

## Chapter 8

# The Future Of Game Design

### Key Topics:

- ◆ *Benefits of design*
- ◆ *Checklist of good design elements*
- ◆ *Where games are now*
- ◆ *Where games are going next*

In this chapter, we will review basic gameplay essentials and then take a speculative look at how game concepts and design may evolve in the future. First, though, we'll debunk some of the myths surrounding formal design and its effect on the creative process.

## The Necessity Of Design

I'm going to start with a quotation that you might think has a familiar ring: "In these companies there is a designer who directs by word alone and who seldom or never dirties his hands writing code. Designers say to the other team members, 'Program this feature in such-and-such a way,' and yet they do no real work themselves."

Does that sound like something you've heard before? Some dyed-in-the-wool developers still think a formal design isn't necessary, and in a moment we'll review the arguments they usually give to support that view. However, first I have a confession. The quotation above didn't quite read like that in the original. Instead of "designers," it referred to "architects," and, for "writing code," substitute "cutting stone."

The statement is from the thirteenth century, and it bemoans the passing of the "good old days" (the twelfth century) when stonemasons could just fling up a new cathedral without having to bother with architects' plans. But back then, as now, there was good reason to abandon the old trial-and-error process. Building a cathedral was a decades-long project that involved a team of several hundred men. Also, cathedrals that were built according to intuitive,

unplanned, “see how it goes” methods were proving unreliable; that is, sometimes they fell down. Unlike software, when cathedrals fell down they killed people.

I guess the quote shows that people have always lamented the need for change, because change normally goes in the direction of making things more formalized, logical, and safe. The pioneers don’t care for the sound of that, be they cathedral builders, movie makers, soldiers, or astronauts. Or even game developers.

## Don’t Be Afraid To Plan

When I first began working in the computer games industry back in 1995, there still was a very reactionary opposition to detailed design and the formal development process. Generally, my experience was that artists were willing to embrace the need for change but that programmers tended to be quite fearful of it. A couple of objections the programmers would raise were, “We don’t want designs handed down from on high,” and, “You can design a business application, sure—but every game is a brand new project.”

*Fear* perhaps is too strong a word. These were intelligent, talented, experienced people, and naturally they were wary of changing the way they did things. As Case Study 8.1 reveals, they resisted the concept of detailed design mainly because game development had historically been a hit-and-miss affair. The idea of writing the design in stone and sticking to it throughout an 18-month project seemed absurd. And there’s a good reason for that: It is absurd. As we discussed in Chapter 4, the gameplay spec is an evolving document. It is not a Holy Writ; instead it is a combination of a vision statement, a blueprint, and a repository of changes.

Let’s return to those objections to the design process: that it kills the fun of creative coding, and that it is logically impossible to plan a new thing anyway.

When George Lucas was making *Star Wars*, you can bet he had a script, storyboard, and shooting schedule. Everybody on the crew knew what they were supposed to be filming every hour of every day. Do you suppose that killed the fun? Do you suppose having a detailed plan stifled people’s creativity? Not a bit. There was room for improvisation. Most famously, in *The Empire Strikes Back*, Han’s reply when Leia says she loves him was an idea that Harrison Ford had on the spot. There would have been many other aspects of the movie that were added by various people during shooting, and other ways it would have changed in postproduction. The script wasn’t a straitjacket: It was a visionary framework that actually aided creative input.

Remember as I wrote in Chapter 4 that the existence of a design doesn’t mean that the designer is necessarily the sole author of the project. Formal design certainly doesn’t mean that developers will become serfs toiling without any control of their destiny. Ideas might

### **Case Study 8.1 Design Saves Time**

A couple of years ago, I was working on a sim-type product (which we'll call *Catastrophe*) in which players had to build and manage a kingdom. Early in the design process, I began to consider having a bunch of initially neutral city-states that would declare allegiance according to various factors, including proximity to a player's units (especially army units) and how well each player was managing his economy.

At that stage, the project had just one programmer assigned to it. I asked him to knock up a quick testbed. "Just dots on a screen that change color with allegiance will do," I said. "I want to get a rough idea of how far the allegiance effect will spread, because that feeds back into the player's economy. If we're going to get a runaway effect, I'd rather start building damping factors into the design now—or drop the idea altogether, if it's not workable."

A few days later, he had a display with dots, but they didn't do anything. "I haven't had much time because I've been busy with DirectX," he told me. "But the game engine will be ready in a few months anyway, so can't we test your idea then?"

"I was hoping to be past the purely experimental stage by then. I need to get a guesstimate now to put in the design spec."

"Oh," he said. "Well, we hardly ever stick to the spec anyway, so why bother?"

come from brainstorming sessions in which all can contribute. The design can and will change during development, as the spec is an evolving document.

And what about the difficulty of writing a design when your game is something completely new? Some would argue this is a logical impossibility, but this is evidently not so. If it were, even a description of the new concept would be impossible, because just a five-page treatment is the first step in the design process.

