks miffed at the lack of ready entertainment, and glares at you with yellow eyes as though wondering whether your pants leg is good for claw-sharpening.";

rule fails.

Every turn:

let the destroyed object be the object produced by the cat behavior rules;
if the destroyed object is not nothing:
remove the destroyed object from play;
say "[line break]Good thing you have no use for [the destroyed object] yourself.[paragraph break]".

Test me with "z / z / open bag / z / z".

This little program creates a small scene in its entirety: a kitchen with a cat, the player, and a number of cat toys. The cat will play with the toys until they are all gone, at which point the cat shoots a nasty glare at the player and the game ends.

This tiny demonstration does not do justice to the full range of capabilities in Inform 7; the purpose of showing this text is to demonstrate that the language used in Inform 7 is very easy to read. It's a bit harder to write, because (like any programming language) you can't use just any old word. The Inform 7 language uses a set of precisely defined keywords, each of which provides a special capability. The prospective author must memorize all the keywords and learn how to apply them productively. This is not an especially odious burden as every development environment requires as much. Inform 7's environment, however, is one of the easiest such environments I have encountered.

Expressive Power of Inform 7

Like every development environment, Inform 7 is optimized to carry out a certain set of tasks. It is possible to execute tasks for which Inform 7 is not optimized, but that requires a lot more work. Inform 7 is optimized for the following tasks.

Space

The author can create an indefinite number of stages (called "rooms" in Inform 7) that are connected to each other in any manner the author wants. Movement from one stage to another is usually instantaneous, but can be delayed as desired. The stages are set up on a handy-dandy map that allows easy editing. Players move from stage to stage with the standard command *Go X*, where *X* is a compass direction. They can also use *Go to Y*, where *Y* is the name of a stage. If possible, Inform 7 will navigate the player through intermediate stages to reach the stated objective.

Time

Play is turn-sequenced: Every time the player enters a command, one turn elapses. Authors can provide for delays of a specified number of turns.

Input

All input is through typewritten commands. The commands are similar to English. The author can define an indefinite number of words the player can use, and those words can be assigned meanings. Inform 7 automatically recognizes the basic properties of certain classes of words. For example, a “room” is automatically understood to be a location that can contain the player and a variety of objects. The player must guess the correct words to use. If the player uses words that the system doesn’t recognize, it will respond with an error message.

Output

The output to the player is also text. Inform 7 boasts many powerful capabilities for making the text as natural and colorful as the author desires. It can automatically alter the text to fit the context, using pronouns as necessary. The text-handling capabilities of Inform 7 are one of its great strengths.

Props

The author can create an indefinite number of props, place them in a room, or arrange ownership. Props have many built-in capabilities: They can be small enough to carry or too large to carry. They can be designed to contain other props. They can be assigned any custom trait the author desires, and that custom trait can then be used to control the way they affect the game. The innate prop-handling capabilities of Inform 7 are another of its great strengths.

Behaviors

The author can establish almost any behavior for any prop or room. It can be triggered by any set of conditions the author pleases, including past activities, the presence or absence of particular props, or the values of numeric variables.

Upsides, Downsides

A computer scientist once proved that any set of operations that could be programmed could also be carried out in a spreadsheet program. In other words, spreadsheets provide a computational environment that can handle any problem. Of course, nobody rushed to start coding things up in their spreadsheet programs. A job that could theoretically be executed on a spreadsheet might well

prove to be so tedious or clumsy as to be impractical. For example, you could equip a spreadsheet program with a plug-in that uses the values of a two-dimensional array to draw pixels on the screen; each cell represents one pixel. You could then write formulae for the pixel cells that calculate what the pixel should show. With a few million of these cells, you could draw graphics on the screen!

The same restriction applies to every development environment. For example, every computer program ever created could be written in an assembly language, because assembly is the low-level language internal to a particular CPU device. But assembly is so difficult to use that the great majority of programs are written in a higher-level language. Nobody uses assembly unless they absolutely, positively must do so.

Every programming tool, whether a language or an entire development environment, was designed to ease the workload for a specific type of problem. FORTRAN is an ancient computer language optimized for scientific computations. BASIC, Pascal, and Pilot are all designed to teach programming concepts. Nowadays, C# is the language of choice for many desktop programs, with JavaScript or PHP preferred for web-based programs.

But every strength implies a weakness; programs that are optimized to do one thing well will surely end up doing something else poorly. Inform 7 is not immune to this ironclad law of programming: It does some things well and some things poorly.

I have already commented on some of the great strengths of Inform 7: its natural-language format, its powerful text handling, and the way that it automatically handles lots of obvious yet mundane tasks involving props. So, on to the weaknesses.

Numeric Calculations

The first major weakness, in My Arrogant Opinion, is its poor handling of numeric calculations. Yes, it can handle the four basic operators (add, subtract, multiply, and divide) as well as square root. You can do a lot with these. However, mathematical formulas are clumsy to write in Inform 7, and anything more than a handful of numbers becomes a nightmare to keep track of. Another development environment for IF, called TADS, does not have this problem with math.

Boolean Bias

Related to the difficulty of creating numeric calculations is Inform 7's predilection for Boolean concepts. This is fine for props that can be either broken or unbroken, light or heavy, hidden or visible, and so forth. But it gets a bit silly when applied to human relationships. Does Jane either like or dislike John? Is Fred either friendly or aloof? Is Tom a coward or a hero? Human relationships are never black and white; they are always shades of gray. The Boolean system, however, preferred in Inform 7 encourages the author to approach everything as either black or white.

I hasten to note that Inform 7 was designed for writers, not scientists, and so we should not expect its computational powers to be Herculean. But neither should we ignore the importance of this weakness.

No Structure

Another weakness lies in its lack of provisions for high-level programming architecture. Modern programming languages reflect the bitter experience of generations of programmers who have struggled with programs that become incomprehensible as they grow larger. These languages make it easier to write big programs without getting lost in a maze of instructions. Not so with Inform 7. Although it has produced some truly elegant structural systems, any world big enough to be interesting will surely be confusing to work with.

For example, the source code for a good storyworld called *Alabaster* is organized like an outline with 12 high-level entries, at least 60 mid-level entries, and a few score low-level entries. Each entry comprises about a page of code. It is about 2000 lines of code—a middling size program by regular programming standards. Reading the code is like traversing a maze that has references to things that you can't readily find. It's a mess.

Is It Interactive Storytelling?

In terms of interactive storytelling, *Blue Lacuna* (2008) is, as of 2012, probably the best that IF has to offer. The player wanders around a small island populated by one other person: a madman. By cleverly constraining the situation, the author makes it possible to provide genuine emotional interaction with the madman.

Since he's crazy, he doesn't have to make much sense—but he still has an impressive palette of emotional states, and he does respond to the player's actions in an emotionally consistent fashion. My impression is that the author had to push the development environment very, very hard to achieve these dramatic effects.

IF has become a serious form of literature. It does not, however, constitute interactive storytelling in my opinion. The answer to the fundamental question, “What does the player do?” always boils down to “Solve puzzles.” The puzzles these days are more literate and more meaningful; sometimes they advance the narrative in a useful manner. But the player doesn't interact primarily with characters; the player interacts with puzzles.

The greatest limitation of IF lies in its weak interpersonal interaction. You never engage a character in open conversation; instead, you traverse a dialogue tree (see Chapter 8, “Branching Trees”). The branching decisions are based on simplistic considerations that seldom take into account the emotional state of the character or his or her relationship with the player. (*Blue Lacuna* constitutes an important exception when it comes to relationships, but only because it relies on the contrivance of a madman who doesn't have to always make sense.) It is possible to establish some weak interpersonal interaction in IF, but doing so is a tedious process that discourages extensive interaction.

A good example of this limitation is in *Make It Good*, another highly regarded work of IF. The player is a detective attempting to solve a murder. He eventually figures out the most likely killer, but he has no proof and needs a confession. To induce the killer to confess, he frames her lover by falsifying evidence. Confronted by what appears to be damning evidence against her lover, she breaks down and confesses. At a superficial level, this all looks like solid drama. But it is all based on a clever contrivance: the faking of evidence. That's a purely mechanical process: The player fabricates footprints in the mud, sneaks a knife past a character to put it in an incriminating location, and wipes away fingerprints. By manipulating *things*, the player indirectly manipulates *beliefs*, which in turn trigger *emotions*. The player never engages in direct emotional interaction with a character. I applaud the cleverness with which the author managed to bludgeon the medium into an indirect reflection of drama—but the achievement is so contrived as to reveal the fundamental inutility of IF for genuine interactive storytelling.

The Future of IF

Interactive fiction will surely continue its growth as a medium of expression. We'll see steady improvements in its handling of language, permitting it to handle more complex expressions. Related to this will be improvements in the artificial intelligence to enable more compound expressions—for example, “Clean up the room” might be understood as a command to wipe up the blood, dispose of the bloody rag, return objects to their original positions, and straighten out the rug.

These improvements will not be enough to empower IF to tackle interactive storytelling. To accomplish that, IF will need a number of improvements. First, it will need ways to easily carry out more complex computations, especially numeric computations. Second, it will need to establish some standard schemes for personality modeling (see Chapter 14, “Personality Models,” for more on this). Third, it desperately needs to impose high-level structural elements on the coding system; the current systems are just too clumsy to build anything as complicated as a storyworld.

All of these things could certainly happen; there's a large enough and experienced enough cadre of IF authors to work toward these next-generation goals. The question is, will the leading members of the IF community start pushing for these new developments?

Conclusion

IF has long tempted artists as a possible path to rich interactive storytelling. Yet, after more than 30 years of development, with thousands of talented people experimenting with it, the medium has yet to produce anything that I would call interactive storytelling. It has its merits: You can make fun puzzles with it, and some artists have refined it as a form of literature. But deep down inside, so far it's about rooms and props and mechanical manipulations of these things. Although the text can burst with feeling, in its interactivity, IF has no heart.

CHAPTER 10

Role-Playing Games

ROLE-PLAYING GAMES (RPGS) predate computers. The first RPG was a paper game called *Dungeons & Dragons* (1974). A group of players formed a party exploring a dungeon under the aegis of the *Dungeon Master*, who served in the role I call Fate (see Chapter 15, “Fate Makes the Storyworld Go ’Round”). The game consisted of a large number of rules specifying how the responses to player actions would be calculated. The Dungeon Master (DM) had considerable leeway in interpreting the rules. As the RPG genre developed, DMs exercised greater editorial control over the game, designing their own dungeons and developing extensions to the standard rules. The genre became wildly popular, with lots of different kinds of role-playing games set in different contexts.

The genre relied on the talents of the DM: A good DM could provide an exciting game experience, but a bad DM could ruin the game for the players. It was therefore natural to attempt to build a software DM by computerizing the game. This was first done, to the best of my knowledge, on the Plato computer network in 1975. Don Daglow programmed a version for the PDP-10 shortly afterwards. The advent of personal computers triggered an explosion of software RPGs; the genre is now one of the most popular among computer games, both online and offline.

Some people believe that RPGs could provide a platform for the evolution of interactive storytelling. Many RPGs have narrative elements woven into the gameplay. Yet so far, we have not seen anything that can be called interactive storytelling emerge from that direction. Why not?

The Verbs

As always, the first thing to do when analyzing a piece of software is to invoke Crawford's First Law and ask, "What does the player do? What are the verbs?" The verbs in RPGs, in order of frequency of use, can best be summarized as:

- ▶ Move around the 3D environment
- ▶ Fight monsters
- ▶ Pick up useful items
- ▶ Obtain useful information from friendly NPCs

That's about it.

Consider now the dramatic content of these verbs. Consult your mental database of great stories. How often in these great stories does the protagonist move around in a 3D environment, fight monsters, pick up useful items, and obtain useful information from other characters? And when the protagonist does such things, how important are they to the story? Sure, such concepts do occasionally show up in great stories, but they never assume much importance.

For example, Odysseus travelled around the Mediterranean Sea, but his travels are little more than segues from one adventure to another. He does a little traveling and stumbles onto the land of the Cyclops, where he has a long adventure. Then he goes to the land of the Lotus-Eaters, then somewhere else. The travel is no more important to the story than a semicolon is to a sentence.

Yes, Odysseus fights monsters: the Cyclops, Scylla and Charybdis. But he doesn't triumph over them; he merely evades them. Ultimately, the monsters are a small portion of the story. He has another big fight at the climax of the story, but it's not a fight so much as a one-sided massacre.

Nor does Odysseus spend any time picking up powerful or even useful items. Yes, he gets the bag of winds from Aeolus, but it serves only to provide one more tragedy to the tale. The same thing goes for obtaining useful information from other characters; his interactions with others are vastly more complex than a simple Q&A session.

I conclude that the things the player does in an RPG don't have much in common with the kinds of things that characters do in stories. The verbs are all wrong.



Why couldn't we expand RPGs to include dramatically interesting activities for the player? There's nothing about RPGs that precludes such refinements.

This has been the fond hope of RPG designers since the 1980s. Despite many efforts in this direction, the genre has stubbornly resisted all such efforts.

Little Room for Creativity

The RPG is now a well-defined genre. Players know exactly what to expect in an RPG. Any deviation from that ideal is resented and resisted; products that drift too far from the mainstream are punished with commercial failure. If RPGs were still a nascent genre, if people had broader expectations of what might be considered an RPG, then it would be possible to inject greater narrative content into these products. **Figure 10.1** presents a visual allegory of the problem.

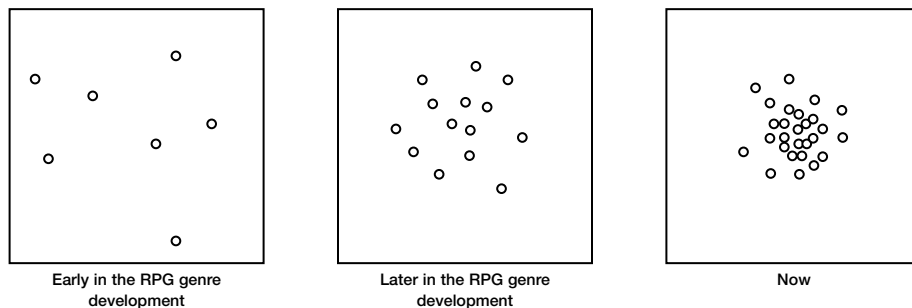


FIGURE 10.1 Design enhancement spaces for RPG games over time

Each box in Figure 10.1 represents a design space—the range of design possibilities for a game. In the early days, nobody quite knew what an RPG game was, so there were lots of crazy experiments, many of them not so good. But with time, people gravitated toward designs that were more entertaining, and the designers zeroed in on basic design elements. Nowadays, the basic RPG design is so clearly established that most designs stay close to that central zone.

One way to illustrate this phenomenon is by looking at how RPG games are described by their designers. I went to a website devoted exclusively to RPG games on the Kickstarter crowdfunding site: <http://rpgkickstarters.tumblr.com>. Here are some unedited self-descriptions from that site:

...an online tactical RPG, a fusion of the real-time strategy and roguelike genres

...a unique RPG system where the core mechanic uses playing cards instead of dice to determine success or failure.

...a storytelling RPG where you and your friends get together and create a haunting story in the vein of HP Lovecraft's works

...a horror role playing game

A tabletop-inspired roleplaying game..,

...a Post-Apocalyptic Zombie RPG that takes place in modern times

...a massive 27-level mega-dungeon more than half-a-million words long!

A high-action high-drama fantasy RPG originally from Japan!

... a rune-based RPG set in the Viking dark age of Ragnarok!

...a gritty fantasy RPG of dramatic action, divine favor, high magic, and epic narrative.

...a competitive fantasy role-playing game, where Sword and Sorcery meets 1980s heavy metal album covers.

...a Fudge RPG that blends cyberpunk with psionics for a fast, simple, fun, and diverse gameplay experience.

My point is that these brief descriptions are written with the expectation that the reader will understand exactly what the game provides, demonstrating just how tightly defined the genre has become. A tightly defined genre doesn't have a lot of room for evolutionary change.

The Dangers of Overspecialization: Koalas vs. Goats

Here's another allegory, this one based on evolution. My starting point is the genus of eucalyptus trees. This remarkable genus has evolved a powerful toxin, eucalyptol, which does an excellent job of keeping herbivores at bay. This stuff is so powerful that few bugs and only a couple of vertebrates can digest it safely. Because of this, eucalyptus trees are quite successful when transplanted to new countries.

One of the few animals that can cope with eucalyptol is the koala, a truly astounding creature. The biochemical cost of separating the toxins from the nutrients in the eucalyptus leaves is just a bit less than the biochemical value of those nutrients, so the benefit of eating eucalyptus leaves is razor-thin. Most creatures that eat toxic substances accomplish their trick by using lots of water to wash the toxins out of their systems.

In most evolutionary settings, the overall caloric cost of eating eucalyptus leaves just wouldn't be worth the effort, but for one factor: The supply of eucalyptus leaves is stupendous. It's like a gigantic untapped market; any entrepreneurial animal that could come up with a way to profitably tap that mountain of nutrition would have it made.

Enter the koala. It evolved a number of mechanisms for managing the problem of eucalyptol toxicity. It developed a super-duper liver specialized for metabolizing the eucalyptol and a metabolism capable of extracting the nutrients from the leaves. Most of all, it developed a super-low-energy lifestyle that balanced all the requirements.

It sleeps 18 hours a day; it moves slowly and requires an absolute minimum of activity. These critters make couch potatoes look frenetic.

Koalas have absolutely no defense against predators other than their arboreal lifestyle. They have no camouflage, no defenses, and they can't run away. They also cannot regulate their temperature like other mammals; they are dependent upon the environment to maintain a safe range of the temperatures. If the temperature changes beyond that range, the koala dies.

The koala is a wondrous creature highly specialized to survive in the harsh environment of the eucalyptus forest.

Now let us turn our attention to goats. While koalas are specialists, goats are generalists. Koalas eat only eucalyptus leaves; goats eat pretty much anything that grows. They eat grass, weeds, scrub, or bark. With their close relatives, the various species of sheep, they flourish all over the world. They aren't particularly good at anything. They aren't fast, they aren't big, they aren't powerful, they aren't well camouflaged, and their digestive systems aren't anything to write home about. They can't climb trees, digest poisonous plants, extract maximum nourishment from grass, or dig safe dens. They have no armor plating, they're not particularly intelligent, and they aren't dexterous. Goats are just about the most mediocre mammals on the planet.

Let us now consider what will happen when the earth's climate changes over the next hundred years. What will happen to koalas and goats?

Koalas will probably go extinct. If it gets hotter, they will no longer be able to maintain the delicate biochemical balance that keeps them alive. If it gets colder, they'll not be able to maintain their body temperatures and will succumb. If it gets drier, the water they get from the eucalyptus leaves will be reduced and they'll die. If it gets wetter, the eucalyptus trees will be supplanted by other species and the koalas will die.

In other words, the koalas are so highly specialized that they have no place to go from where they are. They have reached an evolutionary dead end. They'll be fine as long as nothing changes, but any significant environmental change will drive them to extinction.

Goats, by contrast, have nothing to worry about. If it gets hotter, they'll shed some of their fur and scrape out a living in desert environments just as they've always done. If it gets colder, they'll just grow thicker coats and live like the goats in high mountains. If it gets drier, they'll be fine eating bits of weed and scrub

brush. If it gets wetter, they'll do fine on grass. No matter what happens, goats will be able to cope.

RPGs are like koalas. They are highly specialized to a particular set of expectations. They have a precisely defined marketing environment for which they are perfectly honed. In their own way, they are just as impressive as the koala. But, like koalas, their capacity for evolving in new directions is limited. They sit at an evolutionary dead end, unable to adapt to new environments because they're overspecialized.

What About MMORPGs?



You're ignoring MMORPGs. There's lots of good interactive storytelling taking place in those products.

True, MMORPGs (Massively Multiplayer Online Role-Playing Games) have a lot more dramatic content than conventional computer RPGs. But take careful note of the source of that dramatic content—it doesn't come from the content of the MMORPG itself. The drama comes from the players and it takes place primarily in the game's chat room. Players do mechanical things in the game itself; they attach dramatic significance to those mechanical actions in the chat rooms. All the drama comes from what they write in the chat rooms. Without chat rooms, MMORPGs would not be any more dramatic than regular RPGs.

Such games have been proven to provide dramatic satisfaction to the players; just look at how successful the good ones are. Yet they suffer from two drawbacks: play skills dependency and no author storytelling control.

Player Skills Dependency

For MMORPG, the dramatic quality is entirely dependent on the dramatic skills of the players. If you interact with a group of sophisticated players, you can get some great interactive storytelling. But it only takes one screwup to mess things up. A large vocabulary of terms is dedicated to describing players who detract from the quality of the experience. And even if the players are all competent storytellers, the differences in their dramatic interests can result in narrative chaos.

Many years ago I read a fascinating experiment that powerfully demonstrates this problem. A creative writing teacher had broken up a class into pairs, requiring each pair to write a story by alternating sentences. My worm-eaten memory is too weak to remember the specifics, but the result ended up something like this:

Marie lounged on the couch, staring out the window at the dreary rain and sipping her herbal tea, musing if things with Jason could not have come out better.

Jason checked his equipment for the umpteenth time as the spaceship neared the enemy planet. This was going to be a nasty fight.

Marie shook her head silently. No, she thought, Jason had never loved her; he was too obsessed with proving his manhood to love anybody.

The thought of Marie flashed through Jason's mind. She was just too self-centered to understand the obligations a man must face. A man's gotta do what a man's gotta do.

And so the story went, with the two authors battling for control of the storyline. Jason's author eventually killed off Marie in an enemy attack; Marie's author retaliated with Jason marooned on a desert planet, bereft of human companionship. That's what happens when people with different narrative desires engage in interactive storytelling.

No Authorial Contribution

The second drawback of this chat room storytelling is that the authors don't get any opportunity to express themselves—everything comes from the players. The author's only task is to create an environment that permits exciting events to take place. The author can certainly seed the world with sources of conflict, but the way those conflicts play out is not within the author's purview. These elements provide the grist for the dramatic interaction between players in the chat room. This leads to “averaged out” storytelling, with narrative that tends to follow the same general lines in all cases. The stories coming out of MMORPGs have a sameness to them, like a diet consisting of just hamburgers of differing shapes, sizes, and flavorings.

The Audience

Suppose that a brilliant designer appeared on the scene, a designer who figured out how to get from the here of RPGs to the there of interactive storytelling. Suppose that this designer began to release a stream of such evolutionary products to the world. What would be the reaction?

At first, players would be enthusiastic about the new ideas. They would perceive the products to be RPGs with a bonus. But as the designer moved away from RPGs and toward interactive storytelling, the players would become disenchanted. At some point, they would conclude that these games were no longer RPGs, and they would reject such games. RPG players prefer to play RPGs, not interactive storytelling. They want to wander around a large environment rich with monsters, treasure, and challenges. They want to build their character's strength and experience, amass the wealth to buy powerful weapons, and all the other things that players do in RPGs. As the line of new games evolved in the direction of greater dramatic content, such activities would be subordinated to the more dramatic behaviors, and the players would feel cheated.

Some years ago, the McDonald's fast-food chain made an attempt to move upmarket—that is, to elevate the quality and price of its offerings to shift to a more profitable market. It didn't work; everyone knew that McDonald's made food that was fast, cheap, and palatable. A McDonald's burger that cost more was instead perceived to be fast, expensive, and palatable; nobody would buy it. Marketing people strive to establish a strong market image for their products, but a strong market image cuts both ways: It clearly identifies your product in the public mind for what it *is*. The downside is that a strong market image also clearly identifies your product for what it *isn't*. RPGs have established a strong market image, so their marketing mutability is limited.

Even if we could somehow evolve RPGs in the direction of interactive storytelling, the market would respond negatively. The people who would pay money for interactive storytelling don't play RPGs. The people who pay money for RPGs aren't primarily interested in interactive storytelling. Making the transition from one to the other requires leaping over a gap between the two that is too wide to hurdle.

Conclusions

Role-playing games, like so many other genres of games, are built around fundamentally mechanical manipulations. Players move around in an exotic environment; they build their strength; they fight monsters; they collect gold, jewels, and weapons. But they don't do the actions that are fundamental to interactive storytelling: They don't interact emotionally with other characters. Ultimately, RPGs revolve around things, not people, and stories are about people, not things.

CHAPTER 11

Video Games

VIDEO GAMES ARE BIG BUSINESS that rake in nearly \$70 billion a year. A single game can cost tens of millions of dollars to develop. The competition is intense, as a hit can make obscene profits while an also-ran can lose millions. With such huge stakes, video-game producers are desperate to find any way to gain an edge over the competition, and one of those methods involves stories. Games that are embedded in a dramatic context—that is, they include some kind of story—are usually more successful than their simpler competitors. With so much money involved, game companies have poured money into the challenge of equipping their games with entertaining stories.

As I mentioned in Chapter 7, “Two Cultures, No Hits, No Runs,” game designers have attempted to graft some sort of story onto their games. First, they included the story in the game’s manual, then they started including it inside the game with cut scenes. Modern methods are technologically far superior to the methods of yesteryear, but they remain similar in overall strategy: A story is added to the game. The story and the game are separate creatures sharing the same box; the connections between the two are sparse.

This is important: Interactive storytelling is *not* the goal of video-game designers; their desire is to increase the appeal of their games by adding stories to them. To video-game designers, story is an embellishment and a selling point, rather than an end in itself. A good story is, to them, no different than good music, good graphics, and good video; it's one more factor in the overall success of the game. They want a good story that will enhance the playing experience. That's something else entirely from interactive storytelling.

Nevertheless, I provide this chapter to discuss storytelling in video games to address the many misconceptions that have arisen out of using stories in games. If you have a clearer idea of what video-game stories can and cannot do, you'll make fewer mistakes.

Josiah Lebowitz and Chris Klug wrote an entire book about the stories in games: *Interactive Storytelling for Video Games*. The authors do a good job of explaining how stories are bolted onto video games in the real world. This chapter uses the ideas in that book as its organizing principle; I will compare and contrast stories in video games with interactive storytelling.

The book's authors present five different strategies for weaving a story into a video game; these are my terms for their categories:

- ▶ Constipated stories
- ▶ Multiple endings
- ▶ Branching trees
- ▶ Open-ended stories
- ▶ Fully player-driven stories

Constipated Stories

The least interactive of these strategies tacks a fixed story onto a video game. As you play the game, you get snippets of the story, but nothing you do affects the story. I call this a *constipated story*. Each new fragment of the story is earned by successful completion of a game segment. It's rather like watching a movie

on DVD except that the DVD requires you to jump through hoops before it will show you the next chapter of the story. Sometimes the obstructions are like puzzles; some are more like games. In many cases, a story fragment is presented upon the successful completion of a game level.

The story in these cases is not interactive. Apparently, the design principle is that, if you alternate between an interactive game and a noninteractive story fast enough, it somehow alloys into an interactive story. Like most magic tricks, people quickly see through the trick.

Despite these problems, purveyors of these games are always able to find people willing to pay for them. Most people, however, prefer to pay \$15 to get an entire movie that is unmasochistically encumbered.

Multiple Endings

The next strategy up the authors' ladder of interactivity is just like the previous one, except that the player gets to choose among several endings. Usually, the choice is made close to the end of the game. For example, a game might present the player with a stark choice between noble self-sacrifice and self-aggrandizement. To appreciate just how inadequate this approach is, let's return to my discussion of interactivity in Chapter 2, "Interactivity." There, I likened interactivity to a conversation between two people. Suppose, then, that somebody sits down next to you and tells you a long story while you're playing a video game. At the very end of the story, your interlocutor gives you a choice of two possible endings to their tale. Would you regard this as interactive storytelling?

It is possible, of course, to provide more than two endings. But what difference would that make? The only choice that the player gets to make comes at the very end of the story. The bulk of the game is a conventional noninteractive story; only the final event is interactive. If somebody offered you a graphic novel consisting of 99 pages of plain text followed by a final page with an illustration, would you call it a graphic novel? If you went to see a movie that comprised 99 minutes of a sequence of still images concluded by one minute of beautiful computer-generated animation, would you call it an animated movie? So why should a conventional story capped off with one choice be called interactive storytelling?

The multiple endings approach fails because it violates the need for individuation that I described in Chapter 3, “Interactive Storytelling.” The player is not given any opportunity to create their own story; instead, they are confined to the meager handful of endings that the author deigns to grant them. Yuck!

Branching Trees

Third in the sequence of methods for weaving a story into a video game is the branching tree approach discussed in Chapter 8, “Branching Trees.” The authors of *Interactive Storytelling for Video Games* argue that, because the player can follow a number of different paths to reach the ending, the player enjoys far greater freedom and control than multiple-ending stories. This is true; however, consider the architecture of the branching tree they present as an example, as shown in **Figure 11.1**.

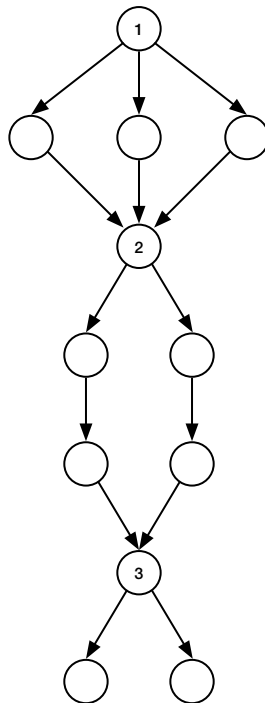


FIGURE 11.1 A branching tree presented in *Interactive Storytelling for Video Games*

Observe that this tree makes heavy use of foldback and provides a grand total of three decisions to the player and only two endings. I suppose that it's strictly correct to observe that three decisions provide "far greater freedom and control" than one decision—but only in the sense that three drops of water are a lot more than one drop of water to a man dying of thirst in the desert.

Open-Ended Stories

The authors of *Interactive Storytelling for Video Games* call the penultimate step in their scheme for adding story to a game "Open-Ended Stories." They are describing games in which the player has a large environment to explore, but a traditional story still plays out as the player makes progress toward the final objective. The environmental exploration reveals bits and pieces of the story alongside the bits and pieces of the game. These aren't open-ended stories; they're disassembled, noninteractive stories set inside an interactive playground, and the player's task is to assemble them into a coherent whole.

The primary challenge in such games, according to the authors, lies in arranging a group of scenes that tell a story when taken as a whole, yet can be experienced in almost any order. The idea of randomly ordered story fragments was first explored in interactive fiction. Although some interesting creations have appeared, this strategy does not provide a sound foundation for interactive storytelling. We certainly haven't seen a genre spring up around the idea; most such efforts have a contrived feel to them and seem to require a great deal of effort to create.

A startling quote unwittingly reveals the authors' attitudes toward player freedom of choice:

Often the best you can do is try to ensure that the main plot itself maintains a good pace, as long as the player follows it in one of the intended ways.

In other words, player choice is not a design desideratum; it's a danger that could compromise the story if players fail to follow the intentions of the designers. This is not interactive storytelling: It's a canned story.

The authors conclude by describing the strength of the open-ended story for those who think that player control is the most important part of stories:

...the real fun in open-ended stories comes from developing your character and exploring every little thing that the game world has to offer...

...For those who don't believe that highly player-driven storytelling is such a good thing, the strength of the style comes not from the story itself, but from the large detailed world in which the story takes place.

Thus, open-ended stories are not in any way a form of interactive storytelling. Open-ended stories are just another way to add a conventional story to a game, although in this case the bits of the story are written on Post-it notes and scattered around a landscape for the player to gather up and make sense of.

Fully Player-Driven Stories

The authors' highest level of interactive storytelling woven into games is "Fully Player-Driven Stories." They offer *The Sims* as an example of this category. The player is free to wander around an interesting environment and do interesting things, but there's no story at all—at least, none in the conventional sense. Certainly there is no inkling of a story designed by an author. This leads the authors of *Interactive Storytelling for Video Games* to look askance at such products. One of them flatly admits that fully player-driven stories aren't actually stories. He's right, of course—but he misses the huge difference between a story (data) and storytelling (process). Computers aren't a medium for presenting canned stories; they're a medium for storytelling. "Fully player-driven story" is an oxymoron; fully player-driven storytelling is our holy grail.

Cost vs. Storytelling

Throughout the book, the authors struggle with the cost of providing rich storytelling. They point out that every scene presented for the developing story must be scripted, acted, photographed, or animated. Because these are usually the most expensive parts of developing any AAA game, adding variations to the basic storyline adds to the total cost of the game.

Worse, from their point of view, is that the more variations there are, the lower the chance the player will see them—which means that all of the money spent on the unseen scenes goes to waste.

Their conclusion is that the prudent designer will tightly constrain the number of variations on the storyline. Whether it's a branching tree, an open-ended story, or a fully player-driven story, the company can afford only a certain number of scenes. Those scenes must be carefully chosen, and the designer should carefully constrain the storyworld so that only those few scenes are possible.

A concern for costs is certainly a necessity in any interactive storytelling project, but the authors' recommendation is entirely wrongheaded. If your orphanage has enough money to feed only 100 children and 150 children show up at the door, you don't send 50 children away to starve: You find cheaper food. If you're building a house and you run low on money before the roof is installed, you don't leave the house without a roof: You find the cheapest way to roof it.

The same thing goes with interactive storytelling. If your budget doesn't permit enough scenes to provide adequate interactivity, you don't accept inadequate interactivity. You find a way to make the scenes cheaper. Interactive storytelling requires vast numbers of scenes, so you'll need to find a way to make those scenes in much, much cheaper ways. The inescapable conclusion is that the scenes must be machine-generated, not scripted and animated by hand. And what's so wrong with that? Isn't the whole point and purpose of computers to automate things, to replace hand calculations with automatic calculations?

What is your priority: beautiful images or rich interactive storytelling? If you put your money on the beautiful images, then your game becomes more like a movie and less interactive. That might sound like a good thing until you realize that, the more movie-like your game is, the more directly it competes with real movies. You simply do not have the budget or the resources to make a better movie than real moviemakers do. If you try to compete with them on their own ground, they'll crush you like a bug. Your best hope is to take advantage of the single thing you have to sell that they don't: interactivity.

Therefore, you want to put your precious dollars into making better interactivity, not better movies. That, in turn, means substituting automated ways of presenting scenes rather than full-blown video or animation.



But anything less than prime video or animation will look crappy next to the competition!

That's the wrong comparison to make. Don't compare a cheap-image game with an expensive-image game. Instead, compare a cheap-image, expensive-storyworld game with an expensive-image, cheap-storyworld game. For the amount of money that others spend on magnificent audio and video, you can use runtime-generated images and provide a richer, more dramatically interesting storyworld. You wouldn't have to constrain the player to scrawny branching trees, limited choices, and crummy interactivity.

Ultimately, what are customers seeking in a game: beautiful video or compelling storyworlds that they're part of? To put it another way, why would they spend \$40 on your beautiful video when they can rent a DVD with better video for \$1?

Conclusions

Interactive Storytelling for Video Games provides the best evidence for the improbability of video games ever providing any foundation for interactive storytelling. The authors clearly and honestly show that video games can include only fixed, planned stories; interactive storytelling is not compatible with the design conventions for video games. If you want a job making the same old canned stories with clichéd heroes fighting clichéd battles as part of clichéd quests, *Interactive Storytelling for Video Games* is the book for you. If you want to create interactive storytelling, the concepts in the book are still useful as a compendium of all the wrong ways to think about the problem.

PART IV

Component Technologies

INTERACTIVE STORYTELLING is complicated! To make it work, you'll need to bring a variety of different software technologies together. Here are some technologies that might prove useful in your quest.

Chapter 12: Modeling is how we represent the real world inside a computer. This chapter is an introduction to how you model things.

Chapter 13: An oddball mathematical system called *bounded numbers* makes modeling easier.

Chapter 14: We infuse our characters with distinctive personalities by building personality models.

Chapter 15: The most important character in every storyworld is Fate, who pushes the player through a dramatically viable sequence of events.

Chapter 16: The most important component of storyworld is the verb, which forms the core of an event.

Chapter 17: You gotta be able to tell the player what's happening, and they have to be able to tell you how they react. But how do you pull this off?

Chapter 18: A good interactive storytelling system needs its own custom scripting language to make the job easier.

Chapter 19: The storytelling engine is the master program that pulls all this stuff together.

Chapter 20: The hardest part of interactive storytelling is the high-level system that keeps the story moving along.

CHAPTER 12

Modeling

A MODEL IS A SMALL VERSION of the real thing. It's a toy version; it is not intended to capture the entirety of its subject. A child's doll has no functioning pulmonary system or musculature; it doesn't need such things to serve its modeling purpose. A novel does not relate the complete history of its protagonist's life; only those events significant to the dramatic goal of the author are included in a novel.

Interactive storytelling is a form of modeling, and like other media, it is neither necessary nor desirable that it capture reality in its entirety; it should model only those aspects of reality that the author intends to communicate.

The Medium

Every medium models one or more dimensions of reality. Painting and photography model the visual appearance of reality; the artist seeks to create an image that clearly and powerfully captures some inner truth. Music models the human condition through the dimension of sound. The musician seeks to express truth through musical notes. Poetry relies on the sound and texture of language to represent truth. Stories use a carefully orchestrated sequence of events to communicate their reality. Cinema is a compound modeling medium that relies on images, like painting; on sounds, like music and poetry; and on a sequence of events, like stories. With three different dimensions in combination, cinema can pack a lot of expressive power.

Obviously, the interactive storyteller must have a clear and firm grasp of the dimensions through which interactive storytelling models the human condition. Here we come to one of the major sources of confusion regarding interactive storytelling. It is implemented on a computer, which can present text as in a novel, or sounds such as music, or two-dimensional imagery, or even three-dimensional imagery. This cornucopia of expressive dimensions dazzles and blinds the novice, making it difficult to see through to the true essence of the computer as an expressive medium. That essence is its capability for interactivity. The difficulty of perceiving the centrality of interactivity is exacerbated by the novelty of this dimension and its abstraction. People perceive only what they recognize: text, sound, and images. They are blind to the unfamiliar: interactivity. Someday, when people are familiar with interactivity as an expressive dimension, it will all seem obvious and natural, and people will look back upon us as ignorant fools unable to recognize the truth staring them in the face. We laugh at the tales of people in the early years of cinema who ran screaming out of the theater upon seeing the first movie of a train coming straight at the audience.

The core of the models we create for interactive storytelling lies in their behavior: It's not primarily important that they *look* like the real thing, nor that they *sound* like the real thing: the sine qua non of interactive storytelling is that the characters *act* like the real thing. It is their behavior, not their looks, that we model in interactive storytelling.

Basics

A computer model has two fundamental types of elements: variables and equations. The variables are numbers that specify what's important about the model. The equations use the variables to calculate useful things. We could create a ridiculously simple model for the motion of a car down a road. There are just three variables: the speed of the car, the duration of a period of time, and the distance the car travels in that period of time. There's just one equation for this model:

$$\text{Distance} = \text{speed} \times \text{time}$$

Wowie-zowie, doesn't that make you just shiver with mathematical excitement?

Most of our simplest models come from physics. However, you don't need to know physics to make models—you just have to understand the ideas it embodies about modeling reality. Like everything else we do, you can just fake it.

For example, suppose you'd like to model something more interesting—and more appropriate to interactive storytelling—than a car driving down the road. Let's try something truly simple: how a person reacts to a compliment. In particular, we'd like to figure out how that person's feelings toward the complimenter will change. For simplicity in the following discussion, I'll replace "complimenter" with "Tom" and "person" with "Mary."

The basic idea is simple: Mary will feel greater affection for Tom after he compliments her. But we want to express that concept in computable terms. To do that, we need just two variables:

- ▶ The "niceness" of the compliment: how generous or flattering it is.
- ▶ The affection that Mary has for Tom.

Remember, these are two numbers representing the magnitude of the quantity in question. It should be obvious that the affection will be increased to a degree concomitant with the niceness of the compliment. So let's just add the niceness of the compliment to the affection:

$$\text{Affection}[\text{Mary}, \text{Tom}] = \text{Affection}[\text{Mary}, \text{Tom}] + \text{Niceness}$$

Remember, this is computer-style language, so we're not declaring an equality, we're defining a mathematical operation. The line above says: "Take the existing Affection that Mary feels for Tom, add the Niceness to it, and make that the new value of the Affection that Mary feels for Tom."



Where do we get the numbers for these values? What's the number for the Affection? What's the number for the Niceness?

Those values would be initialized at the beginning of the program, then changed during the execution of the program in response to events. Remember that they're variables: We assert that this formula will work no matter what values we have for the Affection and the Niceness. So for the moment, we don't really need to imagine the actual numbers.



But it makes more sense if I can think in terms of actual numbers!

Then it's time to start learning to think abstractly. Remember Chapter 4, "Abstraction," and the explanation of the importance of learning to think abstractly? This is the first instance of that kind of abstract thinking. You need to stop thinking in terms of $2 + 2 = 4$ and start thinking in terms of $x + y = z$. You need to transcend this obsession with specific numbers and instead think in terms of all possible numbers. **Figure 12.1** presents a visual metaphor for this kind of abstract thinking.

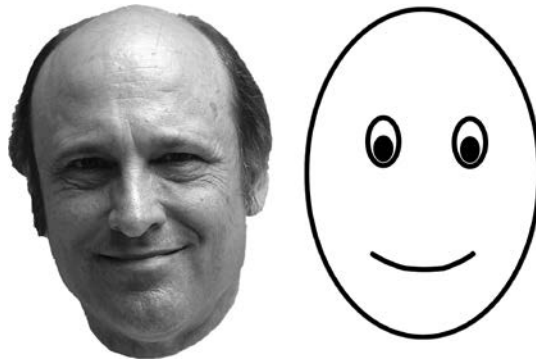


FIGURE 12.1 Abstraction in faces, an idea ruthlessly stolen from Scott McCloud

The face on the left is as direct and explicit as possible: It is a photograph of a single face. The face on the right is a highly abstract representation of a face: It comprises only the absolute essentials. Because it is so very abstract, it also applies to any single person's face.

Variables are to numbers as the face on the right is to the face on the left. They are the abstraction of numbers; one variable might represent *any* number. This is the simplest kind of abstraction that you'll need to grasp in the field of interactive storytelling.

No doubt about it, abstract thinking demands more effort than explicit thinking. I made life a little easier for you by substituting "Tom" and "Mary" for "the complimenter" and "the person." Go back and reread the previous paragraphs, mentally reversing the substitution. You'll find that it's harder to keep track of the ideas when they're expressed with greater abstraction. Let's face it: Abstraction is hard work! If it weren't, they would have solved the problems of interactive storytelling a few months after they got *Pong* working and you wouldn't be reading this book.



What happens if Tom says something uncomplimentary to Mary?

Uncomplimentary is the opposite of nice, so it must be a negative number. If we use a negative number in our formula, the value of `Affection[Mary, Tom]` will be diminished and Mary will feel less `Affection` for Tom. That's what you'd expect, isn't it?

The fact that the negative value has the opposite meaning could cause some confusion, so I have established a standard naming system for obviating such confusion. That naming system requires you to give two names to every variable. The first name describes what the number means when it's positive; the second name describes what it means when the number is negative. Thus, instead of calling it `Nice`, we call it `Nice_Nasty`. That makes it perfectly clear that the positive value denotes niceness and the negative value denotes nastiness.



