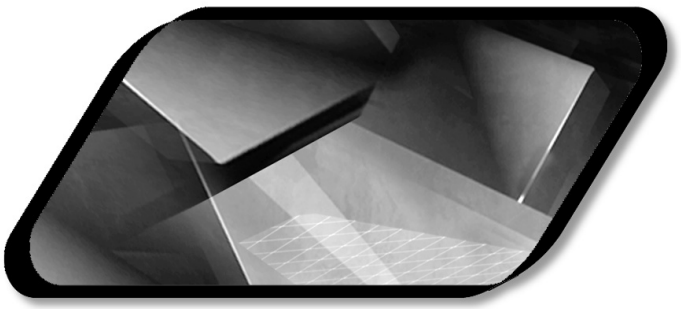


IN the previous chapter, I traced games back to their origins: a team of game developers, working either for a development company or a publisher. In this chapter, we'll go inside that team and see how games are actually built, from initial concept to shipping product.

INTERNAL OR EXTERNAL DEVELOPMENT?

As I mentioned in Chapter 3, publishers either hire their own developers to work in-house (internal development) or they sign a contract with a development company (external development) to build a game. This has certain implications for the development process, depending on the approach used. The financial and administrative details tend to be rather different, so as I describe each stage of development, I'll break out the differences into separate subsections.

STAGE 1: THE BRILLIANT IDEA

A game begins as an idea. You've probably got a great idea yourself, or several of them. Every game developer in the whole industry has ideas for games, and they think and talk and argue about them with their colleagues whenever they have free time. There's very little point in trying to keep a game idea secret; the chances that you have a completely unique idea, never before thought of by anyone, are incredibly small.

INSIDE INFO

"Ideas are easy, production is hard." What matters is the ability to build and complete a high quality, compelling, marketable game.

—Ellen Guon Beeman, Producer, Monolith Productions

Evolutionary or Revolutionary?

The vast majority of games in the industry are "evolved" from earlier games. Their creative, and often their technical, content—what you might call their genetic material—is a mixture of ideas that have gone before. Certain groups of characteristics are especially popular with the players, so games have evolved into different "breeds"—the genres I talked about in Chapter 3.

If a game idea has evolved from earlier games or other entertainment media, it is easily understandable to anyone who hears about it. For example, a friend and I once pitched an idea to a producer as "*Diablo* meets *The X-Files* in 3-D." If you were to cross *Diablo* (fairly fast combat action from an aerial perspective above a small party of people) with *The X-Files* (mysterious supernatural conspiracies set in the present day) you would get our game, which we had decided to call *Psychic Warriors*. We hoped that, by summarizing the game idea in a single, crisp sentence (known as a "high concept"), it would appeal to the producer and the marketing people whose job it would be to sell the game to the public. The longer it takes to explain an idea, the more likely you are to lose someone's attention and interest.

As you can imagine, this evolutionary approach has given rise to a world of games that look somewhat alike. Some are even derisively called "clones"—games with identical play mechanics and nothing but some new graphics slapped on. During the heyday of the Super Nintendo console, a few publishers became notorious for producing clones: side-scrolling games that were all alike except for their appearance.

The alternative, a revolutionary game, is much harder to persuade a publisher to build, and even if they do build it, it's harder to sell to the public. Players like their cozy, familiar genres. When a player puts down his money for a role-playing game or flight simulator, he knows what to expect: what sorts of challenges he will face, decisions he will make, and actions he will take. Asking a player to buy a revolutionary game—a type of game the world has never seen before—is asking him to gamble $50 or so on a game that he may end up hating.

But when the revolution succeeds, the rewards are enormous. Power, glory, riches! A new kind of game is born and you were part of it. To a creative person there is no finer feeling. *The Sims* is such a game—a runaway bestseller whose publisher never really believed in it until its colossal popularity showed them how wrong they were. Electronic Arts only grudgingly allowed Will Wright, the designer, to make *The Sims* because he already had a string of hit games. Interestingly, *The Sims* was *not* the first game of its kind. It borrows a number of ideas from a much earlier game called *Little Computer People*, but *Little Computer People* was ahead of its time; the public wasn't ready for it. Which brings up a point: Sometimes you can only have a revolution when the conditions are right.

How Publishers Hear about Game Ideas

To begin with, whoever has the idea (I'll assume it's you for the moment) has to persuade a publisher to think about it. Not to develop it—we're still a long way from that stage—but just to think about it. Getting a publisher's attention is the first hurdle, and you can imagine, there are an awful lot of people clamoring for it. Publishers get unsolicited submissions—"bluebirds"—sent to them in the mail all the time, but they seldom take them seriously. (In fact, for legal reasons, most publishers won't accept

unsolicited submissions. If you do want to send your game idea to a publisher, you should contact them first to find out whether they will accept your game proposals.)

More frequently, an external development company whom a publisher has already worked with, or has heard of—someone like Monolith, for example, with a track record of creating successful games—phones up a producer and says, "Hey, we've got a great idea for a game that we want to talk to you about." If the development company has a good reputation and the producer isn't too busy, he'll set up a meeting for the developers to come and deliver their pitch.

But publishers don't just wait for ideas to come to them, either. They have a product plan that specifies what kinds of games they want to release in the next year or two, and how much money they want to spend on developing and marketing them. If a publisher is large enough, it usually has several product lines or brands that it's planning to produce games for. In this case, they already know what they want to build, and they go looking for developers to do the work. Sometimes they'll use their own in-house teams; other times they'll contact development companies and offer them the job.

Internal Development

If you work for a publisher, it's not that hard to get someone to at least listen to your idea, because they work in the same office with you. If it meets their product needs they may ask you to do some further work on it; if they really like it, they may even assign a few developers to work with you.

External Development

If you work for an external developer, you can of course talk to your bosses about your idea, but your company is unlikely to have the money to develop it themselves. If they like it enough, though, they may decide to pitch it to a publisher.

Pitching the Game

Since this is a book about how to get a job, not how to get a publishing deal, I'll just give brief highlights of pitching a game. First you have to find someone who's interested and willing to talk to you in person—just mailing them some documents isn't going to cut it. Second, that person has to either have some decision-making authority herself, or have the ear of someone who does. There's no point in pitching your game to a testing intern. Ideally, the pitch should be made to a producer or executive producer.

Pitching a game is more than just talking about it. You should have a prepared presentation, with PowerPoint slides and handouts. You should bring along some early, "high-concept" design documents and some concept drawings that show key visual elements in the game, especially anything radically new. You might have some

animations already, a short video to show, some working code, or ideally, a playable prototype. Obviously, the farther along a game is in development, the more interested the publisher is going to be, because it means they have to invest that much less money to finish it. In rare cases, a developer has actually got a complete game that's ready to ship. Financially, that's the best scenario of all, although few publishers will ship a game without insisting on a few changes.

