there is reason for hope! Now that we've identified these problems, we
have the chance to think about ways to address them and avoid their worst
effects. We also have the understanding needed to create new techniques
that eliminate these problems and help us create fun and interesting new
ways to combine game playing and storytelling.

In this chapter we'll start to cast our eyes forward and consider a few
important topics that will help us understand the qualities of participatory stories. We'll consider the change in nature of the audience,
from passive observers to active participants. Then we'll discuss how these
participants interact with the underlying simulation; this is quite a different
matter from a traditional user interface. Finally, we'll look at the issue
of language.

Fun

Any style of storytelling that allows the audience opportunities to participate
is implicitly asking the audience to be active: that is, to do something.
Most contemporary storytelling media entertain audiences who are passive, though their imaginations are engaged. Watching television requires
no more physical effort than lying on the couch, and reading a book means
only turning pages and scanning words with our eyes.

What then makes us think that people will want to engage in any new
forms of storytelling that requires their active participation? After all, this
is a leisure activity, so why would anyone want to work at it?
The answer is that people are willing to do all kinds of work as long as
it's fun.
Why do people go to professional sports games when they can watch them on television? It's a lot more work to pack up and go to the stadium, park the car, buy tickets, and then get home again. Why do we play with our dogs, make love, take hikes, go canoeing, or make any other physical effort when we could instead be sitting motionless on the couch and watching television? Why do we work for our entertainment and pleasure?

The Most Fun Theory

We put in effort for many recreational activities because they're fun, and we're willing to put in work if we get back enough fun. I call this the Most Fun Theory.

The Most Fun Theory: People are willing to put in as much work as required in any activity if they get back enough high-quality fun.

This is why we are willing to work so hard when we play tennis, go kayaking, play music with friends, or even participate in a physical party game like Charades. We're willing, and even quite eager, to work as long as we're getting back enough good fun. If the fun we get back isn't up to the effort we're required to invest, we'll back off or stop altogether. But if we're having enough fun, we'll keep doing it and even look for opportunities to do it again.

The Fun-to-Work Ratio

If we're willing to work for our fun, why then do people watch television? Much television programming is awful and, frankly, hardly any fun at all. Understanding why people watch TV leads us to the Fun-to-Work Ratio.

The Fun-to-Work Ratio: People want to maximize the ratio of quality fun to invested work.

Let's find the Fun-to-Work Ratio for TV. The amount of fun in many TV shows is very small; often, it's tiny. The amount of effort required to watch television is effectively zero: we need do nothing but lie on the couch and keep our eyes open. The ratio of a tiny bit of fun to zero work is infinite! That's why television is so appealing: the Fun-to-Work Ratio is as high as it can be.

But there's a problem: it's generally not very high-quality fun. Determining the "quality" of fun is a very personal judgment, but we've all had the experience of watching television and, though we're not enjoying it, just being too lazy to turn it off. We're not expending much effort, and there's a good deal of some kind of fun coming our way, but the "fun" is pretty dreary.

The Fun-to-Work Ratio helps us see why we then get off of our couches and do any physical activity that brings us pleasure: it's not just the quantity of fun that matters, but the quality as well. The scale is very subjective and personal, but at some point we will choose to expend effort to get a smaller amount of good fun over a larger amount of lesser-quality fun.

This measure also explains the impressive sales figures and reviewer praise given to games that offer significant novelty. Many people find things that are new or different to be inherently interesting and fun to explore.

The Fun-to-Work ratio is the fundamental reason why people play with toys, which, after all, requires effort. It's because we get back enough fun, of a quality that we enjoy, that the effort is worth it to us. It's the same reason we go on hikes, ski, play with dogs, carve pumpkins, or go dancing.

Storytelling experiences that invite or even require the audience to actively participate, must make sure that they return enough fun of sufficiently high quality that the audience will find the Fun-to-Work Ratio both of high quantity and quality.

Home Drama

In Chapter 8 I presented a number of arguments against the very idea of interactive stories. One of those arguments I summarized as acting is hard, and I discussed it in only one paragraph, promising to return to it later. Let's now look at this issue more closely, starting with simply reading a great script.

Actors often first rehearse a play or movie by just reading out lines while sitting around a table; naturally enough, this is called a table reading, and it helps give everyone a rough idea of the general shape of the work.

Sometimes a playwright will test out a play with an acting company in a very informal presentation. After just one or two rehearsals, the actors walk through the play on stage in front of an audience. Usually there is no lighting or costuming, nor any sets or props, and the actors read their lines from the scripts in their hands as they walk around with only very
rudimentary blocking. Such a staged reading gives the playwright and an audience a rough idea of how the characters come across, and a general feel for pacing and the overall feel of the play.

Table readings and staged readings aren’t restricted to professional actors. Anyone can get together a group of friends to sit around a table and read a play out loud. This is very easy to do, yet it’s rare for people to spend an evening this way.

This may seem something of a surprise, given our perspective from earlier chapters on the importance of stories in our lives. We read books and see movies, and we watch television and listen to the radio, so why don’t we put on readings of plays at home?

After all, all the ingredients are there. We have a great story to work from. There are many popular plays, widely known and even beloved. The characters and plots are fascinating, and the words are already chosen and waiting on the page. We don’t need props or other preparation: just show up, take a script, read along, and speak aloud when it’s your turn.

It’s certainly not necessary that everyone who takes place in a home drama reading group be a good actor. In most groups it’s perfectly acceptable to read the lines with a minimum of inflection, simply so everyone can hear them out loud and enjoy the flow of the play. But most people naturally feel that the experience is better with at least a bit of emotion put into the process, and even this touch of emotion brings us to the world of acting and all the reasons people tend to avoid situations where they’re called upon to perform as actors.

If people don’t widely participate in home drama, it seems unlikely that they would want to take on the role of a fictional character in an interactive story. This reluctance returns us to the discussion in Chapter 8, where I talked about the difficulties inherent in controlling an on-screen character: are we acting as we would, as we think the character would, or as we think the character should? If we choose to forego games in which we act as ourselves, we’re eliminating a huge element of what interactive fiction can do for us. If we include such identification, then we have a challenge. Active participation is the critical element that distinguishes any kind of interaction in a story from static media such as books, records, and movies.

The answer is to look more closely at the elements of home drama. By understanding just what features of this activity are appealing and unappealing, we can focus our energies on devising interactive experiences that keep the desirable elements while avoiding the ones people don’t like.

The positives of the home drama experience are similar to those of any story, and we’ve covered those in detail in Part I: the pleasure of losing yourself in another time and place, of learning about other people’s lives, of seeing how a fascinating character behaves in an interesting and challenging situation, and of discovering something about human nature. These are all reasons why we’d expect people to enjoy table readings at home. We can see why home dramatic readings are rare by applying the Most Fun Theory: the effort involved exceeds the amount of high-quality fun that people experience in return. So let’s look at the nature of that effort, which takes the form of problems that must be surmounted.

We’ll see that the main issues have to do with human nature and emotional risk-taking, rather than technology.

