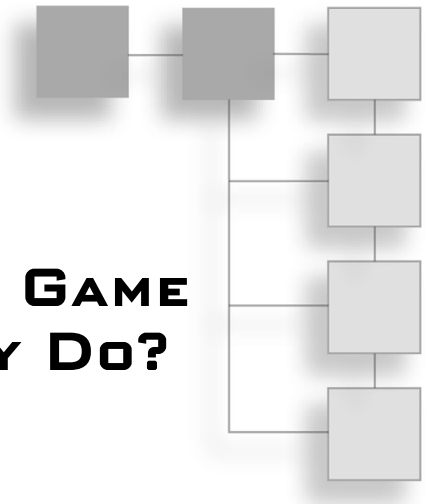
CHAPTER 1



What Does a Video Game Producer Actually Do?

As you've purchased this book, you're probably eager to get straight to the point. I'll get straight to detailing just what a game producer actually does, because for many people (both inside and outside the video game industry), it is a mystery.

So just what does a video game producer actually do? As outlined by Dave Perry during his keynote speech at the 2004 Game Developer's Conference, a video game producer is the person

- Whose primary focus is on the delivery of the video game as a completed project.
- Who knows every person on the team by his or her first name.
- Who works late with the team and is available to provide guidance whenever necessary, any time, day or night.
- Who clearly communicates with anyone who can affect the game, positively or negatively, as it is the game producer's responsibility to bring everyone into the fold of game production.
- Who runs interference with anyone who can affect the game or otherwise sidetrack the product.
- Who does everything possible to sell, promote, and protect the game *and* the team.
- Who has the complete confidence that he or she can cross any obstacle and face any challenge.
- Who does whatever it takes to help the team deliver the game.

A Brief History of Producing

In traditional media and the entertainment business, a *producer* is one who assembles the cast of a play, brings an artist or talent to a studio, or organizes a publicly broadcasted event. The producer has an all-encompassing role; that is, he or she takes primary responsibility for the completion of the event, project, or program. Specifically, the role of a movie or television producer included casting, hiring a director, finding the script, handling contracts, distributing the finished product, financing, scheduling, location management, promotion, marketing, and PR (Public Relations). Similarly, the role of the record producer, an occupation that emerged with the popularity of the phonograph, involved finding talent, hiring the recording studio, securing the distribution and financing from a record publisher, promotion and PR events, as well as contracts and legal agreements for the artist, writers, and musicians.

In the 21st century, the role of producer has evolved, as new mediums of entertainment—most notably, interactive entertainment—have emerged. Today, the role of a video game producer may include all of the responsibilities of a television, movie, or record producer, plus a lot more. Indeed, interactive entertainment includes many aspects and challenges not faced by traditional movie, television, or record producers—for example, finding ways to include new rendering technology or the ideal set of game-development tools for specific product type; devising ways to ensure that the core compelling gameplay is clearly focused, communicated by the Design team, and included into the game's development; or ensuring that a highly addictive and compelling entertainment experience is outlined in the design documentation.

The Diverse Role of a Video Game Producer

If excellence is your goal as a video game producer, expect to experience many challenges. This section is designed to introduce the various types of diverse challenges you can expect to face as a video game producer, as well as some of the common responsibilities enjoyed by any producer, regardless of medium. They appear here in alphabetical order, not in order of importance. After reviewing this list, you should have a basic understanding of some of the challenges faced by producers and what their daily work consists of. As you'll see, a producer requires a wide variety of skills, experiences, and knowledge to meet the challenges they face on a daily basis. Although not every producer position is the same, nor does every producer face all these challenges, it is likely that during the course of your career as a producer, you'll find that every circumstance, skill, or trait listed here will prove valuable.

Actively Contribute

A producer contributes to the team effort, vision, and work required to complete the game. This means that the producer just does not sit in his or her office reworking the Microsoft Project schedule all day, but actively participates in team meetings, design meetings, problem solving, and design ideas, and makes decisions when required. The contribution of the producer should be seamlessly integrated into that of the team, providing the oil that keeps the team running smoothly.

Apply Good Decision-Making Skills

It may seem obvious that good decision making is a critical aspect of game producing. After all, who wants to make bad decisions? The problem is, you can't really know whether a decision is a good one or a bad one until after it's been made, hence the saying, "Hindsight is 20/20." *Good decision making* here refers more to the *process* of making decisions than the decisions themselves. Indeed, there may well be times when it is better to make a decision, even if it's wrong, than to endlessly delay on deciding or to flip-flop on the decision after it has been made.

Specifically, *good decision making* refers to the process of securing all relevant information, asking for recommendations and advice from other stakeholders, setting a deadline before which the decision must be made, and then making the decision and announcing it and the reasoning behind it to all who are involved. Even if a decision is wrong, following this process ensures that the team has an adequately clear direction during the course of developing the game and instills confidence in others about the producer. As an added bonus, if the reasoning behind the decision is sound, then the decision will be right the majority of the time. Of course, no one is a perfect decision maker, but not following a clear decision-making process only compounds the chance that a bad decision will be made for the wrong reasons—and worse, after much delay.

Attend Budget Meetings

At budget meetings, the producer must explain the status of the budget, accounting for how much money has been spent on the project and how much more needs to be spent on the game in order to complete it on time. This may often include an analysis of the profit-and-loss (P&L) statement for the project (or brand).

Be Forward-Thinking

Forward thinking means looking and reasoning ahead—one day, one week, or one month ahead—so that there is no opportunity for a problem to suddenly present itself as an obstacle to completion of the game. This includes investigating and finding ways to solve problems *before* they affect the game's development. Licensing the game-development tools and securing the rights to use third-party software in the game are excellent examples of the forward thinking that is required of a producer.

Other fundamental decisions related to the game's development include the minimum system specifications for the game, what video card it will support, or the number of platforms on which the game will be released. A producer must consider all the issues that can potentially affect a game's development and weigh them in a forward-thinking manner.

Build Consensus

Seeking to build a consensus whenever possible is generally one of the best ways to ensure a harmonious relationship within a team. Building confidence in the team by asking their opinions when forming a decision is one of the ways to build a consensus. Getting others to believe in your ideas as if they were their own is the principle behind building a consensus.

Sometimes a hard decision must be made, one that not everyone agrees with. But before getting to that point, do your best to build a consensus and take other's recommendations. Getting people to reach an agreement as a whole is generally a tough challenge.

Deliver Animation

While a video game is mostly about gameplay, a video game producer is often charged with delivering specific animations for the game to help convey the story, provide content for the marketing campaign or both. The demands created by being responsible for delivering both gameplay and animation simultaneously and in concert with the other requires an extreme amount of enthusiasm for the project. Creating a specially rendered movie trailer for marketing purposes is another good example of divergent tasks that a producer must balance against the other. In each case, whether the animation is used for marketing, in the game, or both, a producer must work closely with the art director and the animator to ensure that the animation is completed on time, is appropriate for the game, and uses conventional film techniques to show the progression of the story and how it relates to gameplay.

