# Chapter 2

# Design Components and Processes

## *Chapter Objectives*

**After reading this chapter and completing the exercises, you will be able to do the following:**

- Understand the player-centric approach to game design and apply its principles to your own design practice.
- Know how the core mechanics and the user interface work together to create gameplay for the player.
- Explain how gameplay modes and shell menus make up the structure of a game.
- Recognize the three stages of game design (concept, elaboration, and tuning) and know what kinds of design work take place in each stage.
- Know the kinds of jobs required on a design team.
- Know the kinds of documents that a game designer is likely to need to make and what they are for.
- Know the qualities required of a good game designer.

## Introduction

Game design is the process of:

- Imagining a game
- Defining the way it works
- Describing the elements that make up the game (conceptual, functional, artistic, and others)
- Transmitting that information to the team that will build the game

A game designer's job includes all of these tasks. In this chapter, we begin by discussing our approach to the task, called player-centric game design. We then introduce the central components of any video game, the core mechanics and the user interface. We show how these components are used in the design process. Finally, we examine the various job roles on a design team and some of the qualities that it takes to be a game designer.

In the video game industry, all but the smallest games are designed by teams of anywhere from three to 20 people. We cannot know what role you will play on such a team, and therefore we write as if you are the lead designer, responsible for overseeing everything. (If your team is small, or if you are designing alone, you may perform some or all of the roles yourself.) Our text is written as if you are designing for a home console machine or personal computer, although much of the material here is applicable to any game device.

## Approaching the Task

Before we can discuss the process of game design itself, we must discuss how to approach it—that is, what you are actually trying to accomplish and how to think about it. Over the years, people have tried many approaches to game design, and some of them are better than others. A few tend to result in catastrophic failures. This book teaches you an approach that seems best to us.

### Art, Engineering, or Craft?

Some people like to think of game design as an art, a process of imagination that draws on a mysterious wellspring of creativity. They think of game designers as artists, and they suppose that game designers spend their time indulging in flights of imagination. Other people, often more mathematically or technologically oriented, see game design as a type of engineering. They concentrate on the methodology for determining and balancing the rules of play. Game design to these people is a set of techniques. Aesthetics are a minor consideration.

Each of these views is incomplete. Game design is not purely an art because it is not primarily a means of aesthetic expression. Nor is game design an act of pure engineering. It's not bound by rigorous standards or formal methods. The goal of a game is to entertain through play, and designing a game requires both creativity and careful planning.

**KEY POINT**

Game design is a craft, combining both aesthetic and functional elements. Craftsmanship of a high quality produces elegance.

Interactive entertainment is an art form, but like film and television it is a collaborative art form. In fact, it is far more collaborative than either of those media, and the designer is seldom granted the level of creative control that a film director enjoys. Consequently, no single person is entitled to call himself the artist. Designing games is a *craft,* like cinematography or costume design. A game includes both artistic and functional elements: It must be aesthetically

pleasing, but it also must work well and be enjoyable to play. The greatest games—the ones whose reputations spread like wildfire and that continue to be played and discussed long after their contemporaries are forgotten—combine their artistic and functional elements brilliantly, achieving a quality for which the best word is *elegance.* Elegance is the sign of craftsmanship of the highest order.

## The Player-Centric Approach

We favor an approach called ***player-centric game design.*** We believe this approach is the one most likely to produce an enjoyable game, which, in turn, will help it to be a commercially successful one. Many other factors affect the commercial success of a game as well: marketing, distribution, and the experience of the development team. Many of these are beyond the control of the game designer, so no design or development methodology can guarantee a hit. However, a well-designed game undoubtedly has a better chance of being a hit than a poorly designed one.

We define player-centric game design as follows:

> *Player-centric game design is a philosophy of design in which the designer envisions a* representative player *of a game the designer wants to create, then accepts two key obligations to that player:*
>
> - *The duty to entertain: A game's primary function is to entertain the player, and it is the designer's obligation to create a game that does so. Other motivations are secondary.*
> - *The duty to empathize: To design a game that entertains the player, the designer must imagine that he is the player and must build the game to meet the player's desires and preferences for entertainment.*

The first obligation, the duty to entertain, can be adapted somewhat if the game is intended for education, research, advertising, political, or other purposes, but for recreational video games it is imperative. If a player is going to spend time and money on your game, your first concern *must* be to see that he enjoys himself. This means that entertaining the player takes priority over your own desire to express yourself creatively. You must have a creative vision for your game, but if some aspect of your vision is incompatible with entertaining the player, you should modify or eliminate it.

The second obligation, the duty to empathize, requires you to place yourself in the position of a representative player and imagine what it will be like to play your game. You must mentally become the player and stand in his shoes. For every design decision that you make—and there will be thousands—you must ask yourself how it meets the player's desires and preferences about interactive entertainment. Note our emphasis on a *representative player.* It is up to you to decide what that means, but this hypothetical being must bear some resemblance to the customers that you want to actually buy the game. You

cannot insist that your typical player be a highly skilled gamer unless you want to restrict your customer base to nothing but highly skilled gamers.

Player-centric game design means thinking about how the player will react to everything in your game: its artwork, its user interface, its gameplay, and so on. But that is only the surface. At a deeper level, you must understand what the player *wants* from the entire experience you are offering—what motivates her to play your game at all. To design a game around the player, you must have a clear answer to the following questions: Who *is* your player, anyway? What does she like and dislike? Why did she buy your game? The answer will also be influenced by the game concept that you choose for your game, which is why we discuss both player-centric design and game concepts together in Chapter 3.

This process of empathizing with your player is one of the things that differentiates games from presentational forms of entertainment. With books, paintings, music, and movies, it is considered artistically virtuous to create your work without worrying about how it will be received, and it's thought to be rather mercenary to modify the content based on sales considerations. But with a video game—whether you think of it as a work of art or not—you *must* think about the player's feelings about the game, because the player participates in the game with both thought and action.

In our opinion, player-centric game design produces the most entertaining, enjoyable games. However, there are two common misconceptions about player-centric design that you must avoid.

**Misconception 1: I Am My Own Typical Player** For years, designers built video games, in effect, for themselves. They assumed that whatever they liked, their customers would also like. Because most designers were young males, they took it for granted that their customer base was also made up of young males. That was indeed true for a long time, but now it is a dangerous fallacy. As the market for games expands beyond the traditional gamer, you must be able to design games for other kinds of players. In the player-centric approach, this means learning to think like your intended players, whoever they may be: little girls, old men, busy mothers, and so on. You cannot assume that players will like what you like. Rather, you must learn to design for what *they* like. (You may also find that you grow to like a game that you didn't think you would as you work to design it!)

One of the most common mistakes that male designers make is to assume that male and female players are alike, when in fact they often have different priorities and preferences. For an excellent discussion of how to reach female players without alienating male ones, read *Gender Inclusive Game Design,* by Sheri Graner Ray (Ray, 2003). With every design decision, ask yourself, "What if the player is female?" Does your decision apply equally to her?

A few game developers argue that they don't want to work on any game that they personally wouldn't want to play—that if a game doesn't appeal to them, they won't have any "passion" for it and won't do a good job. Taken to its

logical conclusion, that means we would never have games for young children, because young children can't build games for themselves. Insisting that you must have passion for your game or you can't do a good job on it is a very self-centered approach—the opposite of player-centric design. Instead of passion, we prefer professionalism, the willingness to work hard to do a good job because that's what you're paid for, regardless of whether you personally would choose to play your game for entertainment. If you are a true professional, you *can* create a brilliant game for an audience other than yourself. The design team on the game *Bratz: Rock Angelz* consisted entirely of adult men, yet the game was a big success because the designers learned how to think like a 10-year-old girl, its intended player. They talked to girls and women, studied other products that girls like, and took seriously their duty to empathize (Elling, 2006).

### COMMANDMENT: You Are Not Your Player

Do not assume that you epitomize your typical player. Player-centric game design requires you to imagine what it is like to be your player, even if that person is someone very different from you.