Social coordination is tough: Reading a play with several characters requires getting a group of people together. Although people can take multiple roles, it’s best when each major character in a given scene is played by a different person, otherwise two characters have the same voice, which dilutes the pleasure of hearing them speak to each other. Some people use accents or otherwise try to distinguish their multiple roles, but that’s hard. If the group can’t get through the whole play in a single sitting, the coordination problem becomes harder, because then somehow roughly the same group needs to come together again to continue the play. This can become a scheduling nightmare for the person who volunteers to coordinate the activity.

There are no sets, props, or costumes: Some people draw inspiration from the physical objects that are associated with performance. Stories are legion about actors who “found” a character when they put on a particular shirt, pair of glasses, or a particularly outrageous wig. Though these physical objects aren’t necessary, some home drama groups keep a small box of props and clothing accessories, from cigars to earrings, for people to go through and draw inspiration from. Sometimes just putting on a particular hat is enough to help us “feel” like a different person.

Performance is stressful: In any group reading a play together, it’s inevitable that some people will get into it more than others. Some people are simply shy or insecure and reluctant to take part in what they perceive as a performance. Even though there’s no formal audi-
ence, they may see the other readers as judging how well they read their lines. They may be nervous about not reading a line with the right inflection, stuttering, or committing some other faux pas in front of other people.

Creating a character is hard. When we accept a role in a staged reading, we can certainly read the lines right from the page, without inflection. But if we want to give them some feeling, we need to understand the meaning behind the words and the feelings of the person saying them.

In other words, we have to find the subtext. Subtext can be very subtle and difficult to discover, or it can be pretty close to the surface. It almost always depends on context and the shared history between the person speaking and the person he’s speaking to.

For example, suppose Stan and Ollie work in an office, and they meet each other next to the coffee maker in the break room. Stan says to Ollie, “The coffee maker is empty.” There might be no subtext at all; it could simply be that Stan is pointing out a fact to Ollie. But Stan probably has a reason for saying this, and that’s the subtext. He might be petulant: “It’s your turn to make coffee, since I did it the last three times.” He could be complaining: “See? I told you that nobody in the office else would ever bother to refill the pot.” Or he could be complimentary: “That new blend you brought in today was great! Everyone’s already finished it off.” Obviously these utterances mean three very different things and would come out in three different tones of voice.

Because the subtext in a conversation depends on the relationship between at least two characters, we need to know something about them to know why they’re saying what they’re saying. Creating a character, with a history and feelings and strengths and weaknesses and all the other attributes that go into making a personality, is hard work. Even for a simple character, you have to make up a lot of background and make sure it all fits together in some reasonable way. For each line, you have to discover an appropriate subtext. Actors call this “building a character,” and it takes time and effort. If it’s not a process that you enjoy, then it can feel more like work than play.

Good acting is hard. We’re all good casual actors. As children, a natural part of development is pretending to feel one emotion while actually feeling another: pretending to be contrite for breaking a rule, for example, while not really feeling badly at all. Or a child might pretend to be brave in front of the school bully, when he’s inwardly afraid. Such casual acting has the quality that we’re trying to give the impression of having an emotion or attitude whether we actually feel it or not.

The modern style of acting builds on this natural tendency with a “naturalistic” technique (this hasn’t always been the case). The “method” approach taught by Stanislavski is based on actors actually feeling the emotions they’re called on to portray [131]. In other words, suppose that you’re an actor in a play and you’ve walked into a surprise birthday party. Working from the script and the director’s guidance, you know that your character is both surprised and very happy.

One way to play the scene is to pretend to be surprised and happy. That is, you try to remember the sorts of things people do when they’re surprised and happy, and you do those things: perhaps you clap your hands to your cheeks and smile widely, take a gasp of air and hold it, and then speak in a rushed and loud voice. These are the sorts of things you’ve seen people do when they’re happily surprised, so you emulate them. Such pantomime is a time-honored form of performance with a long history, but it’s not in fashion today.

The naturalistic school takes a different approach. To greatly simplify a complex subject, you actually feel the emotions that your character is feeling, and then simply execute the lines and blocking that have been given you. So in our scene, you’d think back to a time in your life when you were happily surprised, and you’d bring back that emotional memory so vividly that you relive the moment. With such emotional recall, you’re able to be now, on stage, as you were once before, truly happy and surprised. You don’t have to think about specific gestures or how to make your voice sound: by virtue of feeling the emotions, you’ll naturally do all of the things that you do when you feel that way, and the audience will clearly read your feelings. It’s not quite that simple, and method actors study many other techniques as well [84]. But the basic idea is that you give priority to feeling the emotions you’re portraying, rather than trying to pretend as though you were feeling them.
Interactive Storytelling

This is generally hard to do. As in any art, there are always a few people to whom method acting seems to come naturally. But most actors take classes, and like other artists they study, practice, and work at their craft to develop and increase their skills and range. For most people, becoming a good actor requires the same kind of substantial and sustained commitment required to become a good pianist or a good painter. Because we each have only so much time, we have as priorities the things that we most enjoy. Thus most adults have not put in the work necessary to become good violinists, sculptors, or actors.

Good acting can hurt: I described above how method actors train themselves to actually feel the emotions that their characters are experiencing, and then to perform while in the thrall of those emotions. I picked happy surprise for the example, and many of us would be willing to remember that feeling and let it inhabit us.

But rare is the story where everyone feels good all the time. Remember that the heart of drama is conflict, and conflict easily leads to uncomfortable and unpleasant emotions, such as frustration, anger,rol, and fear.

Most people would naturally be reluctant to deliberately choose to feel these emotions unless there was a great need. Reading a play for pleasure with friends hardly seems to be such a need: why would anyone deliberately want to feel badly as part of his recreation? When we get together with friends, we want to enjoy our time with them. Getting angry or upset, or frustrated or hurtful or embarrassed or any of a million other negative emotions seems to be the last thing we'd want.

Yet that’s today’s acting style. People who see themselves as actors are willing to do this and even to eagerly compete for the opportunity, because it gives them a chance to develop and display the skills they work so hard to develop, and because they inherently enjoy acting. And professionals get paid for it, which doesn’t hurt.

Because as amateurs we don’t want to feel badly when we’re relaxing with friends, many of us are reluctant to throw ourselves into parts that have a lot of conflict. But these are often the best and juiciest parts in modern plays.

First Steps

We also don’t like to see our friends hurt and in pain. It’s hard to sit a table while a friend of yours is crying and sobbing. Even if he’s wrenching out a confession that’s written on the page, if he’s feeling those emotions, he’s still in pain. That’s no fun to watch, particularly since you can’t help him.

As a result of not wanting to feel badly, or to have to helplessly watch friends feel badly, many play-reading groups focus on light comedies, where negative emotions don’t get too deep. But this self-censorship severely limits the range of stories that such groups consider for their get-togethers.

Bad acting is embarrassing: We’ve all seen bad actors perform, and we inwardly cringe for them in sympathetic embarrassment.