That seems like an unnecessary artifice and a waste of effort.

Really? Tell me, should the negative of Angry be Happy, Fearful, or Serene? Should the negative of Excited be Bored, Sleepy, or Disgusted? Should the negative of Empathetic be Callous or Clueless? Many of the English words you'd like to use for variable names have ambiguities that will drive you nuts. The only way to resolve those ambiguities is to clearly specify both the positive and the negative poles of your variable.

Furthermore, we'll have to apply this concept to the one-sided term Affection; the new term will be Affection_Disdain.

Adding More Complexity

Now let's take a further step into complexity. Let's consider what happens when a third person, whom we'll call Dick, hears Tom's comment to Mary. Suppose we'd like to calculate how Tom's comment affects Dick's Affection_Disdain for him. Dick already has some degree of Affection_Disdain for Tom; how will that degree of Affection_Disdain affect our formula?

Because this situation is getting complicated, the first thing we do is lay out the situation with as much clarity as possible. We have three people: Tom, Mary, and Dick. Tom says something that is either complimentary or insulting (that is, Nice_Nasty) to Mary. Dick observes the act and reacts by changing his Affection_Disdain for Tom. How do we calculate that change?

The general idea here is actually pretty simple: Dick will change his Affection_Disdain based on two considerations. Those two considerations are his Affection_Disdain for Mary and the value of Nice_Nasty. If Tom says something nice to Mary, and Dick likes Mary, then Dick will be pleased by the compliment and will have greater affection for Tom. However, if Tom insults Mary, then Dick will be displeased and will have less Affection_Disdain for Tom.

These results are reversed if Dick hates Mary. In that case, Dick will be chagrined by a compliment to the hated Mary and will resent Tom for doing so, reducing the value of Affection_Disdain for Tom. Finally, if Dick hates Mary and Tom insults Mary, then Dick will be favorably disposed toward Tom. "Yeah! You tell her, Tom!"

Let's summarize these results in a simple table of outcomes (**Table 12.1**):

	Affection	Disdain
Nice	Increase	Decrease
Nasty	Decrease	Increase

We can write these results down in Boolean formulae:

If (Affection_Disdain[Dick, Mary]>0) AND (Nice_Nasty>0) then increase Affection_Disdain[Dick, Tom]

If (Affection_Disdain[Dick, Mary]>0) AND (Nice_Nasty<0) then decrease Affection_Disdain[Dick, Tom]

If (Affection_Disdain[Dick, Mary]<0) AND (Nice_Nasty>0) then decrease Affection_Disdain[Dick, Tom]

If (Affection_Disdain[Dick, Mary]<0) AND (Nice_Nasty<0) then increase Affection_Disdain[Dick, Tom]



But we don't want to use Boolean formulae, remember? We want to use numeric formulae! How do we get from Boolean to numeric?

It's time for a trick of the trade that will make formula writing much easier.

It consists of two rules:

- ▶ Convert Boolean OR to numeric addition.
- ▶ Convert Boolean AND to numeric multiplication.

These rules are simple to learn and easy to use. To apply them, we replace the Boolean statements inside the parentheses [$(x>0)$ or $(x<0)$] with the numeric values themselves, and we replace the AND with a multiplication sign. The end result constitutes the net change:

$$\text{Change in Affection_Disdain}[\text{Dick}, \text{Tom}] = \text{Affection_Disdain}[\text{Dick}, \text{Mary}] * \text{Nice_Nasty}$$

This single equation does everything that the four Boolean statements do and more. Go ahead, walk through four separate cases in which the values have different combinations of positive and negative, and you'll see that the change in `Affection_Disdain` goes in the right direction.

But this numeric formula does more than just that! For example, suppose that the comment is neither nice nor nasty. What if Tom merely says, "Gosh, it's hot today!" The `Nice_Nasty` value of that statement is zero. That zero value gets plugged into the formula and—ta-da!—the final result is zero! If Dick sees Tom say nothing important to Mary, then Dick's feelings for Tom are unchanged.

It works the other way, too. Suppose that Dick doesn't know Mary from Adam, so Dick's `Affection_Disdain` for Mary is zero. Plug that zero into the formula and re-ta-da! The result is zero! Dick's feelings for Tom are unchanged because Dick just doesn't care about Mary.

Even better, the results of this formula show proportion to the scale of the action. If Dick has strong feelings about Mary, then his feelings for Tom will change more than if Dick's feelings about Mary are neutral.

If Tom says something really complimentary, or something really insulting, that will have a larger effect than if Tom voices a wishy-washy compliment or insult.

Thus, numeric calculations give us the potential for greater fidelity in our models.

Lesson 21

Numeric calculations are superior to Boolean calculations.

Abstracting the Process

As if we hadn't gotten abstract enough, I'm now going to abstract the process we just went through. I'll present you with a recipe to follow whenever you need to create a model for some process. Here it is in eight simple steps:

1. State in words the situation.

Write down, in plain English, the situation under which you want to make a calculation. It's important to write it down because that forces you to work out the ambiguities and nail down any relevant details. If you don't get this step clear, then everything that follows will be a waste of time.

Let's work out an example as we go. Recall that dramatic moment in *The Lord of the Rings* when Frodo stands on the precipice over the fires of Mount Doom, holding the Ring. He has come all this way to hurl the Ring into those fires, but he hesitates.

2. State in words the answer you seek.

What do you hope to accomplish in this calculation? What are you trying to calculate? Remember: If you don't know where you're going, you probably won't get there.

For our example, we seek to decide whether Frodo will indeed throw the Ring into the fires.

3. State in words the general principle you think should apply.

What's the human truth you're trying to articulate? It could be as obvious as, "If you're nice to someone, they'll be nice to you." It could be "Hell hath no fury like a woman scorned." Or, "A lie can travel around the world before the truth can lace its shoes." Your wording might not be quite as eloquent, but it should get the idea down on paper. You don't want to get mathematical here. Just state the general idea in plain English; we'll convert it to mathematical form later.

In our example, the general question is whether Frodo can overcome the call of the Ring.

4. Decide on and name the relevant variables.

This is the first technical step in the process. Don't underestimate its importance. The problem isn't just coming up with appropriate names. It's more like how you decide to slice the pie.

The challenge is to get the clearest possible expression of the variables you intend to use. Perhaps you want to use the variables that you've already established for other purposes; if so, make certain that they are relevant to this task.

I decide that the two relevant variables in our example are Frodo's strength of will and his virtue. We'll call these `Strong_Weak` and `Good_Bad`. I could have chosen other variables; perhaps you can slice the pie better than this. But let's proceed with these two.

5. Express the general principle in Boolean form.

Now set up a Boolean expression that uses the variables from step 4 to express the general principle you established in step 3. It's easier to start off with the Boolean form, because you can readily articulate simple black-and-white principles. In this case, the Boolean form is trivially simple:

```
IF (Frodo is strong) OR (Frodo is good) THEN Burn_That_Ring
```

6. Convert the Boolean form to number form.

This will seem difficult at first, but once you get some practice, you'll find it fairly easy. Just remember: OR means addition while AND means multiplication. For our sample case, the mathematical expression is obvious:

```
IF ((Strong_Weak + Good_Bad) > 0) THEN Burn_That_Ring
```

7. Check the results using a wide range of numbers.

Here's where you catch your mistakes. You should try your formula with every possible combination of three values of every input variable: a big negative value, a big positive value, and a zero value. If you have two variables, that's nine combinations to check. Check every single combination to make sure you get results that make sense. If you miss one, the algorithm will take advantage of your oversight to blow you out of the water later on. Algorithms are ornery critters—don't trust 'em!

In our example, the test is fairly easy to carry out. When both variables are positive, the result is always positive; when both are negative, the result is always negative. When they're mixed, then the greater absolute value decides the result (**Table 12.2**).

TABLE 12.2 Variable results

Strong_Weak	Good_Bad	Result	Decision	Look right?
+0.9	+0.9	+1.8	Destroy ring	Yes
+0.9	0.0	+0.9	Destroy ring	Yes
+0.9	-0.9	0.0	Keep ring	Yes
0.0	+0.9	+0.9	Destroy ring	Yes
0.0	0.0	0.0	Keep ring	Yes
0.0	-0.9	-0.9	Keep ring	Yes
-0.9	+0.9	0.0	Keep ring	Yes
-0.9	0.0	-0.9	Keep ring	Yes
-0.9	-0.9	-1.8	Keep ring	Yes

Gadzooks, this is so simple. Perhaps you should dump interactive storytelling for something with a little challenge, like calculating how balls bounce off walls.

8. Adjust to taste.

Here's where the artist in you comes out. Are your results too big, or too small? Are they biased in one direction? Are they appropriately sensitive to conditions, with a response that's neither too big nor too small? Nobody in the world can tell you what the right expression is; you just have to decide for yourself what you want to say, algorithmically speaking. Should Frodo be more or less likely to keep the ring than this formula determines? It's up to you to decide.

Fine Points

There are a million ways to make this process more complicated. You can refine it, tune it, adjust it, and improve it in more ways than a cook can fiddle with a recipe. A chef does not achieve greatness by competently executing the basics. No, culinary talent expresses itself in the mastery of myriad variations on the basic theme. Following are just two of the finer points of modeling: weighting factors and addition versus multiplication.

Weighting Factors

In our example of Frodo in Mount Doom, we simply added his `Strong_Weak` to his `Good_Bad`. But think about it: Do these two factors deserve to be treated as equals? Perhaps your own assessment of the human condition is that an iron will is of greater importance than mere good intentions. Perhaps your life experiences have taught you the reverse. In either case, you need to adjust the formula to incorporate this knowledge. To do this, you merely need to multiply the more important factor by some number greater than 1. For example, if you believe that `Strong_Weak` is twice as important as `Good_Bad`, then you would use this formula:

```
IF ((2 * Strong_Weak + Good_Bad) > 0) THEN Burn_That_Ring
```

The problem is more difficult when the two factors are being multiplied together, in which case multiplying by a constant increases the effects of both:

```
(2 * Strong_Weak) * Good_Bad = Strong_Weak * (2 * Good_Bad)
```

You need a way to increase one of the two multiplicands. The most common solution is to use an exponent:

```
IF ((Strong_Weak**2) * Good_Bad) > 0) THEN Burn_That_Ring
```



But this solution is flawed: it works only when `Strong_Weak` is greater than 1. If, say, `Strong_Weak` is 0.5, then squaring it yields 0.25—yipes, that's smaller! As Hitler said to his foreign minister on the day Great Britain declared war, "What do we do now?!?!"

Yessirree Bob, this is definitely a problem, and there's only one way to make sure that this never bites you: Test the value of `Strong_Weak` and use the positive exponent if `Strong_Weak` is greater than 1, and use the negative of the exponent if `Strong_Weak` is less than 1. The formula looks like this:

```
IF (Strong_Weak > 1) THEN Exponent = 2
ELSE Exponent = -2
IF ((Strong_Weak**Exponent) * Good_Bad) > 0) THEN Burn_That_Ring
```

Even this has its problems. First, if `Strong_Weak` is negative, then things get especially messy. Second, exponentiation is a major source of psychotic behavior in functions, so in general, it's best to refrain from using exponentiation until you've gotten your mathematical sea legs.

Addition vs. Multiplication

I gave you a good rule earlier: If your factors are combined with OR in their Boolean form, then add them in the numeric form; if they're combined with AND in their Boolean form, multiply them in the numeric form. That sounds simple enough, doesn't it?

But how do you decide whether to use OR or AND in their Boolean form? Here's an example: Joe Teenager is working on his car in the garage. Since Joe is a teenager, he failed to adequately secure the jack before crawling underneath the car. Furthermore, his teenagerness causes him to accidentally kick the jack while under the car. The jack falls over and the car falls on Joe's leg. Joe yells bloody murder. Joe's mom rushes out from the kitchen and perceives the crisis. She grabs the bumper of the car and lifts mightily. Does she succeed in raising the car enough for Joe to escape?

Two factors apply: Mom's strength and her desperation. We know that in sufficiently desperate situations, humans are capable of superhuman feats. These two factors combine in this crisis. But do they combine with OR or AND? Which of these two statements is closer to the truth:

```
IF ((Mom is strong enough) OR (Mom is desperate enough)) THEN she lifts
the car
IF ((Mom is strong enough) AND (Mom is desperate enough)) THEN she lifts
the car
```

A pretty problem, no? The way to resolve it is to look at the zero case: What happens if Mom has zero strength combined with vast determination or zero determination combined with vast strength?

Consider: Mom is a paraplegic who deeply loves her only son. She is certainly desperate, but all the desperation in the world won't compensate for the inability to move her limbs.

Now, what if Joe is not really her son. What if he's her wife-beating husband who has threatened to kill her if she leaves him. She's a weightlifter who can press 300 pounds, but this car weighs 2000 pounds. She has absolutely zero desperation in this case; there's no way she'll find the strength to save her no-good hubbie. Bye-bye, Joe the Jerk.

Thus, neither desperation nor strength is adequate to compensate for a deficiency in the other factor. That's the crucial distinction: the ability to compensate. If one factor can compensate for the other, then you use OR to combine them; if they can't compensate for each other, then use AND to combine them. In this example (Mom versus the car), we AND them.

Here's an example where they do compensate for each other: The two factors are Mom's strength and the car's weight:

IF ((Mom is strong enough) OR (the car is light enough)) THEN she lifts the car

Here the zero case doesn't work as clearly as we'd like: If Mom is a paraplegic but the car is light as a feather, I suppose that she could blow it off Joe with a puff of air. It's clear, though, that the lighter the car is, the less strength Mom needs to lift it. So the two factors do indeed compensate for each other, and so we add them.

When it comes to matters of the heart, truth is never so clear. In many cases, you'll have to bring your artistic judgment to bear. Is love strong enough to overcome rage? Does the desire for revenge overpower the lust for money? When they're both at work, do these factors add together or multiply together? There's no firm answer; this is where you get to be an artist.

Conclusions

Modeling behavior is the essence of interactive storytelling. If your characters don't behave right, then making them look right or sound right won't do any good. Appearance doesn't compensate for behavior; appearance and behavior multiply together to produce overall storytelling quality. In terms of overall quality, we can achieve, oh, 90 percent perfection in the appearance of characters on computers, but the overall quality of the behavior of our characters is, oh, 1 percent. Multiply those numbers together. Now suppose that, with some effort, we could add 1 percent to either of those factors. Do the math.

This page intentionally left blank

CHAPTER 13

Bounded Numbers



Just a minute here! In the last chapter, you gave this song and dance about how important it is to transcend the pedestrian world of mere numbers and think instead about abstract variables. Now you turn around and start talking about numbers?!

YEP, I'M A LOW-DOWN, LYING HYPOCRITE. Although I think in terms of abstract variables, the computer thinks in terms of actual numbers. There are some problems, however, with actual calculations with normal numbers, and these problems must be dealt with.

For example, suppose you're building a storyworld about romance. We already know that you will need a variable such as `Affection_Disdain` to record how one character feels about another character. What scale do you use? Is the maximum amount of love that's possible equal to 10? Maybe it should be 100. Or how about 1000?

You might decide upon 1 as the top end of your range for love. In other words, the maximum possible love that one actor can feel for another is 1.00. It's as if "perfect love" is 100 percent, which is 1.00. Right? In this system, normal love is a lesser number, like 0.5.

OK, so the ceiling on love is 1.00. So what's the floor? The obvious answer is 0.00, which represents zero love—no love at all.

But wait! What about hatred? Isn't it the opposite of love? So should 0.00 represent total hatred, and 0.50 represent neutral feelings? Or should total hatred be represented by -1.00 , with 0.00 representing neither love nor hate?

It is possible to work either way, but I have learned that the algorithms are easiest to design when -1.00 represents undying hatred, 0.00 represents a completely neutral attitude, and $+1.00$ represents perfect love.



But why even use numbers in the first place? Most people find it much easier to think in Boolean terms (true or false, black or white) than numeric terms.

True, Boolean thinking is easier than numeric thinking. Moreover, computer science has developed a vast library of algorithms for Boolean computation. When you use Boolean terms, you have a big toolkit at your fingertips.

On the other hand, don't forget the old adage "When you have a big enough hammer, everything starts to look like a nail." The fact that you have a huge hammer for Boolean thinking doesn't mean that reality is Boolean.

It's easier to understand the significance of numeric approaches when you realize that Boolean thinking is simply a form of numeric thinking that uses only one bit of data. Remember how everything inside the computer is composed of bits? A Boolean number is one bit in size: It's either a 1 or a 0. A byte has eight bits and so can range (in binary) from 00000000 to 11111111. In decimal, that's 0 to 255. The difference between one bit and eight bits is a matter of resolution. If we ask, "How much does John love Mary?" the answer in Boolean terms is either "He loves her" or "He doesn't love her." But if we represent his love for her in terms of a single byte, then we can say, "He doesn't love her at all," or "He loves her 1/256th as much as total love," or "He loves her 2/256ths as much as total love," and so on right up to "He loves her totally." That's a lot more expressive, isn't it?

Perhaps a visual metaphor will make it easier to understand the difference. Here's a photograph rendered with different numbers of bits per pixel (**Figure 13.1**).



FIGURE 13.1 More bits (less Boolean) means more shades of gray

With 8 bits per pixel, we have 256 shades of gray to use, and the picture looks very good. With only 3 bits per pixel, we have only 8 shades of gray to use, and the picture doesn't look so good. With the Boolean option, using only 1 bit per pixel, we have no shades of gray—just black and white. The picture looks awful. That's what Boolean thinking does to interactive storytelling.

The numbers used in modern computing are 64 bits in size. That gives you zillions of shades of gray, so your palette of personal shades of gray is far bigger than you'll ever need. I'll discuss Boolean calculations more later in this chapter.



I don't need 64 bits of resolution! I could build lots of great storyworlds with just 8 bits! Who cares if Mary's love for Joe is 0.93875 instead of 0.93876?

True, but as far as designing algorithms goes, there really isn't much difference between 8-bit numbers and 64-bit numbers—and that difference is entirely to the advantage of 64-bit numbers. In terms of algorithm design, you must choose between numeric algorithms and Boolean algorithms. If you choose numeric algorithms, then you might as well use the full 64 bits; it can't possibly hurt.

There's another reason to use plenty of bits—something called *round-off error*. Suppose you're adding lots of tiny numbers together to get a big number. This kind of calculation happens all the time in algorithm design. If the ratio of the biggest number to the tiniest number exceeds the dynamic range of the word (derived from the number of bits it contains), then the tiny number will, in effect, get rounded down to zero. You can add millions of these tiny numbers to the big number and get nowhere. Oops!

Overruns

A problem more serious than round-off error is overruns. Consider what happens when you design algorithms for changing the value of a variable. Suppose, for example, that you want to calculate how romantic it is when a boy gives flowers to a girl. That should increase her love for him by some small amount. So you write an equation like this:

$$\text{Increase in Affection_Disdain} = 0.01$$

In other words, when he brings her flowers, her love for him increases by the tiny amount of 0.01. That seems right, doesn't it?

But suppose that a boy meets a girl and then brings her flowers 101 times. If you use the above formula, then her love for him will be 1.01. There are two serious problems with this: First, you previously established that 1.00 represents perfect love, so you can't have a love greater than 1.00. Your formula smashes through the ceiling of the number. Second, romance involves a lot more than just giving flowers to the girl. Sure, it should help at first, but there's a limit on its effect.

Let's tackle the second problem first. How do we reduce the size of the effect as the boy repeats the action? We could use this formula:

$$\text{Increase in Affection_Disdain} = 0.01 / \text{number of times he gave flowers}$$

In other words, her love increases by 0.01 the first time he gives her flowers, by 0.005 the second time, 0.0033 the third time, 0.0025 the fourth time, and so forth. The size of the effect diminishes each time he gives her flowers. That sounds about right, doesn't it?

The first problem remains. Suppose that she already loves him by 0.99—close to, but not quite at the ceiling. Then he gives her flowers for the first time. That bumps up her love to the ceiling, 1.00. And the second time he gives her flowers, her love is increased to 1.005. Once again, your formula smashed through the ceiling. Something has to be done about this formula.



Aw, c'mon! This isn't going to happen! We'll never get a starting love of 0.99, because that's just too perfect. Even Romeo and Juliet didn't love each other that much!

One of the nastiest lessons you learn when working with computers is that you can never rely on probability. If a bad thing can happen, it *will* eventually happen, and when it does, you're screwed. You can't rely on likelihood to keep your design from failing. If it's not bulletproof, a stray bullet will eventually kill it. So this problem is very real.

Bounded Numbers

This brings us to a radical idea: *bounded numbers*. It's a weird system, but it works beautifully for the kinds of problems we encounter with interpersonal behavior.

Suppose that we have an `Affection_Disdain` value of 0.60 and the boy does something really sensational, like proposing marriage. As a result, the girl is so pleased that her `Affection_Disdain` for him will increase by 0.50. If we add 0.50 to 0.60, we get 1.10—again, we blast through the ceiling. **Figure 13.2** illustrates the problem.

Here's the trick: Instead of using 0.50 as the distance we move, we make 0.50 the fraction of the distance to the ceiling that we move. Think in terms of percentages. The girl's `Affection_Disdain` is 60 percent; we increase that by 50 percent, getting the illegal value of 110 percent. But if we say, "Go 50 percent of the way from the current value to the ceiling," then we're guaranteed that we'll never go through the ceiling. **Figure 13.3** illustrates the concept.

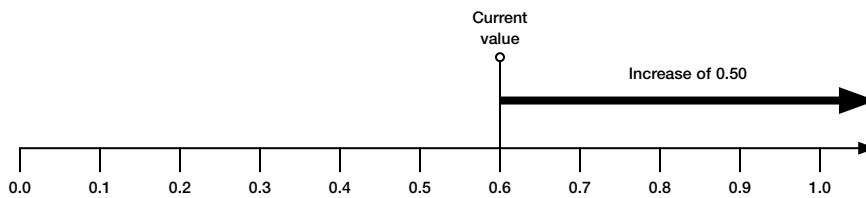


FIGURE 13.2 Increasing a value so much that it overruns our intended ceiling

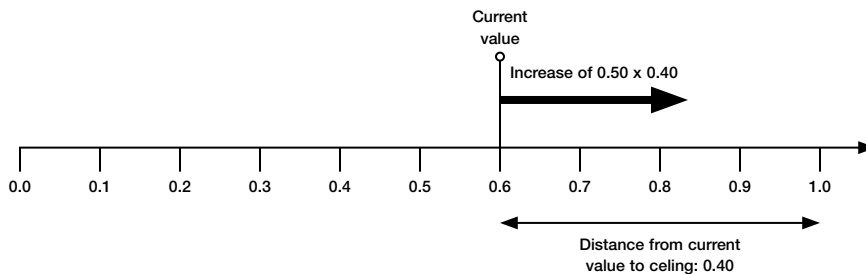


FIGURE 13.3 One way to ensure that we stay under the ceiling

If you have any doubts that this solves our problem, just try some experiments like this: “If we go 80 percent of the way from 0.90 to 1.00, then we’ll end up at 0.98.”



OK, smartypants, what happens when we use negative numbers? Suppose the girl feels Affection_Disdain for the boy of -0.20 and he does something really nice that will increase her Love by 0.2 ? What then?

No problem. **Figure 13.4** shows the diagram of the calculation.

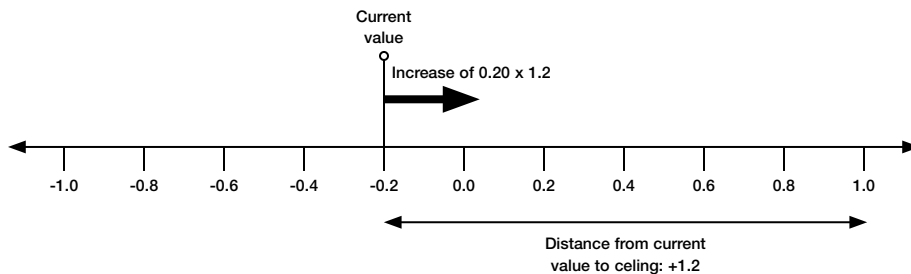


FIGURE 13.4 Starting from a negative number

Because the distance to the ceiling is 1.2, we multiply the change (0.2) by 1.2 to get a net increase of 0.24 . In other words, if you do something nice to somebody who doesn’t like you, they’re really impressed and will alter their attitude toward you substantially.



OK, I admit that works. But what about a negative change? What happens if the girl has -0.2 Affection_Disdain for the boy, and he does something nasty to her that will make her Affection_Disdain 0.4 less? How does that work?

Figure 13.5 shows the diagram for that situation; you should be able to figure it out.

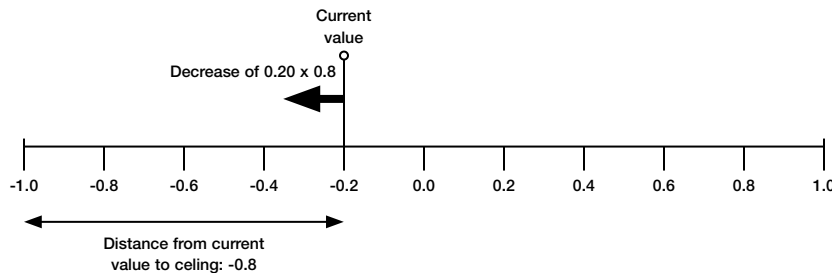


FIGURE 13.5 Moving in the negative direction

See? It works no matter what! This scheme allows you to have numbers describing people that are guaranteed to remain within reasonable bounds. This eliminates one of the biggest headaches I've struggled with over the years.

What I have described so far is bounded number addition and subtraction. A company called Choice of Games has developed a system rather like this called Fairmath. However, it goes only as far as this point in the chapter; it does not include the remaining properties.

Bell Curves and People

Bounded numbers provide another benefit: They naturally fit into the bell curve that most accurately represents human traits and relationships. For example, let's consider a personality trait like `Nice_Nasty`. That's an important personality trait; you'll probably want to include it in your storyworld so that you can show good characters behaving nicely and bad characters behaving nastily. Using bounded numbers, you'll want all the values to fall between -1.0 and $+1.0$. The nicest person in the world (say, Ms. Goody Two-Shoes) would have a `Nice_Nasty` value of $+1.0$, and the nastiest person in the world (say, me) would have a `Nice_Nasty` of -1.0 . The average person would have a `Nice_Nasty` of 0.0 . This observation automatically solves a great many problems in storyworld design.

How do we calibrate personality trait values? Exactly how nice is 0.5 `Nice_Nasty`? Nice enough to share one's ice cream cone? Nice enough to give a kidney to save a life? You can't possibly decide things like this with an absolute system.

By thinking in terms of bounded numbers and bell curves, you don't need to worry about deep philosophical definitions of niceness and nastiness. Instead, you visualize a bell curve, as in **Figure 13.6**.

Most people are close to average; it takes a very special person to push all the way out to the far ends of the bell curve.

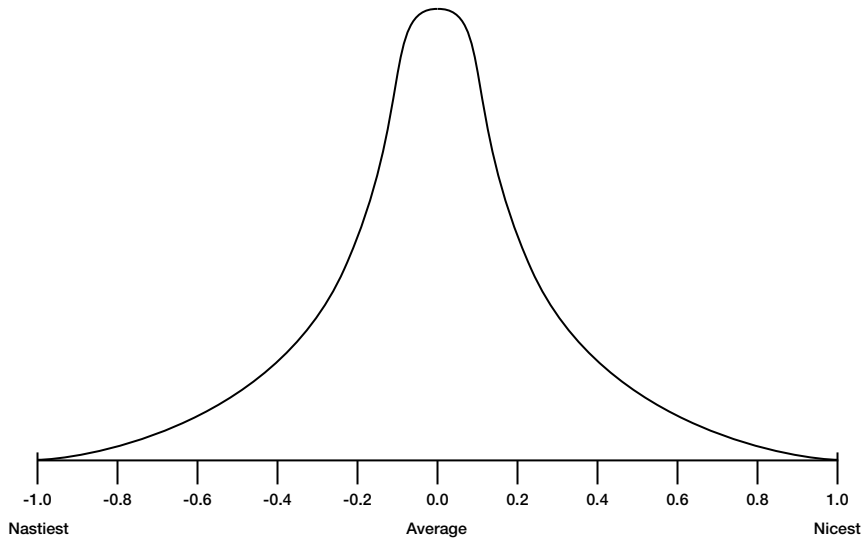


FIGURE 13.6 A bell curve representing the distribution of a common personality trait

Technical Details

Here's how to implement bounded numbers on the computer. This is programmer stuff, but it's simple enough that you should be able to understand it. Here's the Java code for converting a plain old regular (unbounded) number to a bounded number:

```
float boundedTransform(double unboundedNumber) {
    if (unboundedNumber > 0.0)
        return 1 - (1 / (1 + unboundedNumber));
    else
        return (1 / (1 - unboundedNumber)) - 1;
}
```

Here's the code for converting a bounded number to a regular (unbounded) number:

```
float boundedInverseTransform(double boundedNumber) {
    if (boundedNumber > 0.0)
        return (1 / (1 - boundedNumber)) - 1;
    else
        return 1 - (1 / (1 + boundedNumber));
}
```

To perform arithmetic operations on bounded numbers, you convert them to unbounded numbers, execute the operator, then convert back to bounded numbers. However, you should avoid multiplication and division; bounded numbers behave in unexpected ways when you multiply and divide them. For now, it's best to restrict yourself to addition and subtraction. When you're a pro with bounded numbers, you can figure out for yourself the tricks involved in using bounded numbers with multiplication and division.

Lesson 22

Use bounded numbers in your models whenever possible.

The Blend Operator

You can carry out a great deal of work with simple bounded addition and bounded subtraction. However, what happens when you want to weight one of the factors in an equation? For example, suppose that Character#1 apologizes to Character#2 for some previous transgression, and you want to write an inclination equation to determine whether Character#2 will accept the apology. You know that two factors will go into the equation: how Nice_Nasty Character#2 is and how much Affection_Disdain Character#2 has for Character#1. But suppose, also, that you think the Affection_Disdain factor is more important than the Nice_Nasty factor—you believe that friendship is a more powerful force than intrinsic niceness. How do you add extra weight to Affection_Disdain? If you were using conventional numbers, you'd want to multiply by a weighting factor of, say, 2 to double the importance of Affection_Disdain. You could just use a straight bounded multiplication.

There's a better way to mix two bounded numbers together in a controlled fashion: Use the blend operator. Based on experience, I believe that blend is the best way to put together two variables with different weightings, as well as for solving a lot of other problems.

Blend uses three bounded numbers for its input; I'll call them x , y , and $Weight$.

Figure 13.7 shows what blend does with x , y , and $Weight$.

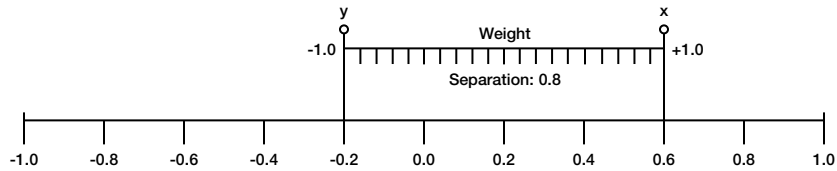


FIGURE 13.7 How the blend operator works

Think of x and y sitting somewhere along the number line. Now look at the horizontal line connecting them, the one marked “Separation: 0.8.” Think of that line as an entire number line in its own right. We set up a custom scale just for that line, with 0.0 in the center and -1.0 at the left edge and $+1.0$ at the right edge. The value of `Weight` slides along that scale to decide what the value of blend is.

For example, if the `Weight` value is 0.0, then blend will pick the number in the center between x and y (in this case, $+0.2$). But if the `Weight` value is $+0.5$, then it will slide to a place halfway between the center and x , which is $+0.4$. Similarly, if the `Weight` factor is -0.5 , then the result will be the number halfway between the middle and y , which in this case is 0.0.

You can now see why we call it blend: It blends together two numbers in whatever proportion you set.

Blend can be very powerful when you want variable weighting. For example, suppose that Joe tells Mary a lie; will she believe him? Three factors affect her decision: her `Trustworthy_Mendacious` feeling towards Joe, the `Likely_Unlikely`-ness of the statement itself (that is, how well it comports with Mary’s existing knowledge), and Mary’s `Gullible_Suspicious` value.

- ▶ The more she trusts Joe, the more likely she is to believe him.
- ▶ The more likely the statement is to be true, the more likely she is to believe him.
- ▶ The more gullible she is, the more likely she is to believe him.

So we set `Trustworthy_Mendacious` as x , `Likely_Unlikely` as y , and `Gullible_Suspicious` as our `Weight` factor. Then, we use blend. **Table 13.1** shows the results based on different values of the three variables.

TABLE 13.1 Weight factors and blend results

Variable	Example 1	Example 2	Example 3	Example 4
Trustworthy_Mendacious	+0.5	+0.6	-0.3	-0.9
Likely_Unlikely	-0.5	+0.7	+0.7	+0.3
Gullible_Suspicious	0.0	-0.6	-0.4	+0.2
Result of Blend	0.0	+0.62	0.0	-0.78



*How did you know to use Gullible_Suspicious as the weighting factor?
Why not one of the other two variables?*

Good question. Let's look at the table of results with the various combinations. We'll use these values for the three variables:

Trustworthy_Mendacious = +0.6

Likely_Unlikely = -0.2

Gullible_Suspicious = +0.4

There are six different combinations. I'll list each one in terms of who plays the role of x, then y, then Weight.

Trustworthy_Mendacious, Likely_Unlikely, Gullible_Suspicious: Blend = +0.04

Trustworthy_Mendacious, Gullible_Suspicious, Likely_Unlikely: Blend = +0.52

Likely_Unlikely, Gullible_Suspicious, Trustworthy_Mendacious: Blend = +0.28

Likely_Unlikely, Trustworthy_Mendacious, Gullible_Suspicious: Blend = +0.36

Gullible_Suspicious, Trustworthy_Mendacious, Likely_Unlikely: Blend = +0.48

Gullible_Suspicious, Likely_Unlikely, Trustworthy_Mendacious: Blend = -0.08

As you can see, we get completely different values for all six combinations. Hence, it really matters what combination you use! Some rules of thumb for figuring out the correct combination are:

- ▶ If a variable intrinsically provides a hard upper limit or lower limit to the desired result, then it should *not* be the weighting variable. In the preceding example, the value of Likely_Unlikely provides such a limit to the final result. Nobody is going to accord something greater credence than they think is likely.

- ▶ If two variables seem to sit in obvious opposition, then they should probably constitute x and y .
- ▶ A weighting factor should probably be a personality trait that shows some sort of preference or inclination on the part of the character. This is why I chose `Gullible_Suspicious` to be the weighting factor.

Here's the formula for the blend operator:

```
uWeightingFactor = 1-((1-WeightingFactor)/2)
Blend = y*uWeightingFactor + x*(1-uWeightingFactor)
```

OK, you may not be able to grasp this formula yet, but someday you should. It's a little messy with all the parentheses, but this is still just high school freshman algebra. Try it out with different sets of numbers for x , y , and `Weight`, and see what you get. If you play with this for a little while, you'll get a feel for it.

One-Factor Use of Blend

The blend operator has many uses. For example, it can be used to scale up or scale down numbers. Suppose you want to scale down the impact of a `Nice_Nasty` trait on a calculation. No problem: just use `Blend(0, Nice_Nasty, 0)` instead of `Nice_Nasty` itself. That will push it from wherever it is to a place halfway toward zero. If you want to scale it down more, use a positive number; a negative number will scale it down less.

If you want to amplify the magnitude of a bounded number to increase its heft in a calculation, the formula is a little messier. You have to use two formulae, one for the case in which the variable is positive and the other for when it's negative:

```
IF (x>0) then use Blend(x, 1, 0)
IF (x<0) then use Blend(x, -1,0)
```

Again, increasing or decreasing the weighting factor (0 in this example) will increase or decrease the amount of amplification.

Two-Factor Use of Blend

Blend is very useful when using two variables as well. In the first example of this section, I showed how to blend together two variables in differing degrees. You can also use blend for variable amplification or suppression of a number. To amplify a number a variable degree, use this:

IF $(x < 0)$ then use $\text{Blend}(x, -1, y)$

IF $(x > 0)$ then use $\text{Blend}(x, 1, y)$

To suppress a number variably, use the simpler formula:

$\text{Blend}(x, 0, y)$

It's not quite true, however, that "All you need is blend." There are times when bounded addition is handier. But once you develop a feel for the flexibility and power of the blend operator, you'll use it frequently.

Lesson 23

Use only blend and bounded addition when calculating with bounded numbers.

Boolean Calculations

Some people think you don't need to dirty your hands with regular numbers. They claim that you can do a "good enough" job using just Boolean calculations. They are wrong.

A Boolean calculation is one using Boolean variables and Boolean operators. A Boolean variable is something that is either true or false. The statement "Socrates was mortal" is a Boolean variable that just happens to be true. "Socrates had bad breath" is a Boolean variable whose value is not known to us: It might be true, it might be false.

All Boolean calculations are ultimately composed of groupings of only four basic Boolean operators: NOT, OR, AND, and Exclusive-OR.

NOT takes a Boolean value as its input and returns the opposite of it. Thus, NOT("Socrates was mortal") is false because "Socrates was mortal" is true. Whatever the Boolean value of "Socrates had bad breath" is, NOT("Socrates had bad breath") is the opposite. Yes, this is so trivial that it seems a waste of space, but it turns out that you really do use this often enough to need to understand it.

OR takes two Boolean variables as input and returns a Boolean value of TRUE if either one of them is true. In other words, it comes out TRUE if one *or* the other Boolean value is TRUE. For example:

```
OR("Socrates was mortal", "Socrates was a hippopotamus") = TRUE
```

It also works if both Boolean values are TRUE:

```
OR("Socrates was mortal", "Socrates was a man") = TRUE
```

AND takes two Boolean variables as input and returns TRUE if—and *only* if—both variables are TRUE. For example:

```
AND("Socrates was mortal", "Socrates was a man") = TRUE
```

```
AND("Socrates was mortal", "Socrates was a hippopotamus") = FALSE
```

Finally, Exclusive-OR takes two Boolean variables as input and returns TRUE if one or the other—but not both—is true:

```
Exclusive-OR("Socrates was mortal", "Socrates was a man") = FALSE
```

```
Exclusive-OR("Socrates was mortal", "Socrates was a hippopotamus") = TRUE
```

```
Exclusive-OR("Socrates loved Cocoa Puffs", "Socrates was a hippopotamus") = FALSE
```

```
Exclusive-OR("Socrates loved Cocoa Puffs", "Socrates was a man") = TRUE
```

That's all there is to Boolean calculation. You take things that are either true or false, then you combine them with various sequences of these four Boolean operators, and you get answers. Of course, whether the answers are any good depends on what Boolean variables you use and what Boolean operators you use. Sometimes, Boolean stuff works great; sometimes, it's garbage. Here are some examples of what I mean:

Hamlet was a man.

Hamlet was a Dane.

Hamlet had no legs.

Hamlet loved his mother, Gertrude.

Hamlet trusted the ghost of his father.

Hamlet loved Ophelia.

Hamlet wanted to die.

Hamlet hated Polonius.

The first two of these statements are simple, straightforward, and true. The third is also simple and straightforward, and it is false. All the others, however, are not at all simple and straightforward. Each and every one of them is only partly true. Yes, Hamlet had positive feelings for his mother, Gertrude, but he also resented her for marrying his uncle. Yes, Hamlet mostly trusted the ghost of his father, but he had his doubts. He probably loved Ophelia, but he said some very unloving things. He seriously considered suicide, but did not act on that impulse. And while he held ill feelings toward Polonius, they were not strong feelings.

Thus, none of the last five statements are true, nor are they false. They cannot be represented as Boolean variables. But there's no problem with these statements when we express them in numeric form:

Affection_Disdain[Hamlet, Gertrude] = +0.4

Trust_Suspicion[Hamlet, ghost] = +0.5

Affection_Disdain[Hamlet, Ophelia] = +0.6

Will_to_Live[Hamlet] = +0.2

Affection_Disdain[Hamlet, Polonius] = -0.3

Thus, Boolean calculations are completely inadequate for dealing with Hamlet. It would be impossible to build a storyworld about Hamlet with just Boolean variables. On the other hand, it is theoretically possible to build a storyworld about Hamlet using numeric calculations. Such a task, of course, remains far beyond our capabilities at any time in the foreseeable future—but it is at least conceivable that this could be accomplished.



Couldn't we make Boolean calculations work by deploying huge masses of Boolean variables? We could get a pretty good definition of Hamlet's personality by assembling a thousand Boolean statements that precisely delineate the nature of his feelings toward others.

But how would you actually *use* all those Boolean statements? Would you really assemble something like this:

If (Hamlet_Loves_Gertrude AND Hamlet_resents_Gertrude_Marriage AND (Hamlet_Understands_Gertrude_Dilemma OR Hamlet_Thinks_That_Claudius_Forced_Gertrude_To_Marry_Him) AND Gertrude_Is_Nice_To_Hamlet AND Hamlet_Is_Nice_Back_To_Gertrude AND...

This is absurd! You simply cannot build a proper Boolean expression that captures human behavior in simple true-or-false statements. It's too complicated.



Couldn't the same thing be said for numeric expressions? Don't numeric descriptions fall too far short of human reality to be of any use in interactive storytelling?

Harken back to Figure 13.1 with the face in shades of gray. Observe that the Boolean image using only black-and-white pixels is completely inadequate to represent my face. The 3-bit image does an acceptable job, and the 8-bit image is pretty good. Nevertheless, not even the 8-bit image truly captures the immense complexity of the face, with its subtle colors and fine textures. Yes, it falls far short of the reality. But is it not adequate for communicating something? And is not the 1-bit image still inadequate?