Again, the fallacy derives from a misunderstanding of what the design is supposed to achieve. I have to stress once more that a design (and especially a design for iterative development) is never intended to be 100 percent correct from the start. Instead, initial design provides you with a first "best guess" that, since it typically takes only 12 man-months out of a total development of 160, pays for itself if it's only 15 percent right. (My own estimate is that a good design, even on a highly innovative project, is actually closer to an 80 percent match with the finished product.)

So, to summarize, why document the design? Because the approach of developers to date has been like medieval alchemy. "Creativity can't be planned for!" they gasp. But in fact it can, and as developers we should strive to be scientific and not let ourselves be held back by superstition. From the outset of your design, you can apply principles based on game theory and storytelling techniques. These principles can save hundreds of hours of development time—and that means tens of thousands of dollars.

## Why Design Is Fine

The advantages of a good design are that it is good for morale, good for the budget, and good for the game.

### *Morale*

Teams thrive on a shared vision and work with more enthusiasm when given clearly defined goals that reward their efforts.

*“When men have a mission, they arrange matters to accomplish it. Without one, they don’t.”*

—Gifts of the Night (*DC Comics, 1999*) by Paul Chadwick and John Bolton

That is what a detailed design is: a shared vision, accessible to the entire team that everyone can analyze and comment on. The tier system of design that this book describes allows the whole team to focus on creating the game in stages. A monthly or bimonthly turnaround means that each tier represents an achievable goal wherein the contributions of all the team members are visible.

This is very different from the kind of process that programmers legitimately fear, where they are simply issued with lists of tasks each week. Incredible as it seems, such a system was in fact applied to all internal developments at a major publisher until very recently. This is the exact opposite of detailed design as I have defined it, as it makes the design opaque and allows none of the coherence and *Gedanken*-experimentation that a tier-based design and development will encourage.

### *Budget*

Robert H. Dunn, in *Software Defect Removal* (1984), estimates that a design error left undetected until testing will take (on average) 10 times longer to fix than one detected at design time. With today’s much bigger projects, the statistic is probably even worse. Imagine all those problems showing up a month before your shipping date. Small wonder that some old-time development managers have gotten the idea that games can be completed only by catching catnaps in between coding sessions. But many of those problems could be avoided—and the developers would get better working hours—if things were handled better at the design stage.

Any detailed design allows scheduling, resource inventorization, control, and tracking. Without these, it is impossible to accurately budget a project, and, without an accurate budget, it is unlikely the project will ever be funded—unless you have a rich and very indulgent uncle.

### *The Game*

An iterative or tier-based design is even more vital for any product that has to incorporate new or untried features. If you absolutely, positively have to ship by a certain date, the tier system will let you fit the design to that date. If development is slipping, you may have to

trim some features (or even drop them entirely), but the modularity inherent in the design allows you to do so while still keeping control. Case Study 8.2 stresses the importance of keeping all designs up to date.

If you don't have a design, you may find yourself frantically tweaking and changing gameplay the day before shipping. Doing so can never be a good idea. If you didn't get it right earlier, how could an eleventh-hour change do the trick? The advantage of a clear design is that, not only does it describe and plan for the gameplay features you expect, but also it gives you a rational framework within which to make any required changes.

## Essentials Of Game Design

Before moving on to consider where entertainment software is headed, let's review some of the essentials of game design. You can rely on a few useful questions as a litmus test.

### Is It Original?

Few games are completely original. As we discussed in Chapter 1, *originality* is a relative term in any case. But if your design is worth developing, it has to have some features that no other game of its type has tried before.

Of course, these features are the ones that will give you the most trouble in development. Even if the feature is one that's well understood in another genre, importing it into your game will have new effects—which is precisely why you should identify and consider those features right at the start of the design process. As I have constantly said in this section, development will entail changes to the design, but knowing what your key features are—and anticipating the development difficulties they might create—is far preferable to blundering about with no plan to guide you.

Original features don't have to be gameplay features. You might simply have decided to do a wargame with funny cartoon characters. And why not? After all, look and feel alone can make a game different. As long as there is something to distinguish your game, you will know where to focus your main efforts during development, and you can be sure the finished game will stand apart from others on the market.

Right after the initial treatment stage, compile a list of unique selling points (USPs) that define how your game concept is unlike any other. These are the things that make it special and will justify a development team spending a year or two creating it. If the concept has no original features, junk it. The world just doesn't need another female archaeologist with two big guns. Do it differently, or not at all.

### Is It Coherent?

Good game designs are built around a core vision that works like a seed crystal. As the game is built, if changes need to be made, the core vision keeps them targeted on the final goal. It ensures that the game features serve a common thematic purpose. For example, if you

### **Case Study 8.2 Keep The Design Up To Date**

A few years ago, Rachel was brought in as design consultant on a CRPG that we'll call *Golem*. This project had been in development for over a year with no real sign of progress, and the feeling was that it was turning into a house of cards, with no cohesion or robustness in the design. The development director's hope was that, if they could tighten up the design, the software issues might also come under better control.