When it comes our turn to act, we remember that feeling so clearly that many of us shy away from taking the risk of being seen that way, even among friends. Even people with naturally outgoing natures can be reluctant to take on a role that they think may require them to do more than be themselves.

Acting is revealing: Whether we’re reading our lines casually or with emotion flooding our veins, acting is personally revealing. Normally we don’t cry in public, but if the role calls for sorrow that befits crying, then we cry to the degree our comfort and skill allow. Feeling an emotion like that and revealing it in public are difficult things for many people to do. In every culture there are norms for what sort of behavior is acceptable from men, women, adults, and children. If a grown man believes that grown men aren’t supposed to cry, that can be a very scary thing for him to do in front of other people. Similarly, if grown women aren’t supposed to be assertive, that can be scary for a woman who needs to perform that role.

Some emotions, like sorrow and fear, can leave us unusually vulnerable to others, so we often hide them in public. Even if we’re simply taking on those feelings from some remembered moment in the past, once they’re inside us, we’re transported (again, to the degree our skill and desire allow) back to that state. So if we’re truly in that open condition, even while reading lines off a page, our hearts are open and vulnerable. If someone in the room chooses to go off the page and ridicule or tease us, it’s going to strike hard. Even if one of
the other people in the room is just playing a character that attacks us this way, it can be just as bad or even worse. Each line read by that player was worked and edited by the playwright until it became a precise and deeply wounding weapon that can hurt us badly, even though it obviously wasn’t written to attack us personally.

In some plays the writing is so intense that feeling pain is almost unavoidable. Sitting around a table and reading Who’s Afraid of Virginia Woolf? is an emotionally wrenching experience, no matter how much distance everyone tries to keep from the material.

Some people try to avoid certain emotions not only because they’re unpleasant, but because they are potentially dangerous or even toxic. A person prone to depression may spend years to learn how to handle feelings of depression and hopelessness and find ways to reduce their frequency and intensity in his life. If a play calls for him to feel such emotions, he’s likely not going to be interested in participating, at least not in that role.

Letting people see the intensity of some of our emotions can be more revealing than we want. If a character is needy and clinging, we might be afraid of how it would look if we were too good at the role: it would imply that there is a needy and clinging part of ourselves that we’re able to tap into. This is true of all sorts of emotions: someone may hide his anger, but if a role calls for rage and he is able to really sink into the role and erupt in fury, his carefully-cultivated peaceful image may be lost as the others in the room see what does indeed lurk inside of him. And the things we choose to hide are often precisely those things we don’t want to show to others or possibly even reveal to our own selves.

Revealing and experiencing emotions, even if just while performing a part, can be personally risky.

So now we can see why most people don’t get together to read plays out loud, even plays that they know well and love. Getting into the part at all means doing some acting. Not only is acting a learned skill, but it’s emotionally risky in the way that picking up cars is physically risky: the activity may be enjoyable in some ways, but it also inherently opens you up to the possibility of injury.

We can now apply the Most Fun Theory and see why play-reading groups are rare: the combination of the logistics of getting people together and the risks of acting, real and perceived, simply exceed the fun that comes back. People need stories, but they can get them in a wide variety of passive media that don’t carry the costs of actively performing a role.

Note that some, but not all, of these emotional risks depend on the presence of other people. Those that are based on exposing one’s emotions to other people are closely related to those people being in the room. Those that don’t depend on other people revolve around our feeling negative emotions that we wouldn’t normally choose to feel, particularly when we’re trying to relax or simply enjoy ourselves.

As we develop forms of stories that call on people to become active participants, we’ll have to be careful not to require most people to take on the risky and unpleasant aspects of acting, even if it’s within the context of a game or online story. There will always be people who will be attracted to acting opportunities and will thrive on them, but they are not the majority. As we seek to develop a form that will be popular to a wide spectrum of people, we need to make sure that we keep our eyes focused on the work and risks involved in participation, and always maximize the Fun-to-Work Ratio with high-quality fun.

System Interface and World Interface

Let’s now consider a very different issue, coming not from the world of human emotions but from the world of technology. The focus is how we interact with the world of the story.

The term “user interface” has traditionally been applied to the hardware and software that is directly manipulated or perceived by someone using a piece of technology. Thus we see user-interface experts studying input hardware (keyboards and mice), output hardware (monitors and speakers), input software (scroll bars and context-sensitive menus), and output software (progress bars and smooth fonts). All of these elements have been extensively studied in the domain of user interface work, sometimes also called human-computer interaction, or HCI. I’ll call these elements the system interface.

There is a second user interface, which I call the world interface, that will be increasingly important as we move forward in designing interactive
stories. Let’s briefly review how to make good traditional user interfaces, so we can apply the relevant principles of the system interface to the world.

The System Interface

User-interface researchers have developed guidelines that they use to characterize a good user interface. Schneiderman’s list of 8 principles is a good summary [121]:

1. Strive for consistency: There should be an overall pattern to the appearance of the interface. Layouts and shortcuts, for example, should be the same from one application to another.

2. Enable frequent users to use shortcuts: As people become more familiar with the system, they should be able to accomplish their tasks more easily or quickly than a novice.

3. Offer informative feedback: People should know that their commands have been properly received and interpreted by the computer, and they should have some idea of how their instructions are progressing over time.

4. Design dialogues to yield closure: People need to know when they have given the computer enough information to carry out the intended process.

5. Offer simple error handling: The system should be designed to offer useful feedback for any possible problem, and the person should be able to undo any command that caused an error. The system should never crash or give an unintelligible error message.

6. Permit easy reversal of actions: Software should recognize that people make mistakes, and sometimes change their minds. The system should make it easy for a person to stop or undo any action, even if it didn’t cause an error.

7. Support internal locus of control: The person should always feel that he is in control of the system.

8. Reduce short-term memory load: People shouldn’t be forced to remember things. The system should provide a list of possible actions and objects whenever something needs to be specified. It should be easy to find the command you want, or the object you want to apply it to, without having to remember it.

The traditional computer user interface is a tool. Its goals are to be as clear and simple as possible, while also being sufficiently powerful and robust to do complex work without undue effort from the user. These criteria require a careful balancing of many design tradeoffs. One can argue that successful user-interface design can benefit from many of the same techniques of artifice and directness that are used in theatrical design [81].

Most UI designers strive to create products that have a feeling of being obvious. They want a novice to be able to sit down and figure out how to get his tasks done without having to consult a manual, and they want experts to be able to work quickly and efficiently.

An automobile’s dashboard and pedals are often used as an example of a very good interface. Running through the list above, we can see that just about everything is handled well, using just a small number of very simple input and output devices.

User-interface designers sometimes speak of the underlying data that a person is interacting with as the model, and any given presentation of it as a view. Thus one might have a model that is a text document, and it may have one view on a large screen at work and a different view on a tiny cell phone display. The interfaces associated with these two views would be very different, but they would both probably support some version of common text-processing operations like cutting and pasting, though by very different means. The game Majestic used a wide variety of devices and views (e.g., fax machines, cell phones, and email) to present a rich multi-media interface to the world of the game.