A pitch is not just a lot of blue-sky enthusiasm about new technology and innovative gameplay, however. You have to convince the publisher that there is a market for the game and it will make them a lot of money. You also have to show them that you can build the game on time and under budget. A well-prepared pitch includes cost estimates and a proposed schedule. You can't do a pitch without being able to demonstrate these things; no matter how brilliant the idea may be, what the publisher wants to know is how feasible and profitable it is.

STAGE 2: PRE-PRODUCTION

If the pitch has gone well, the publisher likes the idea, and they are interested in working with whoever is proposing to do it, then pre-production begins. At this point, the publisher still isn't fully committed to producing and marketing the game. (In fact, a publisher is never entirely committed to publishing a game until it has been built and tested and they've started the marketing for it, for reasons I'll explain later in this chapter.) For the time being they only want to explore the idea, but they're prepared to spend a little money to do so.

There was a time when a publisher said, "Yes, go!" and a developer dived into coding the game the very next day. That time ended about 1985, when a game still cost between fifty and a hundred thousand dollars to develop. It was never good practice even then, and nowadays, with development costs in the millions of dollars, it would be a disaster. Any large project, whether it's building a skyscraper, filming a movie, or developing a computer game requires pre-production: an exploratory and planning stage. It's absolutely essential for building a game on time and within budget.

Once a publisher has decided to go into pre-production, they will assign a producer to it. This person is an employee of the publisher whose job it is to make sure the idea turns into the game the publisher wants. The producer is responsible for making sure it's a fun and, above all, marketable product. I'll discuss his vital role in the development process later, in Chapter 5.

Design Work

The first thing that's needed is a design document, although one may already exist. The developer may have written it on spec, and shown it to the publisher to get their interest, or someone at the publisher may have written one. In most cases, the design

will be incomplete, because there's no point in fleshing out every detail until you know the publisher is interested. Now is the time to finish that work. I'm not going to describe it in detail here, but in addition to thinking about how the game should look, sound, and, most importantly, play, the designer does a certain amount of competitive analysis, checking to see how similar games work, and doing background research—for instance, familiarizing herself with the subject matter of the game. If it's a military flight simulator, for example, she'll go to the public library and check out *Jane's All the World's Aircraft* for inspiration.

Internal Development

Within a publishing house, the producer might write the design document himself, but more likely he will assign an experienced game designer to do it. A producer has other tasks, and modern games are too large and complex to be designed on a part-time basis. Even though the work is taking place in-house, the publisher is already spending money on the project: the designer and producers' salaries.

External Development

Whether the developer has pitched an idea to the publisher, or the publisher has sought out a developer for a project they already have in mind, the publisher will then give the developer a small amount of money—between five and twenty thousand dollars—to write a design document and produce some concept sketches. In the meantime, the producer will be visiting the developer, looking over their facilities and their previous work, talking to their artists and programmers, and trying to get an idea of whether they'll be able to do the job. At this point, a lot hinges on the personal relationships between the people involved. If either side feels they can't trust the other, or just doesn't like the other's way of working, the process can break down right here.

Technical Research and Prototyping

The pre-production phase includes more than just design, however. It's also a time to identify the risks involved in the project, and to do research to minimize those risks when possible. To this end, the publisher will usually ask the developers to assemble a small team, seldom more than ten people, to identify the key challenges in developing the game and start work on them.

Risks fall into three categories: *technical risks*, *production risks*, and *creative risks*.

Technical Risks

Any video game that is not a direct clone of another requires new programming. All new programming represents some technical risk, but certain areas, such as new graphics technology or artificial intelligence, are particularly tricky. Two or three

programmers will build a small demonstration program (sometimes called a *proof of concept*) to test out the new ideas. A proof of concept is not a game at all. It's just a demo, often using "programmer graphics" (old graphic scrap or even just colored dots), that is written to illustrate the correct behavior. The proof of concept is used to show the publisher that the technical issues facing the team are surmountable.

Production Risks

Can the developer's team actually complete the project? This will also need to be demonstrated. Usually, the developer's internal producer/project manager will assemble a document describing the experience of the team, their production methodology, their ability to meet milestones and handle change requests, and other elements that will prove there is a minimal completion risk.

Creative Risks

No one can really be sure what the public is going to like, but if a game design calls for a new kind of user interface, or a type of gameplay never before seen, the publisher is likely to want to try it out before giving the go-ahead for the project. Likewise, they will want to see what the game is going to look like, and maybe even get an idea of how it will feel to play. This is where prototyping comes in. A prototype is a partially working model of the game, normally constructed by a handful of people: a few programmers, maybe a couple of artists, a designer, and a team leader. It doesn't even have to be a stand-alone application; a prototype can be built in Shockwave or Flash as long as it conveys the general idea; though some publishers will insist on more.

In addition to devising a prototype, the artists will be creating concept drawings, a few 3-D models, animations, and backgrounds, and looking in art books and on the Web for background material. It's always valuable to have a lot of pictures around during the prototype phase, to help inspire the team and give everyone a shared sense of what you're aiming to achieve.

WAR STORIES

At one point in my career, I was a lead game designer at Bullfrog Productions, which I had joined because they were famous for some of the most innovative PC games ever made. I was given a prototype team to research a new god game (a subgenre of real-time strategy games) called Genesis *(nothing to do with the Sega Genesis, by the way). It was to be set in a world of pre-industrial tribal peoples, somewhat like the earlier* Populous *games. I had three programmers (two working on technical proofs-of-concept and one on the prototype), an artist, and a level designer to help me with design work. I wanted our game to have spectacular and realistic-looking landscapes, so at our first group meeting I brought big color photographs of actual jungles, deserts, mountain ranges, seacoasts, prairies, and other dramatic geographical features. I told them, "One of our goals is to make our game look like this," which pleased the artist*

WAR STORIES CONTINUED

and made the graphics programmers look rather thoughtful. We hung them all around the walls of our bullpen (Bullfrog had no cubicles) so everyone who worked there, and everyone who walked by, could see them. My artist, Alex Godsill, got into the spirit of the thing and began posting the results of his research as well: pictures of the clothing and weaponry of native peoples all over the world. The place looked amazing, a visual riot of spears and shields, tunics and turbans.

Genesis *never went beyond the prototype stage, unfortunately—the company secured the* Harry Potter *license and realized, quite correctly, that that would be a much more lucrative game to publish. But everybody in the building could tell what the* Genesis *project was about, and we all had a collective idea of how it would look to be in our imaginary world.*

Project Planning

The third aspect of pre-production is project planning. This is an unexciting but absolutely vital part of the process. It's also a black art that can only be learned with experience. Project planning is normally done by the producer working together with a project manager. They'll define the scope of the project: how big and complicated the program will be, and how much artwork, animation, and audio the game will require. From those assumptions, they'll then estimate the size of the staff required to develop the game, and how long it will take.