Develop a Pre-Production Plan

The producer must develop a *pre-production plan*, which is the foundation on which the game's overall development rests. In the pre-production plan, the producer works with the team leaders to establish the critical paths for completing the product and determines the recommended course of action for accomplishing their goals.

Pre-production is the time when the Game Development team prepares to make the game and lays the groundwork for that goal. Ideally, when the team begins production, all of the goals are clearly defined and the course is set.

Pre-production is also used to test and refine art export pipelines and game design documentation, as well as to establish the art asset listing for the game. Detailing the art, design, and feature requirements for the game and including them into a schedule is also part of this process.

tip

Often, I recommend completing a prototype or mini-game during pre-production that establishes itself as a test case for the real game that you're making. In addition to costing less than the final product, doing so enables team members to learn a tremendous amount about the process and to make adjustments as needed before undertaking development on a larger project.

Develop a Production Plan

Just as the producer must develop a pre-production plan, he or she must also develop a production plan, which is the actual documents or set of documents that comprise the plan for the game's development. Although a plan is often believed to remove uncertainty, in reality, the production plan is simply the best estimate of how the game is to be completed. The production plan consists of several smaller plans describing all the elements of the game and how they are going to be completed. This includes plans from each team involved in game creation, including designers, artists, and programmers. The production plan brings these different documents together, enabling interested parties to review the project as a whole, with an understanding of risks, the required budget, a feature list, the schedule, and art assets.

Specifically, a production plan consists of the following documents:

- **Essence statement or executive summary**. Simply put, this document outlines why the game is fun.
- **Creative design document**. This document outlines the creative and artistic vision for the game.
- **Technical design document.** This document outlines the required features of the game as described in the creative design document.
- **Risk-management plan.** This document outlines what the risks are and how to minimize them.
- **Schedule for development.** This can be a detailed schedule or just a monthly milestone schedule.
- **Budget and financial requirements.** This document outlines monthly cost allocations, capital expenditures, and the like.

Generate Game-Design Documentation

The producer must work with the Design team to clarify the game-design documentation and ensure that it is easily producible and cohesive. Game designers have an inherent predisposition to create overly complicated, complex, and disjointed designs, that may require a lot of development time to fix. Game designers are supposed to do this, but the producer's role is to help guide them back to the course of what is producible, possible, and still fun.

Handle Hardware Manufacturers

The producer is the key contact for hardware manufacturers such as Intel, NVIVIDA, ATI, Creative Labs, Microsoft, and console manufacturers like Sony, Nintendo and Microsoft's Xbox. The role of the producer in this context is to develop and maintain good relationships with the representatives of these hardware manufacturers, ensuring that the Game Development team has access to the latest hardware, drivers, technical support, and knowledge required to use the hardware to its fullest potential. This includes obtaining evaluation or pre-release versions of video cards and sound cards, as well as production versions, and ensuring compatibility with the widest range of hardware products, peripherals, and console add-ons, such as steering wheels, pedals, dance pads, or maracas (in the case of *Samba De Amigo*).

Handle Legal/Contractual Issues

A working knowledge of the law related to contracts and business litigation is often required of a producer. Although you're certainly going to have access to the advice of lawyers and other professionals, you need to understand the fundamental principles of contract law, civil litigation, intellectual property ownership, as well as the basic legal principles that go into contracts, such as exclusive and non-exclusive licenses. Although your first project as a producer may not require this knowledge, the longer you're a producer, the more likely it becomes that this knowledge will be very important.

Handle Licensing and Branding

Licensing includes developing and managing the relationship between the licensee and how the product's development evolves when created under license. *Branding* refers to the overall vision for a product (either within a licensed brand or an original brand) such that the product is consistent with the vision for the brand and supports the main strengths of the brand and the brand's development. A brand is a very important part of software marketing, as it includes the distinctive name identifying the product and the manufacturer. A producer must grasp the vision and concept behind both a license and the brand when managing the development of a video game using either or both.

Handle Middleware Issues

Middleware issues refers to the issues and challenges that face the Game Development team when they're using middleware tools, such as those provided by Criterion Software or Gamebryo. These middleware tools give game developers a standard set of tools and features to use in a limited variety of game genres. When the game design calls for a specific feature set or implementation beyond what the middleware can support, the producer must be able to understand and resolve the issues with the middleware. This can be done by contacting the middleware provider and asking for support or by licensing another third-party toolset to provide the required functionality for the game designers and world builders. Other times, it may not be that easy to solve, which is why the producer must devise a range of alternative solutions and help pick what's best for the game.

Handle Platform Transition

Platform transition refers to the period of time in the video game industry when an existing console platform is currently entrenched in the market and doing well but a new console is being readied for commercial release. During this period, game development for consoles becomes extremely challenging because the hardware for the new console plat

form has often not been finalized, nor have video game developers been provided with development kits (specialized computer hardware for this new platform). The platform-transition period requires forward thinking on the part of a producer to facilitate the delivery of the hardware and flexibility in the game's design—not to mention the development schedule.

Handle Public Relations

Public relations involve meeting the press and presenting a pre-release version of the game for demonstration and evaluation. This requires time for a press tour, excellent speaking abilities, a well-honed message, and passionate enthusiasm for the project. Public relations are an ongoing responsibility of the producer—he must provide interviews, screenshots, and related material to ensure interest in the game in development. Excellent interpersonal skills are required when working with a representative of the Public Relations department at the publisher.

Handle Quality Assurance

Many producers, associate producers, and assistant producers are charged with the responsibility of overseeing the quality assurance and testing efforts for their games. In certain cases, this involves interfacing directly with hands-on testers who work with the Game Development team, or with a Quality Assurance department, with the liaison being through the lead tester or QA department managers. Working with the Quality Assurance department is challenging and stressful, yet is rewarding as the Game Development team fixes bugs and gets the product closer to completion. Database management is often required to input and track bugs properly.

Help Sales

The producer does everything he or she possibly can to help the sales of the video game. This includes meetings with the Sales department, buyers, and Marketing and PR departments, as well as working trade shows. The top-selling products require excellent support from their producers so that everyone involved in selling the product into the market will clearly understand the vision behind the game, and know why it is exciting and compelling. Clearly communicating that message to the sales channel, the industry, and the consumer is an extremely large part of a game's success.

Hire/Interview

The producer is largely responsible for hiring new members of the Game Development team. Of course, there are exceptions, but generally the producer is responsible for

screening candidates and ensuring that they will work well with the rest of the team. Finding potential or new team members who will shine is a skill that every producer must develop if he or she is to be successful in the long term. The hiring and interview process usually includes programmer tests, designer questionnaires, in-person interviews, and phone screening. Some producers are responsible for salary negotiations, but all are responsible for ensuring that they hire the right people for the right job on the right team, and that everyone on that team will be able to work well with the new team member.

Interact with Upper (Executive) Management

A producer will often have the opportunity to work directly with upper management personnel and influence their decisions. Honing of this skill is very important because it affects everyone who works with you and, ultimately, your career as a producer. Understanding how executives evaluate opportunities, manage risks, and determine the right course of action is key.