“Intuitive” Design

Before we move on, it will be useful to set aside a term that has been abused for years when it comes to human-computer interaction. Interface designers and marketers sometimes describe their products as “intuitive.” This is at best a sloppy use of the word: intuitive means that something appeals to intuition, which the American Heritage Dictionary defines as “The act or faculty of knowing or sensing without the use of rational processes; immediate cognition” [107]. The term is slippery because the ability to
immediately grasp something without thinking depends on one's experience and what one already knows.

Professor Wilson's courses on theoretical physics were always standing room only. Though he wasn't the world's most lucid teacher, Wilson was a brilliant researcher, and students were eager to sit in his class and learn directly from the master.

One day Professor Wilson was deriving a particularly tricky equation on the blackboard, speaking out loud to the class as he wrote. "So we can cancel this term, and the result simplifies to this expression. Now it's obvious that I can rewrite this equation in the following form." After writing the new formula, he suddenly stopped, stepped away from the blackboard, crossed his arms, and quietly contemplated what he had just written.

After standing silently for almost a full minute, Professor Wilson walked out the door and down the hall to his office, where he pulled out a pad and a couple of heavy reference books. He spent a half-hour deep in study, flipping through his books and writing page after page of equations. Finally he stood up and strode back to the classroom. Smiling at the befuddled students still in their seats, he triumphantly announced, "I was right! It is obvious!" and then turned back to the board and happily continued where he'd left off.

To someone who's used today's graphical operating systems for many years, context-sensitive pop-up menus may seem obvious or even intuitive, as in free of "rational processes." To someone who hasn't seen either of these before, such gadgets are manifestly not intuitive.

Calling any contemporary computer interface "intuitive" without a complete description of someone's necessary prior experiences makes as much sense as calling it "familiar": without a clear description of who's going to use it, what they already know, and what they want to achieve, the adjective simply doesn't mean anything.

The World Interface

Most computer and console games have their own idiosyncratic user interfaces, which players must learn to control. For example, joysticks move a character or a camera, menus call up lists of objects in an inventory or

magic spells at hand, buttons need to be pressed at the right moment to jump or tumble or crouch down, and there are the complex multi-button combinations that trigger sophisticated actions in real-time fighting games such as Tekken Tag Tournament and stunt games such as Tony Hawk's Pro Skater 4 and SSX Tricky. These multi-button sequences are all part of the system interface.

In a direct user interface, a person's actions are turned into specific commands for the computer, which usually take effect immediately. For example, if you're using a word-processing program and select some text and then copy it, that text is immediately placed on the system clipboard. If you paste from that clipboard, you immediately see the text appear in your document at the location you selected. If you command a selection to be set in bold type, or double-spaced, you see those results right away.

Now let's think about controlling something that is dynamic and running on its own, like a simulator program.

To illustrate the following points, I'll use an imaginary game that I've invented just for this discussion. It's a boat-racing game that I'll call Yacht Racer. In this imaginary game you get to design and build your own yacht from a kit of parts, and then race it against other people's boats. The mechanics and physics of the boat itself, as well as the wind and water, will all be as accurate as possible, thanks to a sophisticated simulator.

The game has two basic locations: the workshop and the ocean. In the workshop, we try to build the best yacht we can. There's a limit to the total price of the parts we can use, and to the boat's maximum length, width, and height. Within these limits we can build whatever we desire from any of the parts that are available. Once we've built our yacht, we go out to the ocean, where we take to the water. We can test our design by taking as many practice runs as we want, through any or all of the courses. While sailing, we can adjust anything on our boat that's open to adjustment: we can trim the sails, move the crew around to redistribute weight, and steer with the rudder. We can also vary the environment, trying out our boat in choppy and calm seas, high storms and quiet, windless days. If we want, we can return to the workshop to adjust our design. We can then return to the ocean again and repeat the cycle as much as we like until we're happy with our boat.

When we're satisfied with our design, we can race our yacht against other boats that are created by the computer or against yachts designed by other people. The winner gets all the glory!
Let's consider what happens when we take our yacht out to sea. Suppose we've equipped our boat with a crew of seven able seamen, all of about the same age, weight, strength, and experience. We discover as we sail that a useful move in some situations involves pulling the sails in close and letting the boat lean over on one side. But we risk capsizing this way, so we'll send a couple of sailors off to the high side of the boat to lean over the railing and act as a counterweight.

We might use a traditional system UI to move these men, perhaps by clicking on each one in turn to collect them together as a group, and then clicking on the spot where we want them to go. By doing this, we'd be changing one element in the underlying, abstract state of our boat (the positions of our crewmen) to bring about our intent (establish a counterweight) to the system (the boat physics simulator). Programmers are used to this idea of communicating with abstract objects like a boat simulator through a programming interface, but I'd like to suggest that players can think of their system interface actions as affecting the underlying simulator through the world interface, as shown in Figure 20.

![Figure 20: The system interface and the world interface. The player (a) manipulates the system interface (b) that is made available to him. These actions change the information (c) stored in the computer that controls the simulator (d).](image)

In this example, one of the inputs to the boat simulator is the distribution of weight. The boat simulator doesn't care about how the player selects crewmen or how he indicates where they should go, it just uses the locations of the men, however they got there. So when we move the sailors, we're using the system UI to issue a command to the boat, or world UI: move these sailors to this point in the computer's internal model of the ship.

The world interface is often invisible. In fact, the nature of this interface may or may not be accessible, in the sense that you can easily discover how to do whatever you want. You may not even know what all of your options are when you get started. As I mentioned in Chapter 5, many simulator games are played by people who enjoy solving the puzzle of figuring out what options they have for controlling the underlying simulator and how it works. This task is accomplished by exploring the world interface, which is accessed through the system interface.

For example, in a detective game you may come to a locked door. To open it, you might use the system interface to dig out your key and use it on the lock. The result would be a message to the world interface that this key is turning in this lock, and if they fit, the door would unlock. If on the other hand you decide to shoot the lock off with your gun, this single action in the system interface may trigger many changes in the world. The world interface may take the type of gun and where it is pointed, and calculate the trajectory of the bullet to find that it continues through the door and shatters a vase in the next room. So the vase breaks into many pieces, each of which crashes to the floor, and the resulting noise sets off a burglar alarm, which will alert the security guard to come running.

The reason for thinking of affecting the underlying world of the game through its own interface is that it lets us look at that the qualities of that interface. Considering the list of UI attributes in the previous section, we might say that a good interface is simultaneously consistent, repeatable, and predictable. The system interface is direct, while the world interface is indirect.

Returning to our imaginary yachting game, suppose we have all of our crewmen standing in the middle of the boat, and we select three of them to go stand by the railing to act as a counterweight. Since they're all of roughly the same weight, it shouldn't matter which three men we pick. So the interface is predictable and consistent: as long as we get the right
distribution of weight around the boat, we'll get the proper effect. The underlying world is also repeatable, because if we make the same change to the weight distribution in a repeat of the very same situation, we would expect the boat to respond the same way.