Internal Development

Obviously, a key consideration at this point is whether the publisher is going to have the internal development resources to do the job. Planning the project will include looking to see what other projects are winding up, so you know which people will soon be available. If they don't have the necessary people on board already, and the project looks like a sure thing, that's good news for you: they'll start hiring!

External Development

Project planning is the responsibility of the development company, but it's also very much a process of negotiation. This is because the plan they come up with will be the basis for a very important business agreement: the development contract between the publisher and the developer. The developer will consider everything carefully, then add a certain percentage for slippage, profits, and bargaining room, and tell the publisher that the game will take two years, require 40 people, and cost three million dollars to develop. The publisher will reply that this is totally unacceptable and that any competent development house could do it with 20 people in 12 months for 1.5 million. The developer will remonstrate and point to all the things required by the design

document. The publisher will say that the designer was letting her imagination run away with her, cut a few things out, and raise their offer a little. The developers will reluctantly admit that they don't *really* have to hire the London Symphony Orchestra to record the music, and lower their demands a little. The process will go back and forth until both sides have a budget and schedule they can live with, which will form part of their development contract.

Whether they actually succeed in keeping to either one depends greatly on the quality of their project planning and management.

Going to Full Production

The length of time a project spends in pre-production can vary from about one to six months, depending on just how much design work, technical research, and prototyping needs to be done. If the game is a sequel whose codebase is largely complete, pre-production might take no longer than is required to do the project planning.

Whether internal or external, pre-production ends with the development team giving a presentation to the publisher to show what they've done, and trying to persuade them to "green-light" the product—take it into full production. This presentation is similar to the original pitch, except that now there should be a lot more to show. It's a key moment for the developers, because if the publisher goes for it, they will eat, sleep, and breathe the game for the next year or more of their lives.

Internal Development

Apart from considering the merits of the game as a product, a publisher has to decide if developing it in-house is the best use of its available people. This usually isn't a difficult decision if the prospective development team has been on-staff for a while, because the company knows them and their strengths and weaknesses. If the publisher really likes the product but not the available team, they can always look for an external developer instead.

External Development

Going to full production with external developers is a much more tricky decision for the publisher, because at this point they're preparing to commit hundreds of thousands or even millions of dollars to an outsider whom they have little control over. But before they do that, the two parties must reach agreement on a development contract. The development contract contains far more than just the development budget and the schedule hammered out in the project-planning phase. You can't understand the game business without at least a passing acquaintance with development contracts, so I've written a special section just about them.

The Development Contract

There are two kinds of development contracts; each uses different terms and each is intended to serve different purposes. The first, and by far the most common in developing new games, is called a *publishing contract*. The other is called a *work-for-hire contract*. I'll describe each in turn.

The Publishing Contract

A game publishing contract is, in essence, very much like a book-publishing contract, although there are many differences in the details.

Advances In a publishing contract, the publisher agrees to *advance* money to the developer throughout the development period to cover the cost of building the game. Book publishers do the same with authors: they advance them money to cover the cost of writing the book. But an advance is neither an outright payment for work accomplished (which is what happens in a work-for-hire contract), nor is it a loan that creates a debt from the developer to the publisher. It is, in fact, an advance payment as part of the total deal. Just as you have to pay a building contractor part of his fee in advance so he can buy construction materials, so a publisher pays a developer in advance so he can work on the game.

Royalties The other key part of the contract states that the developer will get *royalties* from wholesale (not retail) sales of the game; that is, a percentage of the money the publisher makes from selling the game to retailers after it's finished. This amount varies enormously, from as little as 7 or 8 percent of the wholesale price up to 40 percent or more in very special cases. 10 to 20 percent is a fairly common range.

However, the publisher doesn't pay royalties to the developer right from the first unit shipped. Since they've already given the developer advance payments, the publisher keeps the money until the amount of royalties earned matches the amount already advanced. Only when that happens does the developer start receiving royalty payments. The exact same thing occurs in book publishing. Once the advances have been repaid out of the royalties, the game, or book, is said to have *earned out*. The whole scheme is called *advances against royalties*.

Here's a simple example. Suppose Publisher P advances Developer D $2 million to make a game. Publisher P has agreed to pay Developer D a 20 percent royalty on wholesale sales of the game, which they'll be selling at $25 apiece. Twenty percent of $25 is $5, so the developer will receive $5 for each copy of the game shipped. However, the publisher needs to recoup their advances first. So they'll withhold the royalties until that $2 million has been paid off. That means the publisher has to sell 400,000 copies of the game before the developer starts receiving royalty money. (This is all in theory. In practice, it's somewhat different, for reasons I'll describe later.)

Milestones The advance money is not all paid in one lump sum at the beginning of the development; rather, it's doled out a little at a time as the project progresses. The development contract will include a schedule with a series of *milestones*—dates by which certain features must be in the game. On the milestone date, the development company will send a copy of the work in progress to the producer at the publisher. If the producer agrees that the required features are in fact present in the game and working properly, then he'll authorize the next milestone payment to the developer. If he doesn't agree, he'll point out what's wrong. This is one of the touchiest areas of publisher-developer relations. Publishers and developers often disagree on whether a feature has been implemented properly. The publisher withholds the milestone payment, the developer thinks they're being unfair, and a squabble ensues. If the developer is absolutely depending on the money coming in, the publisher can drive them out of business (and kill the project) by not making the payment. On the other hand, sometimes publishers make milestone payments even when they're not really satisfied with the quality of the work, because they can't afford to let the developer go out of business. Even if they could find another developer to take over the project in the middle, it will almost certainly be very late and screw up the publisher's product planning.

Settling on the Numbers The amount of the advance and the size of the royalty percentage depends on a great many factors, not the least of which is the negotiating skill of the developer. However, there are a couple of rules of thumb that help to determine their size. If some of the development work is already done when the publisher agrees to publish the game, then the amount of advance money needed to complete it is smaller, and the royalty percentage is correspondingly larger. For example, the developer may already have a software engine that allows them to build the game very quickly. If the developer has done all the work on their own nickel and the publisher doesn't need to pay *any* advances, then the royalty rate is the highest of all because the publisher's financial risk is correspondingly lower. However, this situation is very rare. Most often, little or none of the work has been done, and the publisher has to pay large advances to complete it.