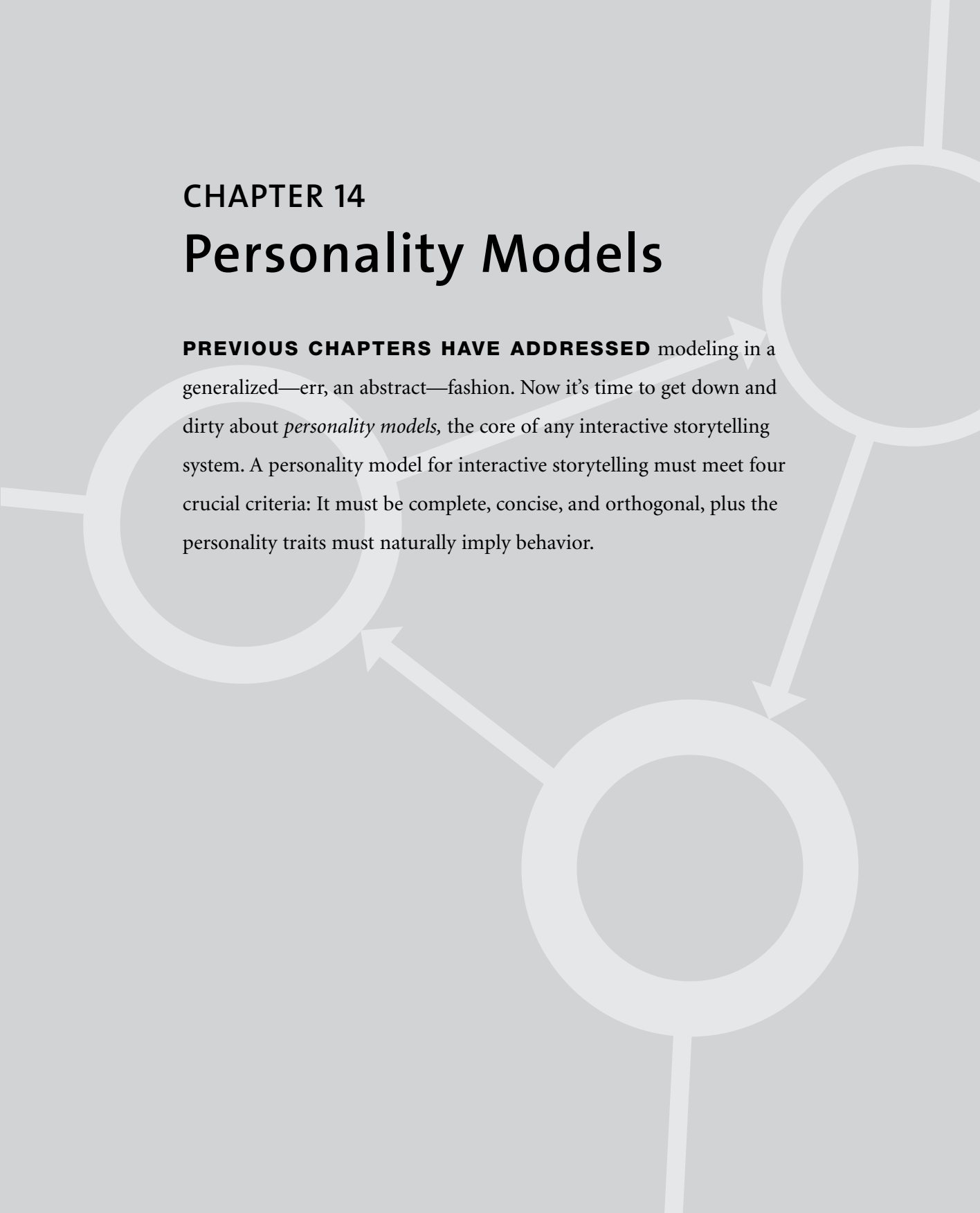
Conclusions

Number systems probably seem like a tedious and unnecessary consideration for the storyworld author, but once you start using mathematical models, you'll quickly realize just how many problems can arise with regular numbers. Bounded numbers are like the metric system—a little strange—but once you start using them, you'll find them a lot handier to work with.

CHAPTER 14

Personality Models

PREVIOUS CHAPTERS HAVE ADDRESSED modeling in a generalized—err, an abstract—fashion. Now it's time to get down and dirty about *personality models*, the core of any interactive storytelling system. A personality model for interactive storytelling must meet four crucial criteria: It must be complete, concise, and orthogonal, plus the personality traits must naturally imply behavior.

A decorative graphic consisting of three large white circles connected by white lines and arrows. One circle is on the left, one is at the bottom, and one is on the right. Arrows point from the bottom circle to the left circle, from the bottom circle to the right circle, and from the right circle to the top right.

Complete

The traits that make up the personality model must completely address all the behaviors we want to evoke in our storyworld. If our storyworld is to include sexual behavior, then we must have traits such as lust and sex appeal. If our storyworld is for children, then we should dispense with such traits. If our storyworld is a guy-type world with lots of action and only a token sexy chick, then we will need only one or two traits concerning romantic interaction. If we're creating a chick-flick storyworld, then we'll need more such variables.

Lesson 24

The personality model must cover the behavioral range of your storyworld.

Concise

Some storyworld authors are tempted to build a huge personality model containing every possible trait they can imagine. Whenever a design problem arises, they throw a new personality trait into their model—and poof! The problem is gone. This ultimately comes back to bite them, because as they add more and more personality traits, it becomes more and more difficult to determine which traits should be applied in any given situation. A good personality model must be small enough to keep inside your head at all times. If you have to consult it every time you use it, then you've made it too big.

Lesson 25

Keep the personality model as small as possible.

Orthogonal

Traits in the personality model must not overlap; if they do, then we will always face difficulties knowing which one to apply in any given circumstance. Suppose for example, that we have built a personality model with the traits `GoodHumored_illHumored` and `Friendly_Aloof`. These two traits are not identical, but they do overlap. That overlap will cause endless problems. Suppose, for

example, that a character with these traits encounters a stranger under inauspicious circumstances. In calculating the character's reactions to the stranger, the storyworld author will need to take both traits into account. It is difficult to imagine any situation in which one trait would apply but the other trait would not apply. If no such situation can be imagined, then there exists no functional difference between the two traits. They should be collapsed into a single trait.

A simple geometric analogy might help those who have never suffered the teachings of vector analysis. Imagine getting around in a city whose streets are laid out in a regular rectangular grid, with numbered avenues running north to south and lettered streets running west to east. If you are at the corner of Third Avenue and B Street, and you need to get to Eighth Avenue and D Street, then you know that you must go five blocks south and two blocks east. That's the only combination that will work.

But now suppose that you live in a city with a three-way street grid, as in **Figure 14.1**.

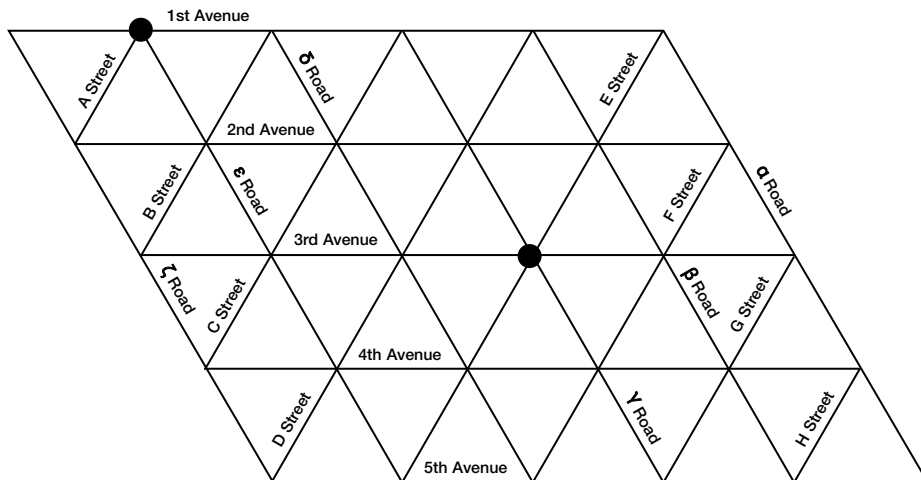


FIGURE 14.1 A city with a triangular grid

You could get from any point to any other point by a number of combinations. For example, you could go from the corner of First Avenue and A Street to the corner of Third Avenue and E Street by going two blocks east and two blocks southeast, or by going four blocks east and two blocks southwest.

Of course, no city is organized in this way, because it's silly. You don't need three sets of roads when there are only two sets of directions (north-south and east-west). You want to collapse those three sets of roads down to two sets. In the process, you'll get roads that are at right angles to one another, and that's what we mean by orthogonality.

This idea of orthogonality is really just a formalization of the concept of conciseness. If two variables are orthogonal, then they have nothing whatsoever in common. Hence, orthogonality is really just a means to achieve conciseness.

Lesson 26

Achieve conciseness through orthogonality.

Behaviorally Discriminating

We do not build personality models as an academic exercise; we use them for the pragmatic purpose of determining the characters' behaviors. Every personality trait in the model must be created with an eye toward the behavior it might control. `Spiritual_Materialistic` might seem like an important personality trait, but if your storyworld does not include spiritual behavior, then it doesn't belong in the personality model. The personality model must be created after the basic content of the storyworld is determined. Once you have answered the question "What does the user *do*?" by creating a list of verbs accessible to the characters, then you can start to build a personality model that permits characters to discriminate among those verbs.

This is probably the most important consideration for designing a personality model. For this reason, you should not design your personality model until you have a clear idea of the verbs you'll be using in your storyworld. You don't need to wait until all the verbs are listed; the verb list usually grows as you develop the storyworld. However, you should have a vivid image of the kinds of behaviors that you'll be expecting of your characters, and that vivid image will guide the design of your personality model. Then and only then can you ask for each possible personality trait, "How useful will this personality trait be in choosing among the options available to a character?" If you decide that the personality

trait in question will help determine a great many choices, then you should use it. But if you anticipate that it will be used in just a few special cases, it's probably better to ditch it. Perhaps we shouldn't call it a "personality model." "Behavioral control model" would be a better term, so as to keep your eye on the ball: behavior, not intrinsic personality.

Lesson 27

The personality model must be derived from the behavioral universe of the storyworld.

Overspecific

Storyworld authors are often tempted to add a new personality trait whenever they run into a brick wall when modeling behavior. For example, imagine Stella Storyworld Author, who is working on her storyworld about the joys and tragedies of parakeet owners. She's got all the usual personality traits in place, but now she's working on handling the verb *Massage*, by which a character massages her aching parakeet. Stella needs a personality trait to limit overuse of this verb, and her current palette of personality traits doesn't have anything remotely applicable. So she concocts a new personality trait, *Finger_Sensitivity*, to act as a brake on uninhibited parakeet massaging. Characters with high *Finger_Sensitivity* will balk at massaging their birds. The problem with this trait is that it is applicable only to massaging birds; it can't be used for anything else. Sure, it solves the bird-massaging problem cleanly, and it seems too insignificant a change to worry about. But custom personality traits are like potato chips: You can't stop with just one. Pretty soon, Stella's storyworld has personality traits like *Sweet_Tooth*, *Loves_The_Beatles*, *Flatulence_Proclivity*, *Stays_Up_Late*, *Sensitive_To_Loud_Tweeting*, and *Doesn't_Subscribe_To_A_Newspaper_So_Has_Nothing_To_Line_The_Birdcage_With*. Now she has a major mess on her hands. Harken, ye sinners, unto the voice of moderation; restrain thy lust for personality traits and stay on point!

Lesson 28

Don't create personality variables specific to individual verbs.

Types of Personality Traits

Four basic types of traits can be used in a personality model. The first type includes the obvious variables: greed, lust, pride, and so forth. They are the intrinsic personality traits that we associate with any character. Ordinarily, they don't change—but there are cases in which it's necessary to change them. For example, consider Darth Vader forsaking the Emperor and saving Luke, Frodo falling under the spell of the Ring, or Neo developing confidence in his skills. Changing these traits, however, should be a rare occurrence.

I call these *first-person* traits because they are represented in many data structures as entries in a one-dimensional array of actors: `Greed_Generosity[Gordon_Gecko]`, `Anger_Fear[Slim_Jim]`, and so on.

The second type of personality trait represents one character's perception of a second person's first-person trait. I refer to these as *second-person* traits because they are represented with two-dimensional arrays. You should probably start off thinking of these as relationships. Later on, you'll realize that thinking in terms of perceived traits rather than relationships will clarify your thinking process.

This notion of first-person and second-person—intrinsic versus perceived—variables might strike you as odd, but it makes a great deal of sense when you approach it in a larger context. For example, consider a relationship such as trust. Could we not say that trust is really one person's perception of another's honesty? Is it not reasonable to say that one character's perception of another's `Good_Bad` trait is really an expression of how much `Affection_Disdain` the first character feels for the second?

Second-person traits are not necessarily symmetric. Mary can hate Jane while Jane only feels slight distaste for Mary. John can trust Jemima even though Jemima doesn't trust John.

This way of thinking about personality variables offers an extra benefit: It can be usefully extended to three dimensions. A third-person trait would then be one person's perception of another's perception of a third person. While that may sound like a mouthful of egregious indirection, third-person statements are made every day: "No, Joe would never trust Marcie." The speaker is expressing his perception that Joe holds a negative perception of Marcie's honesty. Or, "Don't invite both Jeanette and Jason to the party; they hate each other. Invite one or the other." These variables are depicted in **Figure 14.2**.

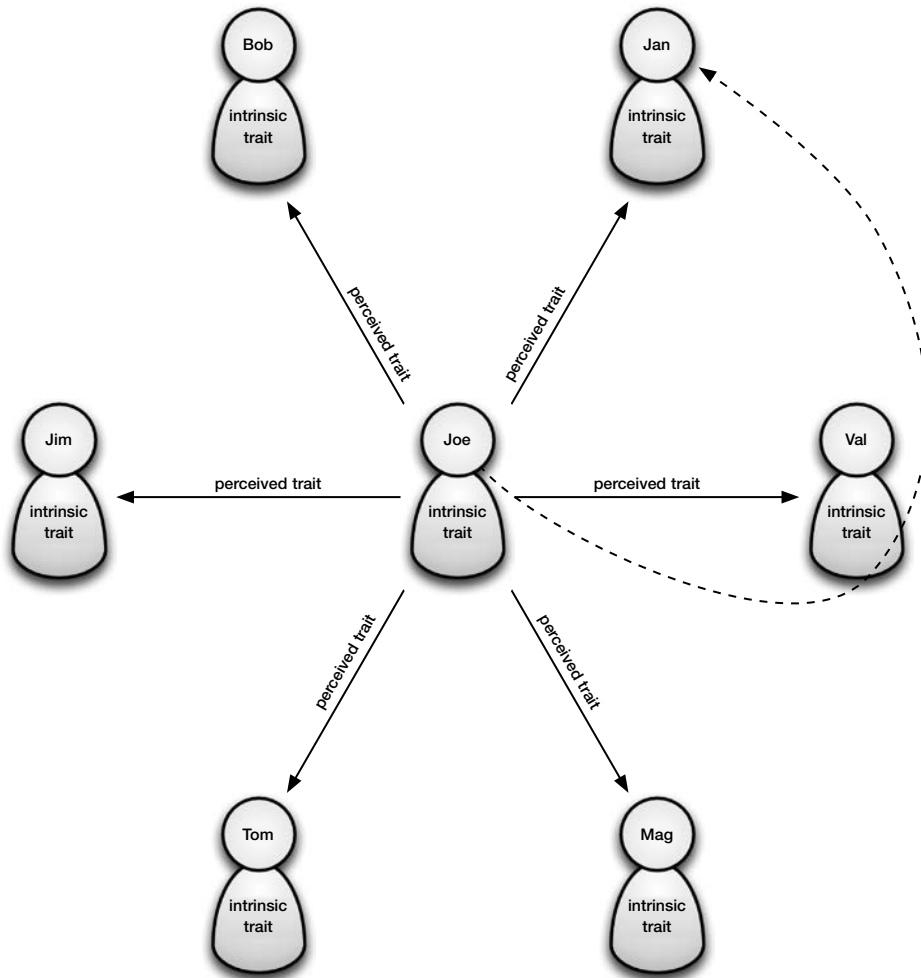


FIGURE 14.2 First-, second-, and third-person traits

The fourth broad type of personality variable is an *accordance* variable; it represents the degree to which a character will readily perceive high values of an intrinsic variable in another. The most obvious example of this is the concept of gullibility—the willingness to accord trust to another. Its opposite, suspiciousness, is a proclivity to accord small amounts of trust to others. Similarly, naive versus cynical would describe the accordance value for Nice_Nasty. Every variable that forms the basis of a relationship should probably have an associated accordance variable.

The Five Factor Model

One of the most common personality models used in psychology is called the Five Factor Model (or the “Big Five” factors). Those traits are:

- ▶ **Open:** Curious, enjoy novelty, creative
- ▶ **Conscientious:** Organized, disciplined, dependable
- ▶ **Extraverted:** Gregarious, sociable, loquacious
- ▶ **Agreeable:** Sympathetic, cooperative, helpful
- ▶ **Neurotic:** Emotionally unstable, anxious, poor impulse control

Here’s how I would rephrase it for interactive storytelling:

- ▶ **Adventurous_Cautious:** Curious, inventive, novelty seeking versus traditional, staid, resistant to change
- ▶ **Disciplined_Sloppy:** Organized, efficient, dutiful versus spontaneous, easygoing
- ▶ **Sociable_Solitary:** Gregarious versus aloof
- ▶ **Friendly_Unfriendly:** Cooperative, amiable versus cold, antagonistic
- ▶ **Anxious_Confident:** Flighty, emotional versus steady, solid

How well does this set of personality traits meet my previously stated set of requirements for personality models? All in all, I think it does a good job with completeness, conciseness, and orthogonality. However, I have reservations about its utility for discriminating behavioral options. The Five Factor Model is a scientific endeavor to determine the elements of human personality, and it does a good job of describing real people. But the needs of interactive storytelling are quite different from the needs of psychology. Real people are more boring than the characters in drama. They make prudent decisions while characters in stories make bold ones; real people hedge their bets while story characters throw caution to the wind. Real people stay well within the envelope of socially acceptable behavior while the characters in stories deliberately push the edges of that envelope to explore the human condition. I have eschewed the Five Factor model, but I’m sure that somebody will use it profitably someday.

Lesson 29

Build your personality model for utility in your storyworld, not accuracy in the real world.



OK, hot shot, why don't you put your money where your mouth is and specify a personality model that fits your criteria?

I have tried many approaches to personality modeling. After all these experiments, I have come to the conclusion that a simple model with just three variables will handle most storyworlds.

Nice_Nasty

This represents the basic goodness or badness of the character. Princess Leia is very nice, the Emperor is very nasty. The perceived value of this variable, which I call $p_{\text{Nice_Nasty}}$, serves to measure the affection that one character has for another. In other words, if you like somebody, you are in effect embracing the belief that the person is nice. If you hate somebody, you necessarily think that they're nasty. The third-person version of this variable is $t_{\text{Nice_Nasty}}$, and it represents how much one character thinks another character likes a third character. The accordancy variable, $a_{\text{Nice_Nasty}}$, represents an actor's willingness to perceive friendliness, amicability, or magnanimity in another actor.



Now just a minute here. You're mixing together a bunch of things that are not identical. Magnanimity is not the same thing as friendliness, and affection is not the same thing as the degree to which you think a person is nice.

You're absolutely right: I *am* crushing some distinct ideas into a single one-size-fits-all category. That's intrinsic to the artistic process. Every artist simplifies in order to clarify. When I was a youngster, I was given only a handful of crayons to draw pictures with. As I got better, I could use a wider range of colors. But if they'd given me a palette of a thousand colors as a child, I wouldn't have known how to use them. We're still in the training-wheels phase of interactive storytelling; it's prudent to keep things simple for now. Once some real artists arise, we can start expanding the palette of personality traits. But for now, I have learned the hard way that we have to keep it simple.

Honest_False

This represents the character's integrity. It's not just whether the character tells the truth; this also covers the degree to which the character honors their promises. Obviously, `pHonest_False` (the character's perceived value of the second character's `Honest_False` value) can be thought of as trust, and `aHonest_False` (the accordant value of `Honest_False`) corresponds to a person's gullibility or suspiciousness. Similarly, `pNice_Nasty` can be thought of as how much one person likes another, and `aNice_Nasty` can be thought of as a person's belief that all people are basically good.

Willful_Pliant

This is the most difficult of the three primary variables to grasp. A person with a lot of will is assertive, pushy, or dominating; a person with a negative value of this trait will be shy, retiring, or submissive. This gets especially tricky when we consider the meaning of `pWillful_Pliant`. The perceived value of this trait is somewhat like fear or wariness: A person with a high value of `pWillful_Pliant` for another will be very careful in their presence, knowing how readily the willful person will assert themselves. The accordant value (`aWillful_Pliant`) is closely related to the inverse of the intrinsic value itself. That's because the intrinsic value is inherently about relationships with others. A dominating person will not be inclined to perceive others as dominating. It's more likely that such a pushy person would see others as pushovers, while a submissive personality would see everybody else as strong-willed.

My experience is that these three personality traits can be stretched to cover just about any situation. They fall short in many areas, but they're clean and simple to use. For example, is `pNice_Nasty`, which I liken to affection, the same thing as romantic love? Did Romeo think that Juliet was just the nicest person in the whole wide world? Not quite, but the fit is close enough that it will work in most applications.

Expanding on the Three Core Personality Traits



This is a pretty narrow set of personality traits; isn't it possible to augment it to get a better overall personality model?

These three personality traits—Nice_Nasty, Honest_False, Willful_Pliant—comprise what is, in my opinion, the best overall personality model for interactive storytelling. However, if you wish to expand upon it, following are some possible additions to consider.

Loquacious_Taciturn

If your storyworld involves a great deal of gossiping, you'll need some way to determine a character's proclivity to spill the beans. The closest approximation to this that you can get with the three core traits is Willful_Pliant. In fact, it's probably best to treat everybody as equally loquacious unless you need to behaviorally differentiate the motor-mouthed characters from the tight-lipped ones.

Generous_Greedy

Greed is an important motivation in many stories, so perhaps it deserves its own trait. It could be approximated with a combination of Nice_Nasty and Willful_Pliant: A greedy person is nasty and willful.

Chaste_Licentious

This is obviously of value in a storyworld revolving around sexual relationships.

Humble_Proud

This might seem like a great personality trait to include, but it overlaps both Nice_Nasty and Willful_Pliant. After all, have you ever known a dramatic character to be both humble and nasty, or both nice and proud? Simultaneously humble and willful, or simultaneously proud and pliant? Those are pretty rare combinations, which suggests that you probably don't want to use this trait.

Ascetic_Sensual

This could be approximated with Generous_Greedy, but I can easily imagine a storyworld in which this particular trait could be useful. Still, it's not something I expect to be common.

Smart_Stupid

This looks like a good idea at first, but consider how you would use that trait to alter the character's behavior? Can you imagine situations in which a stupid character would choose a different option than a smart character? I can think of a few, but not enough to justify this trait.

Physical Traits

A subset of personality traits are actually physical traits, and there are some you might want to include in your storyworld:

Beautiful_Ugly

Strong_Weak

Short_Tall

Young_Old

Skinny_Fat

I warn you, though, to ask yourself this: If your storyworld needs to use such traits, is it really a dramatic storyworld? Did any of these traits play an important role in stories that you recall? Would *The Lord of the Rings* have been different if Frodo had been strong, tall, old, or fat? How about Neo in *The Matrix*? Tom Sawyer? Hamlet? These are dangerous traits to use; if you really need them, ask yourself if you're building a storyworld or a video game.

Moods

We must also account for the variable moods that characters are subject to. These are personality traits, too, but they spontaneously change with time. An event can cause a mood to spike upward, but then it automatically loses strength with time. For example, when Hoss calls Slim Jim a low-down weasel, Slim Jim's anger should jump upward, possibly inducing him to act; then it starts to decline. If Hoss insults Slim Jim a second time, the anger generated by the second insult will be added to the residual anger from the first insult, possibly pushing Slim Jim over the edge.

Like personality traits, moods are bipolar and should be represented as two opposing forces, with the normal value being 0.0. Specific events can then push the mood in either a positive or a negative direction.

I use four moods in my own work.

Happy_Sad

Quod verba? (What need is there for an explanation?)

Aroused_Disgusted

This may seem like an odd combination. But it is impossible to be both aroused and disgusted at the same time; surely this suggests that they are opposing moods. In the previously mentioned movie *All of Me*, Steve Martin plays Roger Cobb, who (by a wild sequence of events) ends up sharing his mind with Edwina, a socially repressed rich woman. At one point, a beautiful woman attempts to seduce Roger, who is all too happy to comply. Prudish Edwina attempts to obstruct the natural course of events by introducing images of dead kittens into their joint mind. The ploy works.

Angry_Fearful

Again, you might question my joining these two moods as opposites—but again, it is impossible to be simultaneously angry and fearful. It is certainly possible for fear to suddenly give way to anger or vice versa, but they don't exist simultaneously in the mind. They are the emotional basis for the fight-or-flight response.

Energetic_Tired

This is really a physical trait, not a personality trait, so it may not be of use in many storyworlds.

Inclination Formulae

Building the trait set for a personality model can be a huge task, but it is only the beginning. The trait set itself is useless; the whole point of personality modeling is to equip our characters with the means to make reasonable, understandable decisions.

Decisions are choices between options. We don't decide "27" or "banana"; we decide to execute one option or execute a different option. We might decide to buy 27 screws or we might decide to eat the banana, but remember that the real options are the verbs: buy and eat. Thus, the next task of the designer is to create a system that links the personality model to the verbs. These links take place through mathematical statements that use the personality and relationship traits to make choices. I call such mathematical statements *inclination formulae*.

Most beginners think about choices in Boolean terms: Will the character execute the verb or not? Like most Boolean formulations, this is entirely too simplistic an approach. Seldom does a character face a choice between simply doing something or not doing it. Instead, characters always face multiple-choice problems: Which of several options does the character choose? Moreover, none of those choices are likely to be obviously right or obviously wrong. If there's any drama in the decision, each choice should be plausible. The real question is always "Which option is best?" The only way to answer that question is to come up with numbers representing how good each option is. The best option is the one with the highest numeric value of "desirable." I call that number the character's inclination to execute that verb.

Inclination formulae always include both personality and context. The character's personality traits are combined with traits intrinsic to the situation to yield an inclination. For example, let's look at a classic barroom brawl scene from a Western movie. Slim Jim has just called Tex a low-down snake. How does Tex respond? Let's say Tex has three options in this situation: He can laugh off the insult, he can counter with another insult, or he can engage Slim Jim in manly fisticuffs. How does he choose among these three options?

We need to write three separate inclination formulae, one for each option. What personality traits would apply? Obviously, `Nice_Nasty` will play an important part: The nastier Tex is, the more likely he is to select the more aggressive

options. His `Willful_Pliant` value will also be important here; a strong-willed fellow won't take any crap. Lastly, we should probably throw in some consideration for Tex's `pNice_Nasty` for Slim Jim; if they're old friends, Tex is more likely to laugh off the insult.

We have three variables to apply to each option, which seems perfect for an application of the blend function. Remember, we want the first two traits to push Tex toward the violent options, while the third trait pushes him toward the laugh-it-off option. Hence, big values of the first two traits should increase the inclinations for the latter two options, while a big value of `pNice_Nasty[Tex, Slim Jim]` should increase the inclination for laughing it off.

Let's start with the option `Laugh_It_Off`. Working in its favor are `pNice_Nasty[Tex, Slim Jim]` and `Nice_Nasty[Tex]`. Working against it is `Willful_Pliant[Tex]`. Let's try this formula:

```
Inclination[Laugh_It_Off] = Blend(Willful_Pliant[Tex], Nice_Nasty[Tex],
pNice_Nasty[Tex, Slim Jim])
```

If `pNice_Nasty[Tex, Slim Jim]` is positive, it will bias the blend toward `Nice_Nasty[Tex]`. If it is negative, it will bias the blend toward `Willful_Pliant[Tex]`. You might want to go through the exercise of trying different values for each of the three variables to verify that this works properly. If you're ambitious, you could even put together a spreadsheet to examine its behavior more closely.

Moving on to the next verb, `Insult`, we see that the thinking used for the previous inclination formula is reversed here. So let's just use the reverse of the previous inclination formula:

```
Inclination[Laugh_It_Off] = -Blend(Willful_Pliant[Tex], Nice_Nasty[Tex],
pNice_Nasty[Tex, Slim Jim])
```

That was easy! But now we face a more difficult problem in the inclination formula for the verb `Fisticuffs`. We need to make it like the inclination formula for `Insult`, only worse. That is, we need to require higher values of one or more of the personality traits to make this option trigger. We have a number of ways to accomplish this, but I'll present the simplest one I can think of:

```
Inclination[Fisticuffs] = -Nice_Nasty[Tex]
```

The idea here is that nice people don't start fistfights, no matter how mad they are.

Two Exercises

If you really want to come to grips with the ideas in this chapter, I suggest two exercises. First, the easier exercise is devising the variables of a personality model. I offered one such list of variables; can you come up with something better?

Try your hand at creating a list of the variables for a personality model for a storyworld about high school dating. Remember the basic desiderata: complete, concise, orthogonal, and behaviorally discriminating. How well can you meet those desiderata? The result of your work should be a list of all the variables used in the model, including complete descriptions of their behavioral implications. When you're finished, evaluate your model with the following questions:

- ▶ How many variables does your model use? (The fewer, the better.)
- ▶ What kinds of behaviors cannot be discriminated by your model? For example, can it decide whether the girl should kiss the guy passionately or just normally?
- ▶ If a storyworld author were using your model, would it always be obvious in any given situation which trait should be used. Or, can you imagine situations in which the author might be torn between two related traits? Might the author feel a need to use lots of different traits in some complicated mess?

The second exercise you might try is revising the three inclination formulae I presented. First, write a little spreadsheet or program that calculates the three inclinations from the values of the three personality traits; then, try out various combinations of values for those traits. Examine those results and decide if you want to change anything. Should you apply weighting to any of these traits?

Conclusions

Although there are a million ways to design a good personality model, there are a billion ways to design a *bad* personality model. At first glance, it seems so easy that some people just slap together a mess of ideas and call it a personality model. When you design your personality model, remember that achieving real-world accuracy is less important than achieving storyworld utility.

CHAPTER 15

Fate Makes the Storyworld Go 'Round

BRENDA LAUREL FIRST COOKED UP the idea behind Fate. In her 1986 doctoral dissertation, she described a software agent she called “the Playwright.” Laurel specified 13 functions that the Playwright should execute:

- ▶ Model the plot in progress.
- ▶ Specify the formal characteristics of upcoming events.
- ▶ Change the storyworld.
- ▶ Modify the goals of characters.
- ▶ Access proposals for future actions of characters.
- ▶ Simulate these proposals to determine their effect on the plot.
- ▶ Evaluate the results of the simulations.
- ▶ Mandate future events.
- ▶ Formulate the script for the next event.
- ▶ Direct the characters.
- ▶ Control its own operation.
- ▶ Remember past events.
- ▶ Learn from past results.

Of course, Laurel was too early to delve into the technical details of how these functions could be realized. Her 13 functions are a wish list, not a plan. Still, it represents the first substantial attempt to consider the problems of interactive storytelling. Interestingly, Laurel was approaching the problem from the artsy side rather than the techie side; it would take years for the techies to catch up with Laurel.

A number of researchers have accepted the challenge of actually building such technology. Nowadays, many of them use the term *drama manager* to refer to a software agent that oversees the development of the story and somehow guides it in desirable directions. I prefer the term *Fate*, because it comes from the world of storytelling, while “drama manager” is a creature of computerdom, a mere variation on file managers, sound managers, and user interface managers.

I believe that Laurel’s laundry list is too long. Although the functions on her list are all desirable, I think it is possible to specify Fate’s responsibilities more succinctly. Indeed, just three fundamental steps are required, all of which spring from my definition of interactivity: “A cyclic process in which two agents alternately listen, think, and speak.”

Listen

Fate must monitor the progress of the story by listening in, and by “listening” I mean much more than merely accepting input. To hear another person speak, you must be able to interpret the noises in terms of the language used. Those squeaks and grunts must all be translated into words and sentences, which must then be analyzed to determine their meaning. In the same fashion, Fate must interpret the raw material of the developing story (its events) in terms of abstract dramatic constructs. It must recognize the patterns of behavior that differentiate drama from tedium.

A simple way to accomplish this relies on *overview variables*. These are numbers calculated by the storytelling engine that assess the overall state of the storyworld. For example, you could calculate the sum of all the affections held by all the characters for each other; this number would indicate the overall level of “good feeling” in the storyworld system. A low value means that everybody hates each other; a high value indicates universal love and brotherhood. Fate monitors this value to determine what to do. If everybody is too happy, maybe it’s time to toss something

ugly into the mix. If the storyworld has degenerated into a dog-eat-dog jungle, then perhaps it's time to bring some soothing sunshine to the cast.

The database for a storyworld will be large; all those numbers can be put together in a staggering number of ways. Following are just a few possibilities: gross trust, social acceptability of violence, dullness, storyworld exploration, conflict, and monitoring individual characters.

Gross Trust

Gross trust is the sum of all the trust that all the characters feel for each other. At low values, this variable signals a complete breakdown of social cooperation. Every social situation can be characterized by the group's overall sense of social solidarity. That sense profoundly affects the behavior of the members of the group.

Social Acceptability of Violence

The social acceptability of violence is a running sum of violent actions. As this sum grows, characters' inhibitions against violent behavior lower. This is another case where a social group develops a sense of what is acceptable and what is not. The first time somebody resorts to violence, it shocks the group and brings opprobrium down onto the perpetrator. As more people resort to violence, the changing atmosphere encourages more violence.

Dullness

Dullness is the sum of verb import values over the last 100 events. If the sum grows too small, then the cast has settled down into a dull period and the storyworld needs some spicing up. Fate should inject some sensational or divisive event.

Storyworld Exploration

Have all the verbs in the storyworld been used at least once? Has the player spent most of the time using a small set of verbs? Here's a simple formula that Fate can use to calculate how thoroughly the player has explored the storyworld's verb set:

$$N * \sum \text{count}_i^2 / (\sum \text{count}_i)^2$$

Where N is the total number of verbs and count_i is the number of times that the i^{th} verb was used. In the theoretically ideal case, which you will never achieve, this formula will yield a result of 1.0. In a real storyworld, it will be greater than 1.0. Fate can swing into action when the result exceeds some critical value. That critical value, however, is different for every storyworld. Therefore, have Fate measure this quantity through many rehearsals and get a good idea of the typical value produced by good playings of the storyworld. You can use that as your criterion for Fate intervening.

Remember, we're not talking about spatial exploration; we're talking about dramatic exploration of the verb set.

Conflict

A more apropos global concept to monitor is dramatic conflict. However, this concept is more difficult to quantify. Fate would monitor how antagonistic each action taken by the protagonist is. This can be done by gauging the nature of the emotional reactions of the antagonist toward the protagonist's actions. In other words, if the protagonist is doing things that displease the antagonist (and vice versa), then there's dramatic conflict.



But what if the antagonist is setting the protagonist up for a dirty trick down the road? The antagonist is all smiles, lulling the protagonist into a false sense of security, until the antagonist can spring the trap. Wouldn't Fate also be fooled by such behavior, thinking that there's little dramatic conflict, when anyone can see there's trouble coming?

Yep, that's a problem, but we're nowhere near that level of dramatic sophistication yet. Someday, this kind of problem will indeed bedevil us, but for now, I'd be ecstatic if we could build storyworlds rising to the level of dramatic power of *Gilligan's Island*.

Monitoring Individual Characters

Overview variables provide us with a gross assessment of the current situation; more illuminating assessments are desirable. Another stratagem arises from monitoring individual behavior. This requires the author to establish beforehand some behavioral expectations for each character; the obvious means for doing so would be to define goals for each character. These goals could be specific, such as "Get the Rubaiyat of Omar Khayyam," "Rescue the Princess from the dragon,"

or “Induce Mary Jane to marry you.” They could also be general, such as “Amass wealth,” or “Earn the affection of everybody,” or “Prove your manhood.” Fate could then monitor the progress of each character toward his or her goal. The thinking part of this stratagem, however, is immensely difficult.

Think

Having developed a coherent dramatic interpretation of the events so far, Fate must then determine how the story should progress from that point forward. The primary means for accomplishing this is the inclination formula, which specifies the best response to any given event. Inclination equations, however, serve only to specify immediate, knee-jerk reactions to events.

Some improvement can be obtained by arranging for Fate to intervene in events. By injecting new events into the stream of events, Fate can influence the development of the story.

For example, suppose the protagonist (Romeo) wants to romance Juliet, and by some weird turn of events, Juliet falls in love with Benvolio. While this could present an interesting digression, it might be prudent for Fate to obviate such a development. This could be done easily enough by Fate having Benvolio loudly belch at the precise moment they begin to kiss (or some similar ruse). This is what I mean by an intervention from Fate.



But shouldn't the storyworld be permitted to progress according to the natural processes built into it? Shouldn't authors honor their dramatic universe by eschewing intervention in favor of a laissez-faire dramatic policy? To use your "author as god" explanation, shouldn't the author minimize the need for miracles?

Let's remember that the algorithms the author uses to define a storyworld are limited and cannot possibly address all situations. There will always be special cases that require special-case treatment. Some of these special cases can be dealt with directly in the inclination formulae, but some can arise from so many different pathways that intervention is the simpler resolution.

A well-designed set of overview variables will make the decision-making task easier, because each overview variable will indicate some significant factor in the development of the story. Each overview variable discussed suggests some action

that Fate should take. Conceivably, Fate would consist of nothing more than a large set of overview variables, each of which triggers a specified event if it goes beyond its acceptable range.

If you choose to monitor individual behavior, your task is much more difficult. For a character to get from his or her current situation to the defined goal, Fate must map out a plan involving a long sequence of actions. This kind of problem has been solved for narrowly defined situations such as plotting a path for a Mars rover across uneven terrain. But the problem we face here is more complex in two ways: First, the range of choices available to the character is much larger, consisting of all the verbs in all their variations. Second, the process requires anticipation of the responses of other characters to the planned actions, which are difficult to predict. While it might be possible to pull this off for short plans, I don't think the creation of long-term plans lies within our grasp.

Speak

Fate's final task is to speak to the user; that is, to translate its determinations into some form that will actually change the storyworld and help the story evolve in the desired direction. You can use any or all of five different schemes to influence the evolution of the story.

Environmental Manipulation

The most overbearing technique is to make environmental alterations in the storyworld. If the player attempts to go over the mountain pass, and Fate determines that the player should instead be traversing the tunnels under the mountains, then Fate need only throw a few snowstorms and avalanches at the player. If that doesn't work, have the trail end abruptly at the top of a cliff. All manner of physical constraints can be applied to force the player into the intended course of action. Such devices, however, are often transparent and insulting to the player. Use them only as a last resort to save the developing story from catastrophe.

In a different set of circumstances, using physical intervention is more satisfying. For example:

It's the climax of the movie. Our hero has driven the villain out of his stronghold and pursued him up the craggy mountaintop. The trapped antagonist turns and

fighters, and the two struggle as the wind tears at their capes and the rain lashes at their faces. A rock gives way under the hero's feet and he slips, nearly plummeting into the yawning chasm below. But somehow his fingers find a handhold and he dangles helplessly at the feet of the delighted evildoer. The villain sneers and picks up the hero's own sword, raising it high over his head to deliver the deathblow. Suddenly, a bolt of lightning strikes the villain, who plunges off the mountain peak, his final scream echoing off the cliffs. The hero pulls himself up, retrieves his sword, and gazes down at the broken body of the villain far below.

Yeah, that works.

Goal Injection

A more subtle approach is to instill some new goal in a character. This new goal, however, must of course be compatible with the personality of the character. It simply won't do to have Gandalf suddenly decide that he wants to make whoopee with Galadriel, for example. In addition, this new goal should be temporary so that it affects only the immediate situation. If your player as Frodo Baggins is just about to do something seriously wrong, make Samwise cross his legs desperately, rock back and forth, and insist that he absolutely *must* go urinate. Then get Samwise into serious trouble, diverting Frodo from his intended course.

Shifting Personalities

The most elegant solution is to alter the personalities of other characters in such a way that induces them to make decisions that influence the player in the desired direction. This is the most indirect of the approaches; Fate makes changes that cause events that will in turn affect the player. That's what makes it so elegant, but it also makes it difficult.

Plot Points

Some special-case verbs, which are meant to advance the development of the story, are triggered by Fate according to some predefined set of circumstances. The easiest technique relies simply on the passage of time, using a sequence of timed plot points that force the story forward. These timed plot points are not the consequence of any action taken by any character; they simply pop up at the preassigned time, forcing the story to move forward.

For example, in my Arthurian storyworld, Mordred intends to usurp the throne. He lays the groundwork for this by fomenting discord within the social group at Camelot. At some point, the clock alarm goes off, causing Mordred to raise the flag of rebellion and begin recruiting an army to attack Arthur. The action of raising the flag of rebellion is the timed plot point that triggers the transition in the story. All of this inevitably leads to the climactic battle resulting in Mordred's defeat, the Long Walk Down the Short Spear, or the utter defeat of Arthur. This simple time-based scheme forces the story forward regardless of the player's actions. The player is kept busy reacting to Mordred's actions and striving to retain the loyalty of the other characters.

Timed plot points aren't the only workable technique; it's also possible to trigger plot points based on the value of an overview variable. For example, the player could be given many ways to solve a problem. After three failed attempts—regardless of what the player attempted—an easier route becomes available. Or, Fate could trigger a plot point only when the player has somehow whipped up some group of characters into a frenzy. Plot points based on some aggregated value open plenty of dramatic possibilities.

Interstitial Stories

Interstitial stories are a variation on what video games call *cut scenes*. The basic idea is to break away from the normal storytelling process and use a completely different presentation style to deliver a short story with limited interactivity. Here's an example of an interstitial story from my game *Trust & Betrayal* (1987).

*Your conversation with Kendra is interrupted by her aunt Dracina. The old Klast used to run the clothing group and was generally feared and hated by all who came in contact with her. Old age has not mellowed Dracina; she is still a tyrant. She gives you hardly a glance and proceeds to converse with Kendra in Klast. This is in itself quite an insult, as it is long-established convention that Eeyal is the only proper language to use in mixed groups, unless pragmatic reasons require the greater precision and expressiveness of a local language. You shrug off the insult, for Dracina is chewing out Kendra over some infraction. Kendra herself is obviously embarrassed by her aunt's intrusion at a very delicate moment and seems to be trying to placate her and get rid of her. Dracina, however, seems determined to press her point; an ugly family fight seems about to explode. What do you do? *

\

Nervously wait and see what happens.

D10-dA8-a Sure enough, a fight ensues; Kendra bursts into tears and her aunt storms out.

\

Excuse yourself to another part of the house and wait there.

D2-dA4+a You can hear them squabbling for a while; then Kendra comes, apologizes, and you both try to pick up where you left off.

\

Tell Dracina that her conversation with Kendra can wait until later.

D4-dA8+a Dracina is outraged; she tongue-lashes you in Klast and stomps out. Kendra smiles appreciatively.

\

Ask Kendra if she will allow her aunt to interfere with campaign business.

A8-aD6-d Kendra looks chastened; her aunt continues to carp for a while and then leaves.

\

The odd characters scattered through the text are part of a simple scripting language I designed for interstitial stories. They divide the text up into chunks delimited by the backslash (\) character. The first chunk is the story presented to the player; the next four chunks represent options available to the player. Each option is presented in two lines: the first line is the text of the option that will be shown to the player; the second line is the response to taking that choice. That response is preceded by a small calculation that alters the relationship between the player and the character Kendra. For example, the first calculation (D10-dA8-a) subtracts 10 from the Dominance value of Kendra for the player, then subtracts 8 from the Affection value of Kendra for the player.

This system integrates the choice made by the player into the development of the story.

Dropping the Fourth Wall

The crudest technique of all is to drop the fourth wall—break out of the story to speak directly to the player as a player, not as a character—and explicitly advise the player as to actions that are inhibiting the satisfying development of the story. I used this technique when using interstitial stories in *Trust & Betrayal*, so I suppose that I can lay claim to having sired the first Fate. Here’s an example of a typical response when it detected a player problem; the following text appeared in a new window on the screen:

Chris Crawford is waiting outside, dressed in a zoot suit, swinging a pocket watch by a fake gold chain.