The world interface to the simulator in many of today's games falls short in one or more of these properties. For example, in many fantasy games your primary weapon is a sword, usually used to hack monsters. But every now and then the sword is useful for something else, such as cutting a rope that holds a trap door closed. When we cut this rope, in effect we've learned one of the ways to manipulate the world through the user interface: ropes can be cut by our sword. The principle of consistency tells us that we should be able to cut any rope we can reach throughout the game. But that almost never happens: the only ropes we actually can cut are the ones that the game designers have explicitly enabled for this operation. Similarly, our sword can chop up a monster, but it can't slice the loaf of bread sitting on a table at an inn.

In terms of the system interface, this discrepancy is like a checkbox that we can sometimes click on to enable, but sometimes can't. System UI designers try to make these modes obvious by graying-out a disabled checkbox or hiding it altogether.

So now we can see why some game designers rely on the magic items I disparaged in Chapter 8: they're a way to distinguish the very limited opportunities to affect the world UI through the much richer system UI. If you can cut a couple of distinguished ropes in the world, but none of the others, then game designers will make those special ropes a different color, cause them to glow, give off a special sound, or otherwise give them a special attribute to tell that player that the checkbox has now become enabled: this rope is recognized by the world UI, and he may (or must) now affect it. The system UI is used to reveal an unpredictable inconsistency in the world UI, and the result is a clumsy mish-mash of unrelated concepts.

To see this problem in a familiar context, imagine a word processor that occasionally let you italicize text. Not all words can be set in italics, and it's hard to predict which ones can and can't be set that way. So as you type, some words appear in red to let you know that they can be set in italics. It would be a frustrating way to work.

In our rope-cutting adventure, once you've cut a rope and it lies in two pieces, in a repeatable system you'd be able to cut those pieces again into smaller pieces. Virtually no game produced today allows that action: once a magical rope has been cut once, it ceases to be magical, and thus cannot be cut any more. And other non-magical ropes in the world cannot be cut at all, so there's no predictability or consistency.

In another common situation, while adventuring through a dark chamber or cave, you sometimes need to light a fire. You may have a torch, a spell to create fire, or a flaming arrow in your quiver, and they can all serve as a flame and thus as a source of light. But you can't burn the wooden chair in the corner, or the old letter you carry in your inventory, or the bottle of cooking oil in the pantry. The system interface allows you to try most of these things: for example, you can move your torch onto the chair or shoot an arrow at it. But the world interface doesn't recognize this as an action it can handle, and thus the simulator ignores that action, and the chair does not catch on fire.

One reason for limiting the world UI so severely is that the world simulator is typically not very robust. If you're to be allowed to set fire to anything, then the system needs to "understand" how every object in the world responds to fire, and how fire itself behaves in an environment. If you set a chair on fire, could it burn down the house? If so, would critical clues to the story be lost forever, and the game effectively be made impossible to win at that point? How would the player know that? Could the hero accidentally set his own clothes on fire? Could a sudden wind blow out his flaming torch? If it started to rain, would it put out the fire? Writing a program that could handle all of these situations would be a monumental task.

Many recent game designs have opened up the world interface in a particular but useful way: your character in 3D environment can walk or stand anywhere that he can get to. So if you can jump up to the top of a house, you can walk on the roof. If you can jump on top of a barrel, you can stand there and jump from that point, extending your vertical reach. If you can carry a barrel that you can stand on, then you can put it where you want and then stand on it. The game Kingdom Hearts uses this freedom of movement both to make the environments come alive and to give players more choices in how to manage real-time fights with the bad guys.

This flexibility has its limits: there are usually walls or cliffs or other barriers to the range available for you to explore. And lots of objects are too high or small for you to stand or climb on. But the beauty of this situation is that the 3D environment is not segmented into the places where you can
stand or walk, and all the others that you slide off or can't even surmount. This part of the world interface is well-matched to the system interface: if you can reach a spot, you can stand there.

Many game designers naturally recoil in horror at the enormous demands implied by a robust simulator and its associated interface. Thus they place artificial boundaries in the system interface: if you walk into someone's office, you cannot pick up the telephone handset sitting on his desk, but you can pick up the pen that was engraved with the telephone number of the corrupt contractor. You can't walk into a well-stocked kitchen and make yourself any meal you'd like, but you can empty the box of baking soda to find the secret microfilm hidden inside. These are not limitations of the system UI, since you can move your character into place and try to do these things; they're limits of the world UI, because your actions don't have an effect on the underlying world of the simulator.

This mismatch is a source of endless confusion and frustration, as players try to discover which actions actually do something, and which look like they ought to do something but don't. The inconsistency between the system interface and the world interface is at the heart of the use of magical items, which I discussed in Chapter 8. Why must you get and use precisely one particular pencil, and no other, to defuse the bomb? Because the game isn't a general simulator, and it can't let you use objects any way you want, on the fly.

A better solution would be a system UI that greatly limited what you could do but was matched to the abilities of the world UI, and thus was consisent and predictable in how all of those actions affected the world. Perhaps you can do only three kinds of things as a player: walk and jump through a 3D environment, carry objects, and cut things with your sword. Then you should be able to walk or jump anywhere that you can reach, carry anything at all that isn't nailed down, and cut anything that your sword can get at. Then the world interface becomes a better user interface: it's consistent and predictable.

There have been a few systems that take this approach. The Incredible Machine offers players a deliberately goofy collection of parts with which to build gadgets. When you start the program, you're given a collection of fanciful mechanical parts and a 2D world in which to arrange them. During the design phase, you can put the parts anywhere you want in the 2D space. Then you turn on the simulator, and the world comes to life.

Gravity runs on and pulls everything downward, mice run away from cats and towards cheese, and helium balloons rise and bump into things.

You can use The Incredible Machine as a puzzle or a toy. As a puzzle, you're given a particular starting setup, a set of parts, and a goal (e.g., use a rope, two pulleys, and a toaster to catch the mouse). Your job is to add your parts to those that are already there to build a machine that, when you start it, will achieve the goal. You usually end up creating a Rube Goldberg-like device that accomplishes a simple goal using a ridiculously complex or unlikely arrangement of parts [143].

When using the game as a toy, you can simply build your own devices and then turn on the world and watch them run.

Although the world of The Incredible Machine is severely limited, it offers an excellent match between the system interface and the world interface. Because all of the objects work the way they ought to, it's great fun to experiment with the different pieces and create little machines to carry out creatively complex means to simple ends.

An action game that tries for the same level of match between the system UI and the world UI is the console game Rocket: Robot on Wheels. In this game you control a little unicycle-like robot named Sprocket as he wheels around an amusement park, trying to clean up the damage done by the park's disgruntled raccoon mascot. The game all takes place in 3D, and its distinguishing feature is a sophisticated physics simulator that applies to the whole world. So it's not just Sprocket that behaves in a physically realistic way; any time he affects the world, it responds in a realistic way as well.