Another thing that affects the royalty percentage is the developer's track record and technological skill. If the publisher has worked with the developer before and has made several hit games with them, the royalty rate is bound to be much higher than if the developer is new and the publisher doesn't know them well. First-class developers are rare and a valuable asset to any publishing company. A publisher who has a relationship with a development company that they like will do a lot to keep it happy and working with them. If the developer becomes *so* vital to the publisher that the publisher's business success would be threatened if the developer started to work with someone else, the publisher might actually buy the developer's company and make it a division of their own.

Finally, royalty percentages aren't always fixed for the life of the product. They often contain escalator clauses that mean the developer gets a higher royalty once the game has sold a certain number of copies. Publishers may also offer bonus percentage points for getting the game done ahead of schedule.

The advances against royalties aren't usually much more than what it will cost to build the game in the first place—there isn't a fat profit built into them for the developer. That's because the developer is hoping to make their real money on the royalties afterward. If the game is a hit, the developer can make millions. On the other hand, if the game is a flop, at least they got paid while they were making it. Unfortunately, it's a sad reality that most games today never earn out (recoup their advances), so the developer never sees any royalties at all. They cost a great deal to develop, and they don't sell well enough to repay those costs. On the other hand, in a few cases, they do fabulously well and the developers make millions. As in Hollywood, the hits subsidize the misses.

What if the Game Gets Cancelled or Doesn't Sell? It's important to realize that the advances are not actually a debt owed by the developer back to the publisher. They're simply an advance payment against anticipated future earnings. If the publisher decides not to publish the product after all, the developer doesn't owe the advances back. Or if the publisher fails to sell enough units to recoup all the advances, the developer doesn't have to pay the difference. Creating the game is the developer's job; selling it is the publisher's. If the developer has done their part, they are entitled to keep the advance money. If the publisher fails to sell the game, that's their fault, so they assume the risk if losing their advance money.

What if the Development Company Wants Out? If the development company doesn't want to finish the project for some reason, they can usually back out by repaying all the advances they have received so far. However, in practical terms this is seldom possible. The developer has been spending that money to pay its employees and subcontractors to build the game; they don't have it to give back. The publisher is counting on them to deliver, so the contractual terms are designed to make sure they do. Of course, terribly managed developers do occasionally go bankrupt, leaving the publisher in the lurch. Back when development companies consisted of only one or two people, they were also known to simply disappear—vanish into thin air, taking the source code for the game with them!

The Fine Print The preceding example gives the bare bones of the advances-against-royalties system. Most arrangements are not that simple, and there are a lot of other things that go into a game development contract as well. Games are big business, and where a lot of money is involved you will always find smart lawyers and fine print. For example, a publisher could try to sell more games by offering a deal to the retailers: buy one game for $30 and get a second game for a penny. If the second game happens

to be your game, the publisher could claim that the wholesale price was one cent, and you would get royalties of 20 percent of 1 cent! Wise developers make sure there are clauses in the development contract that prevent this kind of thing.

There are other things to be aware of as well. As I mentioned in Chapter 3, publishers never actually pay a developer the full amount of the royalties specified by the percentage, because they have to keep a reserve fund of money around in case the retailers want to return the games and get their money back. So although the contract states that the development company will receive 20 percent of the wholesale price of every game sold, after the reserves are deducted this could be 15 percent or even less. This means that it takes even longer for a product to earn out.

Occasionally developers need help from the publisher: assistance with knotty programming problems, or specialized services like motion capture that the developer can't do for themselves. All this costs money, and depending on the terms of the contract, the publisher may consider any money they spend on such things to be part of the developer's advance, having to be recouped later. After all, creating the game is the development company's job; if they need help with it, then should be the ones to pay for that help (goes the theory). But occasionally publishers can be pretty insistent about offering their "help," and an unscrupulous publisher might even overcharge a developer for the value of this "help"—whether they need it or not.

But Don't Worry about It! You're probably wondering why I'm dumping all this rather grim business stuff on you when all you want to do is get a cool job in the game industry and realize your creative potential making great games. The reason is that, although you don't have to negotiate these deals in person, they still affect your day-to-day life as a game developer working for an external development company. When the boss comes in and says, "We hit our milestone two days early! Free beer for everybody!" you'll know what she's talking about. At this point in your career you don't have to *worry* about these things—but what you see go on around you at a game company will make a lot more sense if you *understand* them.

The Work-for-Hire Contract

A work-for-hire contract is a far simpler deal. In this case, there are no advances and no royalty payments. Instead, the publisher simply pays the developer a negotiated fee—again, spread out over a series of milestones—to do the work. This fee is usually higher than the amount a developer receives in advances in a publishing contract, because the developer is giving up the chance to earn royalties. If the game turns out to be a colossal hit, the developer doesn't share in its success.

Publishers seldom sign work-for-hire contracts for original game development. Work-for-hire contracts tend to be used for particular projects like converting an existing game to work on a new platform, or localizing a game into a new language.

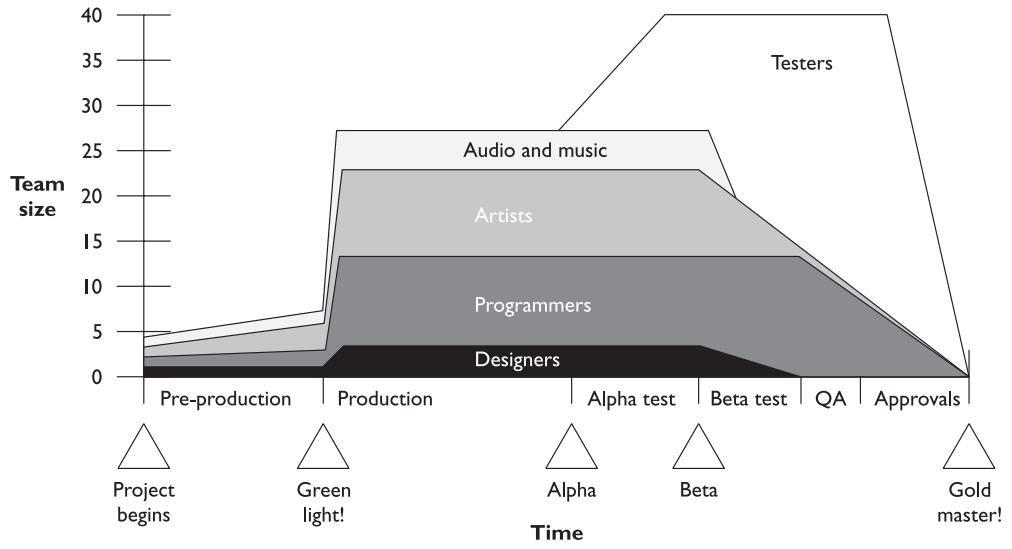
These sorts of tasks are often undertaken by development companies that specialize in them. Since they're not contributing to the game's creative content, they aren't really entitled to share in the rewards of having a hit.