**Misconception 2: The Player Is My Opponent** Because arcade (coin-op) games have been around a long time, some of the techniques of arcade game design have crept into other genres where they are not appropriate. Arcade games make money by getting the player to put in more coins. Consequently, they are designed to be hard to play for more than a few minutes and to continually threaten the player with losing the game. This places considerable constraints on the designer's freedom to tell a story or to modulate the difficulty of the gameplay. The famous Japanese designer Shigeru Miyamoto, who invented the arcade game *Donkey Kong* (and with it the entire *Mario* franchise) eventually abandoned arcade game design because he found it too limiting.

The arcade model encourages the game designer to think of her player as an opponent. It suggests that the designer's job is to create obstacles for the player, to make it hard for the player to win the game. This is a profoundly wrongheaded approach to game design. It takes no account of the player's interests or motivations for playing. It tends to equate "hard" with "fun." And it ignores the potential of creative games, which may not include obstacles at all. Game design is about much more than creating challenges.

If you are working on multiplayer competitive games, in which the players provide the challenge for each other, you're less likely to make this mistake. But it's an easy trap to fall into when designing single-player games because it's up to you to provide the challenges. Never lose sight of the fact that your design goal is to *entertain* the player by a variety of means, not simply to oppose her forward progress through the game.

**COMMANDMENT:** The Player Is Not Your Opponent

Do not think of the player as your opponent. Game design is about *entertaining* the player, not opposing the player. There are many ways to entertain a player.

## Other Motivations That Influence Design

In the commercial game industry, video games are always built for entertainment, but even so, several factors can influence the way a game is designed. In this section, we will examine some of them.

When a company chooses to build a game specifically for a particular market and to include certain elements in its design specifically to increase sales within that market, that game is said to be ***market-driven.*** You might think that any game made for sale should be market-driven. Experience shows, however, that most market-driven games aren't very good. You can't make a brilliant game simply by throwing in all the most popular kinds of gameplay. If you try, you get a game that doesn't feel as if it's about anything in particular. The best games are expressions of the designer's vision, which makes them stand out from other games.

The opposite of a market-driven game is a ***designer-driven*** game. In designer-driven games, the designer retains all creative control and takes a personal role in every creative decision, no matter how small. Usually he does this because he's convinced that his own creative instincts are superior to anyone else's. This approach ignores the benefits of play-testing or other people's collective wisdom, and the result is usually a botched game. *Daikatana* is an often-cited example.

Many publishers commission games to exploit a ***license:*** a particular intellectual property such as a book (*The Lord of the Rings*), movie (*Die Hard*), or sports trademark (*NHL Hockey*). These can be enormously lucrative. As a designer, you will work creatively with characters and a world that already exists, and you'll be making a contribution to the canon of materials about that world. One downside of designing licensed games is that you don't have as much creative freedom as you do designing a game entirely from your own imagination. The owners of the license will insist on the right to approve your game before it ships, as well as the right to demand that you change things they don't like. In addition, there is always a risk of complacency. A great license alone is not enough to guarantee success. The game must be just as good as if it didn't have a license.

A ***technology-driven*** game is designed to show off a particular technological achievement, most often something to do with graphics or a piece of hardware. Nintendo's original *Starfox* was specifically designed to exploit

their FX graphics chip, for example. Console manufacturers often write technology-driven games when they release a new platform to show everyone the features of their machine. The main risk in designing a technology-driven game is that you'll spend too much time concentrating on the technology and not enough on making sure your game is really enjoyable. As with a hot license, a hot technology alone is not enough to guarantee success.

***Art-driven*** games are comparatively rare. An art-driven game exists to show off someone's artwork and aesthetic sensibilities. Although such games are visually innovative, they're seldom very good because the designer has spent more time thinking about ways to present his material than about the player's experience of the game. A game must have enjoyable gameplay as well as great visuals. *Myst* is a game that got this right; it is an art-driven game with strong gameplay.

## Integrating for Entertainment

When one particular motivation drives the development of a game, the result is often a substandard product. A good designer seeks not to maximize one characteristic at the expense of others but to integrate them all in support of a higher goal: entertaining the player.

- A game must present an imaginative, coherent experience, so the designer must have a vision.
- A game must sell well, so the designer must consider the audience's preferences.
- A game with a license must pay back the license's cost, so the designer must understand what benefits it brings and exploit them to the game's best advantage.
- A game must offer an intelligent challenge and a smooth, seamless experience, so the designer must understand the technology.
- A game must be attractive, so the designer must think about its aesthetic style.

Player-centric game design means testing every element and every feature against the standard: Does this contribute to the player's enjoyment? Does it entertain her? If so, it stays; if not, you should consider eliminating it. There's no easy formula for deciding this; the main thing is to make the effort. As we have already quoted designer Brian Moriarty, too many designers "pile on gratuitous features just so that they can boast about them," which means they're not designing player-centrically.

There are sometimes reasons for including features that don't directly entertain: They might be necessary to make other parts of the game work, or they might be required by the licensor. But you should regard them with great suspicion and do your best to minimize their impact on the player.

## The Key Components of Video Games

In the last chapter, we discussed what a game is and what gameplay is. But where does gameplay come from, and how does the player interact with it? In order to create gameplay and offer it to the player, a video game is composed of two key components. These are not technical components but conceptual ones. They are the *core mechanics* and the *user interface.* Some games also use a third important component called the *storytelling engine,* but we will deal with it in Chapter 7, "Storytelling and Narrative." In this section, we introduce the core mechanics and the user interface and show how they work together to produce entertainment. Each of these components has a complete chapter devoted to it later in the book, so our discussion here is limited to defining their functions, not explaining how to design them.

### Core Mechanics

One of a game designer's tasks is to turn the general rules of the game into a symbolic and mathematical model that can be implemented algorithmically. This model is called the ***core mechanics*** of the game. The model is more specific than the rules. For example, the general rules might say, "Caterpillars move faster than snails," but the core mechanics state exactly how fast each moves in centimeters per minute. The programmers then turn the core mechanics into algorithms and write the software that implements the algorithms. This book doesn't address technical design or programming but concentrates on the first part of the process, creating the core mechanics. That process is addressed at length in Chapter 10, "Core Mechanics."

The core mechanics are at the heart of any game, because they generate the gameplay. They define the challenges that the game can offer and the actions that the player can take to meet those challenges. The core mechanics also determine the effect of the player's actions upon the game world. The mechanics state the conditions for achieving the goals of the game and what consequences follow from succeeding or failing to achieve them. In a conventional game, the players are aware of the core mechanics because the players must implement the rules. In a video game, the core mechanics are hidden from the players. The players experience them only through play. If the players play the game over and over, they will eventually become aware of the game's mechanistic nature and learn to optimize their play to beat the game.

One quality of the core mechanics is their degree of ***realism,*** which we define below. A ***simulation,*** in the formal sense, is a mathematical or symbolic model of a real-world situation, created for the purpose of studying real-world problems. If it is to have any validity, the simulation must represent some part of the real world as closely as possible (though aspects of it may need to be simplified). A game, on the other hand, is created for the purpose of entertainment.

Even if it represents the real world to some degree, it will always include compromises to make it more playable and more fun. For example, a real army requires a large general staff to make sure the army has all the ammunition and supplies it needs. In a game, a single player has to manage everything, so to avoid overwhelming him, the designer abstracts these logistical considerations out of the model—that is to say, out of the core mechanics. The player simply pretends that soldiers never need food or sleep, and they never run out of ammunition.

All games fall along a continuum between the ***abstract*** and the ***representational.*** *Pac-Man* is a purely abstract game; it's not a simulation of anything real. Its location is imaginary, and its rules are arbitrary. *Grand Prix Legends* is a highly representational game: It accurately simulates the extraordinary danger of driving racing cars before the spoiler was invented. Although no game is completely realistic, we refer to this variable quality of games as their degree of realism. For the most part, however, we use the terms *abstract* and *representational* to characterize games at opposite ends of the realism scale.

You decide what degree of realism your game will have when you decide upon its concept. The decision you make determines how complex the core mechanics are.

## User Interface

The concept of a ***user interface*** should be familiar to you from computer software generally, but in a game the user interface has a more complex role. Most computer programs are tools of some kind: word-processing tools, Web-browsing tools, painting tools, and so on. They are designed to be as efficient as possible and to present the user's work clearly. Games are different because the player's actions are *not* supposed to be as efficient as possible; they are obstructed by the challenges of the game. Most games also hide information from the player, revealing it only as the player advances. A game's user interface is supposed to entertain as well as to facilitate.