Know Games

The producer must be one of the foremost authorities on video games. This means that the producer must apply his or her knowledge of games and understanding of why games are fun to the current project. Being able to discuss design principles with the Design team, articulate an artistic vision from a competing product, or critique a specific feature set in comparison to the overall market with the programmers are all examples of when a producer's knowledge of video games will be extremely useful.

Learn

Always look for new ways to improve methods, find efficiencies, improve best practices, and otherwise expand the learning opportunities for yourself as well as for the team. Referring to previous experience or knowledge as the ultimate resources limits the effectiveness of a producer. With emerging technology and development processes, producers should always be looking for ways to expand their learning capabilities and opportunities.

Manage Assets

Asset management is the process and method of managing the thousands of assets that must come together to complete a video game. This includes art assets such as models, textures, interface elements, menu screens, cinematic sequences, and special renders. On the design side, this includes world-building tools, multiplayer design, functionality specifications,

use cases, story, script, core gameplay, and adherence to the game's essence statement. On the programming side, this can include tools, functionality, export pipelines, and documentation. Lastly, but certainly not least, asset management involves management of outside delivery of content such as voiceover recording, sound effects, music (ambient and linear), localization (including all the sound and text assets for several different languages), and the creation and delivery of marketing and PR materials for the game.

Manage Big Teams

Managing big teams is a massive challenge and presents its own unique set of challenges, such as the coordination of export pipelines, feature-set integration, and asset tracking. Indeed, merely communicating with your team becomes inherently more difficult when it is comprised of 60 to 100 people, as compared to a team of 30 or 40. The trick here is to break down the large team into several smaller teams and delegate responsibility for managing those smaller teams to other producers. Most importantly, focus on finding the people who work well together and put them in charge of key systems. They'll set the example in terms of productivity and efficiency for others.

Manage Foreign Localization

Foreign localization refers to the process of creating a game in one language and then localizing its content to apply in many worldwide markets. For example, most games are developed in English and then localized to German, Italian, or French. Generally, this means managing the process of including thousands of individual files that have an alternative language's voiceover, artwork, or menu screens in the game before it ships to retail stores. Creating product for worldwide markets is required for almost all successful video games. The localization process is often complicated and time consuming, and requires an excruciating attention to detail and a sound localization management process.

Manage Resources

Resource management refers to deciding when and where resources should be allocated. Obviously, every task cannot be done at the same time, so tasks should be prioritized, and then resources should be assigned to complete that task. This process of resource allocation often requires constant re-evaluation and adjustment in order to ensure that resources are properly allocated across a project that includes dozens of people and often spans several years.

Manage the Art Process

A producer must manage the process of creating artwork for the game. This includes tracking art assets as they are completed and identifying the art assets that are incomplete. Often, art-production resources will need to be reallocated to ensure that the art schedule stays on track. The role of the producer is to work with the Art team to manage this process and to plan for the appropriate risks.

Manage the Audio Process

This topic could be an entire job of itself. Producing audio involves managing the audio contractors who provide voiceover recordings, editing, sound effects, and music (both ambient and linear tracks), as well as mixing or recording in studio if that is required. Being able to produce audio and understand the impact of the sounds and music on the visual is as much an art as it is a science.

Manage Vendor Relationships

Managing vendor relationships is often overlooked and undervalued, but a producer often must contract with outside companies to provide key services that go into the game's development. Products or services provided by outside vendors include software support for 3D modeling applications (such as 3D Studio Max, Maya, and Lightwave), sound libraries, or even third-party software tools such as Incredibuild from Xoreax Software. Even computer manufacturers like Alienware have helped supply hardware used in the development of the games I've produced. Each of these vendor relationships is important.

Often, producers use vendors and contractors on multiple productions once they've developed a good working relationship. As the relationships are maintained, these vendors and contractors are easy to use on the next project, allowing you to skip the process of looking for a qualified vendor who can help make your game.

Manage Your Time

Time management is perhaps the most fundamental aspect of being a producer. Indeed, time management is the single biggest factor that affects whether a game is cancelled. Why? Because the one finite element in game development is time. It is impossible to make time go backward, but it is always possible to spend more money on a game, or to sacrifice the quality of a game. Time management is the process and method of allocating resources on a project to ensure that they have the most effective and efficient impact on the project within the timeline allocated for the project.

Pitch

Pitching is the ability to sell an idea or a concept—specifically, the game concept and development plan. When pitching a game, the producer must be the salesperson for that game to everyone who is listening, whether they be executive management, the publisher, or the press. A successful pitch requires a producer who is excited and passionate about his or her product and can effectively convey that excitement and passion to others so that they agree to buy the product. A game rarely gets off the ground without a good pitch.

Possess Industry Experience

Industry experience is important because it provides an accurate frame of reference for a producer. It should be noted that although there are some similarities, experience in the video game industry is unlike experience in the general entertainment industry. Having never lived the same day twice, an experienced producer in the video game industry is much more likely to be able to effectively problem-solve the common and uncommon challenges that every software project faces. The more years of experience a producer has, especially when coupled with projects on a variety of hardware platforms, the more valuable he or she will be. Experience on a variety of projects sizes is also valuable, as large projects have different problems than do small projects.

Provide Clarity and Focus

Clarity and *focus* refer here to the producer's understanding of the game and the compelling experience it provides to the user. With all the daunting tasks that lie in the path of a game's successful development, providing clarity on which are the most important is critical. Focus on the most important and high-risk tasks first. When the situation becomes daunting, with programming, art, and design requirements apparently on divergent paths, the producer's ability to provide clarity of the final goals of the project, and generate focus to that end, may save the game.

Provide Marketing Support

Providing support to the Marketing department is a challenging task for even the best producers. Demands for marketing assets, like screenshots, special renders, reviews of box cover artwork, magazine ad copy, and sell sheet reviews are just a few of the demands that the Marketing department places on the Game Development team. As a producer, the challenge is to find the best way to deliver these assets and information to marketing without affecting the team or sidetracking their development efforts to make a great game.

Schedule

Scheduling combines the skills discussed under "Time Management" and "Resource Management," and puts them into a plan that is presentable to others and easily understood. Often, updating the schedule can be a large part of a producer's role. Learning to master Microsoft's Excel, Project or even Access is an important part of managing the schedule.

Sow Discipline

Electronic Arts is one of the leaders in today's video game industry. Why? Because EA embraces a disciplined approach to software development and applies it to all areas of its business. Indeed, one of the most critical factors in the success of an organization is the discipline that it applies to its business and production methods. Positive discipline is an important part of an organization, because it ensures the business's long-term success.

As a producer, you can sow the seeds of discipline by doing the following:

1. Set goals for people and encourage them to succeed. Writing down these goals and offering rewards when they are achieved encourages your employees to do even better.
2. Obtain commitments from each team member to accomplish these goals. Obtaining commitments ensures that everyone understands your expectations and agrees to meet those expectations.
3. As work progresses, measure progress and benchmark results from one group against others who are tasked with similar roles. Note the progress of the team and its members, and identify when work can be done more efficiently or effectively.
4. Hold others accountable for their actions and their commitments, especially if they do not seek help when struggling with a task. Of course, several outside influences, external factors, complications, and challenges affect people's ability to complete work, but there are also many avenues to help them achieve their goals and overcome those challenges.