STAGE 3: PRODUCTION

The project is off and running! The development team, whether internal or external, has gotten the green light—that is, received approval to execute the development plan. At this point, the company staffs the project, deciding on all the programmers, writers, artists, animators, musicians, sound engineers, and other creative and technical people that will need to be on board. They may be coming from other projects that have just been completed, or (and this is where you come in!) they may be hired to work on the game. Take a look at Figure 4-1 to see how development teams grow and shrink during the production process. This is only an example; different companies use different approaches.

At this point everyone is working hard. Typically, it'll continue that way for the next 12 to 18 months. Equipped with a task sheet, the project manager will go around making sure people have done the work they're supposed to, and know what they're supposed to do next.

FIGURE 4-1



A plot of team size versus time for a hypothetical product

The Production Process (and Why It's Not Your Problem Yet)

There's an old adage than managing programmers is like herding cats: they're wayward, individualistic, and unpredictable. The field of software engineering is only about 50 years old, so we aren't yet sure how it's really supposed to be done. (Civil engineering, by comparison, is almost 5000 years old, which dates back to the Pyramids.) Game development is even worse, because in the game industry we have to build lots of stuff besides program code: still images, 3-D models and animations, music, sound effects, user interface elements, and so on. Each of these things requires a different procedure to make, so they all take different amounts of time to complete.

It used to be that project managers flew by the seat of their pants, making up schedules and deadlines out of thin air and based on hunches. That kind of approach doesn't work any more. The projects are too big and too complicated to manage by instinct alone, and now we need formal methods. Many smart people have written many large books on the subject. If you're particularly interested, you might read *Game Architecture and Design*, by Andrew Rollings and Dave Morris (now out of print, but there should be plenty of used copies available; a new edition is in the works), and *Rapid Development*, by Steve McConnell.

However, as a newcomer, you really don't have to worry about it now. You're not going to be hired in as a full producer or project manager. If your employers know their business, they will have thought carefully about how they want to run the project, and as it enters full production, they'll be putting their plan into practice. *Your* job will simply be to do your own work.

Jobhunting Tip: Avoiding Incompetent Employers

If all you want is a job, any job, and you don't really care whether the people you work for are competent or not, then you can ignore this (but be warned: companies with incompetent management tend to be short-lived!). On the other hand, if you want to find out if you'll be working for professionals, ask your prospective boss to describe the development plan for the product you're being interviewed for. Don't give him a grilling; just say that you're learning about the business and you'd like to understand his approach. If he seems surprised by the question, or gives you a lot of hand-waving, you'll know he hasn't really thought about it. That's a warning sign that there may be trouble ahead. Conversely, if he draws a neat timeline on the board and explains where the milestones are and who else will be on the team, then you can feel a little more confident. Of course, you don't have any way of knowing if the plan is realistic or not, but at least they *have* a plan. It's a sure way to separate the amateurs from the pros: pros make a plan.

WAR STORIES

At some point in every day there is a moment when we all try to let loose and do something out of the ordinary. When I was at EA, in the early days of my career, we would have shooting contests for money in the lobby with a Fisher-Price basketball hoop that we bought. This would get everyone together, and while most of the people had no ability to shoot, it didn't really matter. A dollar on the line is worth the fun of breaking away and laughing as a team. At Konami, our guys quickly switched this to producer baseball. I, being inept at baseball, didn't have as much success as the others, but it was fun nonetheless, and an important part of our day. Never underestimate the importance of team bonding. A key part of our morale can be directly traced to the bonding that went on as a group.

— Jake Neri, Founder and Partner, Blaze Games

Meetings, Meetings, Meetings!

Every week throughout the production period there will be regular meetings. Subsets of the team, like all the animators, will get together with their lead to discuss issues specifically related to their role. All the leads will get together with the project manager to report on their respective departments. From time to time, the whole team will get together as a group, both for people to learn what the others are doing, and for management to give them information. And these are just the regular meetings. Often, a problem will come up during a larger meeting that requires only two or three people to solve. It's a waste of the others' time to try to solve it there, so the people involved will schedule another meeting to get together and deal with it. And then there are the ad-hoc meetings that arise spontaneously: "Hey, Pat! Let's grab Sandy and Chris and figure out how much disk space we can afford for voiceover audio." A surprising amount of a game developer's life—in extreme cases, 40 percent of a programmer or artist's time and 80 percent of a manager's—is actually spent sitting around a conference table rather than at her desk. Game development is an intensely collaborative activity.

Marketing Activities

Although this is a book for prospective game *developers*, it's useful for you to understand what's involved in *marketing* a game as well. Long ago, all a publisher had to do was buy ads in some gamer magazines, because those were the only media available. Nowadays, there are many more ways to reach gamers, and the marketing department has to cover them all. Here's a list of approaches modern game marketers use:

- **Print advertising** Still the mainstay of game marketing, the marketing department will devise an ad campaign and purchase space in magazines they think will best reach their market. In special cases, they may even buy space in non-gamer magazines and newspapers as well, if they feel a game has a broad enough appeal to justify it.
- **Web sites** Any forthcoming game has to have a web site full of screenshots, information about the game, interviews with the designers, and downloadable items: desktop themes, audio clips, movies, and, above all, a playable demo once the game is complete enough to play.
- **Co-marketing activities** Publishers will often work with retailers to help them sell the game, by providing displays and decorations for their stores and fliers and other material about the games. The publisher may also help to pay for the store's own advertising. Retailers now have enough clout to demand this; a publisher who doesn't provide it may find their games on the bottom shelf, back in the least-accessible corner of the store!
- **Trade shows** A trade show isn't really marketing aimed at the consumer; rather, it's an event intended for the retailers. The Electronic Entertainment Expo (E3) is the big game trade show in the United States. Publishers will spend hundreds of thousands of dollars building a fancy booth and flying their marketing staff and senior developers to a trade show to demonstrate their products to distributors, retailers, and the press. E3 allows consumers in on the last day, so those who want to can get an early look. E3 can be a great opportunity for someone seeking a game development job, but keep in mind that the publishers are spending big bucks to be there to cut deals with distributors and retailers, not necessarily to talk to job applicants.
- **Press events** In addition to buying print advertising, a publisher will also try to get press coverage of the game in as many magazines as possible. They'll invite reporters to visit their offices and send them *press kits*—fliers and CDs full of images and other information the magazine could use if it writes an article about the game. They'll send out early versions for previews to just about everybody they can think of, as well as distribute another version to magazines and other media sources near its release to facilitate reviews.
- **Television advertising** Only used for the biggest of the big hits, such as a surefire winner like *Madden NFL Football*. TV advertising is incredibly expensive just to produce, much less buy.
- **Public events** These are particularly common with sports games, because they can be held in conjunction with major sporting events like the NBA playoffs. The publisher holds a big party, hires celebrities to come and play their game, and, of course, invites the press to attend.