“Struck out again, huh, baby?” he taunts. “Maybe you haven’t figured it all out yet. Let Uncle Chris give you some advice.” He sidles up to you, blocking the escape path that you were eyeing. “I bet you just walk in on somebody and start offering them deals, right? Well, maybe you can get away with that kid stuff in the beginner level game, but in the expert level game, you gotta use your head. Look, how would you feel if somebody came barging into your house talking serious deals about tattling on other people? You’d be pretty reluctant, wouldn’t you? Especially if they jumped right into it without so much as a howdy-do? You gotta break the ice first. You walk into somebody’s house, you say ‘hello.’ It’s a courtesy thing, you know what I mean? Then you make a little small talk, to sorta warm things up. You can really warm up the conversation by offering some free information about somebody betraying somebody else. Just don’t pop the first deal-offer until you’ve got ’em warmed up and ready. So, baby, the name of the game is...”

The sound of a big band playing softly fades into the scene; Chris starts a little soft-shoe and starts singing “Getting to Know You.” By god, is he bad! You stand dumbfounded as he dances to the left, disappearing behind a small shed. A clatter of garbage cans terminates the music.

As you can discern, my claim to fame is not a particularly proud one.

Plot Enforcement

A frequently used ploy attempts to force the player to hew to the author's intended plot. The coarsest of these is the "Kill 'Em If They Stray" approach I described in my discussion of *Dragon's Lair* in Chapter 5, "Verb Thinking." The basic idea is to set up a single storyline and make alternative paths available to the player. However, should the player be so insolent as to actually try one of these alternative paths, the game kills the player. "You can have any story you want," says the designer, "so long as it's mine."

A less heavy-handed approach attempts to steer the player in the right direction. For example, in one example I found, the author wanted the player to enter a diner upon leaving a park. The player was given a simple choice between entering the diner and passing it by. If the player chose to pass it by, Fate informed the player that the aromas emerging from the diner were enticing and renewed the option to enter the diner. If the player still refused to enter the diner, Fate informed the player that his stomach was growling, he was desperately hungry, and there were no other sources of food visible. Presumably, this was enough to get the player to meet his responsibilities and enter the diner. But is it interactive? I think not.

The most subtle form of plot enforcement simply never gives the player the option to stray from the planned plot line. For example, in the diner example, the author would simply declare that the player enters the diner and then give him a choice of different meals to eat. The author frog-marches the player down the defined plot line, giving him meaningless choices along the way as a sop to interactivity. Like the previous schemes, this is not interactive: The player has choices, but they are meaningless.

Plot-enforcement schemes are fundamentally flawed: They attempt to deprive the player of the freedom to make dramatically significant decisions. Interactive storytelling demands that the player be able to pilot his or her own course through the storyworld. Anything less is fraudulent interactivity.

“Correcting” the Player

Some authors have attempted to develop systems that aren’t quite as intrusive as the plot-enforcement methods. Although the styles vary, the central concept in each is to provide hints that the alert player should recognize as guides to the “correct” conclusion of the storyworld. In *My Arrogant Opinion*, the very notion of correcting the player implies the imposition of the author’s will onto the player.

As explained in Chapter 4, “Abstraction,” there is no reconciliation between plot and interactivity—reconciliation can be achieved only by moving to a higher level of abstraction. Thus, the notion of correcting the player derives from an obsession with plot at the concrete level; it is fundamentally misguided and doomed to failure. The author who assumes an adversarial role with the player ruins their relationship before they even begin.

Good software designers take full responsibility for their creations. If the user perceives that a problem has arisen, it’s not the user’s fault—it’s the designer’s fault. Users can never be in error; if the system consisting of user and software fails to carry out the user’s desires, the blame falls entirely on the software design.



That’s not fair! What if the user refuses to read the manual?

The degree to which a manual is required is indicative of poor design. A truly brilliant design needs no explanation; its operation is transparent. This is, of course, a grand ideal never achieved in practice, but it is certainly a desideratum of good software design. We mortals will never achieve such divine perfection, but inasmuch as we are playing god every time we create software, we must acknowledge how far short of true divinity we fall—and we must not blame our users for such shortfalls. To begin the design process with the assumption that the user will make errors betrays the user. The initial assumption should be that the user is always right. So, the software must focus on how to respond if the user’s wishes deviate from what was anticipated.



But what if the player wishes to do something idiotic? Must the designer humor idiocy?

If that's what the player wants, by all means! Remember, the player is paying for this software. If he wants to jump off a cliff, behave boorishly, befriend Darth Vader, or pick his nose, then who are you to defy him? If he jumps off a cliff, let him die—what else can he expect?



But I am the storyworld author, the artist who creates this artistic expression. I'll not be party to perversion and ugliness.

Consider the following statements:

The policeman's beard is half-formed

Evil is beautiful.

Murder is nurturing.

$2+2 = 5$

These statements were created with Microsoft Word and they include gibberish, outrageous lies, and factual errors. Bill Gates can be considered the creative force behind Microsoft Word. Are we to accuse Bill Gates of the aesthetic, intellectual, and moral sins that I have perpetrated with his word processor? Is he an accessory to those sins? Of course not. The software allows me, the user, to do anything I want—so if I exercise my own free will, is there anybody to blame other than me?



But I have no desire to create a soulless storyworld in which any form of depravity is treated as morally neutral.

You don't have to. If the player chooses to murder his sidekick, make sure the sidekick is later necessary to get the player out of a fatal trap. If the player behaves boorishly to others, let them ostracize that player. Your design does not have to be morally or aesthetically valueless; it need only *address* all reasonable player options, not *reward* them. Indeed, the best way to condemn depravity is to let the player experience its consequences.



Must I waste my time devising responses to all manner of crazy player activities?

The key concept here is the differentiation between three very different classes of player behavior: *idiotic*, *reasonable*, and *preferred* (from your point of view). I have no problem with obviating idiotic behavior. It's better to refuse to give the player the option to jump off a cliff because an insignificant number of players will exercise that option; infringing on the players' free will by denying them that option does not comprise a significant intrusion into their free will.

The point I make here concerns the difference between your *preferred* options and *reasonable* options. You should not impose your *preferences* on the player; permit them all reasonable options and then impose the consequences of their choices. Remember the black-and-white image in Figure 13.1 in Chapter 13, "Bounded Numbers"? The white sections of the image are only revealed by the black sections. You can't show the player what is true without also showing them what is false. Don't confine the player to the correct behavior; let them learn the preferred behavior by experiencing the consequences of bad behavior.



What's the difference between this and the "Kill 'em if they stray" technique you so deplore?

Killing the player is a Boolean overreaction. Yes, you want to show the player that some decisions lead to undesirable consequences, but must death be the only undesirable consequence? Isn't it better to show a more realistic undesirable consequence? If the player insults his lover's best friend, should this not lead to an unpleasant confrontation with the lover? Should not adultery lead to a series of possible consequences whose seriousness depends upon the circumstances? Must you paint your picture in black and white? Is there no room for shades of gray?



But then the player will have an unsatisfying experience with my design, which will reflect badly on me.

No, it does not. True, there are immature people who try to blame others for their own errors. But in an interactive experience, the player can always go back, do it differently, and enjoy the improved results—thereby demonstrating beyond any doubt exactly where the original error lay.

Scoring

Fate has another responsibility: It provides feedback on the player's performance as a whole. In its most primitive form, that feedback is nothing more than the declaration that the player won or lost. It is better, however, to communicate the degree of success or failure.



What? You want to turn storyworlds into games?

No. “Game” is an overly broad term that most people now interpret as “video game,” with strong connotations of violence, simplicity, and juvenility. I am not proposing to transform *King Lear* into *Doom* or *The Odyssey* into *Pac-Man*. I am proposing, instead, that player performance be evaluated in dramatic terms rather than game terms. An audience does not reward an excellent dramatic performance by awarding points; it signals its assessment through applause. Thus, the final evaluation of the player's performance should be applause commensurate with the player's dramatic success. The same thing could be accomplished by different musical accompaniments to the end of the story. A well-played storyworld could end with swelling tragic or triumphant music, while a poorly played storyworld could take as coda a bit of elevator music.

In the world of storytelling, the most basic division is between comedies and tragedies. Comedies have happy endings; tragedies have unhappy endings. All video games function as comedies; the player is expected to triumph over adversity. A tragic video game, in which the player is expected to die heroically, is unheard of. This is one of the biggest differences between storyworlds and games: While games are always comedies, storyworlds can just as well be tragedies—but only if applause is the final evaluation, not a score.

Many of our greatest stories are tragedies. We shed a tear for the hero who dies in the end, but we know that it just wouldn't be right if he lived to a ripe old age and died after tripping over a stool. Shakespeare didn't have to kill Hamlet to resolve the core conflict of the story; killing off Claudius would have provided satisfying closure and a happy ending. He didn't have to kill Juliet, either. With Romeo out of the picture, the romance was over and the story could have been wrapped up then and there. But, no, he had to finish her off. This brings us to an ugly little problem: How do we reflect this in a scorekeeping algorithm? Imagine a Romeo and Juliet storyworld, complete with the Montagues and Capulets and their conflicts and intrigues. Imagine that it follows something like Shakespeare's version, and Romeo does himself in upon discovering Juliet's comatose body. The player, as Juliet, wakes up to find Romeo's bloody corpse at her feet. Should the player kill herself or shrug her shoulders and walk away?

Here, drama diverges from game playing. The traditional gamer would certainly not commit suicide; she'd walk away. But drama places a higher value on Juliet's death than on her life. The scoring algorithm must accordingly reward Juliet for suicide and punish her for selfish survival.



This will never work! Nobody's going to commit virtual suicide!

You only think nobody will choose a tragic option because you think of interactive storytelling as an evolutionary advance from video games. If you regard interactive storytelling as a completely new medium, then you will not be shackled by the constraints of the older medium. Remember, cinema did not really begin to develop until moviemakers abandoned all pretense that cinema was an evolutionary step beyond theater. Once they perceived cinema as a new medium with its own aesthetic system, they were able to create movies that successfully entertained audiences.

Feedback

The value of applause as a form of feedback lies in the fact that it shifts the player's self-perception as the protagonist. Game designers frequently talk about the importance of immersion, but they mean it in the narrow sense of providing a perceptually compelling experience. Immersion in interactive storytelling reaches

much deeper: Players must immerse themselves in the role of the protagonist. In the case of Juliet discovering the death of Romeo, genuine dramatic immersion requires the player to feel the emotional logic that drives Juliet to suicide. She must appreciate the dramatic forces that require her death. This is immersion at its deepest level. A player in the role of Juliet in an interactive storyworld should come to precisely the same decision. The player doesn't have to come to that conclusion, of course. The author can only reward, not control.

That reward is provided with applause. The storyworld should end with curtains coming down, the words "The End" appearing on the screen, and an auditory virtual audience reaction. If the curtains descend upon a dead Juliet, the virtual audience should erupt in thunderous applause; if she saunters away, boos and hisses should be her reward. Thus, the player's true role is to perform for a virtual audience that is indistinguishable from Fate. Fate evaluates the player's performance with a scoring algorithm and expresses that score with varying degrees of applause.

Lesson 30

In storyworlds, the reward is applause, not victory.

Algorithms for Scoring

How do you calculate a score to use for applause? The simplest way is to confine the story to a defined number of endings and assign a score to each ending. I expect this will be the primary scheme used in these early days of interactive storytelling. However, such schemes do not take into account the player's behavior throughout the story. The player who acts the clod throughout the story, for example, then somehow reaches Ending #4 will get exactly the same score as the player who accomplishes the same ending by more elegant behavior.

Perhaps the storyworlds will be so well designed as to obviate such undesirable outcomes. Nevertheless, I will show you how to develop more perceptive scoring algorithms. Overview variables provide one handy solution. Have Fate keep a running total of some overview variables that you deem important. A simple example would be to keep track of the overall good will in the storyworld—the sum total of all the pNice_Nasty values in the entire cast or in a selected group of characters. The player's score would then be the increase in that overview variable. This would make dramatic success dependent upon making everybody happy.



OK, smarty pants, how do you calculate a score for Juliet committing suicide?

That's easy; here's the formula:

```
Score = Blend(1, pNice_Nasty[Romeo for Juliet], pNice_Nasty[Juliet for Romeo])
IF (((Juliet dies) AND (Romeo lives)) OR ((Juliet lives) AND (Romeo dies)))
    Score = -Score
```

This formula starts with a blend that pushes Juliet's love for Romeo toward 1 in proportion to his love for her. If they both love each other intensely, then the result of the first formula will be just below 1. But that score is reversed if one of them lives while the other dies. If they both live or both die, then the player gets a big score. If one survives the other, then the player gets a negative score.

This is a simple scoring system. You can build scoring systems to reflect any set of dramatic considerations; complexity is not a problem with computers. The only constraint is your discomfort with algorithms. Ultimately, a scoring system allows you to make your statement with greater precision than a simple Boolean, win-or-lose approach.

Lesson 31

Use scoring systems to reward players rather than using mandates and prohibitions that constrain them.

Conclusion

Fate is your proxy in the storyworld; it implements your highest-level ideas about the storyworld. I assign Fate the first position in my cast of characters—because Fate is just as much a *dramatis persona* as any other. Indeed, Fate is the most important member of the cast. In designing Fate for your storyworld, you will need to implement your greatest algorithmic skills. Someday, when interactive storytelling is a well-established medium, the ability to design dramatically sensitive Fates will be the measure of artistic achievement.

CHAPTER 16

Verbs and Events

ONE OF THE IMPLICATIONS of applying verb-based thinking to interactive storytelling is that the central data structure should be the *verb*. The answer to the question “What does the user *do*?” is “The user does verbs.” It is only natural that the definition of the verb data structure is the central problem in all software design.

Verbs are expressed through mouse and keyboard actions. In a word processor, “double-click on a word” means “select this word.” In a web browser, “click on blue text” means “jump to this linked page.” In all applications, clicking an icon means “do whatever this symbol represents.”

The language of graphical user interfaces (GUI) has evolved into a system of great complexity. Consider all the GUI devices and schemes that we take for granted: menus, icons, dialog boxes, toolbars, tooltips, scrollbars. Your screen is littered with a menagerie of little symbols that do all sorts of useful things; and you’ve memorized all their meanings. What’s important is that each of these little objects *does* something; it’s a verb.

Verb Counts

In most software, verb counts have always been low; but over time, verb counts have inexorably risen. MacWrite, the first true consumer WYSIWYG word processor, did not have a great many verbs. Aside from the obvious verbs for text entry, there were verbs for setting fonts, sizes, and styles; verbs for justifying text and indenting paragraphs; verbs for inserting images; and verbs for setting and clearing tabs. I'd guess that MacWrite had fewer than a hundred verbs. But nowadays, Microsoft Word has so many verbs that students assigned to the task of counting them lose count somewhere above 300. With style sheets, footnotes, index entries, tables of contents, and so on, there are just too many verbs to track.

The same goes for games. The verb counts of games of the early 1980s were generally lower than ten. Perhaps this was due to the simple nature of the joysticks used for input—the only possible inputs were to move left, right, up, or down and to fire. That made five verbs. Combining the fire button with the joystick values gave nine verbs; the theoretical maximum verb count with a joystick alone was nineteen verbs. Since then, games have grown more complex and verb counts have grown. In general, however, the verb counts of today's games seldom exceed 30 and usually average about 15.

No matter how we design it, interactive storytelling requires hundreds of verbs. Consider the verb count in this bit of fiction:

Alas, poor Yorick! I knew him, Horatio: A fellow of infinite jest, of most excellent fancy; he hath borne me upon his back a thousand times. And now how abhorred in my imagination it is! My gorge rises at it. Here hung those lips that I have kiss'd I know not how oft. Where be your gibes now, your gambols, your songs, your flashes of merriment that were wont to set the table on a roar? Not one now to mock your own grinning—quite chap-fall'n? Now get you to my lady's chamber, and tell her, let her paint an inch thick, to this favor she must come; make her laugh at that.

In a single short sequence, we have 14 unique verbs, ignoring tense and number differences. It constitutes less than half a percent of the entire play; clearly the play as a whole boasts hundreds of different verbs. Our interactive storytelling engines must be capable of handling such large verb counts.

A children's book, Disney's *The Little Mermaid*, provides an even more illuminating example. The book is all of 26 pages long, consists mostly of pictures, and has text in a large, easy-to-read font. I went through the entire story and counted 121 unique verbs—this in a story for 5-year-olds!

Lesson 32

Storyworlds need an absolute minimum of a hundred verbs.

Generalized Verb Handling

With low verb counts, designers can craft each verb individually, writing custom code for each. This worked fine when you could count your verbs on your fingers, but once we started needing our toes, things became more difficult. A program such as Microsoft Word is an intricate maze of code, requiring the efforts of hundreds of programmers, and its complexity causes it to break with depressing regularity. The difficulty of managing hundreds of custom-coded verbs has restrained designers of interactive storytelling software. But we in interactive storytelling cannot evade the monster; our work requires us to manage hundreds of verbs. We are forced to dispense with the notion of custom-designing each and every verb. We need a more generalized approach, something more abstract that allows us to create, design, and program hundreds of verbs.

The first step in creating a generalized system of verb handling is to design a data structure for verbs. We need a general-purpose table of properties that define each verb. Following are some of the basic variables that you could put into your verb data structure.

- ▶ **Name:** A simple text label for the verb.
- ▶ **Momentous_Trivial:** A number indicating how newsworthy this verb is. This may not seem that valuable, but my experience has taught me that this value is vital to many calculations. “Take out the garbage” takes a negative value, while “Murder” is almost as close to maximum as you can get.

- ▶ **Audience:** This one can be very messy. Every verb has certain requirements for the presence or absence of other characters. Most of the time, the DirObject must be collocated with the Subject; after all, you can't kiss or punch somebody who's off on another stage. There are (of course) a few exceptions. For example, should the Subject be hatching some dark plan aimed at the DirObject, then the Subject most definitely wants to carry this out without the presence of the DirObject. MixPoison, DigGrave, or PrepareTrap are three verbs that require the absence of the DirObject.



Poison, Grave, and Trap are all the direct objects of their respective verbs.

That's right. But they're *not* the DirObjects of their respective verbs. I define DirObject to be the character that is the intended target of the verb. Besides, what else can you mix other than poison if you intend to harm someone? These elements don't function as direct objects because there's nothing else you could reasonably substitute for them. I'll go into this in more detail shortly in the "Sentence Structure" section.

Some verbs require the privacy of the Subject and DirObject and therefore the absence of other characters. Romantic verbs are often like this. A few verbs require the presence of a third party, either as a witness or as party to the action. And, of course, there are also specifications for whether other characters present can witness a verb being executed. I've tried all sorts of arrangements and, after much hand-wringing, I've come to the conclusion that the best approach is to wildly overdo it. The best way to specify required audiences is to maintain an array of two Boolean flags. The first, if true, requires that the character in question must be present for the verb to be executed; the second, if true, requires that the character in question must *not* be present for the verb to be executed. If both values are false, then the verb doesn't care a fig for the presence or absence of the character in question.

- ▶ **Witnesses:** This specifies who gets to know about the event. In 99 percent of cases, the witnesses consist of everybody on the stage where the event takes place. But that other 1 percent of cases can drive you crazy. For example, what about an event that is so important that every character should be

aware of it? When and how, for example, did you learn about the 9/11 attacks? Here's another weird case: The Subject does something secret that he conceals from others onstage. The assassin quietly unsheathes his knife. The partygoer sneakily dumps his bad-tasting drink into a potted plant. The embarrassed boy tries to zip up his pants. There can also be situations in which two people speak confidentially. They lower their voices and speak softly so that nobody else can hear. Yes, these cases are rare enough that you can probably get away without them, but the day will surely come when you really do need them.

- ▶ `Execution_Time`: This is useful only if you want to have verbs that take a while to carry out—for example, `Mow_The_Lawn`, `Make_Love`, `Bake_Cake`—and it's dramatically significant that they take a while to carry out. In the great majority of cases, your verbs will require just one tick of the clock to execute.
- ▶ `Preparation_Time`: This is used even more rarely than the `Execution_Time` datum. It is useful only if it is important that a character not be able to execute a verb immediately, which isn't often. Take a tip from me: Don't bother with this until you're ready to build your interactive *War and Peace*.

I have even more abstruse variables that you can attach to a verb, but this set gives you a good idea of the kind of thing you need to make verbs work. But I have not yet addressed the most important part of every verb: the facilities by which characters react to events using the verb. That, and the execution of the consequences of a verb, will not be addressed until we get to Chapter 19, "Engines."

Sentence Structure

A verb is never executed all by itself; it is necessarily part of a sentence. Obviously, we have to design our system with some standard definition of sentences, but different verbs require different sentence structures. "Joe insulted Mary" requires just three words: a Subject, a Verb, and a DirObject. But "Joe gave Mary the hamburger" adds a fourth word, an indirect object. What about "Joe traded Mary the hamburger for the book"? Or "Joe promised to give Mary a hamburger on Friday if she would give him the book on Thursday"? As you can see, sentence structures can become very complicated.

My solution to this problem is to endow each verb with a set of *WordSockets*, each of which can hold a specified type of word. Most *WordSockets* for most verbs will be empty. **Table 16.1** shows *WordSockets* for the sample sentences on the previous page.

TABLE 16.1 *WordSocket* examples

Verb	Insult	Give	Trade	Promise
WordSocket 1	DirObject	DirObject	DirObject	DirObject
WordSocket 2		Prop	Prop given	Subordinate verb
WordSocket 3			Prop received	Prop given
WordSocket 4				Time to give
WordSocket 5				Prop received
WordSocket 6				Time to receive



Where are the WordSockets for the subject and the verb?

The *WordSockets* for the subject and verb are built into the *WordSocket* set for every verb. When you create a verb, the system automatically creates those first two *WordSockets*. You can then add other *WordSockets* as required.

Plans and Events

Sentences are used in two ways in a storytelling engine. When the engine is figuring out what a character is going to do next, the engine assembles a sentence into what I call a *Plan*. In other words, a *Plan* is a sentence that has not yet been executed—but is intended to be. When a *Plan* is executed, it becomes an *Event*. An *Event* is just a *Plan* that actually happens. You shouldn't assume that every *Plan* automatically becomes an *Event*.

It's also useful to tack some additional information onto an *Event*, such as where and when the *Event* took place, and which characters witnessed it. This information can be useful for later calculations.

HistoryBooks

A *HistoryBook* is a record of all the Events that have taken place so far in a story. Every time an Event takes place, it's stored in the *HistoryBook*. Here's an example of why we need *HistoryBooks*:

Slim Jim and Tex are at it again, squaring off in the bar. Slim Jim punches Tex. Tex has three options: back down, punch back, or pull out his gun and start shooting. He determines that he's too proud to back down and too prudent to start shooting, so he punches Slim Jim.

Slim Jim now consults his options: They are the same as Tex's. He too refuses to back down but balks at starting a gunfight, so he punches Tex.

Tex repeats. Slim Jim repeats. Tex repeats. Slim Jim repeats.

This is a computer program, and computers can do this kind of thing forever. In the world of drama, this cycle is quickly broken. How can we design the software so as to break up cycles such as this?

Moods

One easy way to solve the problem of an endless loop is to use moods. If we include a mood such as *Anger_Fear*, then each punch increases the level of *Anger_Fear* in the punchee. In turn, *Anger_Fear* serves to increase the inclination to take stronger action. Thus, as the fistfight progresses, one of the combatants will eventually become so angry that he'll grab his gun. This certainly comports with our intuitive expectations.

HistoryBooks and Loops

Some kinds of looping behavior cannot be terminated by recourse to moods. Suppose the situation between Slim Jim and Tex is replaced with one in which two characters compliment each other. It is entirely possible to inadvertently set up a situation in which the compliments run on forever. Moods wouldn't help here; a compliment doesn't alter a character's moods so strongly as to change the character's behavior.

The solution is to refer to past behavior when determining future behavior. In this case, the recipient of a compliment would consult the *HistoryBook* of past Events

to check whether they have already rendered a compliment. If so, then the character refrains from repeating themselves and chooses an alternate course of action.

HistoryBooks can also be used to trigger irritation at repeated behavior. Suppose, for example, that Dick invites Jane out on a date and Jane refuses. The next day, Dick repeats his invitation and Jane repeats her refusal. At some point in this cycle, Jane should become irritated at Dick and respond with a more forceful rejection. Jane can only do this if she can consult her memory (the HistoryBook) to count how many times Dick has already asked her out on a date.

Another benefit of HistoryBooks is their ability to store enabling or disabling information. For example, suppose our hero seeks to rescue the princess by finding the error in the corporate books. If he has taken his accounting course, he'll be able to locate the error; otherwise, the princess is doomed. The traditional way to handle this problem is to have a global flag indicating that the hero has in fact taken his accounting course. With a HistoryBook, this is unnecessary; the software need only search the HistoryBook for the Event in question.



This is incomparably more complicated than simply storing a flag!

True, but such a flag requires custom code, whereas the use of the HistoryBook is a general solution. The HistoryBook can record whether he took a pottery course, a thermodynamics course, or a Chinese language course. It can record whether he cut down the oak tree, obtained the silver key, or destroyed the dragon's diapers. A HistoryBook records everything that has happened and can keep track of everything that matters. Instead of setting up a long list of special-case conditions, the storytelling engine need only search the HistoryBook. Sure, it costs more machine cycles, but it saves brain cycles—which is more valuable?

Backstory

HistoryBooks also permit a neat trick: the inclusion of backstory in a storyworld. Most attempts at interactive storytelling avoid reliance on backstory because it's so clumsy to implement. The most common technique is to insert it as a character's idle recollection at some otherwise quiet moment: "As Elizabeth walked through the meadow, the dew reminded her of the time she nearly drowned in

the river.” The author can set up the HistoryBook with Events that take place before the initiation of the storyworld; these Events can then be recalled at the player’s discretion.

Extra Information

The simplest addition to a HistoryBook is causal information establishing the sequence of Events in logical form and in temporal form. This is easily accomplished in the retrograde direction; each Event records the index number of its proximately causal Event. Forward causation is made more difficult to record by the branching nature of causation. Every Event in drama has but one proximate cause, but can have several consequent events. It is technically possible to keep track of the forward sequence of causation using the proper data structures. However, in practice such design contortions are unnecessary; simple searches of the HistoryBook can run quickly enough, given the small size of HistoryBooks and the high speed of current processors.

Gossip

Drama seldom relies solely on the direct experience of the characters; often, crucial dramatic information is communicated indirectly through the statements of intermediary characters. Sometimes this is done to heighten the dramatic impact (“But Luke, I *am* your father!”) or to permit the recipient of the information the opportunity to react to that Event *ex post facto*. Any adequate interactive storytelling system must provide for the transmission of information among characters. I refer to such information transmission systems as *gossip* systems. This stretches the conventional definition of gossip to include all information related by one character to another, not just the “dirt.” Such systems are inescapably complicated. I’ll first outline the procedure used in the gossip system of my own interactive storytelling technology; then I’ll discuss some of the finer points and variations possible on the basic structure.

Recording the Event

When an Event takes place, it is stored in the HistoryBook. A limited number of characters witness the Event—usually those who populate the stage on which the Event takes place. This is reflected in an array attached to the Event data structure, labeled `I_Know`. Whenever a character witnesses an Event, the `I_Know` flag for that

character is set to true. But that's not all—the character also keeps a list of the most important Events, known as the *headline list*. There's no reason why we couldn't make the headline list for each character long enough to include every single Event to which the character is privy, but in practice such thoroughness would be wasteful. Ten or twenty Events are all that most people need as grist for gossip.

Each Event in the headline list is prioritized according to its `Momentous_Trivial` value (the value assigned to each verb specifying the degree of dramatic intensity arising from the execution of the verb). The engine can then sort the verbs in the headline list by their `Momentous_Trivial` values. If a new Event has a `Momentous_Trivial` value greater than that of the lowest Event in the headline list, it replaces that Event.

Communicating Gossip

A headline is not a story—it's the peak Event of the story, the most salient Event in the story. For example, consider this fragment of a story:

Tom enters the bar.

Tom orders a sarsaparilla from the barkeep.

John makes fun of Tom.

Tom ignores John.

John confronts Tom.

Tom shoves John.

John punches Tom.

Tom hits John over the head with a whiskey bottle.

John shoots Tom.

Dick draws a gun and jumps behind the bar.

Jane screams for help.

Tom dies.

John laughs over Tom's body.

Dick shoots at John.

Jane screams for help again.

John shoots back at Dick.

Dick ducks under cover.

Jane cries over Tom's body.

John shoots at Dick again.

Dick shoots John.

Jane cries over Tom's body.

John dies.

Here's the problem: Suppose that Jane, who witnessed the entire affair, later runs into Mary and wants to tell her about these shocking events. Jane will start off by telling Mary that John shot Tom. That information, by itself, is unlikely to satisfy Mary; she'll want to know more. How can the engine assemble the entire tale from the HistoryBook?

This problem is particularly messy. I have tried many approaches and never come up with a satisfactory solution. The best I can suggest is that the matter be resolved by an interrogation. Upon hearing the news that John shot Tom, Mary has two options: She can ask, "Why did John shoot Tom?" or "What happened next?" Jane can readily answer both questions by looking up the Event in the HistoryBook and finding its antecedent or consequent (that's why we save the causal information for each Event). It's clumsy, but it's the best overall solution I have found.

Initiating Gossip

Storyworlds offer several ways to initiate gossip. It could be an autonomous process, driven by the engine. If a character has nothing better to do, the engine searches for a likely gossip-mate. It could be prioritized according to the value of the gossip; a character that has just come across a particularly juicy item of gossip sets aside low-priority plans and scurries off looking for somebody to regale. It could also be a consciously chosen course of action: A character must actively declare the intent to gossip in order to do so.

Why People Gossip

Before we can build a gossip system, we need to come up with a model of why people gossip. We could get into a lot of psychology here, but remember, this is drama, not real life. Our goal is not to model reality but to come up with a dramatically useful model of gossip.

The primary purpose of a gossip system in interactive storytelling is to move information through the cast of characters. Information is only interesting so long as some people know it and other people don't. Once gossip is known by all, its dramatic value evaporates.

A character is motivated to share gossip by its value to the listener. If I tell you a choice bit of gossip that interests you, your attitude toward me will improve. You will trust me more, because I have taken you into my confidence by revealing the gossip. You will feel more affection toward me for providing you with this small boon. And you will likely heighten your estimate of my overall power, because you will believe that I have access to all manner of useful information. From my point of view, sharing gossip is advantageous.

But characters must share their information parsimoniously. Giving information to a blabbermouth guarantees that the information will quickly spread through the cast, robbing the original source of the value of presenting it directly to each character. The ideal situation from the point of view of the original source is to first inform the least loquacious characters, and then spread the news further. Unless the gossiper's desire is to have the news quickly spread through the cast, the blabbermouths should be the last ones to know; that way they can't steal the originating character's thunder.

A secondary consideration is the degree of affection that the originating character holds for the listener. Affection increases the originator's proclivity to talk.

The Mechanism

The gossip process begins when a non-player character (NPC) enters a stage. The entering character considers, in turn, each of the characters on the stage, evaluating the desirability of gossiping with that character. This decision turns on such factors as the trust and affection the entering character has for the character in question. The loquaciousness of the character under consideration is also a factor, as cited.

Having decided whom to gossip with and whether to gossip with that character, the entering NPC must now determine what information to share. Obviously, the NPC wants to start with the most momentous headline, but everybody has a different evaluation of the significance of a piece of gossip. A critical factor is the closeness of the NPC to the Subject and DirObject of the event. In other words, learning about the murder of some nameless person on the other side of the planet probably won't evoke much interest from you, but the murder of your wife by your brother would be intensely important to you. Hence, the personal importance of any event is the product of the verb's `Momentous_Trivial` of that event multiplied by the magnitude of the relationship values the NPC feels for the Subject and the DirObject.

A few other considerations go into the process of deciding which piece of gossip to talk about. One is whether the listener already knows the information. This provides a perfect example of the difference between physical reality and dramatic reality. In physical reality, you don't know what events another person knows, which leads to all those boring and pointless, "Did you hear that..." and "Oh, you did? Never mind," sequences. We don't need this kind of clutter in our storyworlds—and we don't have to put up with it, because the computer can look inside the minds of the characters and establish what they already know. Therefore, our gossip software should automatically eliminate from consideration any gossip already known to the listener.

After all these calculations, the actual consummation of gossip is disappointingly simple: The engine sets the `I_Know` flag for the Event to be `TRUE` for the listener. The listener can then react to that Event (*ex post facto*) and enter it into their headline list.

Lies

In terms of drama, the yin of information demands the yang of disinformation. Where would storytelling be without the protagonist being misled by some dastardly lie? Any gossip system must include provisions for lying.

In terms of the data structures I have presented, a lie is nothing more than an Event that never happened. It contains all the same data: Subject, Verb, DirObject, and so forth. But one tiny addition must be made to the Event data

structure to accommodate lying: A Boolean flag indicating whether the Event is TRUE or FALSE. Actual Events will have this flag set to TRUE, meaning that the Event is real and true, while lies will have this flag set to FALSE.

The creation of a lie is difficult to figure out but easy to implement. First, the liar has to make the decision to lie. Several motivations might apply.

Intent to Harm Relationships

The simplest motivation for lying is to wreak some injury upon the victim with a besmirching lie. In this case, the first task is to determine whether the goal is to damage the victim's relationship with a particular character or to damage the victim's overall reputation with the group. In each case, the liar selects the most salient relationship to attack. The selection of that relationship depends on how highly the victim is thought of by others (with respect to that particular trait).

The next step is more difficult. The liar must identify the Verb that would trigger the strongest negative reaction among listeners. For example, if the liar wishes to sully his or her victim's perceived Nice_Nasty value, then a Verb such as Drown_Kittens or Laugh_At_Funeral would likely be best. Conversely, should the liar want to attack the trust that people hold for the victim, then a Verb such as Cheat_Grandma or even Tell_Lie would be more appropriate.

Having decided upon the Verb, the only remaining problem is to select an appropriate DirObject. The danger here is that the DirObject might know that the Event never happened, so the liar must select a DirObject who is remote from the group or is not trusted. The simplest solution is for the liar to use himself or herself as the DirObject—if this accomplishes the relationship alteration desired. For example, a female liar could easily besmirch a male victim in the eyes of that victim's girlfriend by claiming that the male victim slept with her. However, there are cases in which the liar cannot use himself or herself as the DirObject. For example, if the liar wishes to impugn the benevolence of the victim, but everybody knows that the victim hates the liar, then circulating the lie that the victim insulted the liar will accomplish little.

Once the DirObject has been selected, the liar needs to create the falsehood as an Event with a Subject (the victim of the lie), Verb, and DirObject. The liar then inserts the lie into the HistoryBook, which sets the Truth Boolean flag to FALSE and the I_Know flag for that character to true. At this point, the liar can simply

let the normal gossip mechanism described above take over, trusting it to spread the word to the appropriate people as opportunity permits.

Gain Benefit Through Misdirection

The more common motivation for a lie is to misdirect its victim in some manner that causes beneficial events for the liar. Drama usually requires that the lie induce its victim into some undesirable situation. Sad to say, such lies require complex inference engines. (An inference engine is a technology from artificial intelligence that is capable of drawing logical conclusions from a collection of data.) Such technology exists, but integrating an inference engine into a story-telling engine is not a first-generation problem. Indeed, the entire issue of gossip might be fairly characterized as second-generation technology.

Defensive Denial

A liar might want to deflect suspicion from himself for some improper action by explicitly denying it. This situation is easy enough to calculate: The liar considers the likely reactions of others upon learning about the improper action. If there's little cost to doing so, then it's best to admit the action; otherwise, make like a politician and lie, lie, lie!

Secrets

Another form of malinformation is the revelation of a secret. Jane tells Mary, "Don't tell anybody, but I'm pregnant by Fred!" Mary then tells Adrienne, who tells Marcy, who tells somebody else, and pretty soon everybody knows the news—much to Jane's chagrin. The telling, retention, and revelation of secrets is a crucial element of drama, yet managing the mechanics of such behavior is complex.

I attach to each Event a Boolean flag labeled, amazingly enough, Secret, which indicates whether the Event is expected to be kept secret. The secret is "owned" by the Subject of the Event, and a character's respect for the secret is determined by the character's relationship with the Subject.

Here we have another interesting example of the distinction between dramatic reality and physical reality. If Fred and Jane have sex and Jane gets pregnant, the physical reality might be reported as "Fred impregnates Jane," because we normally attach the active role to the male. But if the story context makes Jane's pregnancy the important factor, then the event should be recorded as "Jane

becomes pregnant by Fred.” Jane is, in terms of the drama, the more important character in the event and therefore should be treated as the Subject. Moreover, this way, she owns the secret.

At first, only a few characters will know any Event worthy of secrecy—perhaps just the Subject and DirObject. What causes the secret to spread?

Realistically speaking, most people would keep such secrets to themselves. But this is drama, not reality, so it’s important that the owner of the secret divulge it to somebody else. That’s how we get the dramatic pot bubbling. Perhaps nothing will come of it—but it could just as well lead to all sorts of dramatic mayhem. What fun!

The dramatic effervescence arises from a kind of “trust gradient.” Jane may trust Mary only, but Mary trusts Adrienne, so Mary tells Adrienne, who trusts and tells Ann, and so forth. In each case, the bean-spiller advises the recipient to keep it a secret, and in each case the recipient divulges the secret to a trusted friend to gain the benefits of sharing gossip without risking the sanctions of being exposed.

Of course, eventually everybody knows, and when the secret’s owner discovers this, there’s hell to pay. How can the secret’s owner respond to the betrayal?

The Grapevine

What prevents people from telling every lie and divulging every secret? Answer: the likelihood of unpleasant consequences if discovered. The key issue here is discovery; people lie and blab secrets because they believe that they will not be caught. But sometimes they *are* caught. We therefore need a mechanism that permits such improper gossip to be discovered—but only sometimes. In other words, there must be a chance of getting caught, but it must not be so high as to make it overly risky. My solution to this is the *grapevine* (as in “I heard it through the grapevine”).

The basic idea is to keep track of who told what to whom. For each and every item of gossip, we have to maintain a record of every transmission of that item. If Larry concocts the lie that John slept with Margaret, then tells that lie to Dave, and Dave tells Fred, who in turn tells both Mary and Sam, and Mary tells Jane while Sam tells Agnes and Tabitha, and Agnes asks Margaret why she slept with John, then Margaret has to be able to deny the lie and demand that Agnes reveal her source for the lie. **Figure 16.1** illustrates the process.

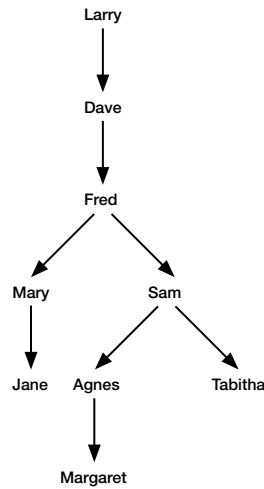


FIGURE 16.1 A lie wends its way through the cast

It isn't difficult to assemble the data structure required to keep track of this tree. Its value is that Margaret, upon learning of the lie told about her, can demand, "Who told you that?" And Agnes can then choose either to spill the beans about Sam or prevaricate, depending on her attachment to Sam relative to her attachment to Margaret. If she does reveal that Sam was her source, then Margaret can go to Sam and make the same demand. In this fashion, Margaret could trace the lie all the way back to its source. This is cumbersome, of course, but the existence of this possibility is the best deterrent to lying.



Such a system would quickly spawn megabytes of data that would strain the capacity of a computer. Most of that data is of little dramatic importance, so why burden the computer with all of it?

These days our computers sag with so much RAM that this shouldn't concern us, but there is an answer for the fastidious programmer. The trick is to keep track of the tree for every Event. Once every character has learned of the Event and planned his or her reaction to it, if any, the storytelling engine dismisses the information. If everybody has heard the news and nobody has anything left to do about it, there's no longer any point in keeping it.



Successfully tracking the malinformation back to its source is a lengthy process that is unlikely to be completed. Why bother?

For Margaret to nail Larry, she needs the cooperation of Tabitha, Sam, Fred, and Dave. Should even one of those people decide to protect Larry, Margaret will be stumped. But the real value of this situation lies in the element of suspicion. Suppose, for example, that she traces the lie as far back as Sam, who insists that he cannot recall who told him (or that he did not tell Tabitha). This provides additional grist for the dramatic mill. It creates tension between Sam and Tabitha (“Why did you tell Margaret that I told you about her? Now Margaret’s on my case!”). It reduces Margaret’s trust in Sam. Moreover, she can guess Sam’s likely informants based on her knowledge of his relationships, thereby getting a step or two closer to the source. Thus, Margaret is able to develop suspicions that she can then act on, and the plot thickens.

Despite these possibilities, the clinching argument for the grapevine is the need for some fair and workable deterrent to lying. Accountability is that deterrent.

Rare is the protagonist who does not stumble at some point in the story over some lack of information, some misinformation, or some unfortunately revealed secret. Information transfer among members of the cast is often an important component of narrative. On the other hand, implementing gossip is a messy business. Beginners should probably wait until they’ve built up some experience before tackling this important issue.

Conclusions

The most important lesson of this book is Crawford’s First Law: Always ask, What does the user *do*? What are the verbs? Organizing your verbs is the most important and most time-consuming task in building a storyworld.

A verb does not exist in isolation. It is part of a sentence, and you’ll need a standardized sentence structure that you can compute with. If you want to keep it simple, just stick with a basic four-word sentence: Subject, Verb, DirObject, and IndObject. Even then, IndObject will be empty most of the time. A character makes a Plan using the sentence structure, then the engine converts that Plan into an Event by executing it.

CHAPTER 17

Languages

FEW PEOPLE NOTICE LANGUAGE. It's so common and so frequently used that it escapes our notice. Language is (almost) the only means we have of communicating with each other. Recall my definition of interaction: A cyclic process between two or more active agents in which each agent alternately listens, thinks, and speaks. That describes a conversation, and interactions with a computer should aspire to be like conversations. Not just any old conversations, but interesting, entertaining conversations. This, of course, implies that computers should use language to communicate with humans. In fact, people already use language to talk to their computers. (That language, though, is a cryptic, mashed-down fragment of real language.) When you play a typical shoot-'em-up, you're using the controller to tell the computer "Forward. Left. More left! Fire! Duck! Right. Forward. Forward some more. Wait. Fire!" And so on and so forth. The computer is talking back to you by showing the scene.

When you use a word processor, you're using the keyboard to tell it what characters to include. You use the mouse to tell the computer where you want to edit the text and what special commands to execute. "Computer, highlight this portion of the text. Now put that text in italics. Increase its size to 14 points. Deselect that text now. Go to the end of the paragraph. Delete the last three words. Replace them with this text." And so on and so forth.

Using Devices to Talk

The devices we use to talk to the computer (controllers, keyboards, and mice) are alien to our instincts; few of us had even seen any of them until maybe 30 years ago. The languages themselves are weird contraptions designed by computer programmers, not linguists. Clicking the tiny red circle means one thing; clicking the tiny yellow circle means something else. A tiny picture of a paintbrush denotes formatting controls; tiny boxes with teensy-weensy lines inside them indicate how I can split the view of my document into several panes.