SMART Goals

SMART Goals is a slick acronym for goals that are

- **Specific**. Be as specific as possible when establishing your goals. Clarity is king in this regard. It's hard to motivate people to complete goals that are non-specific, and even harder to measure their results.
- **Measurable.** Measurable results are what matter. Finishing the project report by Friday or finalizing the functional specifications for the game's design by the end of the month are both measurable and concrete examples.
- **Acceptable.** Set your own goals. No one knows your capabilities better than you do. Determine what is acceptable for your own standards and then live up to or exceed them.
- **Realistic.** Don't plan for a lot of accomplishments if you know that only a few are really possible. Focus on a few big goals rather than many smaller ones.
- **Time bound.** Define when you want your goals to be completed as well as when you're going to have the time to work on them. If you write it down now, it is a lot easier to make it actually happen.

Take Ownership

When a producer *takes ownership*, it means that he or she has a personal sense of pride and accomplishment associated with his or her work and that of the team. *Ownership* refers not to taking credit for the work accomplished, but making it your goal to remove obstacles so that the work can be accomplished. A producer who doesn't take full ownership of his area or set of responsibilities is generally not very effective. Taking ownership of a project, game, or team must be balanced with an objective view of the game's development progress, goals, and marketplace conditions. A producer cannot take ownership for a project without regard for the external factors that affect a game.

Teach Others

Being able to teach others is another required skill. Because communication is a principal part of the job, producers must be able to communicate their knowledge, lessons, and experience to others on the team. Often, simply being able to explain the situation or circumstance or to answer questions from team members ensures that problems within the team are addressed before a noticeable impact on the team's productivity occurs. Being able to share the rationale behind a decision in a clear, concise way shows the team that

decisions are not made arbitrarily. Other times, the producer may be called upon to integrate a new team member and teach him new procedures, methods, or best practices that will make his work more efficient. These types of situations require a producer to share his or her knowledge and to be able to teach to those who are willing to learn.

Understand Cinematic Production

Cinematic production includes the storyboarding, animatic creation, and actual rendering or filming of a game's cinematic sequences. These are the sequences that tie the story together with the gameplay for the user. A working knowledge of or a background in film direction, scene composition, lighting techniques, script, relevance to gameplay, and music scored to visual are important to success in this area.

Understand Development Systems

Development systems refers to the specialized computers required by game developers that allow development on proprietary platforms or game consoles such as the Xbox, Playstation 2, or Nintendo GameCube. Often, these hardware systems are difficult to procure; it is the responsibility of the producer to secure their delivery for the team. Only a limited amount of game development can be done on normal workstations without the use of a development system that emulates the actual hardware for which the game is designed and developed.

Work with the Programming Team

The producer must work with the Programming team to establish key goals early on in the development process and then ensure that the programmers have all the tools they need to succeed. Throughout the development process, a producer's job is to track progress, understand dependencies between workloads and features, establish critical milestones, and help solve (non-technical) problems for the programmers.

Software-Production Methods

All games are not alike, and neither are the methods used to create them. Indeed, there are several ways to develop a video game. This section discusses how some of the common software-production methods are applied. Along the way, you'll get an overview of how a video game comes together and how the process is managed. Further on in the book, I'll discuss the specifics of each portion of the game in more detail: what tools you as a producer can use to keep a project on track and how to apply them.

Code-Like-Hell, Fix-Like-Hell

The *code-like-hell, fix-like-hell* method of game software development, shown in Figure 1.1, is probably the most common and oldest model. Some advance planning is done, but rarely is it followed, updated, or referenced. Programmers code as quickly as they can to implement what they think the design calls for; it is then tested and fixed. This model is prone to failure because of the stressful situations that arise during the development. Programmers cannot work at a constantly frenetic pace, nor can designers and testers. As a result, this process breaks down over time. It leaves room for error, and those errors aren't fixed until after somebody finds them, at which point the code is further along than it was when the errors or bugs were introduced. This model is generally only suited for small projects with simple requirements because the code is difficult to maintain over a longer period (six or more months). This method is also referred to as the *extreme game development method* or the *XP method* and is shown in Figure 1.1.

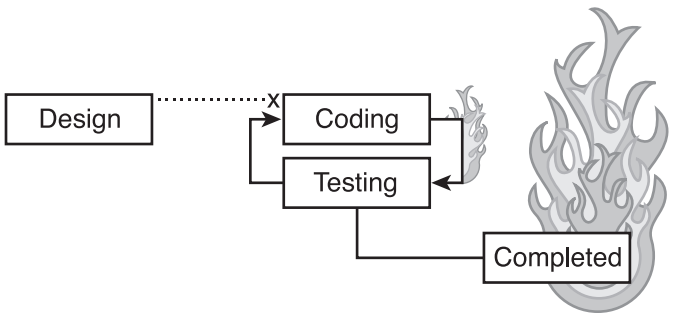


Figure 1.1
The code-like-hell, fix-like-hell approach.

Increments to Completion

Increments to completion is the software-production method that calls for the software to be developed in relatively compact, finite increments. Developing an adventure or first-person shooter (FPS) game using this process might work because once the world engine and tools exist, every piece of the game is simply an increment added to the original core. As the pieces come together from various parts of the team, they are checked against the high-level design document. The specifications of the design and the key requirements of the game are outlined in this high-level documentation, but low-level documentation is not completed until just before or just after the feature is implemented—usually when the designers and the programmers agree on what is possible with a feature and how it should be implemented.

One advantage of using the increments-to-completion model, shown in Figure 1.2, is that various features of the game can be developed in parallel or independently of the rest of the game's parts. This is often good in theory, but it is more challenging in practice to implement successfully without a high degree of coordination and easily modifiable code structures. Although the benefits of this model are not always outweighed by the detractions, a producer should consider that using this model often allows the team to demonstrate a playable game early on in the development process and continually progress as different systems, features, and artwork become available for integration into the game. Often, lessons learned from the first increments (such as a prototype phase) turn out to be quite helpful in the long run.

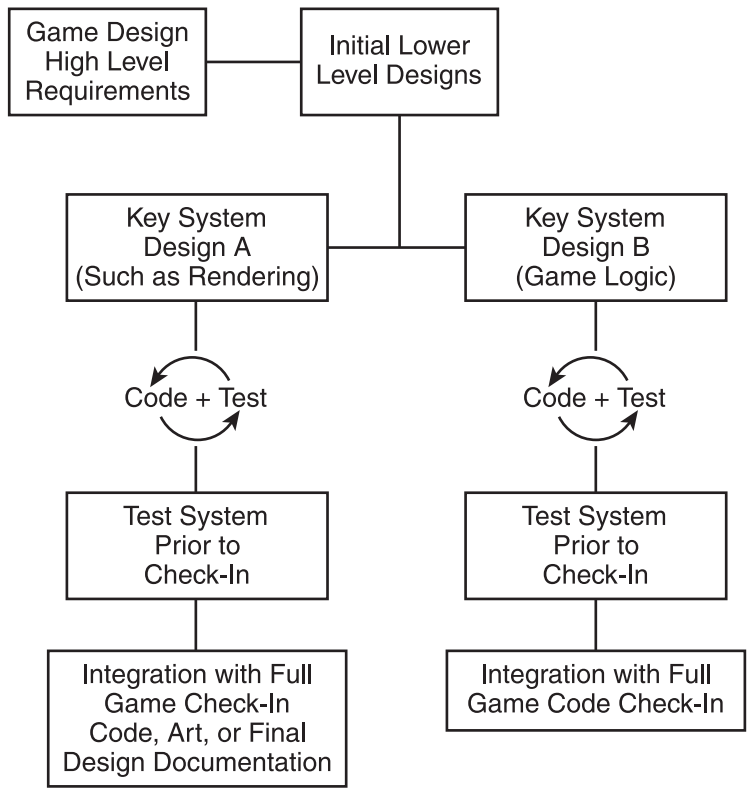


Figure 1.2
The increments-to-completion approach.