The user interface mediates between the core mechanics of the game and the player (see Figure 2.1). It takes the challenges that are generated by the core mechanics (driving a racing car, for example) and turns them into graphics on the screen and sound from the speakers. It also turns the player's button-presses and joystick movements on the keyboard or controller into actions within the context of the game. If it does this smoothly and naturally, the player comes to associate the button-press with the action. She no longer has to think, "I must press button A to apply the brakes." Instead she thinks, "Brakes!" and presses button A automatically. The user interface interprets the button-press as the braking action and informs the core mechanics; the core mechanics determine the effect of the braking and send an instruction back to the user interface telling it to show the result. The user interface adjusts the animation to show the car slowing down and presents it to the player. All this happens in a fraction of a second.

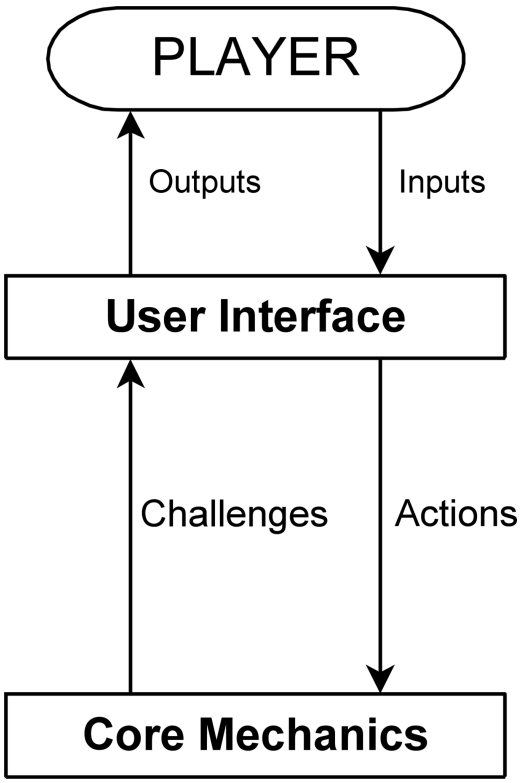


**FIGURE 2.1** The relationships among core mechanics, user interface, and player.

Because the user interface lies between the player and the core mechanics, it is sometimes referred to as the ***presentation layer.***

The user interface does more than display the outputs and receive the inputs. It also presents the story of the game, if there is any, and creates the sensory embodiment of the game world—all the images and sounds of the world and, if the game machine has a vibrating controller, the vibrations of the world too. All the artwork and all the audio of the game are part of its user interface, its presentation layer. Two essential features of the user interface of a game are its *perspective* and its *interaction model,* as shown in Figure 2.2.

**Interaction Models** As we described, the user interface turns the player's button-presses into actions within the game world. The relationship between the player's button-presses and the resulting actions is dictated by the game's ***interaction model.*** The model determines how the player projects her will, her choices and commands, into the game. In particular, it defines what she may and may not act upon at any given moment. Video games use a number of

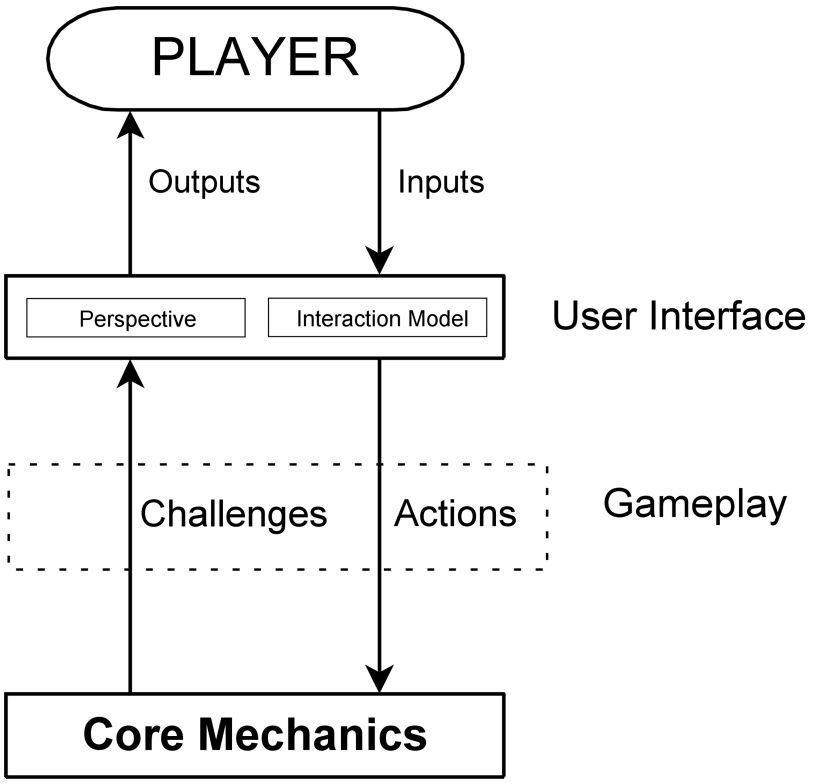


**FIGURE 2.2** Perspective and interaction model are features of the user interface.

standard interaction models, including multipresence, avatar-based models, contestant models, and so on. In a multipresent model, for example, the player can act on different parts of the game world whenever she wants to, reaching "into" it from the "outside." In an avatar-based model, the player is represented by a character who already *is* inside the game world, and the player acts on the world through that character. Just as the visible parts of a game's user interface change during play, a game can have more than one interaction model depending on what is happening at the time. Interaction models are discussed at greater length in Chapter 8, "Creating the User Experience."

**Perspectives** If a game includes a simulated physical space, or ***game world,*** then it will almost certainly use graphics to display that space to the player. The user interface must display the space from a visual ***perspective:*** a particular camera angle or point of view. Even if a game doesn't have a game world, it still may have a collection of data that it displays to the player in tables, charts, or some other form, and a given view of that data may be thought of as a sort of "conceptual perspective," a perspective formed upon the information provided.

We use the term *perspective* in this somewhat broader sense to include both visual and conceptual perspectives.

The most commonly used perspectives are first person and third person for presenting 3D game worlds and top-down, side-scrolling, and isometric for presenting 2D worlds. We discuss the question of game world dimensionality in Chapter 4, "Game Worlds," and address the merits of the different perspectives in Chapter 8, "Creating the User Experience."

**FYI** *Games Without Graphics*

Many of the early computer games were text-based, designed to be played on a printing terminal attached to a mainframe computer. Text-only games still exist in the form of quizzes or trivia games, especially for small devices such as cell phones. ***Interactive fiction***—text-only adventure games—has long since ceased to be a commercial genre, but it is still popular with a small group of hobbyists. Blind players can play text-only games using text-to-speech synthesizers. There is also a very small number of experimental audio-only games intended for the blind.

## The Structure of a Video Game

You now know how the core mechanics of a game work with the user interface to create gameplay for the player. A game seldom presents all its challenges at one time, however, nor does it permit the player to take all actions at all times. Instead, most video games present a subset of their complete gameplay, often with a particular user interface to support it. Both the gameplay available and the user interface change from time to time as the player is required to meet new challenges or to view the game world from a different perspective. These changes sometimes occur in response to something the player has done, and at other times they occur automatically when the core mechanics have determined that they should. How and why the changes occur are determined by the game's ***structure.*** The structure is made up of *gameplay modes,* a vitally important concept in our approach to game design, and *shell menus.* In this section, we define what gameplay modes and shell menus are and discuss how they interact to form the structure.

### Gameplay Modes

If a game is to be coherent, the challenges and actions available to the player at any given time should be conceptually related to one another. In hand-to-hand combat, for example, the player should be able to move around, wield his

weapons, quaff a healing potion (though that may entail some risk), and perhaps run away or surrender. He should not be able to pull out a map or sit down to inventory his assets, even if those are actions he may take at other times in the game. Likewise, a race car driver should not be able to adjust the suspension of the car while driving it or drive the car while it's in the shop.