Seeing how bad the interface languages we use are, it's easy to understand why some people have problems appreciating that they really are languages. Just think of them as languages with lots of latex makeup, like the aliens in science fiction movies. Deep down inside, they really are languages, even if they look like a cross between a moose and a mouse.

The expressive power of our user interface languages has been slowly growing over the last few decades, but we are reaching the upper limits of what can be done with a controller, keyboard, and mouse. Fortunately, the experts who work at voice recognition have gotten pretty good, and it won't be too long before we're literally talking to our computers. If you have any doubt that, someday, computers will converse with humans in natural language, I can provide you with absolute, iron-clad proof of my prediction: That's how computers work in *Star Trek*.

We interactive storytellers can't wait for someday, because language is fundamental to storytelling. Can you imagine a story told without any language? There are a few: silent movies, rare movies such as *Koyaanisqatsi*, and a few oddball comics. But even silent movies frequently included captions, and the rarity of wordless movies and comics demonstrates that storytelling without language, while theoretically possible, requires exceptional amounts of both contrivance and creativity. So we absolutely, positively have to have some way to use language in our storytelling.

That's a problem, because as any computer scientist can tell you, computers cannot understand natural language. It's not a matter of not having enough memory or needing faster computers. The problem is more fundamental than that. To explain it, I must introduce you to two gentlemen from the early 20th century: Edward Sapir and Benjamin Whorf.

The Sapir-Whorf Hypothesis

Edward Sapir and Benjamin Whorf worked in linguistics. Their research into Native American languages demonstrated that language is closely related to the perception of reality of the speaker. The most common expression of this concept is the apocryphal assertion that Eskimos have a hundred different words for snow. While that particular assertion is false, it is nevertheless true that the Inuit language has many more ways of talking about frozen water than the language of Africa's San Bushmen. The different environments in which different cultures live spawn different kinds of languages.



The Sapir-Whorf hypothesis was refuted long ago.

Not quite. You see, Sapir and Whorf never actually declared any explicit version of their hypothesis. They never even called it a hypothesis. Instead, they wrote books and papers arguing different facets of the idea that language and our perception of reality are closely connected. This spawned a lot of confusion because people restated their ideas in varied ways with different words. For example, the strong version of the Sapir-Whorf hypothesis states that language controls our thinking—and that version has long since been rejected. But there's a weaker version of the Sapir-Whorf hypothesis, which states that language reflects our perceptual reality. This version of the hypothesis is generally accepted.

So language mirrors reality. That, in turn, suggests something scary: If we want to put language into the computer, we'll have to put reality into the computer first. Can you even imagine how you would define reality to a computer?

```
Class Reality {
    Planet Mercury, Venus, Earth, Mars,...
    Ice Cream Chocolate, Vanilla, Strawberry, Rocky Road...
    Cable USB, RS-232, FireWire, Centronics, Ethernet...
    Tree Elm, Cherry, Walnut, Oak, Eucalyptus, Sycamore...
    ...
}
```


Where would you even begin? How could you possibly specify in computer language everything you know about reality?

That's the problem: To get language working on a computer, we have to figure out how to encode reality into a computer—and nobody has any idea of how to do that. (A huge project called Cyc, which has been underway for many years, attempts to get the common-sense version of reality into a computer. It's been underway for decades, and so far the results are impressive but nowhere near complete.)

It appears, then, that interactive storytelling is impossible. The computer can't understand language, and we can't make interactive storytelling without language. What do we do now?

Faking Real Language

People have been coming up with all sorts of clever schemes to fake language on the computer—that is, to make it seem as if the computer really can understand language. One of the first such exercises was a program called Eliza written by Joseph Weizenbaum in the 1960s. Eliza used some simple tricks to make it appear that it was a psychiatrist who could carry on a conversation with the user. The following hilarious tale comes from Daniel Bobrow about an early implementation of the program in the 1960s. (For readers born after 1970, a PDP-1 was a kind of computer; a teletype was an early interface device consisting of a keyboard and a very slow printer; and LISP is an old programming language.)

One Saturday morning about 9 a.m., a Vice President of Bolt, Beranek, and Newman in charge of selling our Telcom commercial service arrived at our PDP-1 computer room hoping to be able to use the computer for a demonstration. He found a note on the PDP-1 teletype that read: "TYPE ALL MESSAGES INTO THIS TELETYPE AND END THEM WITH A PERIOD." This note was left there by one of the programming bums who had just succeeded in getting a LISP version of the DOCTOR program (based on ELIZA as developed by Weizenbaum) up and running. However, the VP thought that this note indicated that this teletype was linked with the one in my home, since I [Bobrow] often use the computer all night. Therefore, he started typing to "me," but in fact was typing to this version of the DOCTOR program. The following conversation ensued:

vp: Do you think that I can use the computer this morning?

eliza: Why do you ask?

vp: I might be able to make some additional Telcom sales.

eliza: Why aren't you so sure?

vp: My prospective customers have never seen a demonstration of the system.

eliza: Does this mean very much to you?

vp: Of course it does.

eliza: You seem very positive about that.

vp: This is ridiculous.

eliza: What do you mean, this is ridiculous?

vp: Is there any reason why I can't use the computer now?

eliza: Tell me more about your prospective customers.

vp: Please dial me up on 491-1850

After the last remark, note that the VP did not obey instructions and left out the period. Therefore, of course, the computer didn't answer him. This so infuriated the VP, who thought I was playing games with him, that he called me up, woke me from a deep sleep, and said:

vp: Why are you being so snotty to me?

bobrow: What do you mean I am being snotty to you?

The VP angrily read me the dialogue that "we" had been having, and couldn't get any response but laughter from me. It took a while to convince him it really was a computer.

Eliza was smart enough to fool a lot of people, but if you used it for more than five minutes you quickly saw through the algorithm. It had three simple rules:

- ▶ If you detect one of your keywords (such as “might” or “of course”), then respond with a standard response based on that keyword.
- ▶ Repeat back to the user a portion of what they said with a question mark.
- ▶ Provide a standard content-less response such as “In what way?” or “How does that make you feel?”

Eliza did not really do anything like language comprehension. The program used a set of clever tricks to give the impression of understanding language, but it couldn’t carry on for more than a few minutes before falling apart.

Adventure

In the 1970s, the program *Adventure* carried the idea of faking language on a computer a bit further. Its keywords were few and simple, so that players could type things such as “Go north” or “Search room” to make progress in the game. This ultimately led to what is now called interactive fiction (see Chapter 9). The fractured English used in interactive fiction is definitely much improved over the earliest versions, but even the best implementations of the idea today still quickly reveal their limitations.

Siri

We’ve had fairly good voice recognition built into our computers since the 1990s, but it wasn’t good enough, so it was little used. With Siri for the iPhone, however, Apple has made a determined push into the future.

Of course, the advertisements you see make Siri seem more powerful than it really is. I tried these questions and got Siri’s standard “I don’t understand that” response:

- ▶ Why are guys so clueless?
- ▶ Who was Erasmus of Rotterdam?
- ▶ What color is melatonin?
- ▶ What is wrong with my computer?

Siri can handle only language that is specific to the normal needs of iPhone users: locations of nearby places of business, weather, driving directions, and the like. Anything else (other than cute or funny responses) is beyond Siri. It can handle only a tiny fraction of what we call natural language.

Siri's limitations should not be taken to justify pessimism about the future of vocal interaction with the computer. Apple has solved one of the biggest obstacles in speech recognition: It packs much of Siri's intelligence on its own servers. When you talk to Siri, most of the serious computing is going on far away. That allows Siri to be trimmed down to a reasonable size on your iPhone.

The task now is to expand the scope of Siri's knowledge base. This can be done incrementally. You can be certain that Apple is hard at work adding more and more knowledge to the Siri database. They already have an ideal source of feedback on Siri's shortcomings: Whenever Siri can't answer a question, that question goes into a database of failed requests. By sifting through this database, Apple can identify the most important shortcomings to address. I therefore suggest that, if you have Siri, ask it, "Who was Erasmus of Rotterdam?" If enough people ask, Siri will get an answer!

Façade

Michael Mateas and Andrew Stern created the first genuine interactive storyworld; they are the Wright Brothers of interactive storytelling. Their storyworld, *Façade*, is a masterpiece of artificial intelligence combined with clever schemes to keep the language within workable bounds. The player is an old friend visiting Grace and Trip, a couple whose marriage is coming apart. In the course of the evening, the conflict between Grace and Trip comes to a boil, with four possible outcomes:

- ▶ The player is thrown out by Trip.
- ▶ Grace and Trip achieve an apparent reconciliation.
- ▶ The conflict explodes and one of them storms out, ending the marriage.
- ▶ Grace and Trip achieve a genuine reconciliation.

The player's interactions with the characters determine which of the four outcomes eventuates.

Linguistic Interpretation

The most salient innovation in *Façade* is a powerful parser that can interpret all player input (which is entered through the keyboard) and boil it down to one of these 36 basic verbs:

- | | | |
|----------------------|----------------------|----------------------|
| 1. agree | 13. refer to a topic | 25. intimate |
| 2. disagree | 14. praise | 26. judgment |
| 3. positive exclaim | 15. ally | 27. suggestion |
| 4. negative exclaim | 16. criticize light | 28. misc-custom |
| 5. express happy | 17. criticize harsh | 29. manipulation |
| 6. express laugh | 18. oppose | 30. joke test limits |
| 7. express sad | 19. flirt | 31. inappropriate |
| 8. express angry | 20. pacify | 32. hug |
| 9. maybe unsure | 21. provoke | 33. comfort |
| 10. don't understand | 22. greet | 34. kiss |
| 11. thank | 23. goodbye | 35. physically favor |
| 12. apologize | 24. get attention | 36. wander away |

The artificial intelligence is smart enough to recognize any player entry as a form of one of these verbs—although many entries will fall into the “don’t understand” category. What’s especially clever about this system is how it reduces the semantic volume of its words. Semantic volume describes how many different ideas are encapsulated in a single word. The word “get” has a large semantic volume, because it can be used to mean all kinds of different things. The word “bathykolpian” (look it up) occupies only a tiny bit of semantic volume, because it means just one specific thing.

The linguistic system in *Façade* carves up the semantic space into 36 separate regions, each of which contains one word. Of course, most of this semantic volume is occupied by the “don’t understand” verb, but all the remaining volume is allocated to the other verbs. It’s impossible to say anything that the Artificial

Intelligence (AI) can't categorize, which in turn means that, if the player's entry has any inkling of relevant meaning, the AI can detect it and act on it.

The linguistic AI is even smarter: It doesn't analyze each player entry in isolation, but in the dramatic context of the moment. A comment can be interpreted in different ways, depending on the dramatic context. Saying "I don't believe it" means one thing in the context that you've won the lottery and something entirely different in the context that a friend is angry with you.

Readers interested in exploring these possibilities should google "WordNet" for a good starting point.

Tight Constraints

One of the secret skills in both language comprehension and interactive storytelling is the ability to narrow down the situation and confine the circumstances to a specific context. *Façade* does this brilliantly in several ways. First is the staging: There is only a single stage, Grace and Trip's apartment, but there are several substages where individual interactions can take place. Without much room in which to move around, the characters aren't distracted by unnecessary elements.

Even more important is the tiny cast: Just three characters, one of whom is undefined. This permitted the authors to custom-code the personalities of Grace and Trip rather than write generic code to handle the reactions of different types of characters.

The temporal context is also tightly constrained to a single evening. This keeps things moving along and naturally obviates a great many dramatic possibilities.

This style of story design runs counter to the instincts of many techies, but is crucial to the success of *Façade*. If you can make the box small enough, you can fill it. If your box is too big, then you have to leave gaping empty spaces. Most artists are sensitive to the advantages of applying constraints to the artistic context they create.

Storytellers are especially aware of this necessity. If you want to put the heroine in peril, you don't do it in a busy shopping mall or crowded room. No, you want her all alone in an otherwise empty space, so that she has only her own resources to call upon. Otherwise, the reader will want to know why she doesn't just ask somebody for help. This kind of isolation shows up over and over in storytelling.

The boys are isolated on an island in *Lord of the Flies*. In *The Ox-Bow Incident*, the two groups of men encounter each other in a remote area, away from civilization. In *Star Wars*, the planet Tatooine is a sparsely populated desert region where people have greater freedom to misbehave because of the absence of civilized controls. Even the final confrontation between Luke and the Emperor in *The Return of the Jedi* is confined to a single room with just three people: Luke, Darth Vader, and the Emperor.

Lesson 33

Constrain your storyworld to as narrow a scope as possible; that's the only way you'll ever be able to include everything that it needs.

Generated Video with Facial Expressions

Video-game designers strive to create high-quality video showing people moving around in visually realistic ways. *Façade* used a video-generation system that was, for the time, not quite state-of-the-art but still impressive. What's important about *Façade*'s use of video is that it is generated by the software's assessment of the dramatic situation. The *Façade* characters display facial expressions appropriate to their feelings.

The human face hefts quite a load of emotional content. Movies rely heavily on facial close-ups. Even in the most action-intensive scenes, the face plays a major role. For example, watch the famous TIE Fighter Attack scene from *Star Wars Episode 4* on YouTube. It's the scene in which the Millennium Falcon, escaping from the Death Star, is attacked by Imperial TIE fighters. Han Solo and Luke Skywalker shoot down the TIE fighters in fast-paced combat. But watch the scene carefully: Roughly one-half of the video consists of faces—not of screaming TIE fighters and roaring turbolasers. You see shots of Luke and Han as well as Princess Leia and Chewbacca. You even see C3PO, R2D2, and an enemy fighter pilot. Even in action-packed scenes, the human face is crucial.

It is therefore imperative that any interactive storytelling system includes some means for showing the faces of the characters. *Façade* hit the nail on the head with this.

Canned Output

The biggest mistake in the design of *Façade* is the reliance on canned audio from the two NPCs. This had a number of undesirable effects. First, all the canned audio had to be recorded, which raised the development cost. It also gobbles up lots of memory: The *Façade* download weighs in at 160 MB, much of which is required for the audio. Most importantly, any use of canned material tends to inhibit the expansion of options available to the player. (Recall how in Chapter 11 the authors of the book *Interactive Storytelling for Video Games* advise against providing “too many options” to the player on the grounds that this inflates the development cost.) In Dr. Mateas’s next major effort, *Prom Week*, he reverted to text because it can be manipulated to fit a wide variety of situations. We already have software that converts text to speech, but it lacks emotional emphasis. “How dare you question my integrity?” will come out sounding like a question, not an exclamation. Computer scientists are working on the problem, and I’m sure we’ll have emotionally nuanced text-to-speech someday.

Limited Verb Set

Façade’s greatest strength, its smart parser, is also a source of weakness: The parser accomplishes its magic by boiling everything down to one of 36 verbs. This constrains the player in ways that are both desirable and undesirable. For example, a character can kiss another character, but no inflection is possible. The program can’t distinguish between a peck on the cheek, a gentle kiss on the lips, and a long, passionate kiss. The player can express four emotional states (happy, sad, laughing, or angry), but no others. It’s good that Mateas and Stern were able to boil so much of the complexity of human relationships down to these 36 basic acts, but it’s also a limitation.

Limited Extensibility

The most serious problem with *Façade*, in My Arrogant Opinion, is the difficulty anybody would face in taking its ideas up a notch and advancing the concepts to a new generation. The relationships between the inputs and the outputs were hard-coded; many of the connections in the system had to be explicitly and individually specified. Expanding the system would have entailed a workload expanding exponentially with the size of the verb set. As a springboard into the future, *Façade* isn’t very springy.

Ultimately, *Façade* is in some ways a vastly improved version of the Eliza program. It is, of course, orders of magnitude more sophisticated than Eliza, but the fundamental design is similar: Accept textual input from the user, find the best-fitting internal schema for that input, then send output appropriate to that schema to the user. *Façade* is better than Eliza in a hundred ways, not the least of which is the impressive cleverness the authors used to make *Façade* appear to understand more than it really does. But deep down inside, its lowest-level DNA is based on Eliza's.

Tinkertoy Text

In 1984, while waiting in a supermarket checkout line, my eye was assaulted by the *National Enquirer*. It occurred to me that the headlines were formulaic, and that single word—formulaic—inspired an idea. If the headlines were made using a formula, then I could put that formula into a computer and have it generate *National Enquirer* headlines. That in itself wasn't anything more than a joke. At the time, however, I was working on my game *Balance of Power*, and I realized that I could provide feedback about events to the player in the form of procedurally generated news stories.

The basic scheme is truly simple. **Figure 17.1** is a simplified representation of the concept.

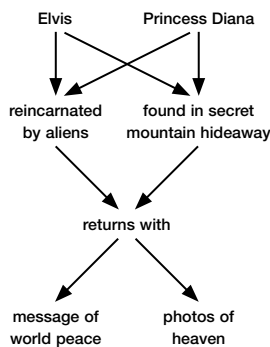


FIGURE 17.1 Conceptual representation of headline generator

Astute readers will recognize this as a variation on the branching tree with fold-back shown in Figure 8.5.



So now you're telling us that branching trees are OK to use?

Remember, this scheme is used only to present a single sentence, not an entire story. A branching tree, as I pointed out in Chapter 8, is capable of handling only a small number of variations—which is fine for a single sentence. A storyworld, on the other hand, needs a lot more variability than a single sentence needs. Finally, as you'll see later, I've developed considerable improvements that greatly enhance the power of the headline generator system.

The first such improvement is to expand the breadth of the tree by concocting additional options at each layer of the tree. The software can then randomly select from those options. We can do better than randomness by inserting what I call an *intensifier*—a phrase that indicates the magnitude of the event. Here's an example taken from *Balance of Power*, way back in 1984:

```
I*Two dissidents*Top critics of the government*Groups of
demonstrators*Thousands of government opponents*
I* escorted to meeting with &* harassed* thrown in jail* executed*
R* by police* by government agents* by / soldiers* by security men.
```

The first two lines of code start with the letter I—this indicates that they are intensifiers and the computer should use a previously specified variable to choose from the four phrases. Each of these intensifier lines uses a different variable for its intensification. The last line of code begins with R, which indicates that the selection should be made randomly. This system was capable of generating 64 different sentences, such as:

Top critics of the government executed by security men.

Two dissidents harassed by government agents.

Thousands of demonstrators thrown in jail by police.

Again, for an entire storyworld, 64 variations would be pathetic, but for a single sentence, the limited number of variations is not a problem.

The next improvement to the headline generator was to include custom symbols that indicate special handling. When such a character is encountered in the text, the software inserts some specified word. Here's a small sample:

?: name of the Subject of the sentence

\$: name of the DirObject of the sentence

#: third person nominative pronoun appropriate for gender of the Subject

For example, the source “% tells \$ that # is feeling sad” would translate to “Fred tells Mary that he is feeling sad” in the case where Fred is the Subject and Mary is the DirObject. Different Subjects and DirObjects would yield a different rendition.

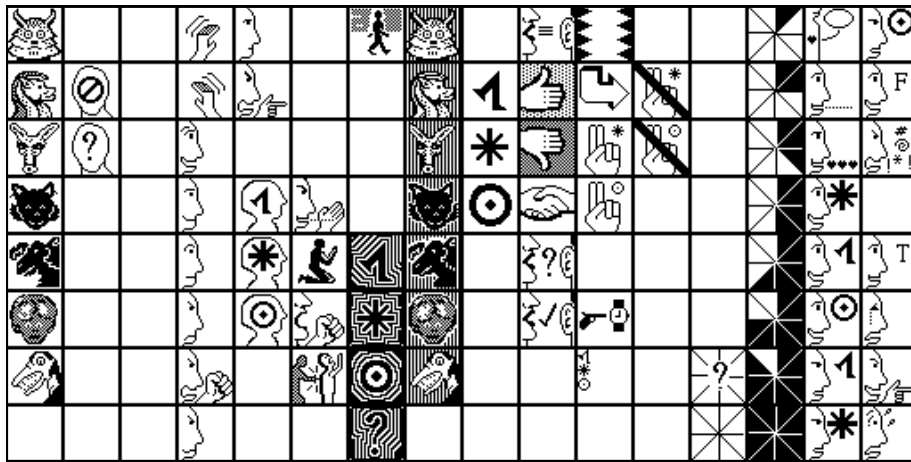
The best improvement is the inclusion of a scripting language for text formation; you'll have to wait for Chapter 18 to see how that's done.

Creating a New Language

Instead of doing a poor job of implementing natural language, we might instead create a completely new language appropriate to our needs. It seems a promising route to follow, but be warned: The task is more difficult than it first appears. I urge you to research “constructed languages” on the Internet before making any decisions. You'll learn that almost anyone can design a language (and several have, including me). Some of the more interesting ideas to learn from are in Solresol and Basic English. I have carried out a number of experiments in creating my own languages to use in interactive storytelling, most of which were failures, but two of my efforts are worthy of consideration. You can see my successes in the game *Trust & Betrayal*.

Trust & Betrayal

My 1987 game *Trust & Betrayal* was based on an extrasensory perception–based (ESP-based) language. The characters all possessed limited ESP powers that they could use to communicate through a standardized system. The language was presented to the player in the form of icons. **Figure 17.2** presents a visual dictionary for that language on the top and translations on the bottom.



Yetvel			greet	formal		go to	Yetvel house		tell			one	say feeling	don't love
Kendra	no-body		good bye	accuse			Kendra house	tanaga	agree	that	broke betray bond	two	small talk	forgive
Gard-bore	who		warn				Gard-bore house	katsin	reject	won't betray	broke attack bond	three	flatter	anger
Wiki			nice	tanaga count	offer reveal		Wiki house	shial	if	won't attack		four	trust	
Lock-sher			sincere	katsin count	beg	tanaga attack	Lock-sher house		ask			five	fear	thanks
Zubi			cooly	shial count	threat	katsin attack	Zubi house		answer	wait	pause	six	love	apology
Skor-dokott			threatening		betray	shial attack	Skor-dokott house				not known	seven	don't fear	deride
			haughty			what attack					zero	eight	don't trust	fright

FIGURE 17.2 Icon dictionary used in *Trust & Betrayal*

Remember, this was created in 1987, when our graphics capabilities were pretty dismal. Even with only 79 words in the dictionary, the language was fairly expressive. The words were joined together geometrically to form sentences; **Figure 17.3** shows one such sentence.

As you can see, the system was crude but effective.



FIGURE 17.3 Yetvel tells Skordokott that Wiki betrayed Skordokott

Inverse Parsers

In addition to the ESP-based language, *Trust & Betrayal* introduced an idea that I call an *inverse parser*. It turns out that Texas Instruments patented the basic idea in the 1970s, which I was unaware of when I reinvented that wheel. The basic idea is to invert the normal operation of a parser.

And what is a *parser*, you ask? It's a hunk of software that scans through text and attempts to figure out its meaning. Text adventures and interactive fiction rely on parsers to understand the user's input. They can understand simple sentences, but the problem that has always plagued such parsers is the difficulty that the player has in figuring out how to word their entry in a manner that the parser can understand. The inverse parser eliminates this problem by placing the computer in the role of servant rather than judge, and the player in the role of commander rather than supplicant.

The underlying idea is that a sentence is composed of words that must be selected in order, so the computer starts off by presenting the player with the first word in the sentence, then offering the player a list of all the words that would reasonably fit into the next slot in the sentence. Each time the user selects the next word in the sentence, the computer places that word into the sentence and offers the player a new set of words that could reasonably be used in the subsequent slot in the sentence. **Figure 17.4** shows how the sentence in Figure 17.3 would have been built.

The top panel shows the initial presentation to the player. The player is Vetvel, the horned creature. Since the player's commands must have themselves as the subject, the inverse parser has already filled in the subject. The inverse parser has also added a blank square to indicate the next slot the player must fill in. That slot requires a verb, so the inverse parser presents all the verbs that are appropriate in this context. Note that the inverse parser excludes a number of verbs from the set. For example, the verb "hello" is not available. That's because Vetvel is already on the same stage with Skordokott and nobody else is present, so it would be silly to say "hello" a second time. Similar considerations have prevented other verbs from appearing in this list.

This is an important differentiator from a conventional parser: The inverse parser must include all the intelligence that the regular parser has, but it must apply that intelligence *before* the player enters their input (not after, as is done with a conventional parser).

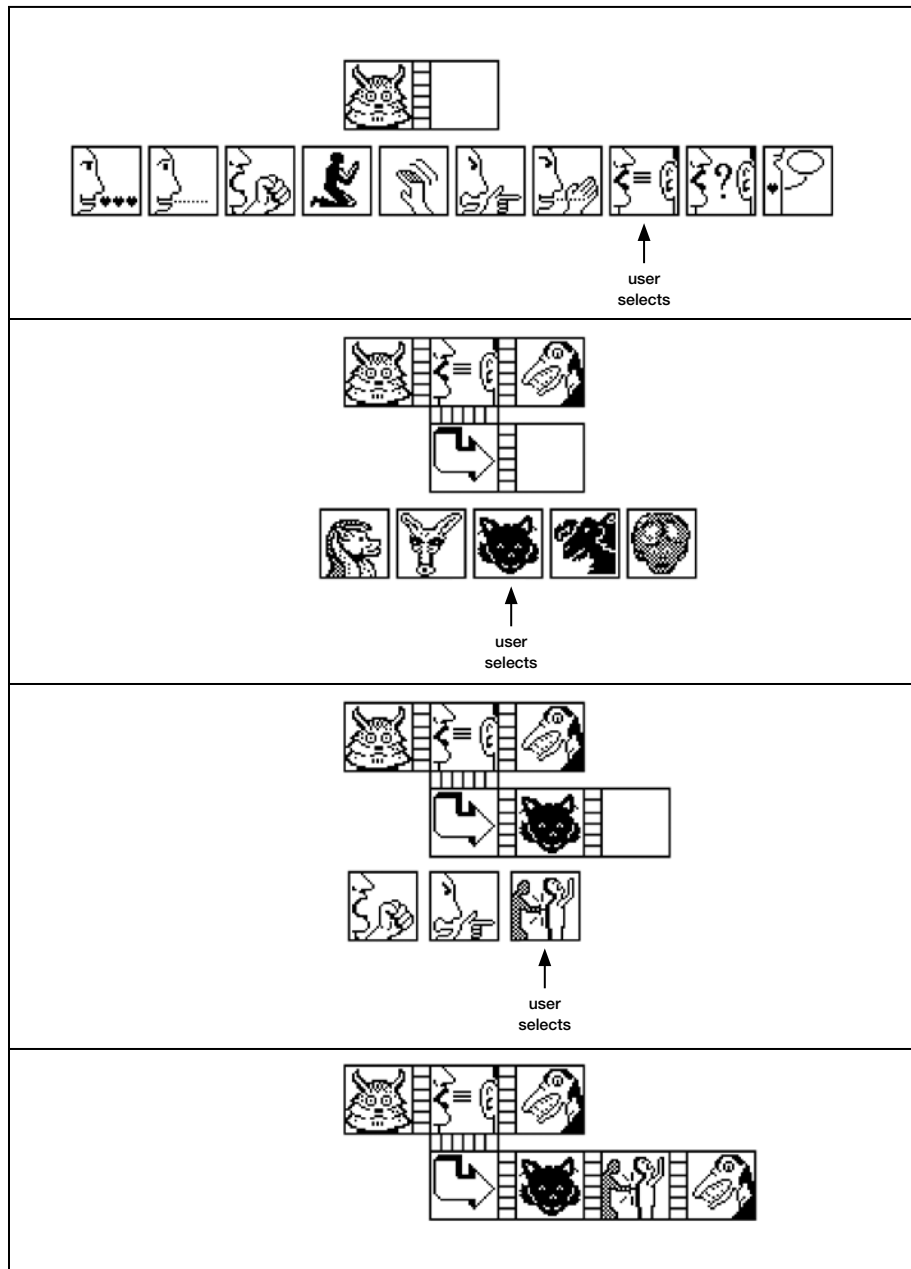


FIGURE 17.4 Process of building the sentence in Figure 17.3

The player scans through the available verbs and chooses “tell.” The inverse parser enters that verb into the sentence abuilding and then adds two more words: Skordokott as DirObject and “that” under the verb. It does so because the player has no choice but to enter those words. Skordokott is the only character on stage, so Vetvel must be talking to Skordokott. The verb “tell” requires the word “that” to introduce the subordinate clause, so “that” is also automatically entered.

Now the inverse parser reaches another point where it cannot know the player’s intentions, so it again presents the player with a blank slot and a list of the available words. In this case, the next word must be the Subject of a subordinate clause, so it must be a character. Obviously Vetvel isn’t going to tell Skordokott about something that Skordokott did, nor about something she did, so those two characters are not included in the list of selectable words.

The player selects the character Wiki, the catlike creature. Once again, the inverse parser fills in the remainder of the sentence. In this case, the inverse parser is wired so that players cannot tell lies, so the inverse parser examines all the truths Vetvel knows in her HistoryBook. The parser determines that there is only one Event that Vetvel knows about, the one in which Wiki betrayed Skordokott to somebody else, so it enters the rest of that sentence.



This is just a glorified system of nested menus!

Not quite. The contents of any given menu are determined by the previously declared menu selections. These are dynamic menus, not static ones. Otherwise, yes, the system is at heart a menu-selection system. An inverse parser can be implemented with text even more easily than the one I used in *Trust & Betrayal*. Later in this chapter, I’ll explain some of the considerations involved in choosing between text and icons.

SympolTalk

SympolTalk is a language system I devised that goes far beyond the capabilities of the iconic language in *Trust & Betrayal*. The original is in color, so the black-and-white rendition in **Figure 17.5** isn’t quite as easy to read as the real thing.

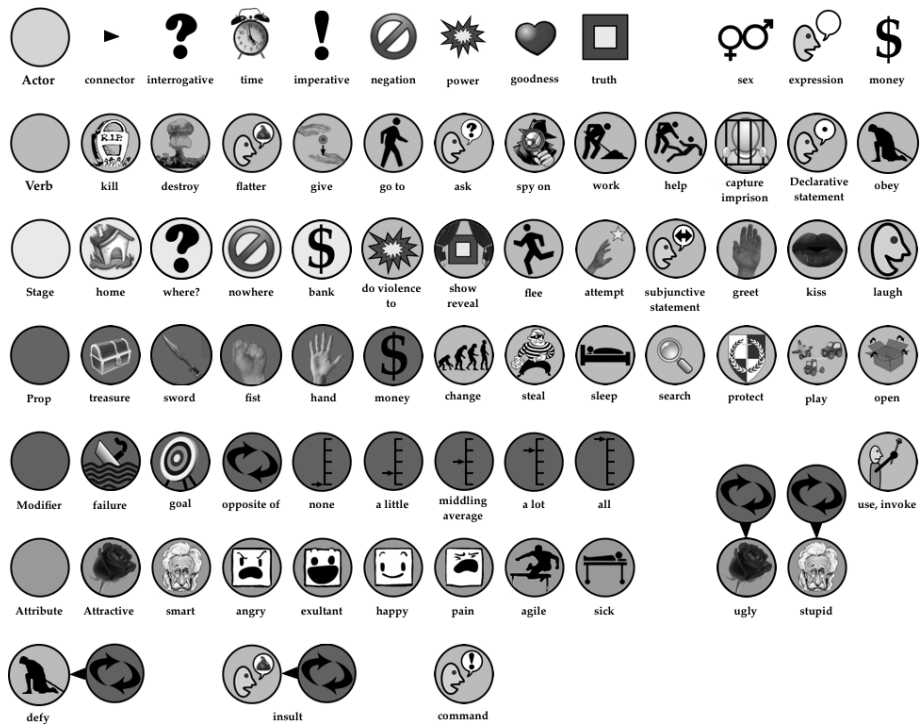


FIGURE 17.5 Partial dictionary for SympolTalk

The icons follow a standard system of background colors: Verbs are green, characters are blue, stages are yellow, props are magenta, adverbs are brown, and adjectives are purple. This iconic language assembles words to make a sentence in the same fashion used in *Trust & Betrayal*. **Figure 17.6** shows a sample sentence.

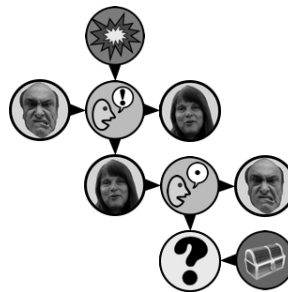


FIGURE 17.6 A sentence in SympolTalk

The word-for-word translation of this sentence is this: “Chris says imperatively (with power) to Gemma that Gemma tell Chris where of treasure.” The meaning is “Chris demands that Gemma tell him where the treasure is.”

Iconic languages have a big advantage over textual language: They’re two-dimensional. Normal language is one-dimensional, which can lead to a lot of confusion about how words connect with words that aren’t adjacent. The computer offers a huge advantage in its two-dimensional representation of information, so it’s easier to communicate syntactical relationships on the computer.



But users have to learn the language. That’s a huge obstacle for them to overcome; people just won’t go for it.

You underestimate the facility with which people pick up new languages. Any tourist can tell you how quickly you start picking up the language when you are in a foreign country and make the effort. We humans are vacuum cleaners for language, slurping up new languages like so many candy bars. Few players had any problems with the iconic language in *Trust & Betrayal*, and those icons were only 32 pixels across and in plain black and white. The icons for SympolTalk are 64 pixels across and in color, providing 96 times as much visual information as the icons in *Trust & Betrayal*. If we equip them with a simple tooltip (the little yellow balloon that often pops up next to an icon to explain its meaning), people will pick up the language readily enough.

Semantic Volume

Iconic languages feature another, even more important benefit: The semantic volume of each symbol is greater than words in natural languages such as English. Consider, for example, this icon from SympolTalk (**Figure 17.7**).



FIGURE 17.7 A verb in SympolTalk

The formal meaning of the icon is “declare.” It can be interpreted as tell, say, declare, aver, reveal, disclose, insist, admit, confess, speak, and many other words. Each of these English words has a narrow meaning; “to declare” means something different from “to reveal” or “to admit.” If we used a single word from normal language, we could end up with sentences that don’t sound quite right:

Sam said that Joan was wrong.

Sam insisted that Joan was wrong.

Sam said that he murdered Joan.

Sam confessed that he murdered Joan.

Sam said that Tom murdered Joan.

Sam revealed that Tom murdered Joan.

This is really a version of the *uncanny valley* problem that states that people tend to perceive any image of the human face as either a cartoon-like drawing or an actual photograph. They’re willing to accept inaccuracies in drawings or artwork because they know that it’s artwork. But suppose you drew a picture of a face that was almost—but not quite—photorealistic. In fact, this was the case with early 3D facial images: The images were close enough to reality that we couldn’t accept them as drawings, but they weren’t accurate enough to look real. Images like that give us the creeps because they look like robots or dolls or something inhuman. They fall into what computer scientists call the uncanny valley.

The same problem arises with artificially generated text. People can accept the brusque shorthand of machine instructions: “Insert coin here” for “Put the coin into this slot” or “Power” for “Press this button to turn on the machine.” When you create text with a computer, it can look great most of the time, but every now and then it betrays its mechanical nature and falls into the uncanny valley.

For my first version of the Storytron technology, I built a powerful system for scripting English output (explained in Chapter 18). It understood pronouns, singular and plural, and gender, and it gave the author lots of power to tune the words to the precise situation. But it still triggered an uncanny valley reaction from players. *Inform 7* also has powerful facilities for correct English phrasing in interactive fiction, but it’s not immune to the problem, either. Language is simply too subtle and too nuanced to be readily computed just yet. For the time being, we’re better off using the deliberate vagueness that we get with an iconic language.

How to Create a Language

The first task some designers tackle is creating characters, props, and stages. This is backward: The first step must be to create the verb list (remember Crawford's First Law). The verbs in interactive storytelling are more complex than the simple verbs in games, as befits the greater complexity of the human condition.

So you're going to start off by creating a long list of verbs for your storyworld. That list will provide you with the information you need to devise the syntax of your language. It should be obvious that every verb will take a Subject and a DirObject. An IndObject is necessary if you want to permit your characters to do something with props. But what if you want people to be able to trade things? Then your sentence structure would look like this:

Subject trade DirObject Prop1 Prop2

This structure provides you with the core sentence, such as "Rick trades Marlene his book for her comb." Be careful here! It's all too easy to come up with huge sentence structures requiring all sorts of complexity. My advice here is to keep things simple by restricting yourself to verbs that require only a Subject, a DirObject, and an IndObject. You can get clever once you've gotten something under your belt.

Symmetry

A fundamental advantage of creating your own language is the ability to make the language symmetric between the user and computer. In other words, the user and computer talk to each other in the same language. This should be perfectly obvious, but our entire experience talking with computers has been with asymmetric languages. The computer talks to us with pixels; we talk to it with a keyboard and mouse. We have to learn two languages: the computer's output language and our input language. What a load of work.

Yes, it's a little harder to design a language that works for both the user and computer, because that language has to meet the needs of both agents, whereas the double-language system now in use can take advantage of the different abilities of each of the agents, computer and human. Nevertheless, the cleanliness of a symmetric language offers such great advantages that I believe it's worth the additional effort.

KISS!

My advice to the beginning interactive storyteller is to start off with a simple storyworld. Start with something closer to “John and Jane play in the sandbox” than *War and Peace*. You want a situation in which actors are naturally limited in the kinds of things they might do. The activities of young persons seem a strong candidate here, because young people do not have many of the experiences that adults do. An ideal test bed is a group of high school students socializing by texting each other: The range of things they are likely to say is limited, their social interactions can bring up a lot of conflict, and there’s a natural turn-sequencing to their interactions.

Finally, you would do well to study basic linguistics. Like it or not, you can’t do interactive storytelling without some kind of language, and until computer scientists build a full-featured natural language comprehension facility, you’ll just have to design your own.

Conclusions

Game designers have it easy: They can do everything they need with simple input devices. But interactive storytelling is far more demanding: Stories are told by means of language, so any interactive storytelling technology will have something in the way of a storytelling language. It is possible to pull this off with fractured English, but experience shows that fractured English is frustrating for most users. For the next few decades, an iconic language is more likely to yield success.

This page intentionally left blank

CHAPTER 18

Scripting Languages

A STORYWORLD MUST BE INTERACTIVE. Interactivity requires thinking on the part of the computer. Thinking can be accomplished only by some sort of programming. Therefore, all storyworld authors must do some programming.



Ack!

That's a succinct way to put it. Let's face it, unless you *are* a programmer, programming sucks. It's like accounting, only the numbers are always changing on you. You must dot every last i and cross every single t, some of which are hidden, and if you miss just one, the whole thing will blow up in your face with no indication as to why. Programming cannot be done without some degree of obsession-compulsion.

Don't panic; scripting languages provide us with a kinder, gentler form of programming. Programming languages are like those super-duper Swiss Army knives with 20 foldout implements; scripting languages are like kitchen knives, where each knife has a single purpose and is optimized for that purpose. Using a monster Swiss Army knife properly requires a lot of study and practice, but mastering the use of a paring knife is simple. So it is with scripting languages.

Scripting languages reduce complexity by eliminating some of the abstraction fundamental to full-featured programming languages. Scripting languages focus narrowly on a single type of problem. Therein lies their value and their purport.

Desiderata

Every system for interactive storytelling must include a provision for programming; this is best achieved through a scripting language. What can we do to ease the burden on our suffering storyworld authors? Following are some guidelines for such a scripting language.

Fixed Typing

Typing is a concept from program design. The idea is to require that every variable have a defined type. Deep down inside a computer, everything is a 64-bit integer, but our software can interpret those integers in all kinds of ways. It can treat that number as an integer, a floating-point number, a text character, a colored pixel on the screen, or as part of a bigger datum that includes many numbers. For example, we can think of a stage as a type of data endowed with many attributes, each specified by a number. The stage might have one number indicating how brightly lit it is; another number for how many people it can hold; a third number denoting whether it is an indoor stage or an outdoor stage; and so on. It can have an array of numbers telling us which characters are currently occupying the stage. All these numbers characterize the attributes for the stage, and taken together as variables, they constitute the definition for the stage.

Modern programming languages permit the programmer to create an infinite number of types, equipping each type with everything it needs, including custom routines. These turbocharged types are called *objects*, and are fundamental to all modern programming languages. Here is a place where we can dispense with some abstraction. We don't need or want objects in a scripting language; we can instead define all the data types that the scripter needs, relieving them of that responsibility.



But I want power. Don't take objects away from me.

If you want power, use the programming language C#. The whole point and purpose of a scripting language is to simplify the programming task, and the only way to ease the burden is to take away some of the programming responsibility. Remember, responsibility and power are two sides of the same coin; increase one and you necessarily increase the other.

Not many data types are required to address all the fundamental requirements of interactive storytelling. Here are some you should consider, listed in order of increasing complexity.

Booleans

In scripting, Booleans are the same as in regular programming; they can be either true or false.

Numbers

Numbers are just plain old numbers. You can use integers or floating-point numbers if you want, but I recommend using bounded numbers, as described in Chapter 13.

Props

Props are the physical objects used in the storyworld. Please be careful here; game designers assume that their worlds must bristle with spare hand grenades, medkits, keys, gold coins, swords, and so forth. Remember, this is storytelling; we don't need all that nonsense. Albert Hitchcock said, "If you show a gun in the first act, it better have been fired by the third act." Include only those props that can be used for dramatic effect.

Stages

As described earlier, a stage is a location where characters interact. All the characters sharing a stage can interact with each other; a character *must* share the stage with another character to interact with that character. Stages should not have any X, Y, or Z coordinates; movement from one stage to another should be instantaneous. The player, of course, should only be able to see the stage where he or she currently resides.

Characters

As described in Chapter 14, a character is any member of the cast—somebody who can be the Subject or DirObject of a Verb. If it can make significant decisions, it's a character.

Verbs

As described in Chapter 16, verbs are the crucial elements of the storyworld, specifying what the characters (including the player) are able to do.

Events

As described in Chapter 16, these are actual cases of a verb being executed. Think of them as sentences, with a Subject, a Verb, and probably a DirObject. Taken in sequence, they're the story that the player generates while traversing the storyworld.

Colored Types

This is a small touch, but most programming editors have now caught on to the value of using colored text to indicate different kinds of things. It helps you see the structure of the code and makes it easier to recognize problems. This little trick works nicely with the fixed data types mentioned. In my work, I have used these color assignments in the code:

Booleans: black

Bounded numbers: red

Props: magenta

Stages: orange

Characters: blue

Verbs: green

Events: French blue

Flow Control

Flow control is such a fundamental element of all programming languages that it has spawned many terms over the decades: GOTO, jump, subroutine, method, function, procedure, loop, branch, and so on.

The basic concept is to control the sequence in which the program executes its various parts; this is one of the first concepts a student programmer learns. It is possible, however, to relieve the storyworld author of the responsibility for flow control, which requires a number of stratagems.