The Cascade

Cascade is used to describe an approach in which the entire team focuses on the next part or parts as one part of the game is completed (see Figure 1.3). Under this approach, parts of the game come together relatively quickly with little time for testing between feature creation and implementation. There is often a need to review and revise a previous part of the game because as more parts are added to it, the function, appearance, or intended usefulness of a particular feature may change. It is difficult to change major parts or systems of a game when using this model, and it requires that everything go correctly right from the start. For this reason, this method is not recommended for use in game development.

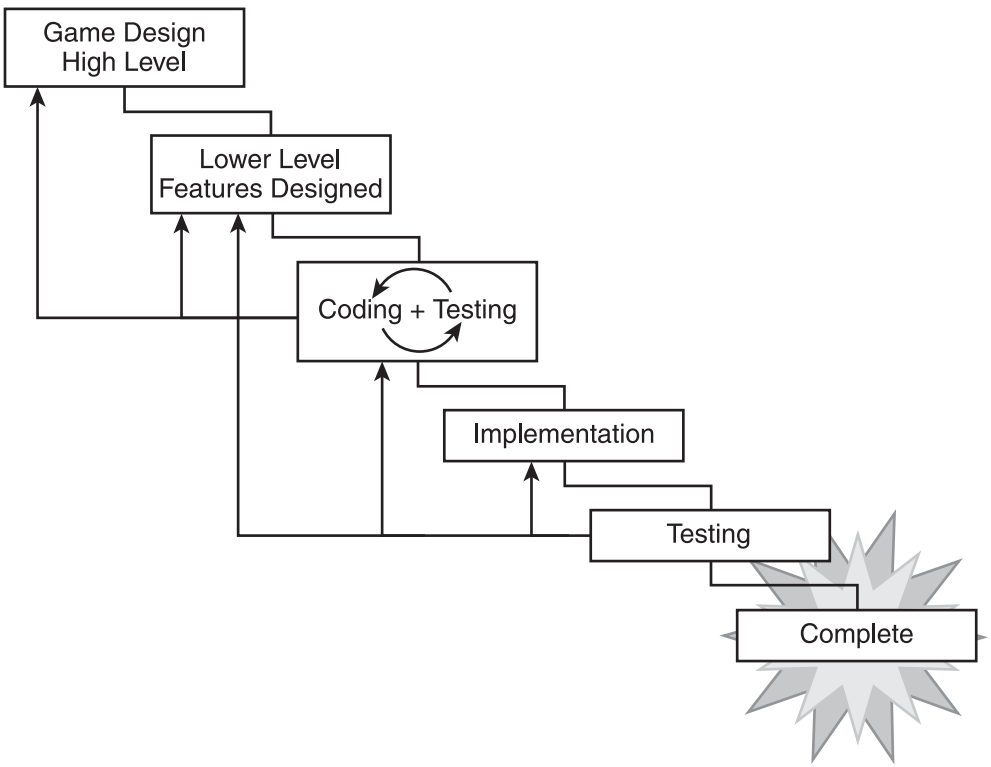


Figure 1.3
The cascade approach.

Iterate Until You Drop

The *iterate-until-you-drop* method is probably the most flexible software-production method in that its entire purpose is to help you, the producer, define the key areas of the game, begin developing them, and then finalize the game's design partway through the development process. This is often beneficial when a game developer is unsure what features will be included with competing products that are scheduled to be released around the same time. It allows the Game Development team to respond to changing market forces or demand, providing the flexibility to quickly implement working code such that the team, the publisher, and the game designers can iterate the fun factor of the game (meaning make the game more fun as they play it more and include more and more gameplay refinements with iteration). This is shown in Figure 1.4. As a generally useful process, iteratation is a process not to be undervalued. Especially when you consider the number of games that have been published in the history of software development that aren't fun.

This method is often useful and is sometimes recommended when the producer has the appropriate tools and understands the methodologies behind object-oriented code and software development. Sometimes, however, a situation's biggest strength can be its biggest drawback when it is not managed effectively. For example, most game designs fail to fully specify a complete list of the elements that make the game fun. Often times, the most fun part of the game isn't realized until the game comes together in some pre-release form. The iterate-until-you-drop method becomes exactly that: a never-ending treadmill of software development that can always be improved. Adapting to changing requirements is the critical benefit of this method. It is the producer's role to ensure that this method doesn't get out of control or become a justification for ever-expanding budgets and development timelines.

When using this method, keep in mind that using the proper tools and tactical methods is critical to completing the project without going over budget or investing many, many years in the same game.

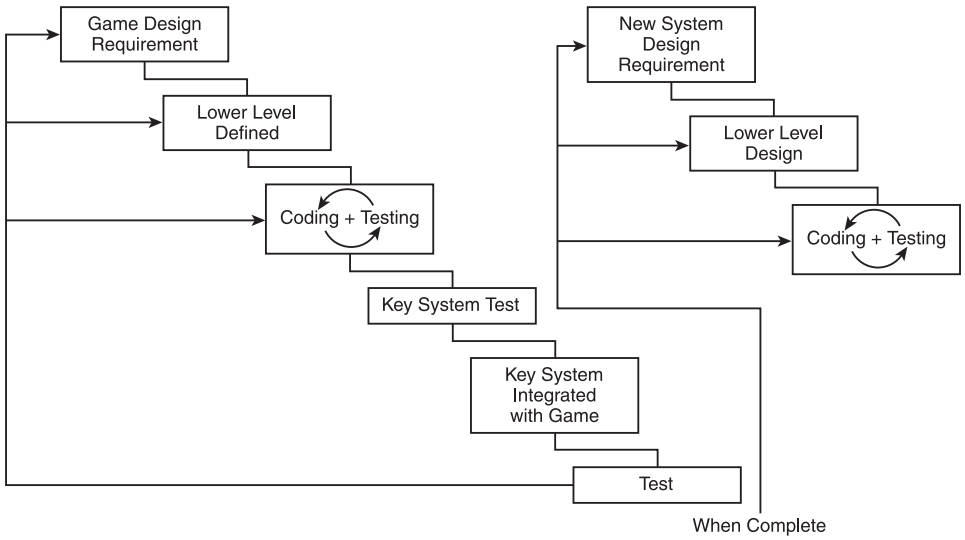


Figure 1.4
The iterate-until-you-drop approach.