In short, unless a game is very simple, not all the challenges and actions that it offers will be available to the player at any one time. The player will only experience a subset of all the gameplay, usually derived from the real-world activity (fighting, driving, constructing, and so on) that the game is simulating at that moment. The user interface, too, must be designed to facilitate whatever activity is taking place. The graphics displayed for driving a racing car are necessarily different from those used for tuning it up in the shop. The perspective and interaction models will be different as well. When driving, the vehicle is the player's avatar on the racetrack, and the player usually sees the world from the cockpit; when tuning up the car, the player has omnipresent control over all of its parts, but the rest of the game world (the racetrack) is not accessible.

This combination of related items—available gameplay and supporting user interface at a given point in the game—collectively describe something that we call a ***gameplay mode.*** See Figure 2.3 for an illustration. In a given gameplay mode, the features of the game combine to give the player a certain experience that feels different from other parts of the game; that is, other gameplay modes. Because the game offers only a subset of all its challenges and actions in a given gameplay mode, the player is focused on a limited number of goals.

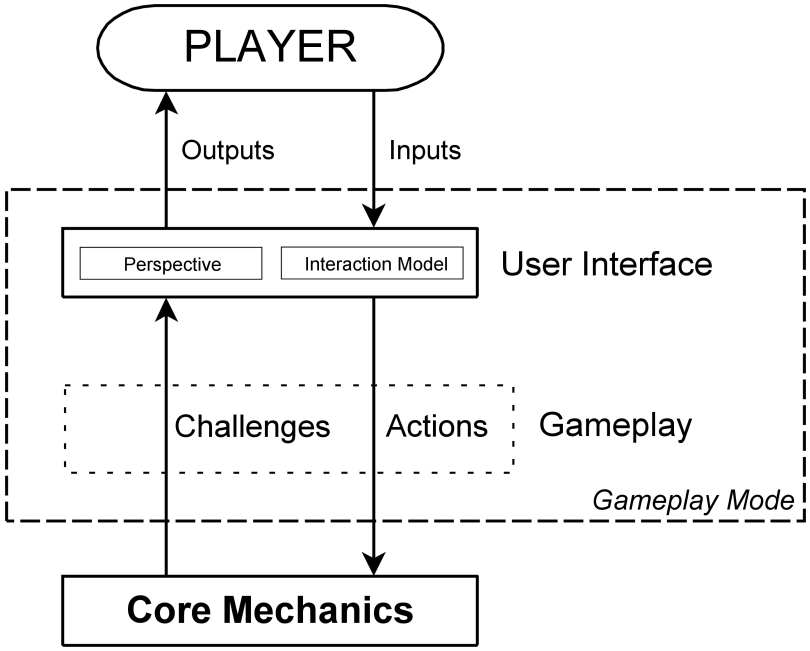


**FIGURE 2.3** The large dashed box represents the gameplay mode.

Gameplay modes are central to our process for designing video games. We therefore provide the following formal definition:

> *A gameplay mode consists of the particular subset of a game's total gameplay that is available at any one time in the game,* plus *the user interface that presents that subset of the gameplay to the player.*

A game can be in only one gameplay mode at a time. When the gameplay available to the player *or* the user interface changes significantly, the game has left one mode and entered another. We include a change to the user interface as a change of mode because such changes redirect the player's focus of attention and cause him to start thinking about different challenges. Also, if the mapping between the buttons on the input device and the actions in the game changes sharply, the player will probably think of it as a new mode.

## **IN THE TRENCHES:** Gameplay Modes in American Football

Video games about American football have many rapid and complex mode switches, especially when playing the team on offense; that is, the team that has the ball. The mapping between the buttons on the controller and the actions they produce in the game changes on a second-by-second basis. Here is the sequence necessary to select and execute a pass play in *Madden NFL Football:*

1. Choose the offensive formation to be used on the next play from a menu.
2. Choose the play to be called from another menu.
3. Take control of the quarterback. Call signals at the line of scrimmage. During this period only one man, who is *not* the quarterback, may move under the player's control. Snap the ball to the quarterback.
4. Drop back from the line of scrimmage and look for an open receiver. Choose one and press the appropriate button to pass the ball to him.
5. Take control of the chosen receiver and run to the place where the ball will come down. Press the appropriate button to try to catch the ball.
6. Run toward the goal line. At this point the ball may not be thrown again.

This process requires six different gameplay modes in the space of about 45 seconds.

Many of the earliest arcade games had only one gameplay mode. In *Asteroids,* for example, you flew a spaceship around a field of asteroids, trying to avoid being hit by one and shooting at them to break them up and disintegrate them. The perspective and the interaction model never changed, nor did the function of the controls. When you had destroyed an entire screenful of asteroids, you got a new screenful that moved somewhat faster, but that was all. From time to time an enemy spaceship appeared and shot at you, presenting new challenges (to avoid being shot, and to shoot the enemy), but because nothing else changed, it wasn't really a new gameplay mode. On the other hand, in *Pac-Man* you were chased by dangerous ghosts until you ate a large dot on the playfield. For a short period after that, the ghosts were vulnerable and would run away from you. Because this represents a significant change to the gameplay (and was a key part of the game's strategy), it can be considered a new gameplay mode even though the user interface does not change. As the designer, it's up to you to decide when the gameplay or the user interface has changed enough to be a new gameplay mode.

Figures 2.4 and 2.5 are screen shots from *Dungeon Keeper 2* illustrating two very different modes. The first, a management mode, shows an aerial perspective of an underground dungeon. It is used for building the dungeon, looking after the creatures who live there, and other strategic activities. The second, called possession mode in the game, is a first-person mode from the

**FIGURE 2.4** Management mode in *Dungeon Keeper 2*.

**FIGURE 2.5** Possession mode in *Dungeon Keeper 2.*

point of view of one of the creatures. You cannot manage the dungeon from this mode; you can only fight enemies such as the ones visible in the picture. Possession mode is essentially tactical.

Not all gameplay modes offer challenges that the player must meet immediately. A strategy menu in a sports game is a gameplay mode because the player must choose the best strategy to help her win the game even though play is temporarily suspended while the player uses the menu. A character creation screen in a role-playing game or an inventory management screen in an adventure game both qualify as gameplay modes. A player's actions there have an influence on the challenges she will face when she returns to regular play.

**KEY POINT**

If a player can take an action that influences the core mechanics—even if that influence is deferred—the game is in a gameplay mode. If he cannot, the game is in a shell menu or shell screen.

## Shell Menus and Screens

Whenever the player is taking actions that influence the game world, that is, actually playing the game, then the game is in a gameplay mode. However, most games also have several other modes in which the player *cannot* affect the game world, but can make other changes. These modes are collectively called ***shell menus*** because they are usually experienced before and after playing the game itself (they are a "shell" around the game, outside the magic circle). Examples of the kinds of activities available in a shell menu include loading and saving the game, setting the audio volume and screen resolution, and reconfiguring the

input devices for the player's convenience. A pause menu in a game is also a shell menu unless it lets the player take some action that affects the game world (such as making strategic adjustments in a sports game), in which case it is a gameplay mode. Noninteractive sequences such as cut-scenes and title screens are called shell screens.

### Forming the Structure

**KEY POINT**

A video game is always in either a gameplay mode or a shell menu or screen. The gameplay modes and shell screens, and the relationships among them, collectively make up the game structure.

The gameplay modes and shell menus of the game collectively make up the structure of the game. To document the structure, you can begin by making a list of all the modes and menus in the game. You must also include a description of *when* and *why* the game switches from one mode or menu to another: what event, or menu selection, causes it to change. Each mode or menu description should include a list of other modes and menus it can switch to and, for each possible switch, a notation about what causes it.

You can document the relationships among all the modes and menus by simply listing them all in a text editor. However, the result isn't easy to follow. We recommend that you document the structure of a game with a *flowboard,* a type of diagram described in the section *Flowboard,* later in this chapter.

Normally, a game moves among its shell menus in response to player actions and nothing else, although arcade games often display an ***attract loop*** that repeatedly shows a title screen, a short noninteractive video of a game in progress, and a high score table. During actual play, a game changes from one gameplay mode to another in response to player actions, or automatically as the circumstances of the game require. For example, in a soccer game, certain violations of the rules result in a penalty kick, in which a single athlete on one team tries to kick the ball past the opposing team's goalie and into the goal and the other athletes on both teams play no role. This is clearly a gameplay mode different from normal play. It is entered not by a specific player choice but by the occurrence of a rule violation.