After chatting to the team and looking at what they had so far, Rachel took the design spec home to read it. When she'd finished, she had a three-page document of queries but wasn't even sure she'd been given the latest version of the spec. "This describes a third-person game," she said to the lead programmer, "but what I've seen is first-person."

"We needed to cut the number of polygons on screen," he explained. "First-person is one less character."

"Fine, but that will change the interface design also."

He agreed. "Bill, the designer, said we'll tweak the interface when we've got the other bits working."

Next, Rachel spoke to Bill. She liked much of his spec and told him so. "There are some great features. I especially liked that the player will be able to prime spells ready for casting. That's a neat tradeoff of versatility against speed."

"Thanks," he said. "Mind you, I can't stand CRPGs. What I really wanted to do was a first-person shooter."

"I thought it was tending that way. Also, one of the artists told me there's a high likelihood of dropping the thief and warrior characters now, so you can only play as a sorcerer?"

"Yes, that's right. The first-person view saves character artwork, but there were too many other issues thrown up, not least being the complication of designing levels that would be challenging to all three character classes."

"It's the old combinatorial explosion problem," Rachel said. "Three character types means nine times the work."

"At least. The new design is a lot simpler."

"Good." They were getting somewhere, she thought. "I was sure there must be a new design. Can you run me off a copy?"

"It's up here!" Bill laughed, tapping his head. "We're so far behind schedule, you didn't think I'd have time to write up every design change, did you?"

The most frightening thing is that Bill wasn't kidding. They never did have any further design documentation from that date, apart from technical specs drawn up by the programmers. Rachel recommended a full retrofit of the game design to date, with further changes to be placed under change control. The company decided against these measures on the basis that they would cost too much time at such a late stage. In fact, it was the failure to implement them that proved costly: About a year and \$750,000 later, *Golem* was canned.

intend to make a strategy game that assists the player to plan attacks easily, you might think twice about a multilayered interface that, although original, militates against the core vision of ease-of-use.

A core vision pulls the look and feel of the game together too. Not only do these “chrome” elements assist each other, but also they should enhance the experience of playing the game. When all the elements of the game are working to the same end, you have a product whose internal resonance guarantees aesthetic appeal. On the other hand, a combination of incoherent game features and artistic styles is simply a mess.

## Is It Interactive?

Ezra Pound had a saying that he was fond of quoting to poets: “Whatever can be said well in prose can be said better in prose.” He was trying to stop poetry from trying to do something it wasn’t suited for, to get poets to instead concentrate on the unique strengths of their own medium. In the same way, the entertainment software of the future will hopefully move away from other media like cinema, defining itself by what it can do best—in particular, by offering maximal interactivity.

We discussed specific kinds of interactivity in Chapter 3. A broader but still useful distinction is between high interactivity and low interactivity.

Most computer games today use high interactivity, as the player has a strongly proactive role. The story should unfold directly from what the player sees and does, because the player’s expectations are that his role is proactive, which means he will be impatient if forced to sit back and be told a story. The best designs avoid lengthy dialogues, chunks of clumsy exposition, and extended full motion videos (FMVs) that remove control from the player. It is possible to tell a story implicitly and with great economy; you do not need lots of dialogue. Think of entering a drifting spaceship. One of the cryo pods is damaged, and a long-dead body is slumped across a table nearby, a bottle of pills in its hand. You don’t need to spell it out; the story is all there.

Low interactivity, on the other hand, is reactive. At its simplest level, it is represented by the audience at a play, hissing and calling out boos and yeahs—as long as the actors take any notice, that is! You use low interactivity when programming a stack of music CDs to play the tracks you want, or when channel-surfing on the TV.

Low interactivity is a way for entertainment software to reach a larger market, since more people choose reactive leisure (such as watching TV or a ball game) over proactive leisure (acting or playing a ball game). In the more story-oriented genres that will evolve out of today’s adventure games, we should see flexible narratives that the viewer can dip into with as much interactivity as he wants.

## Is It Interesting?

In Chapter 3, we examined Sid Meier’s description of a game as “a series of interesting choices.” We also saw that interesting choices are difficult choices. One effect of this is that

the better designed the game, the harder it is to program the AI, because a good game is a game you win by smart playing. Fortunately, artificial intelligence is getting better, as you can see by the uncannily humanlike behavior of the bots in *Quake*, for instance. This is just as well, because the demands of games in the future will push AI to the limit.

An elementary grasp of logic tells us that just because all interesting choices are difficult doesn't mean that all difficult choices are interesting. Being presented with six treasure chests, five of which are booby-trapped and where there is no clue to guide you, is a difficult choice but distinctly poor gameplay. (And poor storytelling, too.)

So don't include features whose only effect will be to annoy the player. The principle of a good design feature is that it presents the player with an upside and a downside, either of which may vary according to other factors. The choice is thus difficult to make, but, if the player gets it right, he is rewarded by success and a further layer of choice.

## Is It Fun?

You can include only so much in a game, so the trick is to make sure that whatever is included will enhance the player's enjoyment.