- **Box design** The design of a game's box is important to help sell the game, so this is traditionally a marketing activity. They'll take all the pictures and lay out the text. In some cases, the marketing department considers everything except the contents of the CD or DVD itself their responsibility, so they'll write the manual, too.

INSIDE INFO You know those "designer diaries" you see online? If one is being produced by a small development company or a one-person shop, it's probably more or less real and tells the unvarnished truth. But if it's on a big publisher's web site, it's essentially a marketing gimmick, letting the public think it's getting a peek into the internals of the design process. Trust me, if there was a colossal screw-up on the project and half the animations had to be reworked, it wouldn't appear in a big publisher's "designer diary." That kind of thing could hurt the stock price!

Marketing a big product is hugely expensive. The general rule of thumb for an ordinary, run-of-the-mill PC game is that the publisher should spend the same amount of money marketing it as they did developing it. But for their AAA products, the top-of-the-line blockbusters, they'll spend three or four times that much—many millions of dollars.

STAGE 4: TESTING

Once the game has gotten to the point where large parts of it are playable, testing begins. Testing is an absolutely essential, but rather unglamorous, part of developing any game. It involves no creativity, only hour upon hour of trying out different features in different combinations to make sure they all work.

Informal testing goes on throughout the development process, as programmers run their code and producers check it over at milestones. Formal testing is normally divided into phases called *alpha*, *beta*, and *quality assurance* (QA). I'll discuss each here in turn.

Alpha Testing

Testing may begin on a game long before the whole thing is assembled and playable; testers can test parts of it as development progresses. But when the game reaches the point that all its *features* are present—even if all its *content* is not—then the game is said to be "at alpha" and ready for alpha testing. Alpha occurs when all parts of the game are functional but not all the graphics or data are necessarily available. For example, in a flight simulator, the plane may be fully functional but not all the landscapes are ready yet. In a football game, the game may be playable, but not all the

stadiums or teams yet created. As a general rule, though, about 80 percent of the content ought to be done, because most of your team's effort after alpha is spent on tuning and bugfixing. There's not enough time left to be creating large amounts of new material.

Alpha is a make-or-break point for the game. For external developers, it's normally an important milestone and triggers a big payment. The publisher's next step will be to commit a lot of resources to testing the game, and to start ramping up the marketing. Before they take that step, they are likely to take a long, hard look and make sure that the game is really fun enough to succeed in the marketplace. A fair number of projects make it all the way to alpha and get killed because, even though they're competently built, the publisher doesn't believe they can compete. Remember, the amount the publisher spends on marketing may be several times what they've already spent on development. If they kill a mediocre product at this point, they save all that marketing money. If they let a mediocre product go forward and it tanks, they've lost much more than just the development costs.

Once a publisher has made the decision to go forward, the game is in alpha-test. This is a period of internal testing by the publisher, developer, or both together. A few weeks before alpha, a testing manager will create a test plan, a master document listing all the tests to be run to check each feature. Testers will be hired, or reassigned from other projects, to execute the test plan. The number varies with the size and importance of the project, but it's not unusual to have 25 or 30 testers working full-time on a single game. The testing manager will also set up a *bug database* to keep track of bugs that the testers have found. Every time a bug is logged, the programmers will have to deal with it, and when done, claim that it has been fixed. The testing manager doesn't take the programmer's word for it, however! The bug remains "open," that is, flagged as a problem, until a tester has re-tested for it and verified that the fix really worked.

Alpha testing can go on for weeks or even months. It's a hard, grueling time. The bugs seem to come in an endless stream, and the programmers develop a secret hatred of the testers and the bug database. In the meantime, the artists, audio people, and other content providers are hurrying to provide the remainder of the data needed to complete the game.

INSIDE INFO If you keep your nose too close to the grindstone, your work gets out of focus. When you are working hard toward a goal, your brain doesn't have time to do any lateral thinking, and you often don't have the ability to step back and clearly evaluate what you've done. Taking a break does more than give your body a rest; it also allows your mind to break out of the rut it gets into during crunch time, permitting you to see your work with a new perspective. Try not to work more than two weeks in a row without a day off.

Localization

If your product is going to be sold in another country, you have to plan for it in advance, as I mentioned in Chapter 2. Localization has an impact across the whole development team, even the programming:

- **Programmers** Programmers must write the software so that all text is read in from files and none is hardwired into the code. Far Eastern languages require two bytes, rather than one, to store each character of text, so if the game is to be localized for one of those languages, the programmers must allocate additional memory for the text. On console machines, the programmers must make sure the code works on both PAL and NTSC television systems.
- **Artists** This group must create multiple versions of any art that includes text, and multiple versions of any art that is culturally sensitive. Nazi symbols are forbidden in Germany, for example, so games about World War II require special artwork for the German market.
- **Audio engineers** Audio engineers have to record separate versions of any voiceover dialog in every language the game will support. The game may even need different music; Japanese and American tastes are somewhat different, for example.
- **Writers** Writers must get their material translated. In addition to the in-game text, the product will need a new box and manual for each country.

All this work should be completed before the end of alpha, and, of course, it all has to be tested.

Beta Testing

When all the content is ready, all the levels designed, and all the art and audio created—including foreign versions—the game is complete. This point is called *beta*, and for external developers, it's another important milestone. The game's still not ready to be shipped, however—not by a long shot. The internal testers are now working on the beta version and the programmers are still fixing the bugs they find. In fact, the internal testers now have a new set of things to test: they not only have to make sure that each feature works, but that it works on every level, with every team or weapon or airplane that the game contains. All the content must be checked to make sure it works with the game. This process is called *beta-testing*.

Once the game is in beta-test, the publisher can, if they want, do *open* beta testing. In open beta testing, the publisher allows members of the general public to test the game—in effect, a field test. It's only possible with PC games; console games cannot be tested this way because the public doesn't have the specialized development hard-

ware necessary to run the game. In order to prevent piracy, the external beta testers are normally given a copy of the game that only works for a limited time; they also sign an agreement not to make copies of the game. Open beta testing is normally restricted to a fixed period.

Open beta testing is of mixed value. On the one hand, ordinary gamers will think of all kinds of weird things to try on the game that the alpha test plan might not have included. The outside beta testers might have hardware configurations that are different from the ones the internal testers have. In those respects, open beta testing is tougher and potentially covers areas that alpha testing didn't. It also has some marketing value; non-employee beta-testers will talk and raise gamers' enthusiasm about the game (as long as it isn't too buggy when they get it).