## Stages of the Design Process

Now that you have learned about the player-centric approach to game design and the key components and structure of a video game, you are ready to start thinking about how to go about designing one. Unfortunately, there are so many kinds of video games in the world that it is impossible to define a simple step-by-step process that will produce a single design document all ready for people to turn into content and code. Furthermore, unless a game is very small, it is not possible to create a complete design and then code it up afterwards. That is how

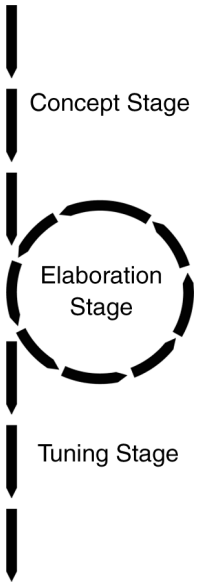


**FIGURE 2.6** Three stages of the design process.

the game industry built games in the 1980s, but experience has shown that large games must be designed and constructed in an iterative process, with repeated playtesting and tuning, and occasional modifications to the design, throughout development. However, not all parts of the design process can be revisited. Some, such as the choice of concept, audience, and genre, should be decided once at the beginning and not changed thereafter. The process is therefore divided into three major parts:

- The ***concept stage,*** which you perform first and whose results do not change
- The ***elaboration stage,*** in which you add most of the design details and refine your decisions through prototyping and playtesting
- The ***tuning stage,*** at which point no new features may be added, but you can make small adjustments to polish the game

We have chosen to use the term *elaboration stage* rather than *development stage* because the latter runs the risk of being confused with *game development.* Figure 2.6 shows the three stages of the design process.

Each of these stages includes a number of design tasks. In the sections that follow, we will look at each stage and the different tasks that you will perform.

Chris Bateman and Richard Boon discuss the relative merits of various design processes in Chapter 1 of their book *21st Century Game Design* (Bateman, and Boon, 2006), and we encourage you to look at it for further discussion on the subject.

**FYI** *A Note on Terminology*

Unfortunately, the game industry has not yet adopted standard names for its design elements, processes, and documents. We have tried to use terms that other professional developers would generally recognize, but we warn you that you cannot expect any given company to use these terms exactly the way we do. If you get confused, please see the Glossary for our definitions of terms.

## The Concept Stage

*Client 2:* *Do I take it that you are proposing to slaughter our tenants?*
*Mr. Wiggin:* *Does that not fit in with your plans?*
*Client 1:* *Not really. We asked for a simple block of flats.*
*Mr. Wiggin:* *Oh. I hadn't fully divined your attitude towards the tenants. You see I mainly design slaughterhouses.*

—*Monty Python's Flying Circus,* "The Architect Sketch"

In the concept stage of game design, you make decisions that you will live with for the life of the project. This stage establishes things about the game that are so fundamental, changing them later would wreak havoc on the development process because a great deal of the work done to implement the game would have to be thrown away. It's like constructing a building: You can revise the color scheme and the lighting design while it's still under construction, but you can't decide that you really wanted an airport instead of a hotel once the foundations are poured.

**FYI** *Concept Versus Preproduction*

Be sure that you don't confuse the concept stage of *design* with the preproduction stage of game *development.* Preproduction is a process borrowed from filmmaking. It's a planning stage of game development during which a developer is deciding what sort of game to make, testing some of those ideas, and figuring out the budget, schedule, and staff requirements. Preproduction ends when the funding agency, usually a publisher, gives the game the green light to proceed to full production. By that time, the concept stage of design is already over and the elaboration stage has begun. In fact, quite a lot of design work gets done during preproduction because that's how the team decides what the game will be. The concept stage of design usually requires only a few weeks; the preproduction stage of development can go on for several months.

**Getting a Concept** All game designs must begin with a game concept; that is, a general idea of how you intend to entertain someone through gameplay and, at a deeper level, *why* you believe it will be a compelling experience. Many different considerations will influence your plans for the game concept. Part of creating a game concept includes deciding what genre your game will fit into, if any. Defining and refining a game concept is described in detail in Chapter 3, "Game Concepts."

**Defining an Audience** Once you know what kind of experience you want to present, you have to think about who would enjoy that experience. In a commercial environment, publishers sometimes define their audience—a "target market"—and *then* think of a concept for a game to sell to them. In any case, the choices you make here will have important consequences for your game because, in player-centric design, you test every design decision against your hypothetical representative player to be sure that the decision helps to entertain your target audience.

**Determining the Player's Role** In an abstract video game, the player doesn't get immersed in a fictional game world and so doesn't have much of a role. He is simply a player playing the game for its own sake. But in a representational game, the player does a lot more pretending. He pretends to believe in the game world, the avatar, and the situations the game puts him in. In such games, the player plays a role, and as a designer it is up to you to define what that role is. It could be an athlete, a general, a dancer, an explorer, a business tycoon, or any of a million other things that people fantasize about doing. Sometimes the roles in a game are multifaceted: In a sports game, the player often changes roles from an athlete on the field; to the coach planning strategy; to the general manager hiring and trading players. You must be able to explain the player's role clearly, because the role a game offers is part of how publishers decide to whether to fund that game, as well as how players decide whether to buy it.

**Fulfilling the Dream** Abstract video games have arbitrary rules, so the player seldom has any preconceptions about what the game will, or won't, allow her to do. Representational video games, however, take place in a world that is at least somewhat familiar, and the player will come to the game with certain expectations and hopes. Representational games are about *fulfilling dreams*—dreams of achievement, of power, of creation, or simply of doing certain things and having certain experiences.

Once you have a game concept, a role, and an audience in mind, it's time to begin thinking about how you will fulfill your player's dream. What is the essence of the experience that you are going to offer? What kinds of challenges will the player expect to face, and what kinds of actions will she expect to perform? Deciding what it means to fulfill the dream is the first step on the road to defining the gameplay itself.

**COMMANDMENT:** Concept Elements Are Permanent

You must not make changes to the concept elements of your game—the game concept, audience, player's role, and dream that it fulfills—once you have started into the elaboration stage of design.

## The Elaboration Stage

Once you have made the fundamental decisions about your game in the concept stage, it's time to move into the elaboration stage of design. At this point, your design work will begin to move from the general to the specific; from the theoretical to the concrete. In the elaboration stage, you normally begin working with a small development team to construct a prototype of the game. If you are planning to incorporate radically new ideas or new technology, your team may also build a testbed or technical demonstration to try them out. From this point on, you may take your design ideas and have the development team implement them in the prototype to see how they work in practice. Based on what you learn, you can then go back and refine them.

At some point during the elaboration stage, your game will (you hope!) get the green light from a funding agency and proceed to full production.

**FYI** *The Danger of Irresolution*

The transition from the concept to the elaboration stage of design is a critical time. At this point, the most important decisions are "set in stone," so to speak; the foundations are poured. Some designers are reluctant to make this transition; they say they're "keeping their options open." They're afraid that they might have made a bad decision or that they might have overlooked something. The consequences of this irresolution are usually disastrous. If the most critical details are still shifting as the game goes into full production, the development team is never entirely sure what it's trying to build. The designer keeps coming around and asking for changes that require huge revisions to the code and content. Production becomes slow and inefficient. It's a sure sign of a lack of vision and confidence. Projects that get into this quagmire are usually cancelled rather than completed.

When you begin the elaboration stage, if you have a team of several people, it becomes possible to begin working on the design tasks in parallel. Once you are all agreed upon the fundamentals of the game, each designer can start work on his own particular area of responsibility.

This process of iterative refinement is not an excuse for introducing major changes into the game late in its development, nor for tweaking it endlessly without ever declaring it finished. Your goal is to build and ship a completed product.

**Defining the Primary Gameplay Mode** The first task after you have locked down your concept is to define the primary gameplay mode of the game, the mode in which the player will spend the majority of his time. Most games have one clearly obvious gameplay mode. In a car racing game, it's driving the car. Tuning the car up in the shop is a secondary mode. In war games, the primary gameplay mode is usually tactical—fighting battles. War games often have a strategic mode as well, in which the player plans battles or chooses areas to conquer on a map, but he generally spends much less time doing that than he does fighting.