Loops

The first trick of relieving the responsibility for flow is to collapse all loops into selection functions. I have learned that the selection function is far and away the most common use of loops in interactive storytelling. A *selection function* is a loop that sweeps through all the members of some set, looking for the single member that best fits some set of criteria. For example, suppose that at some point in a storyworld, a young woman must decide upon the man she wants to marry. This is a simple selection function: It must examine all the men in the cast to find the man who has the best overall combination of traits. In normal programming, it would be executed with a loop. Let's suppose that the ideal man for our girl is unmarried, rich, and handsome. Then our loop for finding him might look like this:

```
Float best_Score = 0;
Int best_Man = -1;
For (i = 0; (i < cast_Size); ++i)
{
    If (male[i] AND NOT married[i])
    {
        Float score = Rich_Poor[i] + Handsome_Ugly[i];
        If (score > best_Score)
        {
            best_Score = score;
            best_Man = i;
        }
    }
}
return best_Man;
```

This entire block of code can easily be built into the scripting language by adding a function that accepts two little scripts as input parameters. I call these two scripts *acceptable* and *desirable*. Pay close heed here, because I have found these two concepts especially useful in many areas of script design for interactive storytelling.

Neither of these two little scripts is a number; they are both bits of script. An *acceptable* script returns a Boolean value; a *desirable* script returns a number value. For our example, the two scripts would read something like this:

```
acceptable = male[Candidate_Character] AND NOT married[Candidate_Character]
desirable = Rich_Poor[Candidate_Character] + Handsome_Ugly
[Candidate_Character]
```

The storyworld author need merely fill in variables in the scripts for *acceptable* and *desirable* and pass them on to a function I call *Pick_Best_Character*. You need one *Pick_Best_XXX* for each of the data types.

Voilà! No more loops!

Branches

It's easy to replace the full-featured IF statement with a simpler, easier-to-understand version:

```
Pick_Upper_If
{
    Boolean script for deciding between upper and lower script
    upper numeric script
    lower numeric script
}
```

Yes, it's still basically an IF statement, but it's easier to understand in this format. You have to know in advance what each term really means. The first term is a Boolean value; if it's true, then the upper numeric script is chosen for execution and the lower number script is skipped over. If it's false, then the program flow is the other way around (lower instead of upper). The important thing is that it presents the issue as a choice between two possible *answers* rather than two possible program paths.

Standardized Sequencing

This is the most difficult design task for an interactive storytelling engine, but it is also the most useful. Chapter 19 explains engines and describes the engine I built, which has a complex and rigorously defined sequence of steps that it executes to interact with the player. Those steps have been carefully organized into fixed parts and variable parts. The fixed parts are hardwired into the engine. The variable parts are controlled by scripts written by the storyworld author. The storyworld author writes bits of script for some of the steps in the engine's operation. Each script has its place. This means that the storyworld author doesn't have to figure out the sequence of operations; instead, the author need only fill in required blank spots.



But this requires the author to memorize every step in the operation of the engine, and how the script for that step alters it. That's a lot of work!

Did you forget Lesson 1: *Stories are complex structures that must meet many hard-to-specify requirements.* The whole point of my engine is to provide the author with a process that addresses many of the requirements of storytelling. Once the author understands that process, they need only fill in the blanks. Yes, that's a lot of work, but it's still less work than building an engine from scratch.

Reducing the process of interactive storytelling to a set of scripts that meet the requirements of each step in the process is an immense design challenge. You've been warned!

Fully Functional Defaults

In every programming language I've ever seen, you can't just plunge in and start writing your code; there are always niggling little specifications that you must satisfy first. Want to use a new variable? Declare it first, and be careful to give it the correct scope. Have an idea for a new class? Good luck: You'll have to specify things like its place in the object hierarchy, what libraries it uses, and other techie goodies before you can actually use it.

Sometimes, the requirements of defaults can be subtle; I've lost much hair dealing with the problems arising from such requirements. Although there are good reasons to include such requirements in full-featured programming languages, a scripting language can be designed to relieve the scripter of such nuisances. The way to do this is to have a standard, fully functional default for anything new the scripter creates. The act of creating a new data structure should not only automatically trigger the creation of all subsidiary elements, but also the initialization of those elements to their most common values. The initialization should be complete enough that the new component can be used in the running interactive storyworld without a scripting error. It might not produce ideal results, but it won't get in the way, either.

It's true that the scripter might not realize that one of the members of a component needs to be corrected. It's easier, however, to notice and correct an incorrect entry than to realize that a required entry is missing.

Providing fully functional defaults dramatically changes the nature of the scripting process. To the scripter, the process becomes one of merely perusing what's already there and deciding if it needs to be altered. There's nothing hidden that must be remembered and plugged in.

Menu-Driven Editing

One of the worst flaws in almost all languages, both programming and scripting, is the reliance on keyboard input. I consider this a blunder of the first magnitude. It's fine for professionals who are happy to commit the vast syntax of a language to memory. But beginners choke on all that complexity—and when it comes to interactive storytelling, *everybody* is a beginner. It is too frustrating to type in something with a small error (a simple typo, misspelling, or failure to include a delimiting character) that ruins everything. It's difficult to maintain your creative optimism when the computer tells you over and over that you screwed up the syntax. Worse, it's impossible to learn all the little details that are there to help you along. You have to consult the manual, which nobody ever does until they are forced to do so. The manual for Java is a huge online monster consisting of several thousand pages of specifications, all written according to a precise system that you must first figure out. I doubt that there's anybody who knows every single capability built into the Java language.

A scripting language should be fairly small. It should be small enough to access by menus. Sappho, the scripting language I developed for my own interactive storytelling technology, comprises 671 distinct operators, such as Blend, ReflexivePronoun, PastVerb, and LookupEvent. You might think that's too many to include in a menu structure, but that's not the case. Here's how editing proceeds in Sappho: First, you click some existing operator in the script that you want to alter. (Remember, everything you need is always there, so you never need to add a new term. You simply need to alter an existing one, usually by making it more complex.) The software figures out the data type of that operator and then rebuilds all the menus to include only those operators that return the selected data type. After all, you can't put a prop into a slot requiring a stage; you must put a stage into a slot requiring a stage. Therefore, the menu system never needs to include all 671 operators; on average, it needs to provide maybe 100 operators.

Sappho has a stack of menu titles along the right edge of the window with 22 major menus; the average menu needs only about 5 menu items. Yes, there are extremes—the longest menu in Sappho has 21 items. There are no secondary menus; you can reach every operator by a standard click-drag-release operation.



Mouse-driven input is for slowpokes; the keyboard is always a faster way to enter information than the mouse.

Yes, you can put bytes into a computer faster with a keyboard than with a mouse, but can you *work* faster with a keyboard than a mouse? Scripting is not just a matter of typing; you have to spend time thinking about what you want to do. If you've already memorized all 671 operators in Sappho, then you don't waste any time thinking; you just type it all in one continuous typing spree. But few people memorize the entirety of any language; we waste time trying to figure out what's available and how to use it. If it's right there on a menu, you don't have to waste so much time. I'm the one who designed and programmed Sappho, so I know it better than anybody, and I still find the menu structure faster to use than keyboard entry.

Furthermore, restricting the menus to contain items of the same data type as the selected item ensures that the user can't commit a data typing mistake such as putting a stage where a prop should be. In fact, Sappho renders syntax errors impossible. The user is never given the opportunity to enter a syntactically incorrect value.

Parameter Prompting

Here's another way to ease the burden of the storyworld author: Provide complete prompts when editing. For example, suppose the scripter has just entered a new term into an inclination formula: the `Nice_Nasty` value of a character. Of course, the scripter hasn't yet entered anything about the character, so we have an unknown value. Instead of putting some unknown value in there, it's better to fill that space with the phrase "Of whom?" That prompts the user to fill in the required information. This can be especially useful with multiparameter operators. If you enter the value `pNice_Nasty`, the code for the story will read `pNice_Nasty[ofWhom? forWhom?]`.



This contradicts your earlier advice to enter fully functioning default values when creating something new.

True, but there is a clear means of differentiating the two cases. When the creation of a new component requires the specification of hidden values, the scripting language should enter fully functional defaults. When the creation of a new component requires the specification of values that are instantly visible, then the user's attention is already riveted on that part of the screen and the user will have no problem realizing that those values must be entered.

Parameter prompting increases in value when you color-code the different data types; the phrase "for whom?" when written in blue clearly says that the slot must be filled in with a character specification.

Benign Error Handling

As previously mentioned, it's possible to design a scripting language to obviate syntax errors. It's *not* possible to design it to obviate run-time errors. For example, suppose your storyworld has a young woman who must select a young man to be her husband. Further, suppose during one particular playing of that storyworld, she dawdles so long that there's no available man left to choose when it's time for her to make a choice. When you call the `Pick_Best_Character` operator, it can't find anybody. What do you do now?

Run-time errors like this arise frequently, and it is impossible to design a decent-size storyworld that is immune to run-time errors. Handling such errors is therefore an unavoidable responsibility.

For most of computing history, programming languages had a standard response to run-time errors: crash. Modern languages include lots of protections against run-time errors, but they still occur. Usually such errors make further processing impossible, which pretty much requires the program to throw up its hands and quit.

But in interactive storytelling, there is an alternative that I call *poison*. When Sappho encounters a run-time error, it doesn't crash or quit; it simply poisons the calculation. This causes every other script tied to the poisoned script to be poisoned as well; Sappho skips over scripts that are poisoned. This poisoning process extends all the way upward to whatever high-level operation used it; that high-level operation is then nulled or erased. This results in the engine refusing to act on a particular situation—but not dying.

In our example of the procrastinating young woman, the failure to find a suitable candidate husband does not wreak havoc on the storyworld. Instead, the poison works its way up only as far as the girl's option to find a candidate husband. That option is erased and everything continues as if she had never had that option in the first place. The run-time error doesn't kill the storyworld; it just makes it a little duller.

Function Analyzer

Suppose you write an inclination formula that looks like this:

```
Inclination = Blend(Nice_Nasty(subject), pNice_Nasty(dirObject, subject),
happy_sad(subject))
```

“That looks about right,” you tell yourself. But do you really know how it will work? Do you understand just how changes in the three input terms will affect the final result? Other than working out a lot of examples with a calculator, or writing a program to perform the evaluation, you really have no way of knowing how any nontrivial algorithm will perform.

What you need is a machine with dials for each of the three inputs. You wiggle the dials around and see how the final result changes in response to your wiggling. That would certainly make clear exactly how the algorithm functions.

A good scripting language would provide just such a machine, but to pull this off you need to implement something unconventional: vertical code layout.

All programming languages are presented to the programmer in a vertical layout; the program is read from top to bottom. It is possible to pack a lot of programming into a single line of code; this squeezes more program into a single-screen view and thus makes the program easier to read. Vertical code layout forbids such space-conserving methods; instead, each line of script contains exactly *one* operator. This makes scripts tall and narrow.



What? To read a program, you'd have to scroll up and down great distances to see what's going on. Your program would look like a roll of adding-machine paper.

Not necessarily; **Figure 18.1** shows how Sappho implemented vertical code layout.

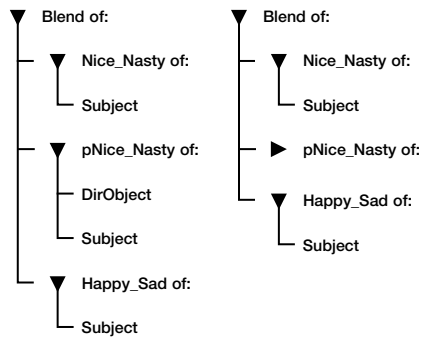


FIGURE 18.1 Vertical layout in Sappho showing collapsed component

This arrangement uses the familiar structure used in file display windows and obeys the same conventions: Click the triangle to toggle it open or closed. The user can collapse everything down to a single line, or open it up to home in on exactly what is interesting. In some ways, this is even more convenient to use than traditional programming editors.

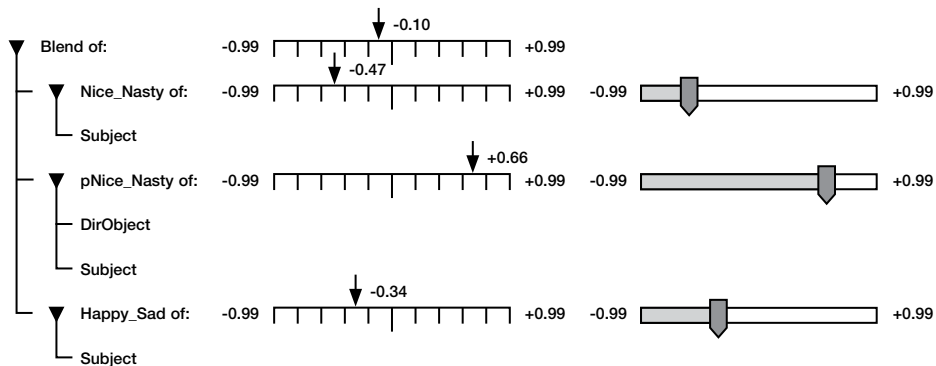


FIGURE 18.2 Function analyzer in Sappho

The real value of vertical code layout, however, lies in the special features it makes possible, such as a function analyzer. **Figure 18.2** shows the function analyzer in Sappho.

The analyzer is broken into three columns. In the left column is the script itself. The right column contains sliders, one for each of the input variables. You see the final result of the calculation at the top of the rulers in the center column. As we wiggle the sliders around, the arrows move back and forth, showing how the final result changes as the input changes.

In my experience, this analyzer makes algorithm design easier, faster, and less stressful for storyworld authors. This system would not be possible without vertical code layout.

Story Recording and Playback

There is yet another benefit of vertical code layout. Facundo Dominguez came up with a brilliant scheme for recording every single decision the storytelling engine made in response to the user's actions during a single playing. This permits the author to play a storyworld, then go back and review every single decision, right down to the details of each calculation, to understand why something happened the way it did. This is, in some ways, better than traditional debugging, because you don't have to catch the problem red handed. You can go back at your leisure and see exactly what led up to the problem and how it arose. **Figure 18.3** shows an example of how Sappho does it.

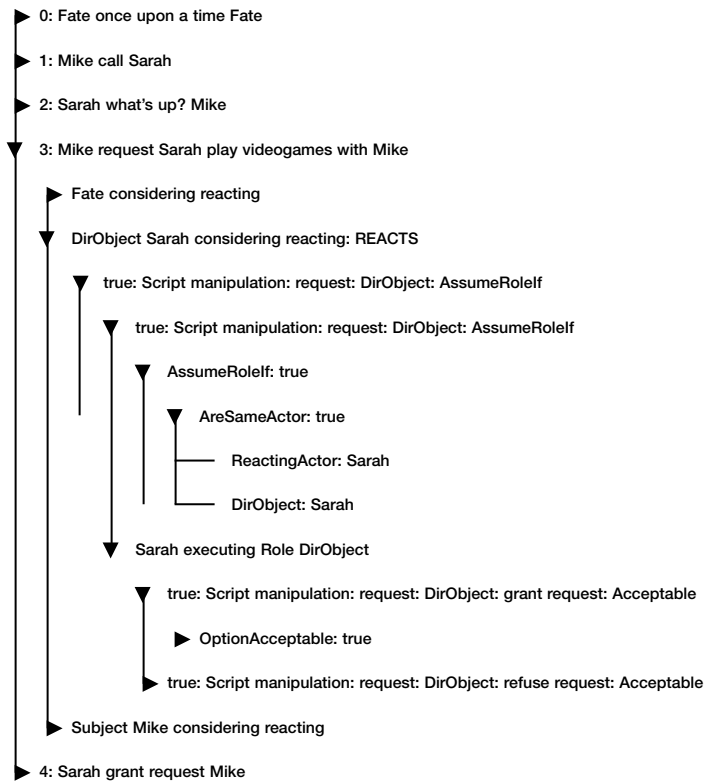


FIGURE 18.3 Record of a storyworld execution

This display is, admittedly, cryptic. Five events are executed, but they are numbered 0 through 4. Each event is presented in shorthand that identifies only the Subject, Verb, and DirObject of each sentence. Event #3 is open for examination. There were three characters who considered reacting: Fate, Sarah, and Mike; of these, only Sarah actually reacted. The details of the script that made the decision for Sarah to react are visible and you can see exactly why the result is true.

What's really impressive about this technology is that it doesn't require megabytes of storage; Facundo devised a truly clever scheme that allowed the engine to reconstitute events as you request information about them.

Extreme Modularization

Modularizing your scripts to the extreme consists of nothing more than creating lots and lots of little scripts rather than writing a few big scripts. The task of programming a storyworld is easier when it is broken into many little steps. In my own technology, a single verb requires perhaps 70 scripts, but most of these are just one or two lines long.



A scripting language of this much complexity would take years of programmer effort to produce.

Welcome to interactive storytelling. Didn't I tell you this was going to be difficult? One way or the other, authors will be creating lots of algorithms for the storyworlds they build. They can use any programming language they want, but how much longer will it take them to master a full-featured programming language than a custom-designed scripting language like this?

Sample Script for Text Generation

In previous chapters, I promised to demonstrate how to use a script to generate fairly intelligent text. **Figure 18.4** presents one such script.

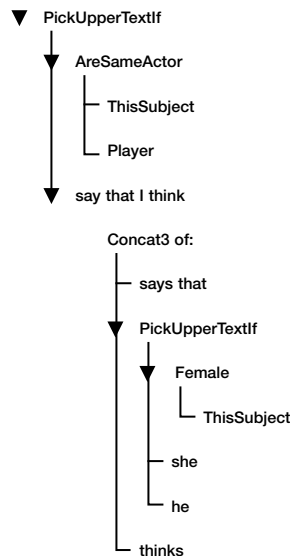


FIGURE 18.4 Simple script for a verb phrase

This little script supplies the verb phrase for a sentence. If the subject of the sentence is the player, then the verb phrase is simply “say that I think.” Since the Subject is the player, the sentence up to this point is written as “I say that I think.”

If the Subject is not the player, then the verb phrase will be the concatenation of three strings. The first string is “says that” and the last string is “thinks.” The middle string will be “he” or “she,” depending on the gender of the Subject. Thus, the verb phrase will read either “says that she thinks” or “says that he thinks.”



That's a lot of work for a trivial bit of text manipulation.

True, but there are other text functions to work with to create more complex sentences, as shown in **Figure 18.5**.

Typical output generated by this script might read as follows: “Morgana reserves one-eighth of the spoils for herself; the remainder she allots evenly among the subkings. In addition, she grants one-quarter of the cattle to Mordred.” The ConcatN operators in Lines 1, 13, and 18 simply concatenate strings together. The only strange operator is CalculatedText5 in line 5, which selects one of the first five strings based on the numeric value in the sixth slot: Quantifier2BNumber. This operator takes a Quantifier type value from the sentence in question and converts it into a bounded number value between -1 and 1. A *quantifier* is a word that expresses the magnitude or degree of some phenomenon. In this example, the quantifier is converted to a number that is in turn broken into quintiles. The lowest quintile is translated into text as “none”; the next quintile is translated as “one-sixteenth,” and so on.

These examples are intended only to demonstrate that text can be scripted. There are quite a few other text functions that can handle verb suffixes, singular versus plural pronouns, and so forth.

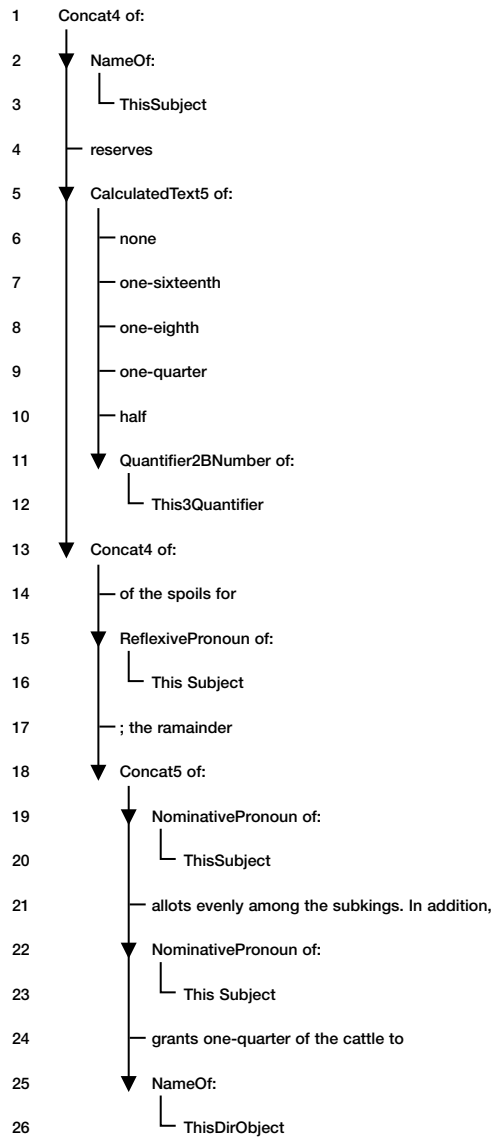


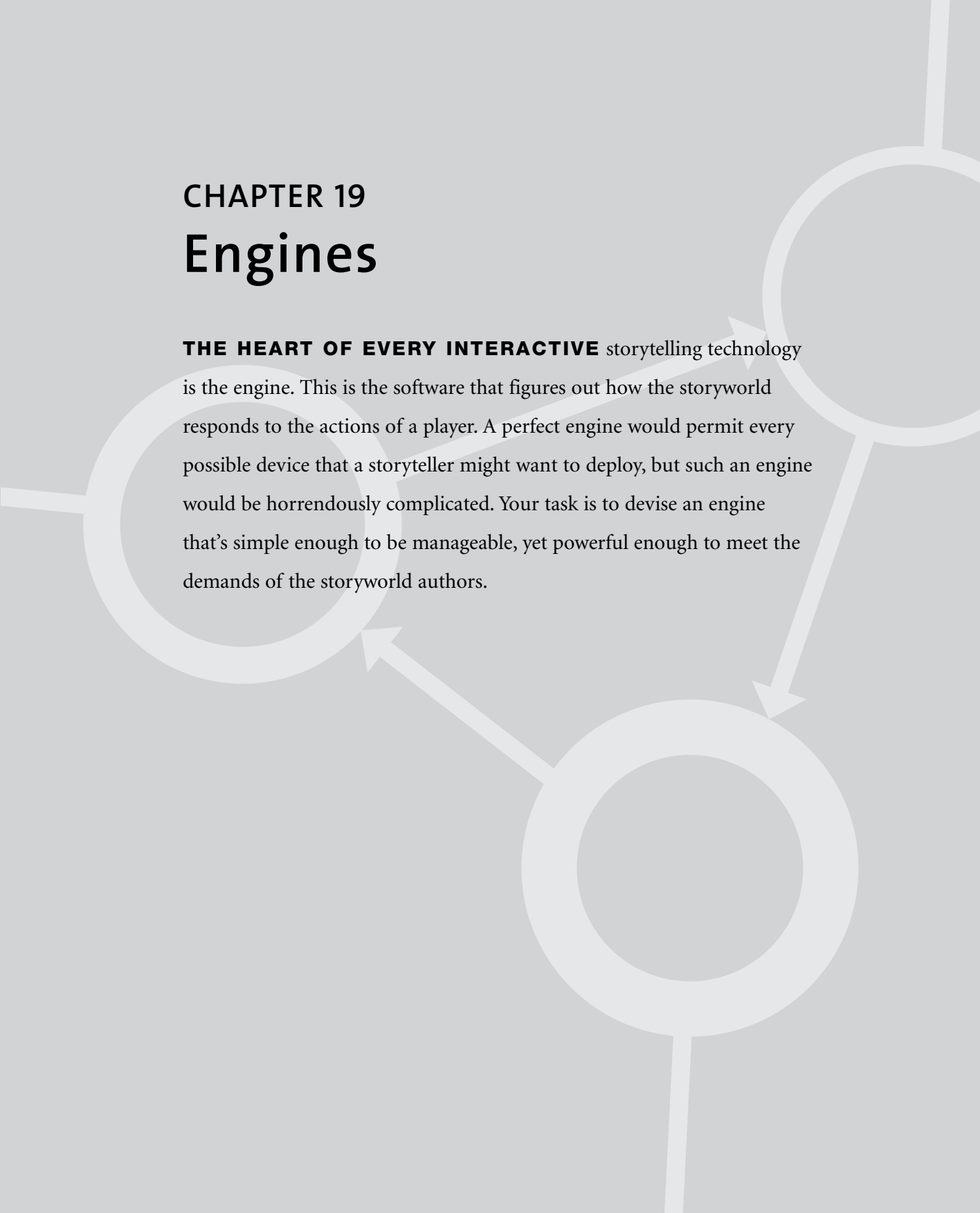
FIGURE 18.5 A more complex script

Conclusions

There's no getting around that storyworld authors must design algorithms and express them in code. We can at least make their task as easy as possible by designing scripting languages that trade computational power for ease of learning and use. There can never be an ideal trade-off between the two; every compromise is a choice of the lesser of two evils. With growing experience, we'll learn more about the nasty trade-offs and how to minimize the pain our authors must face.

CHAPTER 19

Engines



THE HEART OF EVERY INTERACTIVE storytelling technology is the engine. This is the software that figures out how the storyworld responds to the actions of a player. A perfect engine would permit every possible device that a storyteller might want to deploy, but such an engine would be horrendously complicated. Your task is to devise an engine that's simple enough to be manageable, yet powerful enough to meet the demands of the storyworld authors.

Event-Based Engines

The simplest engine relies exclusively on the connections between events to make its decisions. Branching trees use such engines (see Chapter 8). The player triggers an event, which then presents the player with a choice between several new events. The player selects an option, which triggers that event, which then links the player to a new set of choices. Such an engine can be implemented in HTML—just make each page an event and provide each page with links to appropriate subsequent events.

The next step up is to add logical gates between the pages. The player can't get from event A to event B until some logical condition has been met. An improvement upon this is to use numerical calculations to determine the state of the gates. For example, a gate to a new event will not open until the player has executed some action a minimum number of times.

Event-based engines can be souped up in many ways. You can create elaborate Boolean algorithms that open or close the path from one event to another. You can use numeric algorithms for even greater sophistication.

The most complicated engine of this type implements a directed graph rather than a tree. Recall from Chapter 8 that a directed graph is like a tree, except that the connections don't have to go downward; they can also go up or sideways. While a simple tree must be traversed in a limited number of steps, a directed graph could easily provide a path that includes convoluted loops and therefore goes on forever. If these paths involve altering state variables, then the many different passages through the "same" event can take on different significance with each traversal.

The greatest weakness of event-based engines is the difficulty of providing for independent actions by multiple characters. Event-based engines work best for a single character (the player) traversing a series of obstacles. You can get multiple characters working in an event-based engine, but it gets complicated.

Clock-Based Engines

A more convenient way to have multiple characters working simultaneously is to use a clock-based engine (**Figure 19.1**). I call a single tick of the clock a *tick* (I'm a wild and crazy guy). Other people call it a *beat*, a *second*, or a *turn*.

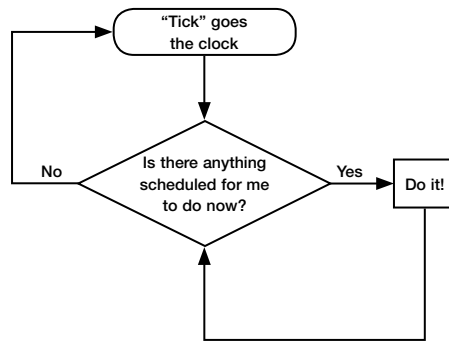


FIGURE 19.1 Basic operation of a clock-based engine

Real-Time vs. Turn-Sequenced

We come to a fundamental decision: Should the engine operate on a real-time basis or on a turn-sequenced basis? In the latter, the engine proceeds until it's time for the player to make a decision, at which point the engine halts and waits for the player's input, resuming operation only after the player has entered his decision. Real-time play, however, requires the engine to continue running regardless of the player's actions or inaction.

Both approaches are technically simple to implement. *Façade* uses real-time play, and it's a good storyworld. However, the reader should realize that real-time play requires a context-free set of verbs. In other words, the player must be free to do just about anything at just about any time. This requires an all-purpose set of verbs that make some sense in almost any situation—a constraint that is difficult to meet.

Imagine, for example, a macho storyworld in which the player can get into manly fistfights with rivals and enjoy romantic scenes with the love of his life. In the fistfights, the player must have a verb for punching somebody in the face. In the romantic scenes, he'll need a verb for kissing. So what happens when he tries to kiss his fistfighting opponent or punch his lover? Sure, you can add appropriate consequences to such actions, but aren't they so absurd as to require pre-emption?

It's possible to apply contextual restrictions on real-time play, but those restrictions must be communicated to the player, which interrupts the flow of play. For these reasons, I recommend turn-sequenced play over real-time play. Let players take all the time they need to decide how they want to react to a situation. However, as Mateas and Stern proved with *Façade*, real-time play works, too.

Plans

For the engine to know when to execute something, it must have what I call a *plan* for that action. In my book—hey, wait a minute, this *is* my book—a plan is an intended event that has not yet taken place. It has a time tag telling when it should be executed. The plans are all time sorted into a list; the engine then starts at the top of list, executing plans until it reaches a plan that is not scheduled to activate until the next tick of the clock. At that point, it gives the clock another tick and looks again for any plans ready to go.



What if the player wants to do something? Where does the player fit into this scheme?

Whenever the situation calls for player input, the engine stops and presents the player with options. The player selects an option, which then becomes a plan, and the engine resumes operation.

Plan Execution

Executing a plan is a ridiculously simple matter:

- ▶ **Move Plan from the Plan List to the HistoryBook.** The engine moves the Plan from the Plan List into the HistoryBook, where it becomes an Event. It also stores some housekeeping information about the Event, including when it happened, where it happened, what Event caused it to happen, and so forth.
- ▶ **Execute Consequences.** The engine executes any Consequences of the Event. A *Consequence* is defined by a script that describes some change in the storyworld. A simple example of a Consequence comes when Tex breaks the whiskey bottle over Slim Jim's head. The whiskey bottle is now broken and cannot be used again as a head-smasher. This change in the storyworld is implemented by a Consequence script.
- ▶ **Offer Fate opportunity to react.** The engine offers Fate the opportunity to react to the Event. (I'll explain the reacting process in a minute.)
- ▶ **Offer witness chance to react.** A *witness* is somebody on the same Stage who is not directly involved in the Event. Witnesses get to react before the participants in the event because their reaction would usually interrupt any actions

by the participants. If the Sheriff is a witness, fires his gun into the air, and shouts “Break it up!” to two fellows engaged in fisticuffs, that pretty well finishes their fisticuff action-reaction sequence.

- ▶ **Offer DirObject chance to react.** The DirObject is far and away the most likely character to react to an Event. After all, if somebody does something to you, what are the chances that you’ll do something in response to their act?
- ▶ **Offer Subject opportunity to react.** The Subject of the Event is the last character to react to the Event. About the only time you want a Subject to react to an Event is when the Event is part of a sequence of Events, as described in the next paragraph.

This particular order is important. Fate always goes first because Fate might want to intervene in a manner that will prevent other characters from reacting. For example, if Mortimer the Merciless has Henry Hero strapped to the operating table and says, “And now we shall remove your heart!” This would be a good time for Fate to blow out a fuse, casting them into darkness from which Henry can escape. If Fate does that, there’s no point in anybody else reacting to Mortimer’s action, is there?



Why would the Subject want to react to the Event? That’s sorta like saying, “Joe punches Tom. Joe reacts to punching Tom by saying, ‘Did I do that?’”

No, silly, the Subject is given a chance to react to permit sequential actions that could be interrupted by somebody else. For example, Mortimer the Merciless might first mix up some poison in secret, and then react to that Event by sneaking the poison into the kitchen, and then react to that Event by putting the poison into the ice cream. Each of those Events could possibly be witnessed by somebody, ruining Mortimer’s evil plan, but we can’t have such intervention unless the entire poisoning plan is broken down into individual steps.

Roles

Each verb has attached to it a number of Roles that define how characters can react to an Event containing that verb. A Role comprises a requirement script, emotional reactions, and a set of options.

Once a character has been given the opportunity to react, it checks the Role requirements for each of the Roles in turn. The Role's requirement script specifies what logical conditions a character must meet to be permitted to execute that Role. Let's suppose that our Verb is "punch in the nose" and Slim Jim has just executed this Verb on Tex. Different people are going to react to this Event in different ways, so they need different Roles. For example, Tex as DirObject will surely want to respond to being punched. But Tex's girlfriend Stella will likely have her own reaction and will need a Role of her own. If Tex's big brother Lamont is present, he'll probably step in as well. Therefore, in this example, we have three different Roles: one for the DirObject, one for a girlfriend, and one for a family member. The Role requirement for the first Role (DirObject) is simple: Is the character in question the DirObject? The Role requirement for the second Role is more complicated: Is the character in question the girlfriend of the DirObject? Lastly, the Role requirement for the third Role is similar: Is the character in question a family member of the DirObject?

Whenever the engine offers a character the opportunity to react to an Event, it scans through all the Role requirements for that Verb, looking for one the character matches. If the engine finds no such Role, then the character doesn't get to react and the engine moves on to the next character. But if it does find a Role that the character fits, then it executes that Role for that character.

Role Execution

The first step in Role execution involves changes to a character's feelings in response to the Event. For example, Tex, playing the DirObject Role, will probably get mad at Slim Jim. But Tex's girlfriend might react by feeling fear for Tex. So each Role starts off with some specifications of emotional reactions, if any.

Now the engine gets to the heart of things: What's the reacting character going to do about the Event? The author assigns to each Role a list of *Options*, which are plans that the reacting character could execute in response to the Event. For example, it would be perfectly appropriate for a character playing the Role of family member to step between the two, join in the fight, or let Tex work it out himself. All three of these Plans would be good to add to that Role as Options. It would not be appropriate for a family member to challenge Slim Jim to a game of chess or kiss Tex's girlfriend, so those Verbs would not be added to the list of Options.

For each Option, the author must specify every part of the sentence required by its Verb. In most cases, this is simple. If Tex selects the Plan “break whiskey bottle over the Subject’s head,” then the Verb is “break the whiskey bottle over head of” and the Option need only specify who the DirObject is—in this example, that would be the Subject of the current Event, Slim Jim. Such specifications are placed in scripts written in Sappho.

However, there is an additional complication. Suppose the fight between Slim Jim and Tex escalates into a full barroom brawl, with everybody fighting everybody else. Suppose Gordon is one of the cowboys in the saloon; who’s he going to pick to fight with? There’s no obvious answer as there was in Tex’s case. So how does he pick his target?

Acceptable and Desirable

Here we come back to a concept first described in Chapter 18, “Scripting Languages”: Every choice is calculated with just two scripts. One script is for the acceptability of the choice, and another is for the desirability of the choice. Thus, for Gordon’s case, his Role will have an Option for “punch” that will have to specify its DirObject. The DirObject slot will then have one script for Acceptable and another script for Desirable. The Acceptable script will probably say something like “a candidate is acceptable if they are on this Stage, male, and not the Barkeep.” The Desirable script specifies how satisfying it would be to execute the Option Verb on the DirObject. In this case, it would probably be the opposite of how much Gordon likes the candidate. This means that the most desirable character to punch will be the one Gordon dislikes the most.

Acceptable and Desirable also apply to the Option as a whole. Perhaps Gordon is the local preacher, and he’s supposed to set a good example. Perhaps Gordon is too chicken to jump into the fight. So the Acceptable term for the Option as a whole might be something like “it’s acceptable as long as I’m not the preacher.” The Desirable term might read like this: “The Desirability is how much I hate the DirObject whom I intend to punch minus how cowardly I am.” If the Option passes the Acceptable test, then its Desirable value is checked. If the Desirable value is greater than zero *and* greater than the Desirable values for all the other Options, then that Option is selected. If the Option is selected, then it forms a new Plan for that character.

Once the engine has calculated all the reactions of all the characters to the Event in question, it goes back to the Plan List, looking for more Plans to execute. Once it has gotten to the end of the Plan List, the turn is over. The engine chalks up one more tick on the clock and starts over.

That wasn't so bad, was it?

Additional Odds and Ends in the Engine

There remain a number of minor features in the engine. One of these is *spying*, which reflects one of the major differences between drama and real life. In real life, people can arrange to speak together with confidence that they are not being overheard. Such is not the case in drama. The two characters lean over the table to each other, whispering their plan to foil the antagonist, then depart. The curtains behind their table part to reveal—the antagonist! Drama oozes with such unlikely circumstances because they permit interesting dramatic convolutions. Sure, it isn't realistic—this is drama, not simulation.

Another minor detail is a special data type, a *group*. A group is a set of characters defined to act as a social unit. For example, a storybuilder might build a group representing a gang, and then treat the gang as a single object. The leader could exhort the group as a whole, accept the cheers and huzzahs of the group, lead the group into a fight with another gang, and so forth. Group interactions are uncommon in drama; most interactions involve only two people. But there are a few interactions demanding group treatment, and so the engine provides for such situations.

The engine also has a provision that permits a character to abort a Plan at the last minute. Suppose, for example, that Mortimer the Merciless secretly pours the poison into Henry Hero's wineglass, and reacts to that event by establishing a Plan to hand Henry Hero the wineglass. But suppose that Mortimer is confronted by a servant who witnessed Mortimer's action and demands to know what he's doing. Exposed, Mortimer can abort his Plan to give Henry the wineglass.

Plot-Based Engines

The third fundamental type of engine is derived from experiments with story-generation programs. The first of these was *TaleSpin*, created by James R. Meehan way back in 1977. This was primarily an inference engine couched in a narrative context. Its basic plot was always the same: A character is presented with a challenge, knowledge of how that challenge might be met is presented to the character, and the character figures out how to meet the challenge.

Plot-based engines have come a long way since then. They are now able to define such abstractions as protagonist, antagonist, objectives for each of these, characters that can assist them in achieving their objectives, and other factors. They can then calculate a story based on these parameters.

This approach is noninteractive: The program accepts basic specifications and uses those specifications to compute its story. It is possible, however, to interrupt the story in progress at key points and alter one of the parameters. This provides a form of interactive storytelling.

So far, such engines have not been successful in delivering adequate interactive storytelling. They face three obstacles. First, they are at heart inference engines, and as such their central task is to determine reasonable behavior. They are fundamentally no different from the software that drives a Mars Rover across the surface of the planet. There's no reason why an inference engine has to infer rational behavior; it could just as well infer dramatic behavior. So far, however, the powerful techniques for producing rational behavior tend to overshadow the more appropriate and ill-developed techniques for dramatic behavior. This hurdle is none too high; we simply haven't quite gotten to it yet.

The second obstacle facing plot-based engines is the problem of inserting player interruptions into the ongoing story. At present this tends to occur only at dramatically salient moments, which come so rarely that the player feels more like a spectator than a protagonist.

A third obstacle with such engines is that they attempt to fit the player's actions into some overarching plot. This can produce rather odd results as the engine desperately tries to force the protagonist to follow the predetermined plot. It's rather like one of those (GPS-driven) journey planners telling the driver where

to turn. Imagine its response when the driver ignores half of its instructions. That's what some plot-driven schemes do.

This third obstacle has been attacked by means of adaptive-plot schemes. The engine attempts to follow one plot, but if the player's actions deviate too far from that plot, then the engine scans through all the other plots in its bag of tricks, finds the best fitting one, and then attempts to follow that plot. This works better, but so far most programs have plot databases too small to perform the job well.

Conclusions

The three broad classes of engines for interactive storytelling (event-based, clock-based, and plot-based) are surely not the only possibilities. I'm sure we'll see new ideas appearing in the future. My overall assessment is that the event-based systems are the easiest to implement, but they lack enough dramatic fidelity to deliver adequate interactive storytelling. I do not rule out the possibility that such engines might someday develop to a level of sophistication that permits fairly good interactive storytelling. After all, the internal combustion engine of the 21st century is a far cry from that of the late 19th century—and vastly more useful. But I am not optimistic about this strategy.

The clock-based engines are currently the leading technology; both *Façade* and my own *Storytron* technology use clock-based engines. The major difference between clock-based engines and event-based engines is that the clock-based engines easily handle multiple active characters, whereas the event-based engines are clumsy at that task. However, clock-based engines are correspondingly more complex to build. *Façade* solved the complexity problem by determined application of the KISS principle; *Storytron* failed because it could not cope with the complexity problem. I believe that clock-based engines provide our most likely shot at getting interactive storytelling working, but they'll need a lot more work before that happens.

The plot-based engines seem a dead end to me. Their basic DNA is noninteractive because they are inspired by the notion of following a plot rather than reacting to the player. Deep down inside, such engines don't remind themselves that "the customer is always right." Instead, they ask themselves, "How can we make the customer like what we have to offer?"

CHAPTER 20

High-Level Structures

IN CHAPTER 19, “ENGINES,” I briefly discussed plot-based engines, which use a top-down approach: Starting with the high-level concepts of plot and story structure, they attempt to figure out the low-level events that should emerge from the combination of the player’s decisions and the high-level plot concepts. So far, such techniques have not been successful. The bottom-up style, starting with individual character decisions and working upward from there, seems more promising. Bottom-up engines, however, suffer from meandering; there’s little impression of story structure. This chapter addresses that problem, suggesting some ideas for imparting a stronger sense of narrative development to the experience.

Plot Points

The simplest trick to providing some structure is to use *plot points*. In traditional storytelling, a plot point is any salient moment in the story; typically, it represents a turning point or sharp departure from the story's trajectory up to that point. In interactive storytelling, I use the term to describe an event that is forced onto the player by Fate. It is a preordained development that forces the developing story to go down a particular path. I recognize three broad classes of such plot points: timed, logical, and opportunistic.

Timed Plot Points

The simplest manifestation of plot points is timed plot points. At some predetermined time in the developing story, Fate triggers an event that forces the story to advance in a desired direction. For example, in my design for *Le Morte D'Arthur*, the story is divided into two major periods. In the first period, the player, as Arthur, establishes his rule over the fractious warlords under his leadership, building relationships and imposing his peace on their petty quarrels. Then Fate triggers the major plot point, in which Mordred raises the flag of rebellion and incites disaffected warlords to join his rebellion. After some political and military maneuvering, the two rivals meet in a final battle, which determines the outcome of the story.

Timed plot points are easy to implement; the author begins the story with a plan for Fate to trigger the plot point at a specified time. They work best as a way of forcing a conclusion to the story. There's no reason why a storyworld couldn't have multiple timed plot points. For example, let's consider the development of the story in *Star Wars Episode 4* as if Fate intervenes with plot points:

1. Fate brings R2D2 and C3PO to Luke Skywalker.
2. Fate induces R2D2 to run away, forcing Luke to chase him.
3. Fate brings Obi-Wan Kenobe to cross paths with Luke.
4. Fate kills Luke's guardians, leaving him no reason to stay on Tatooine.
5. Fate puts the Death Star in Luke's path, giving him the opportunity to rescue Princess Leia.
6. Fate brings the Death Star to the rebel base, forcing Luke to blow it up.

Viewed this way, it would seem that Luke never did much more than react to Fate's plot points.

Timed plot points suffer from a nasty weakness: What if the player isn't ready for a plot point? For example, what would happen if Luke dawdled around with the chicks at Mos Eisley spaceport before heading off with Obi-Wan? The Death Star would have showed up at Alderaan with no Luke in sight.