As an example, consider artificial life, which has hovered around the fringes of the games industry as a buzzword for the last few years. We are now starting to see a clutch of games that make much of their A-life credentials. I'm interested in the technology myself, but a pitfall exists for the overeager designer, because sometimes A-life (like real life) just isn't any fun.

An online RPG world in which all the computer characters are sustaining a real economy is worthless if the player-characters form a subculture that's unconnected to that economy. A strategy game in which you have to keep rounding up your soldiers because they've lost their nerve and deserted isn't clever; it's just witless. In an adventure game, having to remember to keep punishing your hero so that he does what you tell him is...well, actually that could be fun—a true “god game” in the Old Testament style—because it enhances your relationship with the game's central character. But it's for *that* reason that it's worth including, not because it's a nifty example of new technology.

The lesson is that the technology, like everything else, must serve the needs of the game, and thus the needs of the player. Never let yourself get carried away by a cool idea, new technology, or amazing effects. Always ask first: Is this fun?

## The Future Of Design

Suppose that, in order to be a game designer, you needed qualifications in business management, artificial intelligence, creative writing, computer science, and game theory. If that were the case, there wouldn't be many game designers, and we wouldn't get the best game concepts possible: We'd just get the best that could be dreamed up by people with more degrees than a thermometer.

Look at it another way. Suppose that the only novelists in history had been people who trained as typesetters as well. Some of them might have written great novels, works of genius, even. But those works would have been fewer, because the talent pool would have been smaller.

In the 1980s, the programmers of a game were most likely the designers as well. (They tended to do the artwork, too, which explains some of those early Spectrum games.) The classic games of the 1980s were created this way, but no one should lament the fact that formalized roles and more-accessible technology mean that it is now possible to be a designer without also having to be a programmer. It just makes the talent pool wider. The creative programmers will continue to design as well, just as some camera operators also write screenplays.

In the future, we should see more individuals such as screenwriters, artists, novelists, and board-game designers enter computer game design, which will revolutionize what we expect from entertainment software. Many of these people will have no idea of the limitations of the technology: It's not the job of the designer to worry about that, at least in the initial design phase. Not knowing what can't be done, they will demand it, and their development teams, rising to the challenge, will deliver games astoundingly different from anything we've seen before.

*"Most game designers are programmers and very familiar with technology. Programmers are held back in an artistic, creative way. They know the limits of technology."*

—*Roberta Williams, creator of King's Quest, quoted in Game Design 101*

## Making Designs More Generic

If there will be one overriding evolution in game design, I would say it will be a move towards starting out with a generic design. This is consistent with (and indeed a logical consequence of) the software factory model. Such designs will allow designers to start arguing from the general to the specific, which will make the design concepts more portable and robust.

Thus, instead of beginning with a design that specified game resources as food, wood, money, and piety, I now begin by defining attributes of resources. I might say that resources can be localized (physically resident somewhere in the game world) or nonlocalized (held intrinsically by the player and therefore not subject to capture by the enemy). They can also be collectable (waiting to be picked up) or creatable (requiring some action to spawn them), and they can be perishable (decaying over time) or indestructible. And so on.

By this model, piety is nonlocalized, creatable, and indestructible, whereas food might be localized, collectable, and perishable. The advantage comes when I move on to my next game—another wargame, but now set in the modern world and involving guerrillas and military juntas. I decide that one of the resources will be morale, which is spent whenever you want to repair damaged buildings. Broadcasting stations generate morale; friendly losses

or enemy propaganda reduce it; and it declines over time as long as you have soldiers on your own city streets. Rather than having to design this resource entirely from scratch, the architecture group already has a template for including it: nonlocalized, creatable, and perishable. In practice, the resource template is rather more complicated than that, and with more attributes, because it aims to allow completely customizable resources for any game.

Now, this can mean many attribute slots, but they are actually grouped in classes (attributes involving existence, persistence, and so on). In many games, those attribute slots would never be switched on or would simply default to some higher level of classification. The advantage is that you could, in theory, take units out of the terrorist game and drop them into the first game and they would still be ready to function—which obviously would be handy if you are aiming for any level of reuse.

## Nonsymbolic Design

If I throw a ball and take many high-speed photographs of its flight, I'll see that the trajectory the ball took is a parabola. But the ball didn't follow that path because gravity told it to move in a parabola. A parabola is just a symbolic concept in the analytical domain of mathematics, and the universe doesn't know anything about mathematics or analysis or symbols; these are human concepts. In reality, there are just a bunch of physical processes, each of which deals only with the processes and circumstances just before and just after it. So, the ball is at one position, and gravity tells the ball's velocity to change, and the ball's velocity tells its position to change.

This is the opposite approach to that taken in most software applications. There, processing power is at a premium, so the sooner you can go to symbolic constructs, the better. The tradeoff is that software can crash when your symbolic "shortcut" misses something that the one-step-at-a-time approach would have taken in its stride.