On the other hand, however, open beta testing is haphazard. You can't hand external beta testers a checklist and make them complete it; they're not getting paid. You can never be quite sure what they'll cover and what they'll ignore. Also, because they're ordinary gamers and not professional testers, they might not be as observant, or be able to describe the bugs they find in a way that's helpful to the programmers. They might even report things as bugs that are just features they don't like. Open beta testing takes a lot of management, and not all publishers bother with it on all games.

Configuration Testing

Configuration testing applies only to PC games, not to console games. Toward the end of the testing period, when the program is looking pretty stable, the publisher still starts trying it on different combinations of hardware and operating system variants. They'll usually have a *config lab*—a room full of PCs with several combinations of video cards, audio cards, memory, processor speeds, and versions of the target operating system. This lets them discover if the game has problems with a particular manufacturer's hardware, and helps them determine what the minimum acceptable configuration for the machine is. Obviously, they can't test every possible combination of every graphics and audio card; there are just too many. But configuration testing is an essential step before any PC game can be released.

Disney's Christmas Configuration Calamity

In the summer of 1994, Walt Disney Corporation had a mega-hit movie on its hands with *The Lion King*. Seeking to capitalize on this, they brought out a *Lion King* video game for the PC, which they made available in time for Christmas. About that time, a computer industry trade group, hoping to boost sales of CD-ROM drives and sound cards, had the bright idea of defining a standard called the "Multimedia PC"—a PC machine with an 80486 processor, an 8-bit Sound Blaster card or equivalent, and a single-speed CD-ROM drive. Millions of people bought machines

marked "Multimedia PC" in the belief that they were the hottest thing going in audio and video for the personal computer.

Unfortunately, someone at Disney had decided *The Lion King* would sound better on a 16-bit sound card … and that was the machine they developed it for. It didn't work, as shipped, on the much-vaunted Multimedia PC.

Christmas morning 1994 was an unmitigated disaster for Disney. Thousands upon thousands of angry parents called Disney's help line to ask why their game wouldn't work on what they believed was the latest and greatest PC. In the end, over half the games were returned for a refund.

If only they had done a configuration test with an 8-bit sound card…

Content Ratings

In many countries in the world, a video game must be submitted to a ratings body to determine how violent, scary, or sexually explicit it is before it may be sold. This is done after the game reaches beta, when all the content is present. The rating institution takes a few days or weeks to examine the game and return a rating, which the publisher is required to print on the game's box and possibly in any future advertising as well.

In America, rating is done by the Entertainment Software Rating Board. This is not required by law (that would be a violation of the First Amendment), but some retailers refuse to carry games that have not been ESRB-rated. Some also refuse to carry games with an AO (Adults Only) or M (Mature) rating. Although many developers object to having their games labeled in this way, the system does help the consumers know what it is they're getting. It is actually superior to the American movie-rating system, because in addition to a letter grade it also supplies "content descriptors"—short phrases that indicate what sorts of things the player will see in the game.

Obtaining the ratings for every country in which the game will be sold is a major job, and most publishers have a special department set up just to handle the paperwork and keep track of the process for each game.

Quality Assurance

Quality assurance sounds like a fancy term for testing, but in fact it refers to a particular aspect of the process. Normally, a game goes to QA when all the bugs in the bug database are fixed and the producer is convinced the game is ready to ship. The QA department tests it for a number of hours, and gives it a simple pass/fail grade. QA doesn't try to determine whether a game is enjoyable or well-balanced; it is concerned only with whether it works as a piece of software. If a game ever crashes, responds inappropriately to a command, or displays something it isn't supposed to display, it fails QA and cannot be shipped. QA also checks to make sure that all the details in the manual are correct:

that the images on the screen match the pictures in the manual, and that the commands are documented correctly and work as described.

The QA department at a publisher is normally separate from, and independent of, the production department that is responsible for the game. That way they can't be pressured into passing the game even if it has problems.

A QA Failure

In January of 1999, Electronic Arts had to recall 100,000 copies of the PlayStation version of their *Tiger Woods 99* game. Someone had (possibly accidentally) included a data file on the disc that didn't belong there, an AVI movie of the highly blasphemous original *South Park* pilot called "The Spirit of Christmas." There was no way to see it accidentally by playing the game, but all the same it was extremely embarrassing—and costly—to EA. The error was made worse by the fact that EA didn't have the rights to *South Park* anyway; they were held by Acclaim. A simple QA check, verifying the identity and purpose of each file, would have saved the company a small fortune.

Licensor and Console Manufacturer Approvals

If a game is based on a licensed property of some kind—*Nancy Drew,* for example, or Major League Baseball—the terms of the license contract will require the publisher to submit a copy of the game to the licensor for their approval. They want to be sure that the publisher isn't misusing their characters, logos, or whatever it is that the license provides. They can be extremely strict about this, insisting that every color be exactly right and that the game include no inappropriate material for their license. Recently, the National Football League has begun cracking down on video games that represent football as more violent than it really is (which is extreme enough in any case!).

In addition to the property licensors, the console manufacturers also have an approvals process. They have both content and quality standards, because the game is going to go out with their logo on it. In addition to performing their own technical tests, the console manufacturer will make sure the subject matter meets their guidelines, and even the package design gets close scrutiny.

If a game is going to be published on more than one console, each version has to be approved by its own console manufacturer. For example, Activision's *Tony Hawk's Pro Skater 4* is available for the Xbox, PlayStation 2, and GameCube all at once, so Activision had to submit it to all three manufacturers, Microsoft, Sony, and Nintendo, for their approval process—in addition to sending it to Tony Hawk himself! This is not a minor moment in the development cycle of the game: If the product fails quality control, the console company (under the terms of the license the publisher signed with them) may be able to require changes or even force the publisher to kill the project.

A Sample Development Schedule

Here's a sample development schedule for a hypothetical 18-month project. Let's say that we're going to make a console game with a license of some kind—*Pro Linebacker Barbie*®, or something of the sort. We'll have to include extra time in the schedule for the license holder and console manufacturer to check over and approve the product. We'll also assume that this is an English-only game, so it doesn't require localization. This example does not include the marketing or sales effort, which is, of course, going on in parallel with development.

Suppose we pick a ship date of November 15, 2005—in other words, in time for the holiday shopping season. We'll want to leave a month of lead time for the console company to manufacture the games for us, because a lot of other publishers are having their games manufactured around then, too. (Cartridge games for machines like the Game Boy Advance actually require longer than this; for disc-based games it's often less, especially at other times of the year). That sets our actual gold master date at October 15, 2005. Therefore, for an 18-month project, we'll need to start on April 15, 2004.