This consistency is both a blessing and a curse. The uniformity between system interface and world interface is great, but managing to accomplish some of the skill tasks can be very hard. At one point you need to guide Sprocket across a lake by hopping over a series of pillars in a limited amount of time (or else they sink into the water, taking you with them). I found it very hard to get my little unicycle to move fast enough and precisely enough to cross the lake. Because everything was physically based, there was no way for me to choose an easier setting more suited to my ability (or lack of it) to accomplish this challenge.

A close fit between what you seem to be able to do in an environment and what you actually can do in the underlying world is the hallmark of a good match between the system UI and the world UI, and a sign of clarity and conceptual unity.
Language

Perhaps the most frustrating aspect of computer characters today, and one of the greatest challenges facing those who would develop better ones, is the question of language. We expect people to understand what we say, and what they say to us can reveal what they’re thinking and feeling.

The ability to carry on an interesting conversation isn’t sufficient to create a responsive, human-like character, but if we’re to communicate with such a character, some kind of language is necessary. Creating and understanding any human language has been a central focus of research in artificial intelligence for decades, and still appears to be far away. It would be nice to have something that allows us to have meaningful exchanges with computer characters in the meantime.

If you’ve ever spent time in a country where you don’t speak the language, then you’ve experienced firsthand a frustrating mismatch between the system and world interfaces, but not in the way we’ve seen them so far. In most games, the system interface is more flexible than the world’s. But when you don’t speak the language, the problem is the other way around: you’re surrounded by intelligent and helpful people, but you lack the tools you need to communicate with them.

The field of natural languages is enormous, and there’s no way to even broadly describe the field here. Pinker’s recent popular description of the field explores the subject, as well as reporting just how much we still don’t understand [111]. The sheer difficulty of the task is beautifully illustrated by a well-known bit of wordplay: interpret the phrase “Time flies like an arrow, but fruit flies like a banana.”

These kinds of ambiguous sentences also show up in real life. In December 2003, I saw this message on the outdoor marquee of a hotel in Vancouver, British Columbia: “Prime Dates Still Available for Christmas Parties.” In addition to the meaning they probably intended, there are at least three more ways to interpret this message.

Writing a computer program that can figure out what these sentences “mean” is very hard.

Happily, our needs here are more modest. Since computer understanding of language is so limited, we need only look at some examples of languages that match that limited ability and are also easy to learn and use.

There are probably two principal uses of language in contemporary games and in the story environments not too far beyond the horizon: information exchange and chatter.

Information exchange is when two people are talking with the purpose of sharing information. It can be factual information, an emotional outpouring, or even just a casual conversation. A good two-way, natural conversation of this type is hard for computers to fake, since each participant needs to be paying attention and responding to what the other person is saying.

Casual chatter is just idle talk with no real purpose. Computers can fake this much more easily. Starting with the famous 1966 program Eliza, which simulated a Rogerian psychotherapist, computers have been able to carry on better aimless conversations with people [138]. Such programs have come a long way; the Alice program today is able to respond in an engaging way on lots of different topics [136]. Most of these chatbots aren’t explicitly designed for the purpose of sharing information: the goal is that they can carry on a casual back and forth, like two people meeting at a cocktail party.

In recent years, people have begun to apply chatbot technology to commerce. So you might go to a website chat window and think you’re speaking to a customer service representative about a problem you’re having, but in reality you’re talking to a chatbot who’s designed to act as a first line of contact with customers. The chatbot is programmed to try to diagnose your problem and suggest a solution. If it can’t figure out what you’re saying, or doesn’t know how to help, or “senses” you’re getting frustrated, you may be switched to a real person. Sometimes that switch is announced, and sometimes the person simply picks up the conversation where the chatbot left off so that you, the customer, never know that such a handoff occurred. Chatbots are also used as salesmen, answering questions about products in a way meant to be more personal than a product description flyer.

Because children use language very flexibly themselves, they are often less critical of what comes back to them from a chatbot than an adult would be. Some marketing companies have used this observation to create chatbots that masquerade as real children and act as a child’s “friend” on an instant-messaging platform. The purpose of speaking to the child and pretending to be his friend is to use that friendship to market products to the child [85].
Chatbots in general aren’t going to be useful as central characters in storytelling for a while yet. The problem is that they can still be unmasked too easily: their knowledge of the world is so limited that as soon as you ask a question about something they don’t know about, or about the implications of something they’ve said in a larger context, the illusion is shattered.

Chatbots are great for bit parts with very constrained interactions. A chatbot would make an excellent bureaucrat: if you ask it a specific question about its specialty, you’ll get a specific answer. Anything else would be met with a polite shrug of the linguistic shoulders, or a suggestion that you speak to his supervisor. Chatbots also work well in very formal roles, like a waiter in a fancy restaurant, where there are distinct limits on the nature of the interaction they’re supposed to have with clients.

Given these limitations, how then will we communicate with more interesting computer characters in a story? Perhaps the best way to go is to invent a new language.

In 1887, Dr. L.L. Zamenhof proposed a new language for international communication, which he called Esperanto [34]. This language is much too general and sophisticated to simplify our problem; writing computer programs to understand Esperanto would be on the same order of difficulty as writing programs that could understand English, and then there’s the problem of teaching it to people.

The language Loglan [18] (a contraction of logical language) was invented by James Cooke Brown in the late 1950s. The language now has over 10,000 words and algorithms for making new ones. Loglan aficionados have written original essays and stories in the language and have translated poetry and novels into Loglan.

Because Loglan sentences are constructed with very specific rules, it seems initially to be a reasonable candidate for talking to computer characters. However, there are two immediate problems. The first is the learning problem: to use Loglan, you have to learn the rules of construction and amass a large enough vocabulary that you can express yourself and understand what others say to you. With 10,000 words in the lexicon, this is a lot of work, and much more than almost anyone would be willing to go through just to talk to computer characters. The second problem is that Loglan is simply too expressive: although a computer can break down the words of Loglan in a strict manner, the process of interpreting those words (that is, determining what it is that a person is actually trying to say) doesn’t appear to be any easier than with any other human language. In other words, getting a computer to understand what someone means when he speaks Loglan is still an overwhelming problem.

One way to get around this trouble is to greatly restrict the subject matter. If there isn’t much to talk about, then the computer might have a fighting chance.

For example, the computer program Shrdlu [138] (pronounced shirred-loo) was able to carry on very limited conversations about a world of simple colored blocks. The blocks didn’t originally exist in reality, but the computer was able to “move” them around in its memory to simulate what it would do in a real world. For example, you could ask “Can a block be supported by a block?” and Shrdlu would correctly reply, “Yes.” You could command it to “Put a small red pyramid onto the green box,” and Shrdlu would do so. If you then asked, “Is the red pyramid on top of the green box?” the response would be “Yes.” So in this very limited world, Shrdlu could carry on a simple conversation.