At this point it's not necessary to define every detail. The main things to work on are the components that make up the mode: the perspective in which the player will view the game world, the interaction model in which he will influence the game world, the challenges the world will present to him in that mode, and the actions available to him to overcome those challenges. Get those decisions down on paper, and then you can move on to the details of exactly how this is to happen.

**Designing the Protagonist** If your game is to have a single main character who is the protagonist (whether or not the interaction model is avatar-based), it is essential that you design this character early on. You want the player to like and to identify with the protagonist, to care about what happens to her. If the perspective you chose for the primary gameplay mode was anything other than first person, the player is going to spend a lot of time looking at this character, so it's important that she be fun to watch. You must think about how she looks and also about how she behaves: what actions she is capable of, what emotions her face and body language can register, and what kind of language and vocabulary she uses. These issues are discussed in depth in Chapter 6, "Character Development."

**Defining the Game World** The game world is where your game takes place, and defining it can be an enormous task. If the game world is based on the real world (as in a flight simulator, for example), then you can use photographs and maps of real places in order to create its appearance. But if it's a fantasy or science fiction world, you will have to rely on your imagination. And look and feel are only part of the task. There are many dimensions to a game world: physical, temporal, environmental, emotional, and ethical. All these qualities exist to serve and support the gameplay of your game, but they also entertain in their own right. We discuss defining the game world in Chapter 4, "Game Worlds."

**Designing the Core Mechanics** Once you have a sense of the kinds of challenges and actions that you want to include in the primary gameplay mode, you can begin thinking about how the core mechanics will create those challenges and implement the actions. For example, if you plan to challenge the

player to accumulate money, you have to define where the money comes from and what the player has to do to get hold of it. If you challenge the player to play a sport, you must think about all the athletic characteristics—speed, strength, acceleration, accuracy, and so on—that may be required by the sport. If your challenges involve symbolic rather than numeric relationships, as in a puzzle game, you will have to think about what those symbols are and how they are manipulated. Core mechanics are discussed in Chapter 10, "Core Mechanics."

**Creating Additional Modes** As you were deciding upon your game concept, you may have realized that you would need more than one gameplay mode—for example, if you wanted to include separate strategic and tactical modes in a war game, or managing income and expenditures in a business simulation. Or you may have discovered that you will need additional modes while you were defining the primary gameplay mode and core mechanics. Now that you're in the elaboration stage, design the additional modes: their perspective, interaction model, and gameplay. You must also document what causes your game to move from mode to mode—the structure of your game, as described earlier in this chapter.

Do not create additional modes unnecessarily. Every extra mode requires more design work, more artwork, more programming, and more testing. It also complicates your game. Each mode should add to the player's entertainment and serve an important purpose that the game genuinely needs.

**Level Design** Level design is the process of constructing the experience that will be offered directly to the player, using the components provided by the game design: the characters, challenges, actions, game world, core mechanics, and storyline if there is one. These components don't have to be completely finished in order for level design to begin, but there must be enough in place for a level designer to have something to work with. In the early part of the elaboration stage, the level designers will be working to create a typical ***first playable*** level. This level should not be the first one that the *player* will encounter because the first level in the game will be atypical as the player is still learning to play the game. Rather, it's called *first playable* because it's the first one the level designers create.

Creating a working first playable level is an important milestone in the development of a game because it means that testers can begin testing it. We discuss level design in Chapter 12, "General Principles of Level Design."

**Writing the Story** Small video games seldom bother with a story, but large ones usually include a story of some kind. Stories help to keep the player interested and involved. They give her a reason to go on to the next level, to see what happens next. A story may be integrated with the gameplay in a number of different ways. Your story may occur within the levels as the player plays or it may simply be a transition mechanism between the levels—a reward for completing a level. The story may be embedded, with prewritten narrative chunks, or emergent, arising out of the core mechanics. It may be linear and independent of the

player's actions, or it may go in different directions based on the player's choices. We address all these issues in detail in Chapter 7, "Storytelling and Narrative." However you choose to do it, you will be defining the story during the elaboration stage, usually in close conjunction with level design.

**Build, Test, and Iterate** The great game designer Mark Cerny (*Spyro the Dragon, Jak and Daxter*) asserts that during the preproduction process of development, you should build, test, and then throw away no less than four different prototypes of your game. This may be extreme, but the underlying principle is correct. Video games must be prototyped before they can be built for real, and they must be tested at every step along the way. Each new idea must be constructed and tried out, preferably in a quick-and-dirty fashion first, before it is incorporated into the completed product. Cerny also argues that none of the materials you create for prototyping should ever find their way into the final product—or at least, that you should never count on it. By having a firm rule to this effect, you free your programmers and artists to work quickly to build the testbed, secure in the knowledge that they won't have to debug it later. If they're thinking about building maintainable code or final-quality artwork during the preproduction stage of development, the testing process will take far longer than it should.

Once development shifts from preproduction to production, the team is working on material that *will* go out to the customer, and it has to be built with special care. However, you still can't simply design something, hand your design off to the programmers, and forget it. Everything you design must be built, tested, and refined as you go. This is why in modern game development testers are brought in right from the beginning of a project rather than at the end as they used to be.

This is a book about game design, not game development and production, so we won't be discussing the details of managing production here. We recommend that you read *Game Architecture and Design* (Rollings and Morris, 2003) to learn about those processes.

## The Tuning Stage

When you went from the concept stage to the elaboration stage, you locked down the game concept—the foundations of the game. During the elaboration stage, you fleshed out the concept and added new features as necessary. At some point, however, there will come a time at which the *entire* design must be locked—that is, no more features may be added to the game—and you enter the tuning stage. There's no good way to know exactly when this is. It's usually dictated by the schedule. If it would take all the remaining time left to complete and debug the game as the design stands, then clearly you can't add anything more to the design without making the project late! However, this is more of a reactive than a proactive approach to the issue. The design should really be locked at the point at which the designers feel that it is complete and harmonious, even if there is time for more design work.

Once the design has been locked, there's still work for the designers to do. Design work enters the tuning stage, during which you can make small adjustments to the levels and core mechanics of the game as long as you don't introduce any new features. This stage, more than any other, is what makes the difference between a merely good game and a truly great one. Tune and polish your game until it's perfect. Polishing is a subtractive process, not an additive one. You're not putting on new bells and whistles but removing imperfections and making the game shine.

## Game Design Teams

A large video game is almost always designed by a team. Unlike Hollywood, in which guilds and unions define the job roles, the game industry's job titles and responsibilities are not standardized from one company to another. Companies tend to give people titles and tasks in accordance with their abilities and, more important, the needs of a project. However, over the years a few roles have evolved whose responsibilities are largely similar regardless of what game or project they are part of.

- **Lead Designer.** This person oversees the overall design of the game and is responsible for making sure that it is complete and coherent. She is the "keeper of the vision" at the highest and most abstract level. She also evangelizes the game to others both inside and outside the company and is often called upon to serve as a spokesperson for the project. Not all the lead designer's work is creative. As the head of a team, she trades away creativity for authority, and her primary role is to make sure that the design work is getting done and the other team members are doing their jobs properly. A project will have only one lead designer.
- **Game Designer.** The game designer defines and documents how the game actually works: its gameplay and its internal economy. Game designers also conduct background research and assemble data that the game may need. On a large project, these jobs may be split up among several game designers, all reporting to the lead designer.
- **Level Designer.** Level designers take the essential components of the game provided by the game designers—the user interface, core mechanics, and gameplay—and use those components to design and construct the individual levels that the player will play through in the course of a game. Level design used to be considered an inferior position to game designer, but modern level designers frequently need to be able to build 3D models and program in scripting languages. As a result, level design is now a specialized skill, or set of skills, and is considered just as important as game design. A project usually has several level designers reporting either to a lead level designer or to the lead designer.