- ***April 15, 2004: Pre-production begins.*** A small team is doing concept design, technical research, and prototyping work. They're also building the tools they will use during full production.
- ***July 15, 2004: Conceptual design complete; partially playable prototype.*** Three months into the project, the designers will have written a thorough design script. The programmers will use it as a "requirements document" to create a technical design for the final product.
- ***October 15, 2004: Full production begins. Technical design complete; tools complete; playable prototype.*** Using the prototype, the designers can see how the game's mechanics are working and fine-tune as necessary. The team staffs up to full strength.

During full production, there will be numerous milestones to make sure the project is on track. Since these vary with the actual nature of the code and content, I haven't included them here.

- ***April 15, 2005: Formal testing begins.*** Although the game isn't complete yet, the producers have been testing parts of it all along. Now, a testing manager assembles a test plan and assigns a limited number of testers to begin work on the parts of the game that are finished.

- ***May 15, 2005: Alpha. All features are present.*** All aspects of the game should be playable, though the levels, artwork, and audio may not be complete. Testing staffs up to full strength for alpha test.
- ***July 15, 2005: Beta. All content is present.*** All the creative elements of the game—pictures, sound, and text—should be complete, all the levels built, and the game should not crash. Beta testing begins.
- ***September 1, 2005: QA. The game appears to be finished and bug-free.*** Since this is a console product, no configuration testing is necessary, but the game goes through intensive testing by the Quality Assurance staff to be sure it is ready for the approvals process. Beta isn't formally over until QA says it is; they have the last word.
- ***September 15, 2005: Approval process begins.*** Copies of the game go off to the license holders and to the console manufacturer for their approval; they also go to the appropriate government or industry regulatory bodies for a rating evaluation.
- ***October 15, 2005: Gold master!*** The game goes to the console company for manufacturing. Unless there's going to be a sequel or a version on another platform, this project is officially over.

This makes it all look marvelously simple and easy, but, of course, it isn't. Schedules can slip and projects get behind for an infinite number of reasons, but I won't depress you by listing them here. As in this example, schedules are frequently created backward from the desired ship date (the day when the game is first available to the customer). If the ship date is too near to get all the work done on time, one of three things must happen:

- The publisher has to decide to ship it later.
- The developer has to add people to the project.
- Both publisher and developer have to agree to reduce the scope of the game.

There's also a fourth option: making everybody work harder. This is often tried, but almost never works.

STAGE 5: MANUFACTURING

At long, long last—a year or even two years after that initial, brilliant idea—the game is ready to be manufactured and go out to the distributor or retailers. At this point, it has "gone gold" in industry jargon.

In the case of console games, the console manufacturer actually constructs the boxes and presses the discs. Manufacturing console games requires more time than PC games, because the console game companies have to do *every* publisher's games. That means a lot of publishers standing in line, waiting anxiously for their product to come back. Console games also tend to come in standard boxes, so the publisher has little control over what they get back. They design the printing, but the shape of the container is out of their hands.

With PC games, the publisher can shop around to find the best manufacturing deal. Often there are huge economies of scale: a run of 70,000 games can cost exactly as much as a run of 100,000. The manufacturers want large runs so they don't have to reset their equipment for someone else, and they're willing to give discounts to get them.

Pressing a single CD or DVD costs under a dollar in large quantities; the real money goes for the cardboard and those beautiful multicolored sleeves. Box design has a significant effect on the customer's sense of perceived value. A heavy box with a big manual inside feels valuable. A box with a cover flap and more details inside seems nice; there's more to read about the product. A box with very little printing on it is suspect. It gives the impression that there isn't much to the game.

Ultimately, however, manufacturing is one of the places where the publisher wants to keep costs as low as possible, because it's where the rubber meets the road so far as profits are concerned. Every dime they spend on developing and marketing the game, they can consider an investment in its future success; every extra dime they spend on the box comes straight off the bottom line.

WAR STORIES

My first task when I was hired at Electronic Arts was to help reverse-engineer the original Nintendo Entertainment System, or NES. A company named Tengen (part of Atari Games) had announced that they found a way around the Nintendo "lock-out chip" that was in every game cartridge to discourage pirating (and independent publishing). I believe EA thought it would soon be possible to publish their games for the NES, as long as they could legally figure out how the machine worked on their own in order to reverse-engineer it.

EA assigned Jim Nitchals and me to figure out how the NES worked. We didn't even know what microprocessor it used. We desoldered and removed the ROM chips inside a few game cartridges and jury-rigged them up to EPROM burners to read out every byte inside them. There it was, the machine code that we had cut our collective programmer teeth on: MOS Technologies 6502 instructions! (Most of the early personal computers used the 6502 microprocessor, so we weren't that surprised.)

Once we knew what kind of microprocessor was in the machine, everything else fell into place. We could now write programs to try to figure out how the beast worked. Eventually, we were able to discover everything about the machine through looking at what

WAR STORIES CONTINUED

published games did and trying things out with our own programs.

To complicate matters, Jim and I had been working away from all of the other game developers and in fact had been entrusted not to communicate anything about what we were doing to anyone. We had to keep a logbook on how we reverse-engineered the NES, and we had to endure hours of mind-numbing meetings with EA's attorneys to make sure we followed the same "clean room" techniques that Compaq Computer had followed when they reverse-engineered the IBM PC. This wasn't a physical clean room, with bunny suits and air-filters, but a legal *clean room, designed to make sure EA couldn't be sued by Nintendo for violating their copyrights.*

EA didn't end up supporting the NES, but later on the company used some of the techniques we had learned to reverse-engineer the SEGA Genesis. Jim and I ported Populous*, a PC game, to the Genesis. A trade show was coming up in a few months and Trip Hawkins (the CEO of EA at the time) wanted to present a working Genesis game to some SEGA executives at that show. It was a great coup to prove to them that EA could develop games for the Genesis, legally but without their permission, before it was even available in the United States.*

Although EA never actually built unlicensed cartridges for sale, they used this knowledge to force SEGA to give them a better license deal than any other publisher had, and to allow EA to manufacture the cartridges themselves at a substantial cost savings. Of all the publishers who produced SEGA Genesis products, only EA was allowed to manufacture its own cartridges. SEGA put up with this rather than fight because they wanted EA's support to help the Genesis beat the Super Nintendo. Both companies profited handsomely from the arrangement.

—Kevin McGrath, Retired Game Programmer

WRAP-UP

That's it, beginning to end, soup to nuts. You now know how a game goes from a brilliant idea in one person's mind to shelves and shelves full of shiny boxes. Once you have your job in the game industry, you'll have an idea of what's going on and why. In the next chapter, I'll start talking about how you make yourself ready to *get* that job.