Researchers in artificial life have identified an analogous problem:

*"The classical AI approach has been criticised because the symbols and symbol structures on which planning and decision making are based are not grounded in the real world. The problem is that unequivocally decoding sensory data into a symbol and turning a command without error into its intended action may be unsolvable."*

—Luc Steels, *"The Artificial Life Roots of Artificial Intelligence"* in *Artificial Life* (MIT Press, 1997)

Here is an example: Suppose I am putting a monster into my new Frankenstein adventure game, and the idea is that it will jump out of its vat when the player enters the laboratory. Instead of putting in a lot of complicated AI to do with detecting humans and having the goal of wanting to kill them, I just choose the shortcut of placing a trigger tile inside the laboratory door. When the player steps on the trigger, the monster will appear and attack.

Okay so far, but what if the player manages to get onto the tower roof, jumps down, and, by some fluke, manages to land safely on the balcony of the laboratory? Now he can explore the lab, get all the power-ups, and read the journal about the monster (an entry that is supposed to be poignant if he's just fought and killed it, but that is meaningless otherwise). Only when the player goes to leave via the door does the monster climb out of its vat and growl, "You shall not steal my master's secrets!"

In the past, the nonsymbolic, step-by-step approach was not practical. The processing capability wasn't available to deal with that and graphics too. But now much of the graphics work is done by the video card, and computers are doubling in power every 18 months or so. At last, it is starting to be possible to create "uncrashable" games by avoiding the need to design using symbolic shortcuts. Case Study 8.3 compares nonsymbolic and symbolic designs.

## The Future Of Games

Computer games have been around for approximately 20 years. In the form we know it today, the medium is really only some 10 years old. Nonetheless, throughout this section, I've drawn on references dating back to King Solomon and Aristotle, and through Shakespeare to Orson Welles. I've related computer games not only to the technological leap of the last decade, but to 4,000 years of human culture.

I've done so because I don't see computer games as froth for kids, as mere "brain candy." If I felt that way, I wouldn't be working as a game designer. I believe that games—entertainment software—are potentially the most exciting development in the creative arts since man first drew a bison on the wall of a cave and started to tell a story.

But notice I said "potentially." It's that potential which we really haven't even begun to tap. Everything to date has just been the groundwork. It's in the future that those foundations will grow into an entertainment medium that will rank alongside literature, music, and cinema. And the future starts now.

Over the last decade, computers have revolutionized the way we look at home entertainment. However, the revolution so far has been only experimental. It has yet to give birth to a new medium in the way that the fusion of photography and drama created cinema, for instance.

*"No one has ever really looked at what the market wants, or even who it really is. And even when we find out, we often shy away from the lesson. GTI's Deer Hunter... a slideshow of a game, the US mass market lapped it up, fuelling clones from Big Game Hunter to Turkey Hunt until the barrel scraping began.*

*"Deer Hunter revealed that people were starving for game styles that the industry hadn't even considered making. It also showed that people would buy hunting products. Only one of these lessons was learned."*

—Owain Bennallack, writing in MCV, 2 July 1999

### **Case Study 8.3 Comparing Nonsymbolic And Symbolic Design**

In the original *Warcraft*, peasants collected gold by entering a gold mine and bringing sacks back to your town hall. At the start of the game it was always worth spawning peasants because the more peasants you had, the greater your revenue stream. However, there came a point when the peasants started to get in each others' way. Adding more peasants would then lead to "traffic jams" as the peasants encountered each other on the streets of the town and would have to back up to let others get past. The situation was alleviated by leaving wide streets. Additionally, it was not a good idea to place your town hall too close to the gold mine—giving a little more space also helped avoid traffic congestion.

Now, an economist could derive an equation to describe the flow of gold to the town hall. The factors would be the number of peasants, the placement density of the town buildings, and the distance from the town hall to the mine. We can imagine that it would be a pretty complex equation. The point is that the designers of *Warcraft* never needed any such equation. They simply programmed in the basic rules and behaviors and the economic simulation emerged directly from those.

Contrast this with a game like *Caesar 2*, which used underlying equations to create a simulation of an ancient Roman city. This approach is less satisfying because the player is not directly viewing the reasons for success and failure. Instead, when playing a game like *Caesar 2* (or any simulation of its type) you are trying to build an abstract match to the game's underlying equations in your head. The simulated economy and the gameplay are less visible, lessening the sense of immersion.

The designers of the next decade (including, possibly, many of the readers of this book) will take entertainment software from the level of a hobby and make it into a new art form. This is why I have repeatedly said that it doesn't matter if your product is a game. If you like games, all well and good. But, just so long as you create something that people can interact with and enjoy—and that couldn't be done better in any medium other than software—you are doing your job as a designer.

Games on a computer are more attractive, exciting, convenient, and immediate than games played on a board. Multimedia products provide more-interesting presentation and better look-up capability than a reference book. Computer toys let you play with unlimited resources. All these products are worthwhile, but none of them exploits the opportunities inherent in software to create a completely new medium. It is as if we were still at the point where movie theaters showed nothing but short films of onrushing trains and racing meets. What we have to do now is get to the point where a new technology metamorphoses into a new medium.

And what is the really exciting thing about entertainment software? It isn't going to be only *one* new medium; it's going to be a whole bunch of them.