Agile Project Management

In *Agile Project Management* (Pearson Publishing, 2004), author Jim Highsmith discusses an excellent method for combining the best of the iterate-until-you-drop approach with a few key principles that every producer should respect. I've provided the gist of Highsmith's ideas here because agile project management is a principle referenced throughout this book.

Agile project management, an excellent process for organized and disciplined teams, centers on the following key stages of software development:

- Envision
- Speculate
- Explore
- Adapt
- Finalize

note

The name of each stage of the process references both the activities and the intended results of the stage. Highsmith avoids terms like initiate, plan, and direct because these terms are associated with a prediction of relative accuracy—of which video game software projects are probably the antithesis.

Envision

Envision refers to the game designer's vision or the essence statement of the game. Other things to consider during the Envision stage are scope, gaming-community support, and how the team will work together. During this stage, the key selling points of the game are determined, as is just what makes the game fun. This is when the question "What game are you making and who is it for?" is answered. In addition, this is the stage in which the question "Who are you going to use to make this game?" is answered. The Envision stage is where the game designer works with the producer to start spreading the enthusiasm for starting a new game-development project to management, and when key team members envision how they are going to work together.

Speculate

Speculation may conjure up images of reckless actions, gold panning, or playing the stock market. It's actual dictionary definition, however, reads as follows: "To mediate on a subject; reflect." A secondary definition is "To engage in a course of reasoning often based on inconclusive evidence." This describes precisely the actions required of a producer, a designer, an artist, or a programmer when beginning work on a game-development project. A producer should realize that when a plan is introduced, its purpose is to eliminate uncertainty in a highly uncertain process. When the uncertainly fails to evaporate, a producer should re-plan, although some fail to do so. Use of the term *speculate* accurately describes the reality of video game–software development, as well as the volatile market for video games.

The Speculate stage consists of determining the high-level requirements for the game; outlining the work required to complete the game; and creating a development plan (including a schedule with resource allocations), a feature list, risk-management plans, and a budget.

Explore

The Explore stage in agile project management refers to the process of finding and delivering features. Delivering the features required by the game design is the first and foremost objective of the Explore stage. You do this using effective time-management, resource-allocation, and risk-management strategies. Secondly, the team creates a collaborative project community with some elements of self-organization (so that the producer isn't burdened with questions such as who sits where). The producer simply acts as a facilitator during this process. Lastly, during the Explore phase, the producer must manage the team's interactions with management, Marketing, Quality Assurance, and any other stakeholders (like licensees).

Adapt

The term *adapt* refers to the necessary modifications or changes that are required to keep the project focused and on schedule. *Adapt* also refers to the incorporation of lessons learned and the application of those lessons to the project in midstream (generally, responding to change is more important than simply following a plan blindly), and to the life of the project, which means that an adaptation of the Envision stage is also possible as the team learns new information through its adaptations.

In this stage, the team's results are often viewable and open to criticism, both technical and creative. At this stage, the producer should analyze the project status as compared to the published plan. Often, this analysis should focus solely on the budgetary and fiscal impact of the game to date, with a comparison of the features required to play the game. The results of this analysis are included into the adaptation and re-planning stage for the next iteration.

Finalize

Finalize refers to the process of completing the project and doing all that is necessary to document and learn from the mistakes and lessons that this project taught the team and its producer. Often, the goal of a project ending eludes a Game Development team, as patches and constant upgrades or add-on packs are required, but most projects are worthy of celebration once they are completed.

Planning and Scheduling

Now that you have a basic understanding of the process involved in building a game, let's get onto planning and scheduling. There are two basic ways to schedule a project:

- The top-down approach
- The bottom-up approach

Taking the Top-Down Approach

The top-down plan is generally developed by a single person or a small group to provide an overview of what a project schedule *might* look like. Unfortunately, this plan often gets adopted as gospel and is rarely revised without considerable frustration and angst. A top-down plan generally does not involve the participation of those who are going to be called upon and tasked to do the things in the plan. Therefore, the top-down plan should only be considered a goal or a guideline. At best, it is a guess; at worst, it is totally wrong. This type of estimating tool fosters the understanding of what the game's scope and complexity may be. Be careful when creating top-down plans, guarding their release and clearly stating that they are to be revised when more information on the game is available, and when the input from the team is available.

Planning from the Bottom Up

When a producer plans from the bottom up, he or she gathers the relevant team members to work on developing a plan for building the game collaboratively and with a consensus of what is possible by when, and what resources it requires to reach that goal. All art assets, game features, and other project requirements are identified. This can generally only occur after a significant amount of pre-production planning has been conducted (the pre-production phase is discussed later in this chapter).

When planning from the bottom up, the first step is to identify short- or near-term goals. Work to establish these goals on a clear schedule, and then start checking off items on your feature or art-asset list, fitting the appropriate tasks into the schedule as your team concurs. By using this process and involving people from all disciplines necessary to develop the game, you'll share ownership in the schedule. People generally appreciate being asked for their opinions and input, and the scheduling process for video games is not an exception. Furthermore, it lessens the opportunity for a single team member, or a subset of the team, to fall behind schedule because they agreed to the schedule when it was created. (You'll be surprised at the extra effort people exert to protect their pride.) The alternative is much less desirable, especially when team members tell their producer, "I told you so" because the schedule was created without their input.

caution

The detailed, bottom-up plan is only as good as the game design. If you're working with a game design that is non-specific, ill-defined, or otherwise nebulous, be extremely cautious about committing to the plan you're creating.

Scheduling Constraints

When determining what type of schedule you want to follow when developing your game, you'll want to consider the two types of schedule-constraint models:

- Time-constrained model
- Resource-constrained model (including fiscal resources)

Both models provide an assimilation of several smaller plans developed by each of the team groups: designers, artists, and programmers. To establish your schedule, consider both of these models, as described next.

Each team lead (lead artist or art director, as well as the lead programmer and lead designer) looks to identify why and how his team can accomplish the goals of a project. To this

end, he is responsible for creating the small portion of the production plan that relates to his area of responsibility. The lead programmer creates the programming schedule (with the help of the producer). The lead designer and lead artist create the respective Design and Art Production schedules. Then the producer combines these smaller schedules into one larger one and looks for dependencies, critical paths, and resource allocation requirements.

Time-Constrained Model

Reviewing the time-constrained scheduling model is the first step in determining a reasonable production plan. When working with the time-constrained model, focus on building a plan without accounting for the resource requirements. Simply determine the tasks, the features, the owners, and of course, the dependencies of each task. Take a guess at the duration for each task, and then try to link the dependant tasks together in a way that makes sense. The dependencies should be sorted such that the most fundamental and riskiest tasks come first in the schedule, followed by less-risky tasks. The point of this exercise is to determine whether it is even possible for a project of a given size and scope to fit within the timeframe a producer is considering.