Logical Plot Points

The solution to the problem of the wrong-place, wrong-time player is the logical plot point: a plot point that is triggered only when certain preconditions have been satisfied. The Death Star doesn't blow up Alderaan until Luke is on his way to the planet. It doesn't head toward the rebel base until after Luke gets there. Such plot points will usually be preceded by timed plot points that push the player toward meeting the required conditions. If Luke spends too long whispering sweet nothings into the ear of some green-skinned beauty, Obi-Wan can grab him by the ear and drag him to the Millennium Falcon. If he lingers in the rubble field of Alderaan playing real-world asteroids, the princess can slap him and demand that they get to the rebel base.

Logical plot points are powerful but tricky to implement; the author must take into account the possibility that the preconditions might not be achievable. Worse, forcing the player to satisfy the preconditions robs the player of free will. If your player is being dragged by the ear and slapped around for not doing what he's expected to do, the player will resent the intrusions on the fun. Therefore, you should use logical plot points sparingly.

Opportunistic Plot Points

Opportunistic plot points are the most difficult to implement, but they're more subtle than logical plot points. If Luke wants to fool around with the chicks at Mos Eisley spaceport, let him; just use a plot point that gets him involved in some twisty subplot that eventually returns him to a desirable place in the story. The chick turns out to be an Imperial spy; she springs her trap, and Luke flees to the Millennium Falcon. The Falcon has already left without him, so he steals another ship and goes racing off after it, with the Imperial spy in hot pursuit. And so on and so forth.

Sometimes a storyworld can settle into a dull period, during which not much of interest happens. This can be the player's fault or the author's fault. Achieving a proper balance between an avalanche of crises and a snorer—in all possible situations—is almost impossible. One guardrail you can erect against such dull periods is what I call the *rabble-rousing plot point*. Let Fate monitor the `Momentous_Trivial` values of the verbs being executed, and if the running average of these values falls too low, toss a firebomb into the story. Let the player discover that he's actually the son of the villain (nah, nobody would believe that...). Or, bring word that a long-lost sibling is being held prisoner by some unsavory person.

In general, plot points are useful at the highest structural level of the storyworld. But you should never forget that plot points are artifices, clever tricks jammed into the storyworld to impose your intentions on the player. Used sparingly, they won't irritate the player, but the more plot points you use, the more linear the story becomes and the less free will the player can exercise. Use plot points as you would use strong spices in a meal: just enough to be interesting but not so much as to overwhelm the palate.

Goals

A goal is a more delicate means for kicking the player off his or her duff. In most stories, the protagonist's ultimate goal is quickly established; you'll need to do the same in your storyworlds. Usually the final goal is fairly obvious: marry Mr. Right, have an exciting romance, get out from the domination of an overbearing person, and so on. The tricky part lies in setting up secondary goals that, taken together, achieve the final goal. In an ideal storyworld, those secondary goals should be obvious to every player, but the limitations we now face make that unlikely. It's therefore worthwhile to have some means for prodding the player with suggestions.

You can use a supporting character to nudge the player. Traditional storytelling often uses such characters, especially when the protagonist is young. Sometimes the supporting character is an older person, acting as a mentor. Sometimes the character is a buddy who suggests things the player might not have considered. Sometimes the character is a subordinate or assistant character, such as a friendly robot or a younger character. Whatever ploy you use, remember that the supporting character can play both an active role (suggesting goals that the player might not have thought of) and a passive role (serving as a sounding board for the player's thoughts).

Anticipation

I learned a hard lesson in my early work with interactive storytelling. I was building my engine around an Arthurian storyworld. It included Arthur, Guinevere, and Lancelot, and in keeping with the conventions of the basic story, I provided for the possibility of Lancelot falling in love with Guinevere. While I was running one of numerous tests on the storyworld, Lancelot did indeed fall for Guinevere, and the two of them consummated their love in the forest. The verb `Make_Love_To` had a high value of `Important_Trivial`, and of course the gossip section of the engine calculated that, because Lancelot was Arthur's best friend and Guinevere his wife, Arthur would have high interest in any event involving both of them. Accordingly, when Lancelot next encountered Arthur, he searched through his headlines and found the headline that would most interest him: "Arthur, did you hear? I had sex with your wife!"

It is in moments such as this that you realize how astoundingly stupid an apparently reasonable algorithm can be.

The factor that I had left out was anticipation: Lancelot should have anticipated Arthur's likely reaction to the news that he had slept with Guinevere, and held his tongue. But the algorithm for anticipation, it turns out, was far more difficult to design than most of the other algorithms in the engine.

The root problem is twofold. First, to anticipate another's reaction to something, we must put ourselves in that person's place. Lancelot must, in effect, pretend that he is Arthur, and then imagine how he would feel in Arthur's shoes. This requires a considerable amount of computational legerdemain. The second problem is that we are attempting to determine Arthur's reaction in a subjunctive sense: *If* Arthur knew of the event, how would he react?

The Problem

I had already built into the engine the capability for characters to have emotional reactions to events; the code was already in place to ensure that, had Arthur discovered Lancelot's betrayal, he would become very angry. The problem was that this emotional reaction was part of the *real* reaction to the event. In other words, the only way to figure out how Arthur would react was to have him actually react, in which case, he'd actually kill Lancelot. The whole point of anticipation is to see these things coming *before* they happen. I needed to create a virtual

world *inside* the virtual world of the storyworld—a temporary imaginary world inside Lancelot’s head in which Arthur could react to the news, and Lancelot could observe his reaction in safety, and then act accordingly in his real world. Are you starting to get confused?

This is an instance of a general type of problem known in computer science as *recursion*. When a calculation must be carried out inside of itself (Lancelot’s reaction anticipates Arthur’s reaction inside of itself), we must use this confusing method. Unfortunately, recursive code places stringent requirements on the design of a program, and those requirements would have forced me to completely rewrite most of the engine code, so I had to find another way.

The Solution

The solution involved a great many convolutions, but inasmuch as they were strictly programming problems specific to my own design, I’ll not drag you through that muck. Suffice it to say that it took a lot of work. There are three possible approaches here. In the first, we calculate the listening character’s likely reaction (that is, the verb that the listener will likely choose upon hearing the news), and then determine whether that reaction is desirable from the point of view of the speaker. I rejected that approach for my own engine because that simply defers the consideration of the emotional reaction, which is the factor that really matters here. We could follow an endless logical tree of possible reactions, but the further we go down that tree, the hazier our calculations grow and the messier everything gets.

The approach I settled on goes directly to the emotional reaction. It calculates the emotional reaction of the listener and determines whether that emotional reaction is desirable from the point of view of the speaker.

This requires us to specify an emotional reaction for every verb. When somebody witnesses or learns of an event, they must be able to calculate their emotional reaction. Moreover, characters should react only to Subjects, not DirObjects, of events. The Subject is the character actually performing the action; the DirObject can’t be blamed for being acted upon. This saves the storyworld author the trouble of specifying the target of the reaction. The great majority of verbs will give you no problem with such a restriction. Those few verbs that might cause trouble are easily dealt with. In the case of Lancelot sleeping with Guinevere, you need only precede

that event with a verb by which Guinevere explicitly agrees to sleep with Lancelot. Arthur can react to this verb in forming his reaction to Guinevere, and react to the second verb in forming his reaction to Lancelot. This keeps the calculation of reactions simple: the character reacts to one person at a time.

Thus, when a speaker considers whether to reveal information to a listener, the speaker need merely evaluate the listener's emotional reaction to make his decision. The speaker adds up the changes in $p\text{Nice_Nasty}$ and $p\text{Honest_False}$ toward him. A positive sum means that the listener, upon hearing this news, will be more favorably inclined toward the speaker, and so the speaker will proceed to reveal the gossip.

However, this calculation applies only to the case in which the speaker is also the Subject of one or more events in the thread. In cases where the thread contains events in which other characters are the Subject, then the speaker should take into account the listener's emotional reaction toward those characters. Even if Lancelot were, by some odd fluke, unconcerned with Arthur's reaction toward him, he would still want to take into consideration Arthur's likely reaction toward Guinevere. The loss of $p\text{Nice_Nasty}$ and $p\text{Honest_False}$ that Arthur would feel toward Guinevere would be bad for Guinevere, and because Lancelot retains high $p\text{Nice_Nasty}$ for Guinevere, this is undesirable from Lancelot's point of view. Therefore, we must add a term to the calculation of the desirability of telling Arthur about the tryst thread. That term should be similar to the first term: the sum of the changes in Arthur's $p\text{Nice_Nasty}$ and $p\text{Honest_False}$ for Guinevere upon learning of the tryst. However, this sum should be weighted (not weighed: weighted) in proportion to Lancelot's $p\text{Nice_Nasty}$ for Guinevere. If his love for her is the maximum possible value, then Arthur's reaction toward Guinevere should be just as important to Lancelot as Arthur's reaction to himself. If his love for Guinevere is, say, only 50 percent of the maximum, then Arthur's reaction to Guinevere should receive half-weight.

Solution Challenges

Moreover, there is a negative side to these reaction calculations. Suppose that Mordred is the one in possession of the information about Guinevere's adultery. His calculation shows that Arthur will have a negative reaction toward Lancelot and Guinevere. Since Mordred hates both Lancelot and Guinevere, he calculates that Arthur's reaction to the news will be harmful to both of them, so he is highly motivated to reveal the news.

There is another complicating factor to consider: Suppose Lancelot prudently declines to mention his adultery to Arthur, but then encounters Mordred. Mordred's reaction is of no concern to Lancelot, so he might be tempted to reveal the fatal truth—which of course Mordred would be eager to learn so that he could use it to weaken the bonds that tie the trio together. It is conceivable that the same factors that would make Lancelot unwilling to talk to Mordred about anything would protect him in this case. But in the absence of explicit handling of this problem, the engine could still screw up.

The solution to this problem lies in calculating the likelihood of the information reaching Arthur from the listener. This can be a tricky calculation, but it uses the same algorithms that underlie the gossip system, so no new algorithms need be developed. The only new factor is the consideration of the likelihood that a character will betray a secret; this factor is calculated from that character's loyalty to the secret-holder relative to his loyalty to the character from whom the secret is being kept. Be warned—these calculations can become quite intricate.

Choosing Verbs

There is one other application of anticipation: the original decisions taken by characters. After all, Lancelot and Guinevere should have taken Arthur's likely reaction into account when they first decided to have a tryst, not just when they decide whether to talk about it. The basic approach is to add an automatic term to the inclination formula for the verb in question. That term is similar to the term used in determining the desirability of revealing the information, but there are two additional considerations. First, in the gossip case, we are considering a single character's likely reaction; in the case of a verb option that a character is considering, the likely reaction of every character must be taken into account. In other words, when Guinevere decides whether to sleep with Lancelot, she must consider the reaction of every character in the cast. Furthermore, she must then weight (not weigh: weight) each character's reaction by her own relationship with that character. All those reactions could be added up, perhaps, but it might be desirable to use nonlinear addition (say, weighting the reactions by the square of the other character's relationship). This would give her the information she needs to calculate that adultery might not be such a good idea.

The second consideration is the probability that the event can be kept secret. If, for example, Guinevere is contemplating adultery with Lancelot, she knows that it is very much in her interest as well as Lancelot's to keep it secret. If they therefore carry out their affair in secrecy, she can be confident that Lancelot will not reveal the secret and Arthur will never know—in which case the consequences of the choice need not apply.

Inference Engines

The toughest problem with anticipation is the inclusion of logical inferences. Suppose, for example, that Guinevere tells Arthur that she is going to spend the night at her mother's house, and Arthur later discovers that Guinevere did not spend the night at her mother's house. In your mind and mine, that is immediate grounds for suspicion; how are we to equip Arthur with the logic he needs to infer the possibility of hanky-panky?

AI researchers have developed a technology known as an *inference engine* to deal with this kind of problem. An inference engine is a general-purpose program that can take statements about reality that have been coded in the proper form and use those statements to draw inferences. In its simplest form, an inference engine does little more than arrange syllogisms into sequences. Advanced inference engines can carry out extensive logical calculations and collate many factors to draw surprising conclusions.

In practice, inference engines require a great deal of information to be coded into their databases before they can do anything useful. For example, let's take the example of Guinevere failing to spend the night at her mother's house. How would the inference engine come to the conclusion that something fishy was afoot? The starting point would be the following first two statements:

1. Guinevere said that she would spend the night at her mother's.
2. Guinevere did not spend the night at her mother's.

The inference engine must first detect the discrepancy between the two statements, an easy task. Having established that discrepancy, the inference engine must next attempt to explain that discrepancy. To do this, it must have at its fingertips a great deal of information about human behavior.

3. When there is a discrepancy between what somebody says and the truth, then that discrepancy is due either to a lie or to an extraordinary event.

So now the inference engine must examine two paths: Was it a lie or an extraordinary event? To determine which is more likely, the inference engine must use a statement rather like this:

4. People readily gossip about extraordinary events.

And this:

5. Guinevere did not gossip to Arthur about any extraordinary events.

To conclude:

6. Guinevere was not prevented from spending the night at her mother's by an extraordinary event.

Which, when combined with statement #3, leads to this conclusion:

7. Guinevere lied about spending the night at her mother's.

To carry out the calculations necessary for an inference engine, a prohibitively large amount of information about the world must be entered into the inference engine's database. And, in fact, AI researchers have learned that inference engines are practical only in those special cases in which the domain of applicable knowledge is small and tightly defined. For general purpose use such as we face in interactive storytelling, too much information is required to build a proper inference engine. Nevertheless, this remains an important field of effort. Someday, when we can afford to build such large databases of human behavior, inference engines will play a role in interactive storytelling technology.

Tree Analysis

Another approach to inferring the behavior of others is to explore the tree of possibilities that the character could follow. If the storytelling engine uses a definable system of links between events, then it should be possible to trace those links to determine what might happen. This will likely entail a considerable amount of programming effort; it is similar in overall structure to the tree-searching methods used in chess-playing programs. The good news is that this

kind of tree-searching will likely entail fewer branches at each branch point. The bad news is that each decision is intrinsically more complex than a decision in a chess game, because there are so many more variables involved.

The killer problem is that the decisions made at each juncture have to be made based on the calculating character's perception of the relevant variables, not their actual values. In other words, if we desire Mordred to scheme to bring Lancelot and Guinevere together for an illicit tryst, he must use his own estimates of their love for each other. If he is unaware of that love, then he cannot so scheme. This requires using third-person traits, described in Chapter 14, "Personality Models."

Third-person traits are especially difficult to manage, because they cannot be directly observed but must instead be inferred by the third person. If Tom is to assess Dick's $p_{\text{Nice_Nasty}}$ for Harry, he has three different sources of information:

1. **Tom's observations of Dick's actions toward Harry.** This requires Tom to work backward from Dick's behavior. For example, suppose that Harry greets Dick warmly and Dick returns the greeting warmly. Tom would have to examine the inclination formulae for the greeting verb to estimate the value of Dick's $p_{\text{Nice_Nasty}}$ toward Harry. Since the $p_{\text{Nice_Nasty}}$ value would likely be mixed in with other values, each of which has its own uncertainties, Tom's calculations would be so full of uncertainty as to be nearly useless.
2. **Tom's conversations with Dick about Harry.** The best way to learn somebody's feelings toward another is to talk about those feelings. Tom says to Dick, "Boy, that Harry sure is a jerk, isn't he?" and then pays close attention to Dick's response. This works well, unless Dick doesn't trust Tom, in which case his response may be guarded.
3. **Tom's conversations with other people about Dick's feelings toward Harry.** Tom says to Jane, "I just heard that Dick really hates Harry!" This will, Tom hopes, induce Jane to offer her own estimate of Dick's $p_{\text{Nice_Nasty}}$ for Harry. For this to work, the storyworld must include a great deal of such activity. Such activity makes up a good portion of many soap operas.

The many uncertainties arising from such considerations compel the inclusion of uncertainty measures for each third-person trait and the inclusion of uncertainty considerations in the calculations of the characters.

If this sounds impossibly complicated, you might wish to examine my bagatelle *Gossip*, a short, simple game focusing solely on the processes described in the second and third items in the list immediately above. I built it as a demonstration of how such processes can be modeled.

Nevertheless, all of this makes tree analysis immensely difficult; I recommend that the first generation of interactive storytelling engines refrain from using this technique.

Conclusions

Bottom-up engines are vulnerable to meandering, meaningless development. Storyworld authors must therefore apply schemes for tying things together and giving some direction to the story developments. This chapter has offered some suggestions on how to do this. At the highest level, don't think in terms of a plot that must be followed; think of your storyworld as a slow-moving river that naturally eases the player along without frog-marching him.

PART V

Wrapping Up

ACADEMICS HAVE ATTACKED the problem of interactive storytelling from a variety of directions. Here are some of the results of their work, followed by my predictions for the future.

Chapter 21: A lot of academic research focuses on interactive storytelling and exploring the problem from many different angles.

Chapter 22: What does the future hold for interactive storytelling? Nobody knows, but that doesn't deter me from making predictions.

This page intentionally left blank

CHAPTER 21

Academic Research

THROUGHOUT THIS BOOK I have struggled to present a chaotic and rapidly changing field in an organized manner. In the process of putting everything in neat little boxes, I have left quite a mess lying on the floor. In particular, I have written little about the progress made in the world of academia, even though the most impressive advances have been made by academics. This chapter attempts to redress that deficit.

One reason it is difficult to credit academics is the lack of continuity in the various research projects. To demonstrate the problem quantitatively, I compiled lists of all the authors appearing in a set of proceedings from four closely related conferences in 2001, 2006, 2009, and 2011. About 300 different authors contributed to these papers. Of these, only 34 names appeared in more than one year. In other words, more than 250 of these authors made a single appearance and were never heard from again. This reflects a problem I think is common in the field: People drop in, try something interesting, realize just how difficult the problems are, and wander off to try something else. There are a few dozen determined practitioners who have been slogging away for years; they are making steady progress, but much of the work in the field has a flash-in-the-pan quality. Many interesting ideas and projects come and go at a speed that would make Paris fashion designers dizzy.

One other problem slows progress in academic work on interactive storytelling: There's no real incentive to get it working. Remember, an academic career succeeds or fails on a scholar's output of papers, not on functional software. An academic could well have a highly successful career in interactive storytelling without ever actually designing anything that works. As long as the papers keep showing up in the journals, all is well.

One indication of the slight mismatch between academic performance and productive output is the way that papers are judged. I have twice had submissions rejected for the sole reason that my papers failed to cite the work of other academics. The referees had no objections to the content of my papers, and in fact each of the papers offered unique and worthwhile ideas. The problem was that I didn't fit my work into the corpus of existing academic literature on the subject; that flaw by itself disqualified my papers.



Sour grapes!

I understand the appropriateness of the rejection decisions; after all, academia operates under a system that requires each scholar to add to the existing structure of knowledge in the field. My papers didn't fit into that structure, so they didn't fit in a scholarly journal. I'm not complaining about academic standards; I'm pointing out that those standards cause the path of academic progress to diverge somewhat from the path of commercial progress. In their defense, I'll add that a number of academic journals have bent the rules to include my writings, and I have often been invited to speak at academic conferences.

Drama Managers

A major result of efforts in academic research into interactive storytelling was the creation of *drama managers*, a concept first articulated by Brenda Laurel in the 1980s. These bubbling cauldrons of software are similar to the concept I call Fate, but they take a different approach. Actually, they take a number of different approaches. Most of these approaches attempt to apply existing concepts from computer science to the problem of interactive storytelling.

The basic structure treats the drama manager as a software module that intervenes in the storytelling process, ostensibly to correct mistakes or keep the story on track. The problem these schemes face is this: “How do we calculate good drama?” After all, if you can’t calculate the difference between good drama and bad drama, you certainly can’t manage it to ensure good drama.

As I explained in Chapter 3, attempting to force the player along an intended plotline destroys the interactivity. The better drama managers therefore use a lighter touch, seeking only to ensure that the developing story maintains conflict or some other element of good storytelling.

My impression is that most drama managers suffer from violations of Crawford’s First Law: Always ask, “What does the user *do*? What are the verbs?” Once the drama manager has come to some conclusion regarding the changes required in the dramatic situation it is considering, it has no regular means of implementing those changes, so the intervention is designed specifically to that problem. The execution of such changes tends to have an abrupt or artificial feel.

I believe that Fate, described in Chapter 15, provides the most facile implementation of the concept of the drama manager. Because Fate is treated as a character by the engine, its operation fits more cleanly into the overall flow of the processing. Because Fate executes verbs like any other character, the implementation of Fate’s decisions is more natural. There is some clumsiness in the display of Fate’s actions to the player; we can’t tell the player, “Fate causes the bullet to ricochet harmlessly off the medal you’re wearing on your chest.” But we can certainly tell the player, “The bullet miraculously ricochets off the medal you’re wearing on your chest.”



So you’re going to have a verb called `Bullet_Miraculously_Ricochets`? Why can’t the player use that verb himself?

Because the player isn’t Fate. You’re right that Fate needs custom solutions just like any other drama manager, but Fate uses verbs that have the same basic structure and execution that any other verb has. The overall process is cleaner and easier to implement.

We cannot rule out the possibility that someday somebody will get more conventional drama managers working. A lot of work is being done and steady progress is being made. Don't forget, though, that we're still a long way from success with drama managers.

Goal-Seeking NPCs

A popular approach to the problem of interactive storytelling builds on artificial intelligence methodologies involving goal execution for robots. The software that drives the Mars rovers is a high-powered example of this kind of software. When the rover is given a target to approach, it must figure out the path it will take to get there. This requires it to survey the ground ahead, detect rocks, and figure out how to bypass them. Plenty of other considerations, such as the roughness of the ground along different paths, its slope, and the energy requirements for traversing the ground, must be taken into consideration. These goal-selection techniques have been around for decades now and have become quite intelligent. Such powerful tools naturally attract the attention of workers in interactive storytelling, and perhaps they could use these methods to drive the behavior of NPCs. If the characters can pursue dramatically interesting goals that conflict with those of the player, some real drama could emerge.

In practice, these efforts have not yet been fruitful. These AI techniques are based on clear, mechanical requirements that don't work well in the complicated world of interactive storytelling. They were developed to control the behavior of robots, and the jump from robot to dramatic character is much further than some of the more technical researchers realize. Characters in stories aren't robots, and they don't calculate the optimal strategy to meet their goals. Their mistakes are not due solely to lack of information; sometimes the most serious blunder a character makes is due to a personality flaw. If characters in stories pursued their goals according to the precepts of artificial intelligence, Neo would have taken the blue pill, Frodo would never have volunteered to take the Ring to Mount Doom, and James T. Kirk of the starship *Enterprise*—well, he wouldn't have done *anything*!

I remain skeptical that goal-seeking methodologies will successfully solve the central problems of interactive storytelling. At heart, they are just too mechanistic in structure to provide dramatically satisfying storytelling. However, there is some promising new research into rendering goal-seeking algorithms in harmony with human cognition.

Comme il Faut

The most impressive work to emerge from academia, in *My Arrogant Opinion*, is a new kind of interactive storytelling engine that goes under the sexy French name *Comme il Faut*, which means something like “conforming to the standards of propriety.” It’s an apt description, for the *Comme il Faut* system (hereafter referred to as “CiF”) does not confine its computations to a character’s internal drives; it also considers the social rules under which the character interacts with others.

In laudable fashion, the CiF creators went back to basics (as opposed to simply modifying an existing technology) to consider how to design their engine. They began with narrative and settled on what’s called Goffman’s Dramaturgical Analysis, a complicated system for describing the evolution of a story. This is not a computational system; it’s a way of chopping narrative into parts and labeling each of the parts. This provided the theoretical foundation for their ensuing work.

Next, after studying many works on social interaction, the CiF creators decided to use Berne’s Transactional Analysis, a system for describing social behavior in terms of a prescribed set of “games” available for social interaction. Berne provides a library of such games, defining the roles involved, the motivations required, and the options available to each actor in the game. Think of it as role playing with a dungeon master possessing no artistic flair and a PhD in psychology.

The CiF designers figured out how to implement Berne’s ideas in code, but they still weren’t finished. They then turned to Reiss’s Motivation Analysis, which posits 16 basic needs for every person. This serves as their personality model; a character is defined by the values (between 0 and 1) of each of the 16 basic needs.

After implementing all this in code, the CiF designers then integrated the two systems (Transactional Analysis and Motivation Analysis), along with a little goal-seeking code, a system for providing input and output, two bell peppers, and a dash of cinnamon. The end result is a complex and powerful engine that can address a wide range of social situations.

Most important, the CiF technology is both editable and extensible. Its operation affords a decent degree of custom manipulation to address particular storyworlds, and it can be improved indefinitely by adding more specifications. There’s no reason why, 20 years from today, CiF won’t lie at the heart of a major interactive storytelling technology, vastly expanded and improved.

Prom Week

CiF is not some laboratory freak. You can already see CiF in one published game, *Prom Week* (<http://promweek.soe.ucsc.edu>). This impressive work defies all my categories and doesn't fit into any other chapter. It does use a clock-driven engine, but that's about the only recognizable aspect of its design. *Prom Week's* greatest deviation from convention is its unique asymmetric interaction system, which uses a little language and a little point-and-click. Input from the player is all point-and-click. The player clicks a character to examine that character's attributes; clicking a second character specifies that person as the DirObject of the commands now arrayed around the Subject's head in a daisy-petal structure. The player selects one of the daisy petals, which triggers a short sequence of events (a "game" from the Transactional Analysis system) expressed primarily in the form of verbal acts between the Subject and the DirObject.

The theme of the storyworld is a high school prom and the social machinations leading up to it. There are 18 characters in the cast, each with a fixed initial personality and relationships. The player selects one of the cast members to play, and is given a set of goals to achieve in the week leading up to the prom. Goals are achieved by manipulating social relationships.

The storyworld benefits from the tightly confined set of behaviors available to the characters; this keeps the verb count down to about 60. Its narrow focus also helps clarify the design. It has been received very favorably.

Release the Kraken!

The CiF system is packed with potential. Now that I've built it up, it's time to tear it down. In playing a demonstration version of *Prom Week*, my character needed to convince a girl to like him. To accomplish this, I was advised to approach another girl who was hated by the first girl, and then insult that second girl. Having done so, I was now in the good graces of the first girl. This was, of course, the simplest possible demonstration of the operation of the CiF system. But this mechanistic style pervades the entire system. These are not dramatically interesting characters; they're little robots with buttons you push to make them smile or frown.

This problem arises from the overly Boolean style of the CiF design. The CiF engine occasionally uses numeric calculations, but they're quite primitive: simple

additions and subtractions. It should be possible, even easy, to provide more subtle arithmetic calculations, but the problem isn't in the technology, it's in the underlying thinking.

Computer scientists, I believe, see the world in Boolean, black-and-white terms. That's not due to simple-mindedness on their part; it's the result of years of experience in making brilliant logical contraptions out of simple Boolean parts. A computer is, at heart, a gigantic Boolean machine with literally billions of little yes-or-no components that work together to make movies, play games, and run the Internet. Confidence in the vast power of Boolean logic is certainly justified.

In theory, Boolean methods could solve any problem, but that doesn't make them ideal for every problem. It's true that Boolean methods are the rock bottom, theoretically fundamental basis for all computer processes. It's equally true that assembly language is the rock bottom, theoretically fundamental language for programming. Few people use assembly language because programming in assembly language is a huge, complicated task. You can get more work done with a high-level language than with assembly language. Numeric calculations are to Boolean calculations as high-level languages are to assembly language. Some calculations are just easier to work out with numeric approaches than with Boolean approaches.

Computer scientists have difficulty appreciating this because computer science has spent its entire life building up from the fundamentals. Scientists have spent decades immersed in the intricacies of Boolean techniques and, in the process, have internalized a style of thinking that has successfully solved most of the fundamental problems they face. They're now expanding into more numeric approaches, but such techniques are not as well developed as the highly polished Boolean techniques.

But with interactive storytelling, we face new problems of greater subtlety, problems that are not well addressed with the yes-or-no, black-or-white fundamentalism of Boolean calculation. Recall my explanation in Chapter 13 of the advantages of numeric calculations.

My second razzberry is directed at the overly technical approach underlying CiF. C'mon, folks, we're talking storytelling, not scholastics. You don't have to pull everything out of the research literature. Render unto the arts the things that are artistic, and unto computer scientists the things that are, err, computer sciency. I

think you would have done better to design one big, integrated system that captures what you believe to be the essence of storytelling. If you don't grasp the big picture of storytelling, whole and complete, in your mind, what in the purgatory are you doing working on interactive storytelling?

I have no doubt that the CiF technology will continue to improve and show better performance in handling subtler aspects of dramatic behavior. But as long as it is fundamentally a work of computer science, as opposed to a work of art, it will never bridge the uncanny valley.

The IRIS Project

From 2008 to 2011, the European Union, along with a host of technology companies, funded the Integrating Research in Interactive Storytelling project, known as IRIS. This ambitious project involved ten universities in seven countries coordinating their efforts to create a solid foundation for future research in interactive storytelling. The effort was broad-based, including everything from authoring tools to storytelling engines to evaluation techniques. Scores of scholarly papers were produced, along with final reports detailing the results. The final results did not, I think, achieve as much as the researchers might have hoped; the storytelling engine in particular was not up to my own expectations from such a stellar group of people. Still, they seem to have accomplished their basic goal of establishing a firm, integrated foundation for pursuing research on interactive storytelling. If they are able to build on this foundation, I think that we'll be seeing wonderful things coming from the European researchers.

Conclusions

Academic research got off to a slow start, but it has steadily gained momentum. I've seen fewer papers talking fecklessly about "narrative versus agency" and more papers delving into meaty issues. The academics are already way ahead of anybody else, and will surely make even greater progress in the future.

CHAPTER 22

The Future

SO, WHERE IS INTERACTIVE STORYTELLING HEADED?

How soon can we expect to see it blossom into a genuine medium of artistic expression? When will it become commercially viable? And what will it look like? Answering questions like these is a surefire way for me to make a fool of myself—but I've already done that, so I have no reservations about proceeding. Here are my predictions. Save this book; it'll be good for a laugh in 20 or 30 years.

It Will Be Big

When we finally start delivering decent interactive storytelling, the new industry will leave video games in the dust. Interactive stories won't eliminate video games, but the difference in scale will be dramatic. To get an idea of the difference, walk the aisles of your local supermarket. Compare the amount of aisle space devoted to candy and ice cream with the amount devoted to other foods. Candy and ice cream are like video games: They're fun food. They pack an intensely pleasurable gustatory wallop. Other foods aren't as intensely pleasurable, but they offer more subtle tastes and greater diversity. As children, we love the fun taste of candy and ice cream. But, after a few million calories worth of fun food, we start to yearn for greater subtlety and diversity. We consume fruit, vegetables, Chinese food, German food, French food, Mexican food, Thai food, Indian food, Japanese food, English food (such as it is), Italian food, Greek food, barbecue, chili, breads, cheeses, wines, spirits, fish, pasta, and chips—and we flavor these things with all sorts of spices and sauces. As adults, we still enjoy candy and ice cream, but we spend more money on the other foods. It'll be the same way with interactive storytelling—vis-à-vis video games.

Video, including both television and movies, provides another example. There's a good supply of "fun video" in cartoons, comic-book movies, and action movies. But there's a lot more video that doesn't fit into the fun category: romantic comedy, drama, mystery, police shows, reality TV, news, science shows, documentaries...it's a long list. Sure, kids and teenagers spend most of their video time on the fun stuff, but when you look at the big picture, fun video occupies a niche in a vast universe of video appealing to the broad range of human tastes. Interactive storytelling addresses a much larger range of human tastes than video games.

I'm not finished beating this dead horse; let's talk about literature. Go into a bookstore and walk the aisles. You'll find a comic book section, I'm sure, but it will likely occupy a small corner of the bookstore. Much more space will be devoted to less fun forms of literature: cookbooks, romance novels, travel books, science fiction, self-help books, thrillers, science, humor, and much more. Interactive storytelling doesn't cover all these subjects, but even if you count only fiction in the bookstore, you'll still find a lot more entertaining fiction than fun literature.

These days, video-game people are like kids in candy stores. They see the grand diversity of candies on sale and think their world is infinite. Sure, there's a huge panoply of products catering to their taste for fun stuff. But it's just one galaxy in a vast universe of products meeting a range of tastes that boggles the minds of the inexperienced. Interactive storytelling doesn't span that universe, but it comprises a lot more galaxies in the video-game cosmos than the fun ones do.

Negative Predictions

It's easier to make negative predictions than positive ones (but it's just as easy to make a fool of yourself with negative predictions). Here goes, starting with the ones I'm most confident of.

Interactive Storytelling Won't Come from Existing Fields

I say this with confidence: Existing fields of study or professions will not produce great interactive storytelling. The video-game industry is too deeply caught in its own rut. The industry has a firmly established business model dedicated to selling testosterone-soaked products. On top of this, the public has a firmly established market perception that video games should be testosterone-soaked. The video-game industry is too big to focus attention on a nascent market such as interactive storytelling; they figure they can wait until it has grown to a respectable size before moving in and taking over from the small fry who got it rolling. That's what IBM, CDC, DEC, and other big computer makers thought about personal computers.

In much the same manner, Interactive Fiction and Role-Playing Games will not evolve into interactive storytelling. They'll continue doing what they already do, which is not interactive storytelling.

Interactive Storytelling Won't Come Directly from Academia

The academic world is making the greatest contributions to the development of interactive storytelling, and its efforts will expand. However, I believe the emphasis on publishing scientific papers rather than building working technologies will prevent the development of commercially viable interactive storytelling technology by academia.

I do not dismiss these academic efforts as a waste of time; I consider them necessary to the evolution of the medium. The big successes will come from the students of these academics, who will be equipped with the basic mental tools to tackle the problems in their own ways. After leaving school, they'll come up with lots of wild-and-crazy schemes for building interactive storytelling, most of which will fail miserably. But a few of those wild-and-crazy schemes really will work, and those will form the kernel of the new industry.

Positive Predictions

Here's where I really go far out onto a limb. Predicting what will happen in the future is a lot harder than predicting what won't happen. Accordingly, these positive predictions are not as specific as the negative ones.

Women Will Be a Major Driving Force

Since 1980, games people have been talking about “games for women.” The first manifestation of this was the game *Ms. Pac-Man*, sometimes called “Pac-Man with a bow on his head.” The sarcastic comment from developers about “games for women” was “Let's give them pink guns.” Everybody recognizes that women constitute a huge market that remains beyond the reach of games. Despite more than 30 years of hand-wringing over the matter, little has been accomplished. The reason for the failure should be obvious: Women are more interested in social reasoning than spatial reasoning (see Chapter 1). Games are relentlessly spatial in nature and hopelessly asocial.



Are you crazy? What about social games, all the rage these days?

Remember Crawford's First Law of software design: Always ask, “What does the user *do*? What are the verbs?” The verbs in social games are primarily mechanical and spatial in nature. The social interactions are not the stars of the games; they play a supporting role.

The most important difference between interactive storytelling and conventional games lies in social reasoning. Interactive storytelling incorporates social reasoning—and games don't. Women generally prefer entertainment based on social reasoning over entertainment based on killing things. That's why interactive storytelling will appeal to women much more than games do.

Interactive Storytelling Will Come from Indie Developers

Small-fry independent developers will break the commercial ground on interactive storytelling. While I'd put most of my money on start-ups created by students graduating from interactive storytelling programs, I wouldn't rule out other groups. Hey, if I stopped writing books and resumed working on my own stuff, perhaps even *I* will get something working!

Storyworlds Will Have Weak Cosmetics, Initially

Because interactive storytelling will be the work of independent small-fry companies, it will not be stuffed to the gills with expensive eye-popping visuals. It is possible that the first good commercial efforts at interactive storytelling will use little more than plain text augmented with facial expressions.

Interactive Storytelling Will Likely Come from Europe

Nobody learns from success; failure is the best teacher, which is why I have learned so much about interactive storytelling. Americans dominate the video-game biz. They make billions of dollars that way, and they're certainly not going to risk money on wild-blue-yonder ideas such as interactive storytelling when they have a lock on profits in video games. The Europeans, on the other hand, are lean and hungry. They know they can't beat the Americans in the video-game biz, but they've got plenty of talent and energy. Furthermore, they aren't handicapped by the Two Cultures prejudice that affects Americans.

A rough indicator of European dominance is the number of papers published in research publications. I scoured the proceedings of the leading conference about interactive storytelling (the *International Conference on Interactive Digital Storytelling*, or ICIDS) and found 52 papers from Europe, 35 from North America, and 15 from elsewhere around the globe.

The IRIS project (see Chapter 21) is an ambitious effort by the European Union to develop interactive storytelling technology. The well-funded project encompasses ten universities in seven countries. Nothing like this exists in the United States.

Lastly, my conversations with many Europeans convince me that they have the right attitude—the right stuff—to do interactive storytelling. I hope these comments encourage European students to go for it and incite American students to prove me wrong.

Time Frame

Now for the biggest question: How long will it take? I have been overly optimistic; here's what I predicted in the first edition of this book in 2005:

...the first genuine interactive storytelling product will make an appearance sometime between 2006 and 2008...by 2010 we'll definitely have a nascent industry, with several small companies jockeying for position and developing the market. The decade until 2020 will be devoted to market consolidation as big companies buy out little companies, the market matures and expands enormously, and interactive storytelling becomes a mass medium.

Pretty far off the mark, eh? What I have learned in the intervening years is that interactive storytelling is immensely difficult, and people have been slower to understand the basic concepts of interactivity than I had expected. At my Phrontisterion conferences, I often find myself defending basic principles.

Perhaps the most sobering evidence is provided by the results of the 2012 Phrontisterion conference. After discussing the many technological strategies under investigation, the group concluded there was no basis for recommending any particular approach over any other at that time. They could not agree on a definition of interactive storytelling. They could not even agree on basic terminology, with one exception: Everybody agreed that the active agents in interactive storyworlds should be called *characters* instead of *actors* or *agents*. Only 11 people attended—some of the top minds in the field—and even this small of a group could find little common ground. With so much confusion about what we're doing, it is obvious that interactive storytelling is still in the early stages of development. We're nowhere near ready to be thinking in terms of implementation; we'll need another decade to iron out the conceptual issues.

I therefore take a more pessimistic stance in this second edition. My guess is that we'll see many precursors and flawed attempts at interactive storytelling, but nothing that we can agree constitutes the real thing at a commercial level until at least the year 2020.

Conclusions

I conclude with some recommendations for you, the reader. The only reason you have put up with my sesquipedalian baloney throughout this book is that you would like to learn about interactive storytelling, perhaps with an eye to building something.

On the bright side, I can promise you that this is the new frontier of entertainment software. Video games are dominated by huge soulless corporations that are only interested in making money this quarter. A creative spirit has difficulty finding satisfying work in such places. But interactive storytelling is vaguely like video games were when I joined Atari in 1979. Back then, nobody knew what they were doing, so headstrong clods like me just made it up as we went along, and somehow we got an industry going. I swear, if back then I had to compete with some of the talent in the industry now, I would not have lasted long. My success is due to getting started in a new and therefore easy field. You face a similar situation with interactive storytelling.

The monetary rewards will be analogous to those of the early days of video games. A lot of pioneers of the video-game industry got filthy rich, something that rarely happens these days. If you're one of the successful pioneers, and you don't screw up too badly, you've got a good chance of getting rich enough to embarrass old fogies like me.

It won't be easy; the early days of video games were also a bloody time. Atari collapsed and 10,000 people lost their jobs. You'll have to find answers for many of the problems I've sketched here, and those problems are immensely difficult. If they weren't killer problems, I'd have solved them already! You can build on the work others have done, and perhaps make the breakthroughs that will make interactive storytelling come alive.

Make no mistake: The odds are against you. This book was published because there are thousands of likely readers, and you're in competition with every one of them.

Despite these challenges, interactive storytelling remains the best field for those with the talent and energy to make their mark in a world dominated by money and connections.

Go get 'em!

For Further Reading

Since the first edition of this book, a great many sources of information have appeared. Here's a list of the material I'm aware of, along with My Arrogant Opinion of their merits.

Books

Character Development and Storytelling for Games, by Lee Sheldon, 2004, ISBN 1-59200-353-2

This is a great book about storytelling from a genuine expert. You should definitely read it to improve your understanding of the storytelling process. However, it doesn't have anything to say about interactive storytelling. Its focus is on building conventional stories into conventional games.

Interactive Storytelling for Video Games, by Josiah Lebowitz and Chris Klug, 2011, ISBN 978-0-240-81717-0

This book has received excellent reviews. It's great if you want to slip a low-interactivity story into a conventional game, but it doesn't cover much of the material in this book, *Chris Crawford on Interactive Storytelling*.

Hamlet on the Holodeck, by Janet Murray, 1997, ISBN 0-262-63187-3

The classic work on narrative and computers is 15 years old and still just as pertinent to today's problems as it was when first published.

Expressive Processing, by Noah Wardrip-Fruin, 2009, ISBN 978-0-01343-7

An exposition on computer games, narrative, and their relationship.

On the Origin of Stories, by Brian Boyd, 2009, ISBN 978-0-674-03357-3

The first half of this book is great; it presents an excellent integration of evolutionary psychology with narrative theory, suggesting that art is a form of cognitive play.

Computers as Theatre, by Brenda Laurel, 1991, ISBN 0-201-51048-0

This is a strange book by a brilliant woman with a doctorate in theater who has worked at computer companies for many years and combines her two fields of expertise in startling ways. Not quite germane to interactive storytelling, but interesting nonetheless.

Tell Me A Story, by Roger Schank, 1995, ISBN 0-8101-1313-9

The author is an eminent computer scientist who devoted much of his career to applying narrative concepts to the development of artificial intelligence.

The Literary Mind, by Mark Turner, 1996, ISBN 0-19-510411-0

This is a solid work, but it's not light reading; give yourself time to reread portions. Lots of great ideas.

Narrative: A Critical Linguistic Introduction, by Michael J. Toole, 1988, ISBN 0-415-00869-7

This is even heavier duty than *The Literary Mind*. I recommend it only to die-hard students wanting to learn everything they can about storytelling.

On Narrative, Edited by W.J.T. Mitchell, 1980

Super-heavy-duty academic theory. Read this to prove your prowess.

The Thirty-Six Dramatic Situations, by Georges Polti, 1977, ISBN 0-87116-109-5

Originally published in 1921, this book is now a classic. It presents 36 basic plot structures, along with variations on these. Computer science people have longed to use this system as the foundation for a plot-generating system, but so far nobody has succeeded.

Morphology of the Folktale, by Vladimir Propp, 1996, ISBN 0-292-78376-0

Propp wrote this back in the 1930s, and it has attracted a lot of favorable attention from people working on interactive storytelling, because it digests an entire class of stories (Russian folktales) down to a simple system of components that can be linked together (it seems) to generate folktales. Although a number of researchers have used it indirectly with some success, so far nobody has succeeded in getting good results from a direct application of Propp's system.

The Folktale, by Stith Thompson, 1977, ISBN 0-520-03537-2

First published in 1946, this represents a thorough and detailed study of folktales. It includes a concise index of tale types and another concise index of tale motifs that could conceivably form the basis of a dataset for story generation systems. The full index occupies six volumes and is available from the Indiana University Press; the electronic edition is cheaper.