- **User Interface Designer.** If a project includes this as a separate role, the role may be filled by one or more people responsible for designing the layout of the screen in the various gameplay modes of the game and the function of the input devices. In large, complex games, this can easily be a full-time task. An otherwise brilliant game can be ruined by a bad user interface, so it is a good idea to have a specialist on board. Large developers are increasingly turning to usability experts from other software industries to help them test and refine their interfaces.
- **Writer.** Writers are responsible for creating the instructional or fictional content of the game: introductory material, backstory, dialog, cut-scenes, and so on. (Writers do not, generally speaking, do technical writing—that is the responsibility of the game designers.) Few games require a full-time writer; the work is often subcontracted to a freelancer or done by one of the other designers.

Two other positions have a large amount of creative influence on a game, although they do not normally report to the lead designer. Rather, they are people with whom a game designer can expect to have a lot of interaction over the course of a project.

- **Art Director.** The art director, who may also be called the lead artist, manages production of all the visual assets in the game: models, textures, sprites, animations, user interface elements, and so on. The art director also plays a major role in creating and enforcing the visual style of the game. Within the team hierarchy, the art director is usually at the same level as the lead designer, so it is imperative that the two of them have a good working relationship and similar goals for the project.
- **Audio Director.** Like the art director, the audio director of a game oversees production of all the audible assets in the game: music, ambient sounds, effects, and dialog or narration. Typically there is not as much of this material as there is of artwork, so the audio director may be working on several projects at once. Audio is critical to creating a mood for the game, and the lead designer and audio director work together to establish what kinds of sounds are needed to produce it.

### **COMMANDMENT:** Don't Design by Committee

Do not treat the design work as a democratic process in which each person's opinion has equal value ("design by committee"). One person must have the authority to make final decisions, and the others must acknowledge this person's authority.

## Documenting the Design

As part of their job, game designers produce a series of documents to tell others about their game design. Exactly what documents they produce and what the documents are for vary from designer to designer and project to project—but they usually follow a common thread.

### Why Do We Need Documents?

Beginning programmers, especially those who want to get into the game industry, often make the mistake of thinking up a game and then diving in and starting to program it right away. In modern commercial game development, however, this kind of ad hoc approach is disastrous. Different projects require different degrees of formality, but all serious game companies now insist on having some kind of written documentation as design work progresses.

As we said before, a key part of game design is transmitting the design to other members of the team. In practice, a lot of that communication takes place not through the documents themselves but during team meetings and conversations over lunch. That doesn't mean that there's no point in writing design documents, however. The documents record decisions made and agreed upon orally; they create a paper trail. More important, the process of writing a document turns a vague idea into an explicit plan. Even if no one reads it at all, an idea written down is a decision made, a conclusion reached. If a feature of a game is not described in writing, there's a good chance that it has been overlooked and that someone will have to make it up on the fly—or, worse, that each part of the team will have a different idea of what they intended to do. It's far easier and cheaper to correct a design error before any code is written or artwork is created. Depending on the size of the game, wise developers will allot anywhere from one to six months for pure design work before starting on development, usually in combination with some throwaway prototype for testing gameplay ideas.

#### IN THE TRENCHES: Game Idea Versus Design Decision

Here's a game idea: "Dragons should protect their eggs."

Here's a design decision: "Whenever they have eggs in their nests, female dragons will not move beyond visual range from the nest. If an enemy approaches within 50 meters of the nest, the dragon will abandon any other activity and return to the nest to defend the eggs. She will not leave the nest until no enemy has been within the 50 meter radius for at least 30 seconds. She will defend the eggs to her death."

See the difference? This is what creating design documents is about.

## Types of Design Documents

This section is a short introduction to the various types of documents a game designer might be asked to create. This isn't an exhaustive list, nor will every project need all of the items on the list. Rather, these are some of the most commonly used ones.

The high concept and game treatment documents are sales tools, designed to help communicate the game concept to a funding agency such as a publisher. They are usually written in a word processor such as Microsoft Word and distributed in paper form. The other documents used to be written on paper as well, but it is increasingly common in the game industry to create them as pages on a Web site. As long as you can keep the Web site secure, it's a good way of documenting a game design so that all the members of the team can access it, and you can update it easily. You'll need a good site construction tool such as NetObjects Fusion. Another, even easier, alternative is to create a wiki, a tool for letting anybody read and edit hypertext documents. Once the wiki software is installed on the company server, the whole team can edit the content using only a browser. It's important to make sure you have revision control and backups so that you can revert to a previous edition if someone deletes a page by accident.


You can find samples (or pointers to samples) of design documents on the Companion Website.

**High Concept Document** The high concept document is not a document from which to build the game. Just as the purpose of a résumé is to get you a job interview, the purpose of a high concept document is to get you a hearing from someone, a producer or publishing executive. It puts your key ideas down on paper in a bite-size chunk that he can read in a few minutes. Like a résumé, it should be short—not more than two to four pages long.

It's also worthwhile to write high concept documents for yourself, to record ideas that you might want to work on in the future.

**Game Treatment Document** A game treatment presents the game in a broad outline to someone who's already interested in it and wants to hear more about it. Like a high concept document, it's primarily a sales tool. The treatment is designed both to satisfy initial curiosity and to stimulate real enthusiasm for the game. When you give a presentation about your game to a publisher, you should hand him the game treatment at the end so he'll have something to take away and look at, something that will float around his office and remind him of your game. Your goal at this point is to get funding of some sort, either to create a more thorough design or a prototype or (preferably!) to develop the entire game.

The initial treatment is still a simple document—almost a brochure that sums up the basic ideas in the game. A good way of picturing what to write in a treatment is to imagine that you are making a Web site to help sell your game; then throw in some business and development details for good measure.

**Character Design Document** A character design document is specifically intended to record the design of one character who will appear in your game, most often an avatar. Its primary purpose is to show the character's appearance and above all her ***moveset***—a list of animations that documents how she moves, both voluntarily and involuntarily. It should include plenty of concept art of the character in different poses and with different facial expressions. In addition, it should include background information about the character that will help to inform future decisions: her history, values, likes and dislikes, strengths and weaknesses, and so on. Character design is discussed in Chapter 6, "Character Development."

**World Design Document** The world design document is the basis for building all the art and audio that portray your game world. It's not a precise list of everything in the game but, rather, background information about the kinds of things the world will contain. If you have a large landscape or cityscape, for example, the world design document should include a map. You need not supply every detail, just a general overview. This information will be used by the level designers and artists to create the actual content. Be sure also to note the sources of ambient sounds in the world so the audio designers can build them in at the appropriate locations.

The world design document should also document the "feel" of the world, its aesthetic style and emotional tone. If you want to arouse particular emotions through images and music, indicate how you will do so here.

**Flowboard** A flowboard is, as the name suggests, a cross between a flowchart and a storyboard. Storyboards are linear documents used by filmmakers to plan a series of shots; flowcharts are used by programmers (though rarely nowadays) to document an algorithm. A flowboard combines these two ideas to document the structure of a game.

Although you can create a flowboard in an editor such as Visio, it's actually quicker and easier to make one on several sheets of paper and stick them on a large blank wall. Each sheet of paper is used to document one gameplay mode or shell menu. On each page, write the name of the mode clearly at the top. Then, in the center of the page, draw a quick sketch of the screen as it will appear in the mode, showing the perspective (if appropriate) and user interface items that will appear on it. Leave plenty of space around the edges. Off to the sides of the sketch, document the menu items and inputs available to the player and what they do. You can also list the challenges that will arise in that mode, although that's less important—the key thing is to indicate the player actions that will be available. Then draw arrows leading to the other gameplay modes or shell menus and indicate under what circumstances the game makes a transition from the current mode to the next one. By creating one mode per page and putting them up on the wall, you allow everyone in the office to see the structure of the game. You can also easily make revisions by adding new sheets and marking up the existing ones.

**Story and Level Progression Document** This document records the large-scale story of your game, if it has one, and the way the levels will progress from one to the next. If you're making a small game with only one level (such as a board game in computerized form) or a game with no story, you need not create this document. However, if the game will have more than one level, or the player will experience a distinct sense of progress throughout the game, then you will need such a document. You're not trying to record everything that can happen in the game, but rather a general outline of the player's experience from beginning to end. If the game's story will branch based upon the player's actions, this is the place to document it and indicate what decisions will cause the game to take one path rather than another. Here also you indicate *how* the player will experience the story: whether it's told via cut-scenes, mission briefings, dialog, or other narrative elements.