But there were all kinds of limits to what Shrdlu could understand. For example, you could completely confound Shrdlu by typing in a question like this: “If I were to take a red box and balance it on top of the green box but just a little bit off to one side, so that it was just barely balanced, and then I stacked up some other blocks and they fell down and shook the table, would the red box fall on the blue one?” Understanding a question like this, and all it implies about physics and blocks and the rest of the real world, is still far beyond the capabilities of even today’s systems.

So narrowing down the domain of discussion isn’t enough; we have to narrow down how we can talk about it as well. This suggests a language with a small vocabulary, and controls on how sentences can be built.

One of the easiest ways to do this is with a language that is symbolic.

By a “symbolic” language I mean that words are expressed by simple pictures or drawings. The simple drawings that distinguish men’s and women’s bathrooms are symbolic, as are the no-smoking sign and the image of an airplane taking off used to mark the departure gates at an airport. You can make little sentences with these signs. For example, putting a man’s bathroom sign next to an arrow creates a two-word sentence (made of one verb and one noun), which we might verbalize as “men’s bathrooms are in this direction.”
There are other ways to create a symbolic language: Egyptian hieroglyphics and Mandarin Chinese both use visual symbols for words rather than collections of letters, as in English.

A different kind of visual approach is taken by American Sign Language, where many English words have a direct corollary. Though some ASL symbols represent composite concepts, I think ASL is closer to a translation of English into a different representation than a new language of its own.

I find new symbolic languages to be particularly appealing for today’s interfaces for a variety of reasons. First, they can be designed to be very simple. As we’ll see below, some languages can get by with just a couple of hundred words for limited domains. And if there’s lots of context to help players understand the words, and the visual images of the words are well executed, then players can read and write sentences without the need to first memorize the entire vocabulary.

Second, symbolic languages normalize everyone’s experience of the environment. For any two players who are communicating, it doesn’t matter if they are both people speaking the same natural language or people speaking different languages, or even if one is a computer: everyone’s using the same words and the same construction rules. So native speakers of Japanese and Italian can converse in this new and limited, but common, language.

Third, symbolic languages let us communicate with less ambiguity. In many artistic expressions carefully-controlled ambiguity is essential, because it’s in the undefined places that the audience can exercise their imagination and enter the work. But at this stage of computer language, just making ourselves understood when we’re trying to be as direct as possible would be a step forward.

There are other benefits to these languages, including the simple fact that playing with a visual language is fun, and that there’s something appealing about communicating with pictures.

Written languages have a standard layout: for example, English is written left-to-right, top-to-bottom. Each visual language has its own rules as well. Generally each word must be adjacent to a previous word, so that a “sentence” is formed by a cluster of symbols. Often compound words are created from simpler words by overlapping them. For example, a single stick figure might mean a man, and a baseball could literally mean a baseball, while a baseball in the man’s hand would indicate a baseball player.

The pictures that are used for words in a visual language make a world of difference both in how easy it is to use the language and in the aesthetics of the resulting sentences. An attractive-looking language is simply going to be more pleasant to use than one that isn’t as nicely designed.

Another aspect of the pictures is that they need to be representational enough that they bring the right idea to mind, but not so specific that people assume they mean the very same thing as their closest natural-language counterpart.

Visual Languages

The visual language Bliss was created by Charles K. Bliss and described in his 1965 book [11]. The language has about 2000 symbols and is used by for communication by paraplegically disabled people. Figure 21 shows a few words in Bliss, and Figure 22 shows a couple of sample sentences.

Bliss is too rich for our purposes. With a 2000-word vocabulary, it’s both too much for a player to learn for casual entertainment and too expressive. And since it’s meant for human-to-human communication, Bliss allows people to express themselves in ways that would go right over the head of the computer we’re talking to.

Another visual language with simple construction rules is The Elephant’s Memory [70], designed by Timothy Ingen Housz. The Elephant’s Memory has wonderful aesthetics, and it manages to get by with only 200 basic words in the vocabulary. Figure 23 shows a few words in this language, and Figure 24 shows two sample sentences. The way words go together isn’t precisely defined: there is freedom in how the words are combined to construct sentences. This freedom provides some expressive nuance for people communicating with each other, but it makes it more difficult for
the computer to unambiguously determine what a player is trying to say. Compound words in this language are almost like poetry, rewarding and sometimes requiring sophisticated reasoning and interpretation.

A simple visual language was used in the computer game Trust & Betrayal: The Legacy of Siboot. In this game, the player communicates with the other players in the game (all computer-controlled) by creating short sentences in a graphical language called Eoyal. A very pleasing aspect of this design is that this is exactly how the computer characters speak back to you. You also find that the type of information that the computer characters share among themselves follows the same lines, so it feels like everyone in the game is using this language. Since it's the only way characters can express themselves, using this language gives you the feeling you're on a level playing field with the other characters.

Figure 25 shows a couple of sample sentences from the game.

The language of Trust & Betrayal only allows you to construct legal sentences. When you start to write something, you're given a choice of all the symbols that can appear at the start of a sentence. When you've chosen one, the list is then replaced by all the symbols that are allowed to follow that word. In this way you build up a sentence by choosing the next symbol that expresses what you want to say. Since you don't transmit your
sentence until you’re done, you can always back up and rephrase yourself if it’s not coming out the way you like.

The language has a vocabulary of only 80 words. Since words are only available when they can be legally used, it’s easier to remember their meaning from context, and you don’t have to memorize them all before you play the game.

This kind of language has been used more recently in the online browser-based exploration game Banja. The computer-controlled characters in the game speak to you (as you play the title character Banja) using an icon-based language. The Banja language is one-way: you can’t create your own sentences. A nice touch in the design is that some of the Banja icons are animated. For example, a watch face that goes from night to day indicates the passage of one day. When a barman offers you a drink, the offer is represented by an animation of a glass travelling from his hand to yours.

Icon-based languages work for people who share the cultural ideas they represent, regardless of which human language they speak, and are able both to piggyback on, and be independent of, the written words they roughly correlate to. Someone looking at a representational icon can generally get a good rough idea of what it represents, but he’s aware that it might be different from the natural-language term that it resembles.

Languages with strong construction rules make it easy to create sentences that are structurally sound, particularly when the system only allows words that are legal in the sentence that’s been made so far.

These types of languages aren’t the only techniques for communicating with computer characters. Semaphore, drumming rhythms, gestures, music, and any other form of structured interaction can form the basis of a language. The trick is to make sure that the language is so closely matched to the underlying character’s abilities that both the player and the character are able to make themselves understood, and the player is able to say the sorts of things that he might reasonably want to say. Then the system interface’s language matches the world interface’s interpretation.

Procedural and Random Stories

As we’ve seen, writing good stories for general audiences requires skill and craft. If we could shift some or all of the work in this process onto the shoulders of a computer, it might free us up to attend to other interesting questions.
Interactive Storytelling

Many people have thought about this possibility, and they've produced a wide variety of theories, projects, and stories. A recent bibliography can be found in Lang [79].