## The Next Decade

As the entertainment software industry matures, there will be more formalization, creating a clearer distinction between genres. Why should products as diverse as *Tomb Raider*, *Alpha Centauri*, and *Riven* all be classified as “games”? Hard-core gamers may think there’s considerable overlap in the markets for those games, but the wider buying public would find them all quite different.

*“Another hugely derided game [is] Riven, the sequel to Myst. ‘It’s not a game!’ we cry. ‘Then stop giving us games and give us more of this instead,’ many punters respond.”*

—Owain Bennallack; MCV, 2 July 1999

We will see an increasing trend towards more-specialized entertainment software magazines and towards defining genres for display in stores. Just as in bookshops, the mass market won’t want to search through the equivalent of thrillers, sci-fi, and bodice-rippers to find what they’re looking for. You will know the genre you’re interested in, and there will be magazines and Web distribution sites devoted to just that genre.

And what will these genres be? We might expect them to be evolutions from the existing game genres, plus a few more that nobody has thought of yet. The demands of a wider market will mean that, in comparison to today’s games, they will be:

- ◆ **More accessible**—People will want products they can play right away.
- ◆ **More flexible**—The user will have more choice in how to use the product.
- ◆ **More realistic**—Vastly improved artificial intelligence, physics, and graphics will transform the products of the future.
- ◆ **More fun**—No more struggling against the game system; the mass market will not have the patience for underdesigned products.
- ◆ **Completely different**—Only the diehards will continue with games as we know them today.

## The Strengths Of Software

Let’s look at what entertainment software can do really well. If we were compiling a list of USPs for the medium, we could say that it has the edge over other media in terms of:

- ◆ **Depth**—The background can be far more fleshed out. The inhabitants of the game world can have their own independent existence.
- ◆ **Freedom**—The true payoff of interactivity is that the user can make the product deliver what he wants.
- ◆ **Persistence**—You can get engrossed for hundreds of hours, experiencing the ultimate in escapist entertainment.

- ◆ **Multiplay**—Entertainment software empowers groups of people with the ability to create a mutual narrative.

These are the areas in which we may expect to see real advances, as entertainment software (“playware”?) defines itself as different from other media. So, we expect to see an increasing trend towards multiplay and for game worlds to feature better artificial intelligence and physics that will enhance the verisimilitude of the setting. A much greater range of interactivity is also required, allowing the player to choose how the game is used. (Better artificial intelligence will assist here, too.)

It’s possible even today to wring an additional level of interactivity out of games. That’s what we’re doing whenever we use cheat codes. Although the cheat codes give some extra degrees of freedom, the downside is that they tend to be used only by the inner cadre of hard-core gamers (the very people who need them least) and are in any case unsupported. In most cases, using cheat codes doesn’t enhance the game; it breaks it.

Instead, games should try to empower the user with real choice. My ideal for a strategy game, for example, would be one in which I could choose my own role within the world: commander of an army, ruler of a civilization, or Populous-style god. This way, the same product could be both a wargame and a sim-civilization game. I’d also like to decide the kind of opponents I’d face, both by selection of the computer player AI and by altering constants in the game world. (Changing the likelihood of famine occurring could thus switch a computer player from peaceful farmer to desperate raider.) This is a whole other level of interactivity, letting the player choose *how* to play, instead of constraining them with the preconceptions of the designer. With the release of *Dungeon Keeper 2*, which has several options to let you customize the game you want, we are finally beginning to see this kind of interactivity being made available.

In the case of a storytelling or adventure game of 10 years hence, you might decide one evening to watch a software movie starring Victor Virtual. Later, you could decide to switch characters, or demand longer fight scenes. Or you might interfere more directly, by feeding clues to the hero. So you can experience *The Odyssey*, say, from the hero’s own viewpoint, or as Poseidon (the god he had made an enemy of) or as Athena (the goddess who occasionally helped him).

Okay, it’s probable that the specifics of these new media won’t turn out quite as I’ve described them, but the principle is clear: Interactivity will be about degrees of choice, as well as the choices themselves.

## The Crossroads Of Creativity

To get some idea of the different forms into which computer games will evolve, it’s helpful to look at the ways that they entertain.

### ***Games As Stories***

I recently heard about several development companies that have hired psychologists to help them inject emotion into their games. In many cases, these are the same companies that are treating development like an art when it should be a science, and now they are treating creativity as a science when it should be an art!

If you want emotion in your game, hire screenwriters, playwrights, novelists, poets, directors, or actors. But don't hire psychologists. This is so idiotic that it's laughable.

In Chapter 6, we looked at some perennial storytelling techniques that can be usefully employed to intensify the gaming experience. However, I have been saying that entertainment software must become a new art form altogether. Simply borrowing the techniques of other media such as cinema will not do the trick.

Famously, Walt Disney made it his goal to produce a cartoon that would make people cry, and he achieved this goal most notably with the death of Bambi's mother. One ending of the adventure game *Outcast* (Appeal) is, if not quite tear-jerking, at least poignant and emotionally mature. However, it achieves this effect entirely through cinematic techniques. The last 10 minutes of the game include a couple of sequences in which the player is proactive, but those sequences have no effect on the final outcome. The story in *Outcast*, although moving, is told by noninteractive cut scenes.