Resource-Constrained Model

After you've created a time-constrained model for the production plan, convert it to a resource-constrained model. Focus your efforts (and that of the team leads) on assigning tasks appropriate to the skills of the available resources. Identify where and when the resource may complete the task, keeping in mind that you are still in the Speculate stage. Also identify key tasks for which no appropriate resource exists. That means you'll need to hire a new team member with the appropriate skills or recruit an existing team member to learn the required skills for this task. Clearly outline the work days, vacations, and weekends, and provide an allowance for sick days, meetings, and general administrative overhead (for all team leads).

This is where the scheduling gets challenging. Your next task is to include contingency buffers, such as working weekends and flexible days, in each part of the schedule and allocate your contingency equally across all areas of the production plan. Then look for ways in which to divide the production plan into major pieces, called *milestones*. This helps track the progress of the team and provides clear, measurable ways for management to review the status of the game's development.

note

The producer may have to re-create this constraint model several times during the game-development project as the game's development progresses. Realize, too, that the production plan you create is actually the assimilation of several smaller team plans, so focus on breaking down tasks into finite and measurable definitions.

When you switch to the resource-constrained model, you may realize that the game that's designed can't be produced due to various time and resource constraints. That means it's time to look for efficiencies or to start cutting features or subsets of the game's design.

Critical-Path Planning

The *critical path* is the series of tasks or events that make up the start and end of a project (see Figure 1.5). This series of tasks has no available schedule cushion or buffer. If you want to shorten the duration of a project, you must focus on the tasks that are on the critical path. Generally, only a relatively low number (less than 25 percent) of the total tasks of a game are on the critical path, but the items on the critical path generally *must* follow a specific order and sequence of events.

The critical path is the shortest route to completing the game, so it is important to properly plan for the tasks on it. Critical-path planning involves understanding and knowing the sequences of events and ensuring that all potential problems or hurdles associated with accomplishing the tasks on the critical path are addressed prior to reaching that task on the schedule.

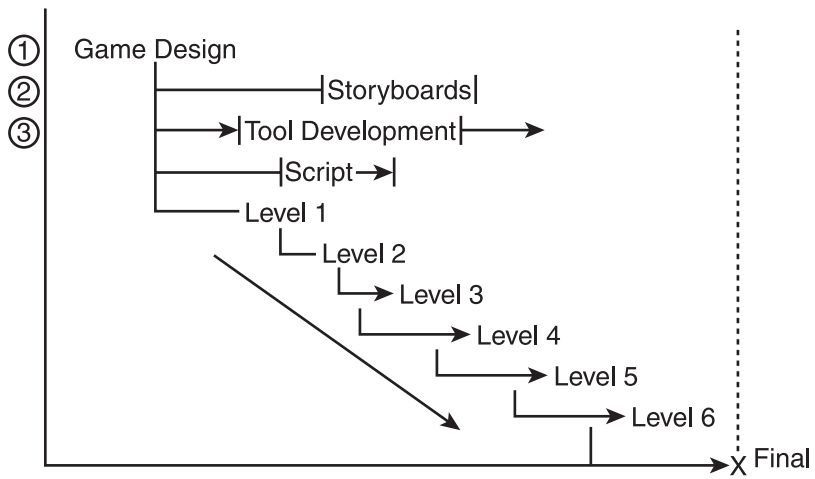


Figure 1.5
The critical path of this project is on the world and level creation, not the storyboards, tool development, or script.

Contingency Planning

Even the best plans go awry. That's why it's important to plan for contingencies before you launch into developing a game. Following are a few ways to plan for those unpredictable events that can throw a monkey wrench into any project.

Plan for Overtime

This is an industry standard in the video game industry; it's assumed that long hours will be required to keep the project on schedule. This has no fiscal impact on the developer or publisher of a video game because most employees are classified as professionals and are therefore exempt from overtime pay.

Hire Additional Personnel

Be careful when considering adding personnel to the team to ensure that it stays on schedule. Although at first glance it may seem that doing so is a great way to keep things on schedule, it often works in reverse. For one thing, additional people require additional management. Secondly, the addition of a new team member inevitably involves an introductory period during which the new hire becomes familiar with his or her role as well as the procedures and goals associated with the project. During this introduction, the new team member often slows others, preventing them from completing their work on schedule. Lastly, the addition of personnel always costs more money, not only in wages but also in equipment.

Work During Holidays and Vacations

Although some game-development and publishing companies have liberal policies regarding time off in lieu of other days worked, it is generally bad form to require the team to work holidays on a regular basis. Indeed, I do not recommend requiring team members to work on holidays or vacations as a contingency. Time off is important for employees for much-needed rest periods during times of high-intensity activity. Although I have occasionally persuaded employees during a busy period to work on a holiday or a vacation in exchange for a day off in the future, I suggest you to plan around holidays and urge your employees to take them. Just work with the team to ensure that vacations are on balance, and are scheduled at a mutually agreeable time for the employee, the team, and the project. In practice, this is not that hard; it just takes time, negotiation, and understanding.

Use a Formula

I'll discuss this later on in the chapter.

Don't Schedule Work for Team Leads

Don't schedule the team leads—that is, the lead programmer, the lead artist, and the lead designer—to do much work related to the actual production of the game; Allow them the time to delegate to their teams and to help manage the process; doing so ensures that the leads are free to jump into their team's work as required. Often, this is not entirely practical, because the team leads usually have many tasks that only they can complete. That said, if a producer works with each lead to establish a framework for minimizing their direct contributions and maximizing their indirect contributions, and refrains from scheduling a lead on any task that is on the critical path, it ensures that the project has ample flexibility to respond to changing parameters, requirements, or conditions.

note

The idea of not scheduling work for team leaders may give you pause. After all, the whole reason your team leaders became team leaders in the first place is because they produced excellent work. If the team leader for the Art team is actually your best modeler, wouldn't it make sense to put him on the job of *modeling*? Yes and no. If you put the lead artist in a managerial role rather than a production one, it enables that person to convey his or her skills to other team members. This cross-pollenization of skills can only help the project. As a compromise that enables you to, for example, enjoy the modeling skills your Art team leader possesses as well as to enable him or her to teach others, consider using the lead artist to set the standard for the 3D models and then have him oversee the rest of the 3D modelers to help them achieve the same standard.

Make Time for Testing After a Task is Completed

As a producer, you should plan for every programmer, designer, and artist to include some time in their daily schedule to test the completion of their—and others'—work. There are few bigger hassles than having team members turn in incomplete or erroneous code or art assets to a game; this breaks the game and prevents the team from playing it, not to mention affecting the team's ability to stay on schedule. Often, I assign a buddy to double-check a peer's assets before they are checked into the main source control being used by the game engine.

Set Aside a Contingency Reserve Fund

The producer should create a contingency fund for the inevitable day when the project faces a significant hurdle, or is threatened with going over budget or over time. It's a great feeling to be able to answer the heated question "How are you doing to pay for that?" with a very pleasant "Out of my contingency fund, of course!" I often show the contingency fund as a line item in my budget proposals and create it as soon as I start working on a game's budget.