The Uses of Enchantment, by Bruno Bettelheim, 1989, ISBN 0-679-72393-5

This is about the role that folktales play in childhood, which is not important for interactive storytelling people, but has some interesting ideas.

The Human Face, by John Liggett, 1974, ISBN 0-8128-1754-0

This is important! The face is crucial to human communication. If you're thinking about including graphics in an interactive storytelling system, you absolutely *must* begin with the human face.

The Art of Interactive Design, by Chris Crawford, 2003, ISBN 1-8864-1184-0

This is the most brilliant book ever written in the history of the universe!

The Cambridge Encyclopedia of Language, by David Crystal, ISBN 0-521-55967-7

This is a fascinating tour of language in all its dimensions, organized as a set of two-page essays.

The Language Instinct, by Steven Pinker, 2007, ISBN 0-688-12141-1

This is a classic work that delves into the way the human mind creates and uses language.

Homo Ludens, by Johan Huizinga, 1950, ISBN 0-8070-4681-7

You must read this book.

Conference Proceedings

The following are all conference proceedings from a variety of academic conferences held over the years. Generally speaking, the older they are, the less useful they are. A lot of good ideas are sprinkled through these pages, but they're written in academic English, an arcane dialect known for its impenetrability.

Virtual Storytelling, Proceedings of the ICVS Conference in Avignon, France, 2001, ISBN 3-540-42611-6

Proceedings of the Third International Conference on Technologies for Interactive Digital Storytelling and Entertainment, 2006, ISBN 3-540-49934-2

Proceedings of the First Joint International Conference on Interactive Digital Storytelling, 2008, ISBN 3-540-89424-1

Proceedings of the Second Joint International Conference on Interactive Digital Storytelling, 2009 ISBN 3-642-10642-0

Proceedings of the Fourth Joint International Conference on Interactive Digital Storytelling, 2011, ISBN 3-642-25288-4

Internet

As always, we can expect the Internet to change, so these URLs might not go anywhere, but they were good in late 2012:

<http://xenia.media.mit.edu/~mueller/storyund/storyres.html>

An extensive collection of links to academic sources on just about everything that an interactive storytelling person might possibly want to know.

<http://stagconf.com>

A site devoted to a conference in 2011 in Vienna on storytelling and games, which includes video of some of the lectures.

<http://gamesandnarrative.net>

A blog by a group of researchers working on interactive and computer game narrative.

<http://grandtextauto.org>

A strong blog about lots of things, some of which are relevant to storytelling.

www.erasmatazz.com/TheLibrary/Library2.html

The library on my personal website, which contains a collection of essays about game design and interactive storytelling.

www.storytron.com

The website for my interactive storytelling game engine Storytron, launched in 2010 but too complex to succeed. There is lots of material explaining exactly how the technology works, but the technology itself is no longer available. By the time you read this, the next generation of my technology may be up and running.

www.inform7.com

The website for the Inform 7 development environment for interactive fiction.

www.tads.org

The website for the TADS development environment for interactive fiction. TADS is more powerful than Inform 7 in its programming abilities, but it's also more difficult for people without some programming skills. It's free.

www.choiceofgames.com

A commercial operation that has its own approach to something very close to interactive fiction.

www.storybricks.com

This is a fascinatingly fresh new approach to a massively multiplayer online game. It has a great point-and-click interface for building the rules for the storyworld. It's too early to say whether this system will work.

www.en.wikipedia.org/wiki/Aarne-Thompson_classification_system

The Wikipedia entry on the Aarne-Thompson system for classifying folktales.

By the time you read this, there will probably be half a dozen new sources that are just loaded with great information. The Internet giveth, and then it giveth some more, and then some more...

Index

Numbers

3D graphics, 90, 107. *see also* graphics

A

abstraction

- Comme il Faut system, 315
- computer modeling and, 163–166, 169–171
- constructing storyworlds, 76–78
- finance and, 71–74
- higher levels of, 74–76
- justice and, 68–70
- science and, 70–71
- translating principle into instance, 78–80

academic research

- difficulties in, 311–312
- drama managers and, 312–314
- goal-seeking NPCs and, 314
- interactive storytelling from, 321–322
- IRIS project, 318
- limitations of CiF system, 316–318
- Prom Week* and, 316

Acceptable scripts, storytelling engines and, 293–294

accordance personality variable, 199

addition

- bounded numbers and, 185, 189
- converting Boolean OR to, 167–168, 170–171
- multiplication vs., 173–174

Adventure (game), 128, 250

adventures, text. *see* text adventures

Affection_Disdain variable, modeling and, 166–168

agency. *see* interactivity

Aha! insights, 15

Alexander the Great, 85

algorithmic design

- for anticipation factor, 302–304
- Boolean vs. numeric, 178–179
- bounded numbers in, 181–183
- function analyzer and, 279–281
- overruns in, 180–181
- scoring performance and, 225–226

All of Me (movie), 104, 205

all-encompassing laws, 69–70

anatomy drawing, 94–96

AND (Boolean operator)

- converting to numeric multiplication, 167–168, 170–171
- function of, 190
- OR vs., 173–174

Angry_Fearful variable, 205

anticipation of reactions

- example of, 301
- inference engines and, 305–306
- problem of, 301–302
- solution challenges, 303–304
- solution for, 302–303
- tree analysis and, 306–308
- verb options for, 304–305

applause, for player's performance, 223, 224–225

Apple Inc., 250–251

arcade games, 89–90. *see also* video games

Aristotle, 21

Aroused_Disgusted variable, 205

artificial intelligence (AI) techniques

- involving goal-seeking NPCs, 314
- linguistic interpretation and, 252

artistic community

- cultivate logical thinking, 109–110
- learn programming, 111
- pattern-based thinking and, 105
- scientific community vs., 103–104
- second-person insight in, 30–31
- standardize terminology, 110–111

Ascetic_Sensual variable, 203

assembly language

- for computer programs, 137
- difficulties of, 317

associative memory, 32–34

Audience variable, in verb data structure, 230

audiences

- ego control and, 35–36
- plot interaction and, 52
- for role-playing games, 149
- second-person insight and, 30–31, 34–35

audio, for *Façade* storyworld, 255

Austen, Jane, 57

- authorial control
 - interactivity vs., 50–54
 - MMORPGs and, 147–148
- B**
- babel fish puzzle, 130–131
- backstory, in HistoryBooks, 234–235
- Balance of Power* (game), 256, 257
- BASIC programming language
 - concepts taught with, 137
 - interactivity speed with, 38–39
- behavior(s)
 - Fate monitoring, 212–213, 214
 - Inform 7 and, 136
 - modeling. *see* modeling behavior
- behaviorally discriminating personality models, 196–197
- bell curves, distribution of human qualities and, 183–184
- Berne's Transactional Analysis, 315
- bill of exchange, 72–73
- biochemistry, 70–71
- bits, algorithmic design and, 178–179
- blend operator
 - inclination formulae and, 206–207
 - one-factor use of, 188
 - two-factor use of, 189
 - for weighting factors, 185–188
- Bless the Child* (movie), 62
- Blue Lacuna* (interactive fiction), 139
- Bobrow, Daniel, 248
- Boolean bias of Inform 7, 138
- Boolean calculations
 - changing to numeric, 167–168
 - for event-based engines, 288
 - features of, 189–192
 - limitations of, 316–317
 - numeric vs., 177–179
 - of relevant variables, 170–171
 - in scripting languages, 271
- bottom-up engines
 - anticipation factor in, 301–308
 - goals and, 300
 - limitations of, 297
 - plot points and, 298–300
- bounded numbers
 - bell curves and, 183–184
 - blend operator with, 185–189
 - computer implementation of, 184–185
 - in modeling algorithms, 181–183
 - scripting languages and, 271
- brain function
 - environmental knowledge and, 5
 - language and, 5–6
 - mental module interaction in, 7–9
 - social reasoning and, 5
 - thinking mechanisms and, 6–7
 - in visual-spatial module, 4–5
- branching trees
 - dialogue trees and, 118–119
 - for event-based engines, 288
 - foldback schemes in, 121–122
 - geometric growth of, 115–117
 - interactive fiction and, 139
 - maze trees and, 120–121
 - with state variables, 123–125
 - text formulation and, 256–258
 - in video game stories, 154–155
- bytes, algorithmic design and, 178–179
- C**
- C# programming language, 137, 271
- Cameron, Andy, 51
- canned audio, for *Façade* (storyworld), 255
- capabilities of Inform 7, 135–137
- Carmack, John, 52
- causal relationships, mathematics and, 96–97
- CD-ROMs, multimedia craze and, 88–90
- Chaplin, Charlie, 21
- characters
 - event-based engines and, 288
 - Fate monitoring, 212–213
 - in interactive storytelling, 324
 - reactions of. *see* anticipation of reactions
 - in scripting languages, 272
- Chaste_Licentious variable, 203
- chemistry, 70–71
- Choice of Games, 183
- choices
 - interactivity and, 40–42
 - in stories, 20
- CiF. *see* Comme il Faut (CiF) system
- Clausewitz, Carl von, 40
- clock-based engines
 - Acceptable/Desirable scripts and, 293–294
 - overview of, 288–289
 - Plan execution by, 290–291
 - Plans and, 290
 - real-time vs. turn-sequenced, 289
 - Role execution by, 292–293
 - Roles and, 291–292
- coinage, abstraction and, 72

colored text, scripted languages and, 272
 Comme il Faut (CiF) system
 features of, 315
 limitations of, 316–318
 Prom Week (game) and, 316
 communication of gossip. *see* gossip
 competitive advantage, of interactivity, 42–43
 completeness, of personality models, 194
 computer graphics. *see* graphics
 computer modeling for interactive storytelling. *see*
 modeling behavior
 computer programming
 assembly language for, 137
 for bounded numbers, 184–185
 high-level architecture for, 138
 noun prejudice in, 86
 noun-verb relationships in, 82–83, 84
 scripting languages and. *see* scripting languages
 vertical code layout in, 279–281
 conciseness
 of personality models, 194
 through orthogonality, 194–196
 conflict
 Fate monitoring, 212
 role of, 18–19
 conflicting laws, justice and, 68–69
 Consequences, Plan execution and, 290
 consequences, player behavior and, 220–223
 constipated stories, video games and, 152–153
 conversation
 computers and, 245–246. *see also* language(s)
 interactivity in, 28–30
 cosmetic factors. *see also* graphics
 in games vs. stories, 48
 in interactive storytelling, 323
 cost, of video game stories, 156–158
 Crawford's First Law of Software Design, 42, 54,
 90, 313, 322
The Creation of Adam (artwork), 36
 creative options, 63
 creativity, role-playing and, 143–145
 credit cards, abstraction and, 73–74
 cultural knowledge, storytelling and, 9–10
 curiosity, verb thinking and, 91–92
 cut scenes
 development of, 106
 inclusion of video in, 107
 Cyc project, 248

D

da Vinci, Leonardo, 94–95
 Daglow, Don, 141
 data capacity, 86–87
 data structure, verb, 229–230
 data types, as objects, 270
 decision making
 balanced choices for, 55
 frequent but minor, 55–57
 significant, 54–55
 unsatisfying outcomes and, 58–59
 defaults, fully functional, 275–276
 deities, environmental knowledge and, 7–8
 denial, lies and, 241
 depth, of interactivity, 39–40
 Desirable scripts, storytelling engines and,
 293–294
 determinism, free will vs., 53–54
 development environments
 Inform 7 and, 133–138
 for text adventures, 129
 dialogue trees, 118–119
 dictionaries, for interactive fiction, 132
 digital cameras, speed and, 39
 digital storytelling, 44
 directed graphs
 for event-based engines, 288
 features of, 123–125
 DirObject(s)
 creating languages and, 266
 handling verbs and, 230
 with intent to harm, 240
 NPC relationship to, 239
 reaction to Events and, 291
 Role requirement for, 292
 sentence structure and, 231
 disabling information, HistoryBooks and, 234
 division, bounded numbers and, 185
 divisiveness, Two Cultures and, 104
 Dominguez, Facundo, 281–282
Doom (game), 52, 107, 223
Dragon's Lair (game), 89–90, 219
 drama managers
 limitations of, 312–314
 story development and, 210
 dramatic conflict, Fate monitoring, 212
 dramatic time, 26
 Dramaturgical Analysis, 315
 dropping the fourth wall, Fate and, 218
 dullness, monitoring of, 211
Dungeons & Dragons (game), 141

E

economics, verb thinking and, 82–83, 85
 editing, menu-driven, 276–277
 ego control, interactivity and, 35–36
 Einstein, Albert, 71
 Eliza computer program, 248–250
 empathy, second-person insight and, 31–32
 enabling information, HistoryBooks and, 234
 Energetic_Tired variable, 205
 engines, storytelling
 bottom-up, 297
 clock-based, 288–294
 event-based, 288
 plot-based, 295–296
 entertainers, second-person insight and, 31
 environmental knowledge module
 function of, 5
 pattern-based thinking and, 7
 social reasoning and, 7–8
 environmental manipulation, Fate and, 214–215
 Environmental Protection Act, 69–70
 equations, modeling behavior and, 163–166
 error(s)
 handling, scripting and, 278–279
 round-off, 179
 ESP-based (extra sensory perception-based)
 language, 258–259
 eucalyptol, 145
 Europe, interactive storytelling in, 323–324
 evaluation, of player's performance, 223–224
 event-based engines, 288
 Events
 Consequences for/reactions to, 290–291
 HistoryBooks and. *see* HistoryBooks
 in interactive storytelling, 232
 recording, 235–236
 scripting languages and, 272
Excalibur (game), 106
 Exclusive-OR (Boolean operator), 190
 Execution_Time variable, in verb data structure, 231
 exponentiation, weighting factors and, 172–173
 extra sensory perception-based (ESP-based)
 language, 258–259

F

Façade (storyworld)
 facial expression in, 254
 limitations of, 255–256
 linguistic system for, 252–253
 overview of, 251

 real-time play in, 64, 289, 296
 tight constraints of, 253–254
 facial expression, *Façade* storyworld and, 254
 factory work, verb thinking and, 82–83, 84
 Fairmath, 183
 FALSE Boolean flag
 handling verbs and, 230
 indicating lies, 240
 Fate
 drama manager comparison, 313–314
 dropping the fourth wall, 218
 environmental manipulation and, 214–215
 goal injection/personality shifting and, 215
 interstitial stories and, 216–217
 listening function of, 210–213
 overview of, 209–210
 plot points and, 215–216
 reacting to Events, 290, 291
 scoring by, 223–226
 thinking function of, 213–214
 feature bloat, 41
Final Fantasy (movie), 21–22
 finance, abstraction and, 71–74
 first-person personality traits, 198–199
 five factor personality model, 200
 floppy disks, 88, 90
 flow control
 branches for, 274
 loops and, 273–274
 standardized sequencing for, 275
 foldback schemes
 branching trees and, 121–122
 with state variables, 123–125
 text formulation and, 256–258
 FORTRAN programming language, 137
 free will, determinism vs., 53–54
 Freitag, Rosa, 64
 fully functional defaults, in scripting process,
 275–276
 fully player-driven stories, video games and, 156
 function analyzer, in scripting process, 279–281
 functional significance, of choices, 41
 future, of interactive storytelling, 320–325

G

games. *see* video games
 Gardner, Martin, 15
 gender bias, in styles of thinking, 105, 322–323
 Generous_Greedy variable, 203
 geometric growth, of branching trees, 115–117

Gilligan's Island (TV show), 212
 Glassner, Dr. Andrew, 51
 goals
 injection by Fate, 215
 seeking, AI methods for, 314
 in storyworlds, 300
Godzilla, 48
 Goffman's Dramaturgical Analysis, 315
The Gold Rush (movie), 21
 gossip
 anticipating reactions to, 301–304
 communication of, 236–237
 grapevine and, 242–244
 initiation of, 237
 intent to harm in, 240–241
 lies and, 239–240
 mechanism for, 238–239
 for misdirection/denial, 241
 motivation for, 238
 recording Events and, 235–236
 secrets and, 241–242
 solution challenges, 303–304
 verb choice and, 304–305
Gossip (game), 308
 grapevine, in interactive storytelling, 242–244
 graphic adventures, 128–129
 graphical user interfaces (GUIs), as verbs, 227
 graphics
 3D, 90, 107
 development of, 22–23
 in games vs. stories, 48
 interactive storytelling and, 323
 Griffith, D.W., 99
 gross trust, Fate monitoring, 211
 groups, storytelling engines and, 294
Guns & Butter (game), 106

H

Hamlet, 190–192, 224
 Hammurabi of Babylonia, 68–69
 hand-eye coordination, 48–49
 Happy_Sad variable, 205
 headline generator system, 256–258
 headline list, gossip and, 236
 history, of storytelling, 4–6
 HistoryBooks
 backstory in, 234–235
 extra information in, 235
 gossip systems in. *see* gossip

 looping behavior in, 233–234
 moods in, 233
 moving Plans into, 290
The Hitchhiker's Guide to the Galaxy (game), 130
The Hollow (book), 49
 Honest_False variable, for behavior modeling, 202
Huckleberry Finn (book), 25
 human qualities
 bell curves illustrating, 183–184
 Boolean expression of, 191–192
 Fate shifting, 215
 AND vs. OR for, 173–174
 quantifying with numbers, 99–102, 163–166
 types of, 198–199
 weighting factors for, 172–173, 185–189
 human understanding, model for, 32–35
 humanities community
 cultivate logical thinking, 109–110
 learn programming, 111
 pattern-based thinking and, 105
 scientific community vs., 103–104
 standardize terminology, 110–111
 Humble_Proud variable, 203

I

I_Know Boolean flag
 gossip and, 239
 lies and, 240
 in recording Events, 235–236
 ICIDS (International Conference on Interactive Digital Storytelling), 323
 iconic languages
 semantic volume in, 264–265
 SympolTalk, 262–264
 for *Trust & Betrayal*, 258–262
 id Software, 107
 IF. *see* interactive fiction (IF)
 IF statements, in scripting languages, 274
 if-then statements, 5, 7
 image, role of in stories, 22–23
 inclination formulae
 anticipating reactions and, 304
 determining story progression, 213
 exercises for, 208
 with personality models, 206–207
 independent developers, interactive storytelling and, 323
 individuation, importance of, 61–62
 IndObject, 244, 266

- inference engines
 anticipation factor and, 305–306
 lies requiring, 241
 plot-based engines and, 295
- Inform 7
 capabilities of, 135–137
 language of, 133–135
 uncanny valley problem and, 265
 weaknesses of, 137–138
- Ingold, Jon, 131
- initiation, of gossip, 237
- input, for Inform 7, 136
- instantiation, of higher principles, 78–80
- Integrating Research in Interactive Storytelling (IRIS) project, 318, 324
- intensifiers, text formulation and, 257
- interactive fiction (IF)
 Adventure program and, 250
 beginning of, 129–130
 development environments for, 133–138
 example of, 131–133
 future of, 140
 graphic adventures and, 128–129
 limitations of, 139–140
 text adventures and, 128
- interactive storytelling
 abstraction and, 74–78
 academic research into. *see* academic research
 branching trees and. *see* branching trees
 creative options in, 63
 decision-making in, 54–57
 different angles in, 59–60
 employing mathematics for, 94–96, 99–102
 free will vs. determinism and, 53–54
 future of, 320–325
 generalized verb handling in, 229–231
 gossip systems and. *see* gossip
 grapevine in, 242–244
 HistoryBooks and. *see* HistoryBooks
 immersion in, 224–225
 individuation in, 61–62
 interactive fiction vs., 139–140
 language and. *see* language(s)
 lies in, 239–241
 in MMORPGs, 147–148
 modeling for. *see* modeling behavior
 other media for, 98–99
 Plans/Events in, 232
 plot vs., 50–52
 principle into instance in, 78–80
 scripting languages for. *see* scripting languages
 secrets in, 241–242
 social reasoning and, 57–58
 Third Option and, 62–63
 three cultures of, 108
 tragedy in, 223–224
 turn-sequenced play in, 63–64
 unsatisfying outcomes in, 58–59
 verb counts in, 228–229
 video game comparison, 47–50
 women and, 322–323
 WordSockets in, 231–232
- Interactive Storytelling for Video Games* (Lebowitz & Klug), 29, 152–158
- interactivity
 competitive advantage of, 42–43
 definition of, 28, 210, 245
 depth of, 39–40
 ego control in, 35–36
 human understanding and, 32–35
 movies incorporating, 50
 power of, 43–44
 process-driven narrative and, 53–54
 quality of, 28–30
 revolutionary nature of, 43
 richness of choices in, 40–42
 second-person insight in, 30–32
 speed of, 37–39
 verb thinking and, 87–88
- interactivized movies, 50
- International Conference on Interactive Digital Storytelling (ICIDS), 323
- Internet, speed and, 39
- interstitial stories, Fate and, 216–217
- intuitive thinking. *see* pattern-based thinking
- inverse parsers, language construction and, 260–262
- iPhone, Siri for, 250–251
- IRIS (Integrating Research in Interactive Storytelling) project, 318, 324
- J**
- JavaScript programming language, 137, 184–185, 276
- Jurassic Park* (movie), 18, 20, 44, 48
- justice
 all-encompassing laws and, 69–70
 conflicting laws and, 68–69
 maintaining, 68

K

“Kill ’em if they stray” technique, 220–223
King Lear (play), 223
 Klug, Chris, 152
 koalas, overspecialization in, 145–147
Koyaanisqatsi (movie), 17, 246

L

language module
 function of, 5–6
 mental module interaction and, 7–9
 sequential thinking and, 7
 storytelling and, 9–15
 language(s)
Adventure (game) and, 250
 computers and, 245–246, 247–248
 creating, 266–267
 Eliza program, 248–250
 ESP-based, 258–259
Façade storyworld, 251–256
 headline generator system and, 256–258
 of Inform 7, 133–135, 136
 inverse parsers, 260–262
 noun prejudice in, 85
 Sappho scripting language, 277
 scripting. *see* scripting languages
 semantic volume and, 264–265
 Siri language user interface, 250–251
 SympolTalk, 262–264
 Laurel, Brenda, 209–210, 312
 laws
 all-encompassing, 69–70
 conflicting, 68–69
Le Morte D’Arthur (game), 298
 Lebowitz, Josiah, 152
 lies
 intent to harm in, 240–241
 in interactive storytelling, 239–241
 for misdirection/denial, 241
 listening functions
 of Fate, 210–213
 in interactivity, 28–30
 prevention of Two Cultures, 108–109
The Little Mermaid (book), 229
 logical plot points, 299
 logical thinking. *see* sequential thinking
 loops
 controlling, 273–274
 in HistoryBooks, 233–234
 Loquacious_Taciturn variable, 203

Lord of the Flies (book), 254
The Lord of the Rings (book/movie), 17, 18, 49,
 169–171

M

Macbeth, 20
 MacWrite, 228
Make It Good (Ingold), 131, 139–140
 market image
 role-playing games and, 149
 video games and, 321
 Martin, Steve, 104, 205
 massively multiplayer online role-playing games
 (MMORPGs), 147–148
 Mateas, Michael, 64, 255
 mathematics. *see also* numeric calculations
 abstraction and, 71
 causal relationships and, 96–97
 Inform 7 and, 137
 for interactive storytelling, 94–96
 as metaphorical, 98–99
 quantifying human qualities, 99–102, 163–166
The Matrix (movie), 20, 92
 Maxwell, James Clerk, 71
 maze trees, 120–121
 mechanism, for gossip, 238–239
 Meehan, James R., 295
 mental modules
 development of, 105
 environmental knowledge, 5
 interaction of, 7–9
 language, 5–6
 social reasoning, 5
 thinking mechanisms and, 6–7
 visual-spatial, 4–5
 menu-driven editing, 276–277
 metaphors, mathematical modeling as, 98–102
 Michelangelo, 36, 60, 94–95
 military science, verb thinking and, 82–83, 84,
 85–86
 misdirection, lies and, 241
Mixed Emotions (storyworld), 64
 MMORPGs (massively multiplayer online role-
 playing games), 147–148
 modeling behavior
 abstract thinking in, 163–166
 adding complexity to, 166–168
 addition vs. multiplication in, 173–174
 Boolean calculations for, 189–192
 bounded numbers for, 181–183
 computer as medium for, 162

- interactive storytelling and, 161
 - steps to follow for, 169–171
 - weighting factors in, 172–173
- modularization, scripting process and, 283
- Momentous_Trivial variable
 - headline list and, 236
 - rabble-rousing plot points and, 300
 - sharing gossip and, 239
 - for verbs, 229
- moods
 - HistoryBooks and, 233
 - personality traits and, 204–205
- motivation, for gossip, 238
- Motivational Analysis, 315
- movies
 - earliest years of, 98–99
 - interaction and, 29–30
 - language and, 246
- Ms. Pac-Man* (game), 322
- multimedia craze, software development and, 88–90
- multiple ending stories, video games and, 153–154
- multiplication
 - addition vs., 173–174
 - bounded numbers and, 185
 - converting Boolean AND to, 167–168, 170–171
- music, scoring performance and, 223
- Myst* (interactive fiction), 129

N

- Name variable, for verb data structure, 229
- Napoleon, 85–86
- narrativity, interactivity vs., 50–52
- national banks, abstraction and, 73–74
- The National Enquirer* (tabloid), 256
- negative predictions, 321–322
- Newton, Isaac, 71, 85
- Nice_Nasty variable, for modeling, 201
- non-player characters (NPCs)
 - canned audio and, 255
 - goal-seeking, 314
 - sharing gossip, 238–239
- NOT (Boolean operator), 190
- nouns
 - bias toward, 85–88
 - as complement to verbs, 82–83
 - verbs vs., 82–83
- NPCs. *see* non-player characters (NPCs)
- numbers
 - indicating verb value, 229
 - in scripting languages, 271

- numeric calculations. *see also* mathematics
 - Boolean vs., 191–192
 - bounded numbers in, 181–185
 - changing Boolean to, 167–168
 - for event-based engines, 288
 - inclination formulae as, 206–207
 - Inform 7 and, 137
 - for interactive storytelling, 177–179
 - modeling behavior and, 170–171
 - overruns and, 180–181

O

- objects, data types as, 270
- The Odyssey* (poem), 25, 142–143, 223
- one-factor use of blend operator, 188
- opportunistic plot points, 299–300
- optical disk games, 89–90
- Option(s)
 - acceptability/desirability of, 293–294
 - creative, 63
 - for Roles, 292–293
 - Third, 62–63
- OR (Boolean operator)
 - AND (Boolean) vs., 173–174
 - converting to numeric addition, 167–168, 170–171
 - function of, 190
- orthogonality, of personality models, 194–196
- output, with Inform 7, 136
- overruns, algorithmic design and, 180–181
- overspecialization, role-playing games and, 145–147
- overview variables
 - description of, 210–211
 - determining story progression, 213–214
 - plot points based on, 216
 - scoring algorithms and, 225
- The Ox-Bow Incident* (book), 254

P

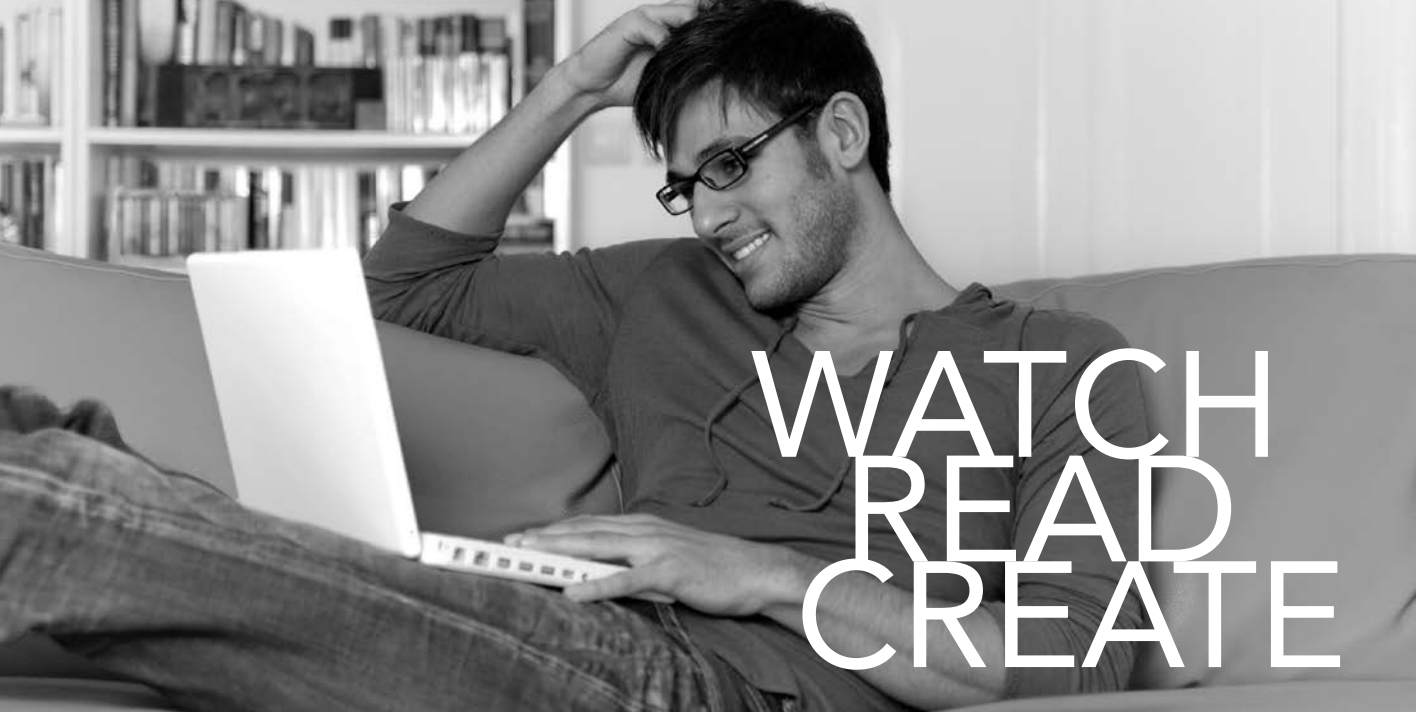
- Pac-Man* (game), 123–125, 223
- parameter prompts, in scripting process, 278
- parsers, inverse, 260–262
- Pascal programming language, 137
- pattern-based thinking
 - in arts/humanities, 104
 - cultivating logical thinking, 109–110
 - development of, 105
 - features of, 6–7
- people, role of, 17–18

- personality models
 - additional traits for, 203–204
 - behaviorally discriminating, 196–197
 - as complete/concise, 194
 - exercises for, 208
 - five factor, 200
 - inclination formulae with, 206–207
 - moods and, 204–205
 - as orthogonal, 194–196
 - overspecific, 197
 - physical traits for, 204
 - three core traits for, 201–202
 - types of traits in, 198–199
 - personality shifts, Fate and, 215
 - personality traits. *see* human qualities
 - PHP programming language, 137
 - Phrontisterion conferences, 324
 - physical intervention, by Fate, 214–215
 - physical traits, behavior modeling and, 204
 - physics, verb thinking and, 82–83, 84, 85
 - Pilot programming language, 137
 - Plans
 - aborting, 294
 - execution of, 290–291
 - in interactive storytelling, 232
 - for storytelling engines, 290
 - player behavior, correcting, 220–223
 - player skills dependency, MMORPGs and, 147–148
 - player-driven stories, video games and, 156
 - the Playwright, 209–210
 - plot
 - enforcement of, 219
 - higher abstraction and, 74–75
 - interactivity vs., 50–54
 - plot points
 - definition of, 298
 - Fate and, 215–216
 - logical, 299
 - opportunistic, 299–300
 - rabble-raising, 300
 - timed, 298–299
 - plot-based engines, 295–296, 297
 - Poetics* (Aristotle), 21
 - poison, for run-time errors, 279
 - Polyclitus, 36
 - predictions, for interactive storytelling, 320–325
 - preferred player behavior, 222–223
 - Preparation_Time variable, for verb data structure, 231
 - Pride and Prejudice* (Austin), 57
 - principle, translating into instance, 78–80
 - print stories, 106
 - process-driven narrative
 - abstract control and, 76–78
 - interactivity and, 53–54
 - processing capacity, video games and, 86–87
 - Prom Week* (game), 255, 316
 - prompts, parameter, 278
 - props
 - IndObjects and, 266
 - Inform 7 and, 136
 - in scripting process, 271
 - Ptolemy, 70
 - puzzles
 - babel fish, 130–131
 - in games vs. stories, 49
 - role of, 19–20
- Q**
- quantifiers, scripting process and, 284
- R**
- rabble-raising plot points, 300
 - reactions
 - anticipation of, 301–304
 - interaction vs., 29
 - verb choice and, 304–305
 - reality
 - language reflecting, 247–248
 - modeling and, 161
 - real-time play
 - clock-based engines and, 289
 - in traditional storytelling, 63–64
 - reasonable player behavior, 222–223
 - recorded video, early, 107
 - recursion, 302
 - Reiss' Motivation Analysis, 315
 - religion, social reasoning and, 7–8
 - repeated behavior, in HistoryBooks, 233–234
 - representational art, visual-spatial modules and, 8, 9
 - requirement script, of Roles, 291–292
 - resource management skills, 49–50
 - Return of the Jedi* (movie), 254
 - revolutionary nature, of interactivity, 43
 - The Rise of the Image, the Fall of the Word* (book), 22
 - robots, goal-seeking, 314
 - role-playing games (RPGs)
 - audience and, 149
 - creativity/change in, 143–145

- MMORPGs and, 147–148
 - overspecialization in, 145–147
 - overview of, 141–142
 - verbs in, 142–143
 - Roles, in clock-based engines, 291–292
 - Romeo and Juliet* (play), 59, 224
 - round-off errors, 179
 - run-time errors, 278–279
- S**
- Sapir, Edward, 247–248
 - Sapir-Whorf hypothesis, 247–248
 - Sappho scripting language
 - editing process with, 277
 - run-time errors and, 279
 - story recording/playback in, 282
 - vertical code layout in, 280–281
 - Saving Private Ryan* (movie), 22
 - science
 - abstraction and, 70–71
 - sequential thinking and, 8, 104–105
 - scientific community
 - cultivating intuitive thinking, 109–110
 - terminology standardized for, 110–111
 - scoring
 - algorithms for, 225–226
 - by Fate, 223–224
 - feedback and, 224–225
 - scripting languages
 - colored type in, 272
 - computer programming for, 269–270
 - flow control in, 272–275
 - fully functional defaults in, 275–276
 - function analyzer for, 279–281
 - limiting data types in, 270–272
 - menu-driven editing in, 276–277
 - modularization in, 283
 - parameter prompting in, 278
 - run-time errors and, 278–279
 - sample script, 283–285
 - story recording/playback, 281–282
 - second-person insight
 - facets of, 31–32
 - human understanding and, 32–35
 - in interactive storytelling, 30–31
 - second-person personality traits, 198–199
 - Secret Boolean flag, 241
 - secrets
 - calculating betrayal of, 304
 - in interactive storytelling, 241–242
 - verb choice and, 305
 - semantic volume, in iconic languages, 264–265
 - sentence structure, verbs and, 231–232
 - sequence control, in scripting process, 275
 - sequential thinking
 - cultivating intuitive thinking, 109–110
 - development of, 105
 - features of, 6–7
 - science and, 8, 104
 - The Seventh Guest* (game), 107
 - Shakespeare, William, 224
 - Show, C.P., 103
 - Shrek* (movie), 21–22
 - simplicity of themes, 48
 - The Sims* (game), 156
 - Siri language user interface, 250–251
 - Smart_Stupid variable, 204
 - Smith, Adam, 85
 - social acceptability of violence, Fate and, 211
 - social conflict, role of, 18–19
 - social reasoning module
 - environmental knowledge and, 7–8
 - function of, 5
 - interactivity and, 40, 57–58, 323
 - pattern-based thinking and, 7
 - storytelling and, 8, 9
 - women and, 322–323
 - Socrates, 190
 - software development
 - of Fate. *see* Fate
 - multimedia craze and, 88–90
 - space, Inform 7 and, 135
 - spatial thinking
 - storytelling and, 24–26
 - video games and, 322
 - speaking functions
 - dropping the fourth wall, 218
 - environmental manipulation, 214–215
 - goal injection/shifting personalities, 215
 - in interactivity, 28–30
 - interstitial stories, 216–217
 - plot enforcement, 219
 - plot points, 215–216
 - spectacle, in stories, 20–22
 - speed, interactivity and, 37–39
 - Spielberg, Steven, 18
 - spying, storytelling engines and, 294
 - stages
 - with Inform 7, 135
 - in scripting languages, 271
 - storytelling and, 24–26
 - standardized sequencing, in interactive storytelling, 275

- Star Trek* (movie), 25–26, 246
- Star Wars* (movie), 18, 20, 24, 25, 48, 49, 50, 254
- Stern, Andrew, 64
- stories
- choices in, 20
 - conflict in, 18–19
 - constipated, 152–153
 - cost of adding, 156–158
 - interstitial, 213–214
 - multiple ending, 153–154
 - open-ended, 155–156
 - player-driven, 156
 - puzzles in, 19–20
 - recording/playback of, 281–282
 - role of people in, 17–18
 - spatial thinking in, 24–26
 - spectacle in, 20–22
 - structure of, 16–17
 - temporal discontinuity in, 26
 - video games adding, 106–107, 151–152
 - visual thinking in, 22–23
- storytelling
- development of, 9–15
 - engines. *see* engines, storytelling
 - history of, 4–6
 - purpose of, 3–4
 - social reasoning and, 8, 9
 - traditional elements of, 62–64
- Storytron* (storyworld), 265, 296
- storyworlds
- anticipation in, 301–308
 - exploration of verb set in, 211–212
 - goals in, 300
 - plot points in, 298–300
 - scripting for. *see* scripting languages
- structure
- Inform 7 and, 138
 - sentence, 231–232
 - of stories, 16–17
- structured memory, 32–34
- Subject(s)
- creating languages and, 266
 - handling verbs and, 230
 - NPC relationship to, 239
 - reactions of, 291
 - reactions to, 302
 - sentence structure and, 231
- subtraction, bounded numbers and, 185
- supporting characters, in storyworlds, 300
- sylogisms, mathematics and, 96–97
- symmetric language, 266
- symmetry, creating languages and, 266
- SympolTalk, 262–264
- T**
- TaleSpin* (story-generation program), 295
- teachers, second-person insight in, 32, 34
- temporal discontinuity, 26
- The Terminator* (movie), 48, 49
- text
- colored for scripted languages, 272
 - handling capabilities of Inform 7, 136
 - label for verbs, 229
 - sample script for generation of, 283–285
- text adventures
- development of IF, 129–130
 - example of, 131–133
 - features of, 128
 - as puzzles, 130–131
- thinking functions
- of Fate, 213–214
 - in interactivity, 28–30
 - as pattern-based/sequential, 6–7
 - shifting to verb thinking, 90–92
- Third Option, in interactive storytelling, 62–63
- third-person personality traits, 198–199, 307–308
- three cultures, in interactive storytelling, 108
- time
- dramatic vs. physical, 26
 - with Inform 7, 135
- time frame, for emergence of interactive storytelling, 324–325
- timed plot points, 215–216, 298–299
- Tomlin, Lily, 104
- traditional storytelling
- creative options and, 63
 - real-time play in, 63–64
 - Third Option in, 62–63
- tragedy, in interactive storytelling, 223–224
- Transactional Analysis, 315
- tree analysis, anticipation and, 306–308
- TRUE Boolean flag
- gossip and, 239
 - in handling verbs, 230
 - lies and, 240
 - recording Events and, 236
- Trust & Betrayal* (game), 106, 216–217, 258–262
- truth, translation into instance, 78
- turn-sequenced play
- clock-based engines and, 289
 - with Inform 7, 135
 - in interactive storytelling, 63–64

- Two Cultures* (Snow), 103
- Two Cultures problem
 divisiveness of, 103–104
 games/stories and, 106–108
 neurophysiological basis of, 105
 prevention of, 108–111
- two-factor use of blend operator, 189
- U**
- uncanny valley problem, 265
- user interface languages
 Sapir-Whorf hypothesis and, 246
 Siri as, 250–251
- V**
- variables
 Boolean, 189–192
 deciding on/naming, 170–171
 modeling behavior and, 163–166
 overruns and, 180–181
 overview, 210–211, 213–214, 216, 225
 personality types as, 198–199
 in verb data structure, 229–231
 weighting factors for, 172–173, 185–189
- verb(s)
 anticipation factor and, 304–305
 complementing nouns, 82–83
 counts, 228–229
 creating languages and, 266, 272
 designing personality models and, 196–197
 for *Façade* (storyworld), 252–253, 255
 generalized handling of, 229–231
 intent to harm and, 240
 in interactive storytelling, 57–58
 inverse parsers and, 260–262
 noun prejudice and, 85–88
 nouns vs., 82–83
 role-playing games and, 142–143
 Roles attached to, 291–292
 WordSockets for, 231–232
- verb-based thinking
 adapting to, 90–92
 importance of, 227
- vertical code layout, function analyzer and,
 279–281
- video games
 adding stories to, 106–107, 151–152
 branching tree approach to, 154–155
 compared to interactive storytelling, 47–50
 constipated stories and, 152–153
 cost of stories in, 156–158
 definition of, 47, 48
 fully player-driven stories for, 156
 market image of, 320–321
 multimedia craze and, 89–90
 multiple ending stories for, 153–154
 noun prejudice and, 86–87
 open-ended stories for, 155–156
 plot vs. interactivity in, 53–54
 video-generated facial expression, in *Façade*
 (storyworld), 254
 violence
 Fate monitoring, 211
 in games vs. stories, 48
 role of, 19
 VisiCalc spreadsheet, speed and, 37–38
- visual-spatial modules
 function of, 4–5
 pattern-based thinking and, 7
 storytelling and, 22–26
 visual art/writing and, 8, 9
- W**
- weaknesses, of Inform 7, 137–138
- The Wealth of Nations* (Smith), 85
- weighting factors
 blend operator for, 185–189
 in modeling behavior, 172–173
- Weizenbaum, Joseph, 248
- Whorf, Benjamin, 247–248
- Willful_Pliant variable, for behavior modeling,
 202
- Wing Commander* (game), 106
- Witnesses variable, for verb data structure,
 230–231
- witnesses, to Events, 290–291
- Wolfenstein 3D* (game), 107
- women, interactive storytelling and, 322–323
- WordSockets, for verbs, 231–232
- writing, visual-spatial modules and, 8, 9. *see also*
 scripting languages
- Y**
- Yar's Revenge* (game), 106
- yin-yang relationships, verb thinking and, 82–83
- Z**
- Zork* (game), 128



WATCH READ CREATE

Unlimited online access to all Peachpit, Adobe Press, Apple Training, and New Riders videos and books, as well as content from other leading publishers including: O'Reilly Media, Focal Press, Sams, Que, Total Training, John Wiley & Sons, Course Technology PTR, Class on Demand, VTC, and more.

No time commitment or contract required!
Sign up for one month or a year.
All for \$19.99 a month

SIGN UP TODAY
peachpit.com/creativeedge

creative
edge