For entertainment software to make its mark, we need to adopt a new approach to interactive storytelling. This will come via improved artificial intelligence and full physics systems that do not *tell* a story but rather allow the player to participate in the creation of a story.

I have said that entertainment software must move away from old models like films, novels, and plays to find a new way to tell stories. In fact, face-to-face role-playing games could provide a useful template.

I have written role-playing games professionally, and I also run them as a hobby. Each week, when preparing a session, I begin by devising a setting—a town by a lake, for example, under quarantine because of a plague. I populate this with the main nonplayer characters (NPCs), whose goals I define. Thus, the notes might read: “Lord Shonu: crafty, cautious, extremely wealthy; wants the Key of Time but isn't prepared to die for it,” and so on. When I have all the NPCs, I can infer the story that would occur *if not for the players*. The players' characters (PCs) perturb the situation with their own actions, aiding some NPCs and opposing others. Because I know the NPCs' motivations, goals, and capabilities, I can then decide how they will respond, and so on.

The point is that the role-playing session itself becomes a process of story creation. None of the participants decides the plot in advance. Instead, it emerges from the actions that everybody takes, which means that it would be possible to run the same scenario with two player groups and get completely different stories. (For instance, in one game, the players

might kill the bad guys and rescue the princess and the king rewards them. In the other, they act so evil that the bad guys come to work for them, and they jointly collect the ransom.)

This is the kind of storytelling process that we will see in the computer role-playing games (CRPGs) of the next decade and, in modified form, in non-CRPGs also.

### ***Games As Visual Arts***

Entertainment software, although rich with unique potential of its own, shares elements in common with other arts. In particular, we might look at two terms used in cinema: *mise en scène*, which is the organization of images in space, and *montage*, which is their respective organization in time. Case Study 8.4 provides an example of *mise en scène*.

To illustrate the difference, consider a very famous movie moment: the shower scene in *Psycho*. The murder scene takes approximately a minute, and there are at least 60 different shots used. This is *montage*. By organizing the shots, Hitchcock creates dislocation and panic. Now suppose he had simply used a single shot for the entire scene. Instead of identifying with the victim, we would become onlookers at the scene of a crime. Instead of empathic terror, we would feel objective horror.

Theater uses *mise en scène* but not montage. Novels use montage as a matter of course, simply because they are forced to describe a scene one point at a time. However, they cannot employ *mise en scène* without evoking montage, because the order of describing objects and people in a room inevitably becomes significant. You would need a picture to detach that from montage. As an extreme simplification of a complex subject, we can say that montage creates narrative and intense emotion, and *mise en scène* creates setting, ambience, and reflexive emotion.

What about computer games? Except in FMVs (which are little movies anyway), the computer game evidently uses only *mise en scène*. This is because montage requires the viewer to be a spectator and not the controller of the action. Montage would work only in a weakly interactive game, say a murder mystery game in which your only choice was which character to follow. In a strongly interactive game, moments of montage can be used very briefly for effect, such as in the marvelous sequence early in *Alone in the Dark* when a monster runs across the lawn and crashes through a window. However, these must be used sparingly, or they cease to work and become only an annoyance. You do not want to keep destroying the feeling of immersion and forcing your player to sit back and wait while he watches snippets of FMV.

The huge advantage that entertainment software has over other visual arts lies in what is going on outside the frame. In an adventure game, I could give a message to a character and he walks off. He later returns with a reply from his lord. This can happen in a film, of course, but there it is a scripted event. In the case of the adventure game, it's possible for me to construct a complete environment so that the character might be waylaid, or lose the mes-

sage, or give it to the wrong person. (Okay, this wouldn't have been so easy in the past as it would have eaten up a lot of processing power, but we're talking about the next 10 years now.) You get the same effect in a simple way when you see a priest in *Age of Empires* walk out of sight behind a wall and you race units to either end, knowing you have him trapped. It works because, even though the priest has moved "off-screen," you know he's still there.

One of the unique strengths of entertainment software is evoking a world that persists even outside the player's immediate vicinity. This is a cut above the fictional worlds of cinema, which are merely credible. These worlds will be more than credible; they will, in a sense, be real.

### ***Games As Sports***

The ability to accommodate multiple participants makes entertainment software perfectly suited for sports. And these don't necessarily have to be real-world sports. Sports that would be impractical because of lethal danger levels (combat golf, lion wrestling) or size and cost (hundred-a-side football) are easy to stage by the medium of software.

Software sports stars may become as well known as real-life baseball and football players, and not everyone need be an active participant. The old game model would insist on a virtual sport in which the user took part, but why shouldn't I be a spectator instead? Perhaps in 10 years' time we might come home from work, get out the beer and pretzels, and connect to view the final of the World Quakeball Tournament.