Using a Formula to Calculate a Schedule

When a producer evaluates the list of the tasks required to complete a game, the first question he or she should ask is "How long will it take to reasonably complete this task?" The problem is that this is harder than you might think to calculate. The easiest way is to assume that one person is working at it full time until it is completed, but this is rarely the case because individuals often get sidetracked or required to perform other tasks during the work day (helping others, filling out insurance forms, attending meetings, completing report, writing e-mails, and so on). Believe it or not, a formula exists that can account for time off and help plan for other contingencies that inevitably interfere with the completion of any task. It's one that has been used in other industries, but that I've modified based on my experience with the game industry and call the "Extremely flexible project planning formula". Here's how it works:

Task Name:	Direct X Compatibility and Rendering
Best Case:	10 days
Worst Case:	25 days
Most Likely Case:	15 days
Formula:	2 (Best Case) + 3 (Worse Case) + Most Likely = X/6
Result:	2(10)+3(25)+15=110/6, or 18.33 days,
Practical Application:	18.33 provides a buffer of 3.33 days over the most-likely case scenario

Although this formula is not infallible, it does help to provide a buffer and to quantify a normally very difficult question. The best part is that a producer can go back to the task list, make adjustments, and re-estimate using this formula partway through the task or the project (as mentioned in the discussion about the agile project management theory's Adapt stage). You can also adapt the formula to reflect the specific circumstances of your team. For example, a team lead may not use this formula because he or she is able to concentrate only on actual game-production work 50 percent of the time, while the rest of the time is invested in administrative overhead. Therefore, you may wish to modify the formula to reflect this, showing that the team lead can only contribute to the project at 50 percent of the estimated 40 hours per week. Therefore, it may take a lead up to 36.66 days to complete this same task (assuming all other things are equal).

Software-Factory Efficiencies

A *software factory* is an organization that uses a set of processes and methods that work like a factory does—with each set of tools or technologies being specialized, but remaining interchangeable and reusable depending on the needs of a specific project. The software factory is built around the core understanding that the code, tools, and documentation from certain features and game engine systems are to be reused. In this way, they are in constant need of updating, but are to remain useful and independent components.

note

The concept of the software factory is simply summarized here for your easy reference and understanding. Read Chapter 11 of *Game Architecture and Design,* by Andrew Rollings and Dave Morris for complete details relating to the software factory.

There are many advantages to using a software factory, including the following:

- The average length of a project is shortened if the team already has a set of familiar tools.
- Cross-platform releases are inherently easier because the team is already familiar with how common libraries are used for different platforms.
- Often, the code that is written using these tools is more stable and reliable because it uses components that are tried and tested. Indeed, although doing so can be difficult, using the software-factory approach makes it is possible for code to be reuseable, maintainable, and well-documented.
- The software-factory method enables the dissemination of information about core systems of the factory. This is helpful in the event a team member leaves; the project does not grind to a halt or become seriously jeopardized because the departing team member is taking valuable knowledge with him or her.
- It is often easier for the producer to estimate and track progress on a project that is using the software-factory approach because he is familiar with how similar system were implemented on previous projects using the same factory methods.

That said, there are a few disadvantages to using this approach that should not be understated. For one, setting up a software factory can be as expensive and time-consuming as making a game itself. That's because the factory requires tools such as libraries, world-building tools, sound-placement tools, level editors, engine architecture, object placement and preview tools, and key-rendering libraries.

Other disadvantages include the following:

- The first project undertaken using this method often takes longer because the factory is still being set up. Wrappers libraries (small tools used by the team to ensure the hardware specific functionality of the game on a particular hardware platform) must be developed for each platform, allowing the code and libraries to be used on multiple platforms.
- Although the code is generally more generic in a software-factory environment, it is more difficult to develop and account for all the possibilities and potential uses of a particular routine. As such, this code often takes longer to develop in the initial stages.
- If you add new people to the team, there will be a learning curve for them when it comes to using the software-factory methods and libraries, as these can be quite specialized.
- Generally speaking, more administration and forward thinking is required when creating a software factory.

Stages of Game Development

Now that you have a sense of the various issues surrounding game creation, let's look at a hypothetical game project from beginning to end and identify the major stages of the process. Of course, each publisher has its own specific procedure for creating games, but nearly all boil down to the phases outlined here, which I've tried to simplify somewhat.

Concept

The *concept* phase is when the game concept is written down. It's when brainstorming occurs, and when ideas are generated.

Prototype

During the prototype phase, a prototype of the game is developed so that users can start to experience the fun as described in the concept documentation. The prototype phase typically lasts 2–4 months, depending on the tools available to the team to create a prototype.

Pitch

During this phase, game developers pitch their game to management or, if the developers don't currently work for a game publisher, to a publisher's representatives. The pitch explains why the game is a great concept, why it is ready and right for the video game market, whether it is producible, and how it will be developed.

Green Light

This phase begins after the pitch is approved for production, and involves gathering a team to begin working on the game. This phase may involve interviewing and contacting several video game–development companies to assemble the right team for the game. Often, a game cannot enter pre-production until business and legal issues around the project are resolved, such as who owns the technology and the creative intellectual property such as the main character and story.

Pre-Production

Pre-production is when the Game Development team works on defining the production pipelines, identifying the needs and uses of the tools they'll need to make the game, and outlining and fleshing out the details behind the game's design.

Production

Production is when the game building actually begins. 3D models are created, worlds are built, sound is recorded, textures are applied, cinematics are filmed, game logic is authored, and all the other pieces of the game are made and put together. This is often a long process—at least 12 months, and often much longer.

Quality Assurance

The *quality-assurance* or *testing* phase often occurs at the final stage of production, about 3–4 months before the game is scheduled to go to manufacturing. During this stage, the game is tested for bugs, errors, deficiencies, or incompatibilities.

Final Gold Master

The Final Gold Master phase is when the master CD is burned and sent to the manufacturing facility for duplication. In the case of a console game there's the "Submit to Hardware Manufacturer" process, which requires attention to detail and potential revisions to meet their approval guidelines. Each hardware manufacturer has their own rigorous testing guidelines and each product must meet or exceed their own QA requirements of functionality and playability.

In the case of online games on either console or PC, teams must continue to work together even after the Final Gold Master phase to fix bugs and prepare patches for release prior to the game being available in retail stores.

Video Game Development Process Models

There are two basic models for a video game's development. The standard model and the Forward loading risk management model. The advantages and disadvantages of each are discussed in detail in Chapter 8. However, in order to provide some context of how this model works and its practical application, I've included the standard model here for consideration and familiarization.

The standard development model shows the common approach to video game development of the past few years. By following this model, the game development team has the most risk in the project at precisely the time in which the project is the most expensive on a daily or weekly basis. While this is often unavoidable, when it is possible to avoid the convergence of risk and expense, a producer should do so.

Since this is the common method, it is widely accepted that there are few alternatives to this model. However, a new model was proposed by Mark Cerny at the DICE conference in 2002. Avoid the standard model whenever possible, focusing on defining the fun in your game before the costs start increasing. By managing risk a producer excels at their job and ensures that the financial investment in the project is responsibly managed.

The Final Word

Now that you've had a chance to skim the surface of all that a producer is called upon to do in the pursuit of excellence and success, turn to the next chapter to understand the specialties of each role and how they differ by company, project and specialty area.