Looking at the output of these prose-generating programs, we can see that they still have a long way to go. The structural pieces are present to different degrees, but the stories lack subtlety and nuance. They have the form of stories, but they don't seem to actually be about anything.

It's going to be a long time before computers produce any kind of story that we would actually want to read as a story, rather than as a novel. For example, the book The Policeman's Beard is Half Constructed [25] is a collection of poetry and prose created by a program called Racter, under the control of its developers. It's interesting from a computer-science point of view, but hardly a coherent piece of writing.

Some more modern systems are able to produce plausible fables and very short stories, but they still require a very generous reader to see these as anything more than proofs of concept.

There's nothing new about structural approaches to writing and story construction. Aristotle started it in the Poetics [3], and many people have picked up the ball since then. The book Seal This Plot [72] presents stories in terms of a few general categories and variations on them. The book (and related software) Plots Unlimited [119] presents over 1400 plot fragments and rules for stitching them together into a single story. These books, and others like them, can be useful tools for stimulating your imagination, but they don't actually write stories.

Whether or not computers will ever be able to write stories that will actually interest us as people is a difficult question. Because stories are such human creations, this challenge is similar to the problem commonly referred to as **hard artificial intelligence**, which in this case means creating a program that can converse like a person in all the ways that one chooses to care about. Debate rages on whether or not such a program is on the near horizon, the far horizon, or is in fact even possible.

The Frankenstein Effect

The main problem with computer-generated stories is that when it comes to people and art, analysis does not easily lead to synthesis.

This is what Dr. Frankenstein learned. He studied anatomy deeply enough to enumerate all of the pieces that go into a human being. He reasoned that this analysis meant that if he brought all the pieces together in the right arrangement, he'd have himself a human being.

This approach works just fine in the physical sciences, like chemistry: if you bring the right molecules together in the right proportion and mix them the right way, you'll get the same result every time. That's why we can have huge manufacturing plants cranking out vast quantities of synthetic chemicals.

But people certainly don't work like that, and neither does art.

If it did, we'd have very few flop movies, boring poems, or ugly paintings. And the world is filled with such things. People don't usually make unpopular art on purpose. Consider film studios: they would love to do everything in their power to make sure that every film they make is a smash hit. We know a lot about movies; as I mentioned before, Citizen Kane was made in 1941, and there have been tens of thousands of films since, resulting in runaway hits, flops, and everything in between. Everyone involved in making movies wants the work to be successful, are serious about the craft, and understand it as well as anyone. Why then do filmmakers sometimes make movies that are generally agreed to be awful?

It's because art doesn't get made by putting some of this next to some of that and heating it in a test tube. Art is just about impossible to predict.

The principles of storytelling and gaming are at their strongest when they tell us what we know does not work. People are turned off by lead characters that are boring, stories without internal conflict, and games that are based entirely on luck. If we think of artistic works as fishing boats in an ocean, the principles tell us something about the boundaries of the islands and continents. They tell us how to avoid getting beached on submerged reefs, but we have to hunt for the fish on our own.

This is why computer-generated art of all kinds is sterile. People who write programs to create stories, poems, paintings, music, or any other art are falling prey to what I call the Frankenstein Effect. They believe that because they have analyzed the pieces of something, they can create new variations of it by assembling the right pieces, and somehow, magically, art will emerge. It's easy enough to make works that are similar to art. You can get very close on all sorts of statistical measures, and sometimes you can even fool people into thinking that a computer-generated piece of art was made by a person.

But this is the artistic analog of junk food. Junk food is usually very tasty, and if you're hungry, it will fill your belly and make you feel well-
fed. It might even give you a shot of energy. But it has little of the nutrition your body needs to survive, and eventually you need to eat food with real nutritional value. Junk art can fill your imagination and make you feel like you've had a good artistic experience, but it has none of the human qualities that artists put into their work to give it spiritual and emotional value.

Just as junk food lacks in nutrition, junk art lacks in meaning.

Some computer-generated art can have value. A person may look at the stories, poems, paintings, or other output of a self-guided program and select works that he thinks are interesting or beautiful. Like someone recording the sound of a thunderstorm, he may see beauty in these forms that was not consciously put there by any artist with something to say. Finding beauty is always something to celebrate, but if we're thinking of writing programs that require this interpretive step, we have to think about why we're doing so. There are many skilled and talented artists who have something important to say and the tools with which to say it. I believe that appreciating intentional works that have been created to communicate an idea is ultimately a more rewarding use of our time than searching for value in works that have no inherent meaning.

Research and analysis of the arts will continue, and slowly we will come to understand these very human activities better. But for now, the best art comes from people, and there's just no reason to pick computer-generated junk art when high-quality human art, and the artists to make it, are both so readily available.

In the meantime, we should recognize the magnitude of the gulf that exists today between the analysis of art (which gives us principles) and the creation of new art.

Audience Interpretation

I believe we're best served today by actually writing our own stories, rather than creating programs to make them for us.

We might alternatively try to create stories by using random events, in the way some modern composers use random sounds and events in their work. For example, the concert piece Imaginary Landscape No. 4 [21] is scored for 12 radios tuned at random.

There are two levels of meaning in these accidental, or uncontrolled, works. One comes from the creator of the surrounding work (e.g., the instructions for when and where to play highway sounds, or how to tune the radios). The author may, for example, be saying something about the inherent artistic qualities in the natural or man-made world. But obviously, the person responsible for executing the work is not expressing anything specific in the random elements themselves, since that work is by definition uncontrolled and therefore without intentional meaning except when interpreted in context.

The other meaning comes from the audience, who may choose to create their own personal interpretations. As humans, we are remarkably good at detecting patterns, whether seeing the face of a friend in a crowd or hearing the sound of our name being called out in a noisy airport. We often detect patterns where none exist, seeing shapes in the clouds and hearing songs in the wind. If we find structure in computer-generated poetry or in the sounds of traffic, it's because we as the audience are choosing and arranging what our senses are reporting to us. In other words, we extract meaning from these random presentations by imposing our own structure upon them. Without that structure, they remain meaningless in themselves and carry only whatever contextual meaning they have from their relationship to the other pieces of the work.

So we can piece together a series of events as a coherent narrative, but only if we do a lot of the work. In toys like The Sims, the events that happen to the simulated people in the world are not strictly random: they result from each person's state of "mind," what he's been up to recently, and his economic status. People without a penny don't build swimming pools, and those without friends don't throw wildly popular parties. When people tell stories about the lives of Sims, they're acting as creative storytellers, weaving a narrative out of the raw material of observed events. Shared toys like The Sims Online allow us to play together, but they still don't explicitly generate narratives.

As I discussed at the start of Chapter 8, authoring a story is very different from making one up for ourselves as we play with a toy. Similarly, making up a story from random events can be fun, but to make it entertaining for others, we need to use craft and the tools of authorship to arrange these events to shape a coherent story.

Random or unpredictable events can be fun, but great stories are still the work of great storytellers who have something to say.