### ***Games As Toys***

A toy is something that we play with in order to have fun, which is why *entertainment software* and *playware* are better terms for what we do than *computer games*. All games are

#### ***Case Study 8.4 An Example Of Mise En Scène***

About halfway through Appeal's *Outcast*, there is an FMV where the hero, Cutter Slade, responds to his lady friend's parting words, "We'll have words about this later," by saying to himself, "I'm all tingly with anticipation."

Suddenly, he seems to feel he's being watched. He turns to look towards the impregnable fortress in the heart of the city. In one continuous shot, the camera pans up over the wall of the fortress and continues up and up until it finds the villain of the story standing on his balcony, gazing at the city where he suspects Cutter Slade is hiding.

Although FMVs are not really what entertainment software is about, it's impossible to deny that this is a remarkable case of *mise en scène*. To bring us close to both hero and villain in one uncut sequence is something that even cinema would rarely attempt. And, if it happened in a movie, it still would not impress us as being as "real" as in the context of an adventure game where we have at least an illusory sense that these characters are bustling about their lives at all times.

toys, but not all toys are games, and our aim should be to create something that is interactive and fun whether or not it is a game.

I'm not predicting the end of games, as they are a perfectly valid way of having fun. Gameplay will continue to be an absolute priority in the strategy genre, for example. Players of strategy games demand a challenging game experience, and the evidence indicates that they are interested only in the best. This is why the superior strategy games (such as *Starcraft* and *Age of Empires*) are million-sellers while the rest are nowhere. Having found a satisfying game, it seems that strategy gamers will stick with it until something better comes along (*better* in this case being judged purely on the basis of interesting gameplay). Consequently, strategy gamers are loyal consumers, because the better product they are waiting for is usually the sequel to the game they've already enjoyed. To lure them away, you have to pull out all the stops and deliver the very best gameplay experience that the technology allows. (This was my thinking with the design for *Warrior Kings*, incidentally. Time will show if I was correct.)

But is gameplay needed in other genres? At the moment, gameplay is present in a peripheral sense in products like *Ecstatica* when it comes to choosing your weapons and tactics, but gameplay here is only part of a much greater whole: It helps to enhance the product, but it is not central to it. We might suspect that overt gameplay elements are emphasized only in such products by historical accident: Early text adventures like *The Hobbit* found it easier to emulate the form of a game than a story.

The real innovation came with *SimCity*. That was the point in the evolution of entertainment software when people could say, "This is very definitely a toy, it appeals to adults, and it's very popular." Whether it was a game or not didn't enter the equation. (Simulations tend to contain gameplay simply because real life does, and those "interesting choices" are one of the things that make a simulation rewarding. But simulations are not defined by their gameplay content.)

Everybody accepts that software toys (like *Pokemon*, for instance) are fine for children:

*"Encourage exploration. Make your environment free and comfortable enough that children will be willing to try things out until hitting on what you want them to do. While they're figuring it out, they'll still be having fun."*

—Tzvi Freeman, "Power to the Kids!" in *Game Developer*, September 1997

If you've ever played a game like *Dungeon Keeper* or *Age of Empires* with a small child watching, you'll know that children really appreciate the fun aspects of the game. "Make the lion chase that man," they'll say, or "Pick up the goblin and make him squirm!"

But it isn't just children who respond that way. Adults can enjoy entertainment software in the same way. A friend of mine who doesn't play games enjoys *Age of Empires* because he likes ordering his little men around to build a city. What annoys him is when the enemy players storm in and destroy it. For him, that spoils the fun.

I think we adults like to have gameplay just to make what we're doing seem respectable. We're not quite sure whether it's okay just to play. But play isn't the same thing as tomfoolery. Play can educate the mind, stimulate the imagination, sharpen the wits, gladden the heart, and enrich the soul. In the future, we'll see much more awareness of entertainment software as a new kind of toy, and the products will be all the more diverse and better for it.

## The Way Forward

A novelist said to me, "I hate computer games, they're not art."

Well, art isn't a rarefied concept for the elite few. All forms of entertainment give rise to arts. They may not be formal arts, but they are arts nonetheless. Computer games may not be there yet, but it's an inevitable trend.

So far, the technology of entertainment software has overshadowed the art. Now, armed with ever-improving graphics, physics, and artificial intelligence, we can look forward to a decade of exploring the medium's artistic potential. It will move away from the tropes of cinema and literature to define itself in ways that we are only starting to anticipate.

The way forward will come from design because designers are the artists of the new medium, just as medieval architects were the artists of stoneworking. As designers, we must insist that what has been achieved in computer games is not good enough. We must devise new concepts to make us worthy of our role at this, the most exciting moment in the history of human entertainment.

I'd like to end with two quotations. The first is from George Bernard Shaw: "The reasonable man adapts himself to the world; the unreasonable one persists in trying to adapt the world to himself. Therefore all progress depends on the unreasonable man."

The other quotation is from Sir Christopher Cockerell, inventor of the hovercraft: "If it wasn't for the silly chaps, we'd still be in the Stone Age."

Among the readers of this book are, perhaps, those "unreasonable" men and women who will pioneer the transition of entertainment software from a niche hobby to a whole series of brave new media.