Bear in mind that the story or level progression is not the same as the game's structure. An entire story can take place in only one gameplay mode; likewise, a game can have many different gameplay modes but no story at all. Although the game changes from mode to mode over time, and the story also progresses over time, the two are not necessarily related.

**The Game Script Document** Back when games were smaller, it was common to incorporate all of the preceding documents except for the high concept and treatment into a single massive tome, the game script (or "bible"). As games have gotten larger, the industry has tended to break out the character, world, and story documentation into individual documents to make them more manageable. The game script is still used, however, to document a key area not covered by the others: the rules and core mechanics of the game.

As a good rule of thumb, the game script should enable you to play the game. That is, it should specify the rules of play in enough detail that you could, in theory, play the game without the use of computer—maybe as a (complicated) board game or table-top role-playing game. This doesn't mean you should actually sit down and play it as such, but it should theoretically be possible to do so, based solely on the game script document. Sitting down and playing paper versions of game ideas is a very inexpensive way of getting valuable feedback on your game design. For designers without huge teams and equally huge budgets, we heartily encourage paper-play testing.

The game script does not include the technical design, though it may include the target machine and minimum technical specifications required. However, it does not address how the game is built or implemented in software. The technical design document, if there is one, is usually based on the game script and is written by the lead programmer or technical director for the game. Technical design is beyond the scope of this book. If you want to know more about technical design, read *Game Architecture and Design* (Rollings and Morris, 2003).

## Anatomy of a Game Designer

Like all crafts, game design requires both talent and skill. Talent is innate, but skill is learned. Effective game designers require a wide base of skills. The following sections discuss some of the skills that are most useful for the professional game designer. Don't be discouraged if you don't possess all of them. It's a wish list—the characteristics we would like to see in a hypothetical "ideal designer."

### Imagination

A game exists in an artificial universe, a make-believe place governed by make-believe rules. Imagination is essential to creating this place. It comes in various forms:

- Visual and auditory imagination enables you to think of new buildings, trees, animals, creatures, clothing, and people—how they look and sound.
- Dramatic imagination is required for the development of good characters, plots, scenes, motivations, emotions, climaxes, and conclusions.
- Conceptual imagination is about relationships between ideas, their interactions and dependencies.
- Lateral thinking is the process of looking for alternative answers, taking an unexpected route to solve a problem.
- Deduction is the process of reasoning from a creative decision you've made to its possible consequences. Deduction isn't ordinarily thought of as imagination, but the conclusions you arrive at will produce new material for your game.

**FYI** *Imagination Includes Adaptation*

Imagination does not consist only of the ability to invent new things. It's also valuable to be able to look at an old idea and breathe new life into it with a fresh approach. J.K. Rowling does this brilliantly in her *Harry Potter* novels. She still has witches flying on broomsticks, but she invented the sport of Quidditch, which is played while flying on them.

## Technical Awareness

Technical awareness is a general understanding of how computer programs, particularly games, actually work. You don't have to be a software engineer, but it is extremely valuable to have had a little programming experience. Level designers, in particular, often need to be able to program in simple scripting languages. Get to know the technical capabilities of your target platform. You must also be aware of what your machine cannot do (for example, understand ordinary speech) so that you won't create unworkable designs.

## Analytical Competence

Analytical competence is the ability to study and dissect something: an idea, a problem, or an entire game design. No design is perfect from the start; game design is a process of iterative refinement. Consequently, you must be able to recognize the good and bad parts of a design for what they are.

One example of an analytical task is detecting dominant (that is, unbeatable or nearly unbeatable) strategies at the design phase and weeding them out before they get into the code, as in the infamous *Red Alert* "tank rush." In *Command & Conquer: Red Alert,* tanks on the Soviet side were so much more effective than any other unit that an experienced player could dedicate all production to cranking out a few tanks and then immediately storm the opposition base before the enemy had a chance to get a production line set up.

## Mathematical Competence

Designers must have basic math skills, including trigonometry and the simpler principles of probability. Balancing games that feature complex internal economies, such as business simulations or real-time strategy games, can require you to spend a lot of time looking at numbers. You don't need a PhD in mathematics, but you should be comfortable with the subject. You may be able to handle most of the requirements with a spreadsheet program such as Microsoft Excel.

## Aesthetic Competence

Although you need not be an artist, you should have a general aesthetic competence and some sense of style. Far too many games are visual clones of one another, depending on stereotypes and clichés rather than real imagination. It's up to you (along with your lead artist) to set the visual tone of the game and to create a consistent, harmonious look.

We encourage you to expand your aesthetic horizons as much as you can. Learn a little about the fundamentals of art: the principles of composition, and which colors coordinate and which clash. Find out about famous art movements—Art Nouveau, Surrealism, Impressionism—and how they changed the

way we see things. Watch movies that are famous for their visual style, such as *Metropolis*. Then move on to the more practical arts: architecture, interior decoration, industrial design. The more aesthetic experience you have, the more likely you are to produce an artistically innovative product.

## General Knowledge and the Ability to Research

The most imaginative game designers are those who have been broadly educated and are interested in a wide variety of things. It helps to be well versed in such topics as history, literature, art, science, and political affairs. More important, you must know how to research the subject of your game. It's tempting just to use a search engine on the Internet, but that's not very efficient because the information it presents will be haphazard and disorganized and might not be reliable. The encyclopedia is a better place to start for any given subject. From there, you can increase your knowledge of a particular area by moving on to more specialized books or TV documentaries.

## Writing Skills

A professional game designer actually spends most of his time writing, so a designer *must* have good writing skills. This means being clear, concise, accurate, and unambiguous. Apart from having to write several detailed documents for each design, you might be expected to produce the story narrative or dialog—especially if the budget won't stretch to a scriptwriter.

Design writing comes in several forms:

- Technical writing is the process of documenting the design in preparation for development. The essential mechanisms of the game have to be answered unambiguously and precisely.
- Fiction writing (narrative) creates the story of the game as a whole—a critical part of the design process if the game has a strong storyline. Some of this material may appear in the finished product as text or voiceover narration. The game's manual, if there is one, often includes fictional material as well.
- Dialog writing (drama) is needed for audio voiceovers and cinematic material. Dialog conveys character, and it also can form part of the plot. A class in playwriting or screenwriting will teach you a lot about writing dialog.

A designer must be able to convey the details of the design to the rest of the team, create the textual and spoken material that will appear in the game, and help sell the idea to a publisher. Good writing skills are essential to accomplish these things successfully.

### Drawing Skills

Some skill at basic drawing and sketching is highly valuable, although not absolutely required for a designer if you have a concept artist to work with. The vast majority of computer games rely heavily on visual content, and drawings are essential when pitching a product to a third party. Game-publishing executives will be interested in a hot concept, a hot market, or a hot license, but only pictures really excite them. The images will remain in their memories long after they forget the details.

### The Ability to Compromise

A professional game designer must be able to compromise on details and integrate a variety of opinions while preserving a consistent, holistic vision of the game. Different people on the development team and at the publishing company will have concerns about their own areas of expertise (programming, art, music, and so on), and their opinions will pull and push the design in different directions. As the designer, you may be tempted to seek sole ownership of the vision and insist that things must be exactly as you imagined them. You must resist the temptation to do that, for two reasons:

- First, you must allow your team some ownership of the vision as well, or its members won't have any motivation or enthusiasm for the project. No one builds computer games solely for the money; we're all here so that we can contribute creatively.
- Second, a designer who can't deliver in a team environment, no matter how visionary she may be, doesn't stay employed for long. You must be able to work successfully with other people.

Compromise means more than just negotiating with other people, however; it also means working within the prevailing circumstances. In many cases, you'll be given a task that limits you to designing a genre clone or a heavily restricted licensed property. On a commercial project, you will almost certainly be told, rather than get to choose, the target hardware upon which your game will run. Your project will always have a desired budget and schedule that it is expected to meet. A professional designer must be able to work within these constraints and to make the compromises necessary to do so.

## Summary

This chapter puts forward the view that game design is not an arcane art but rather a craft, just like any other, that can be learned with application.

Video games are not created by a mysterious, hit-or-miss process. Instead, they are recreational experiences that the designer provides to the players through

rules and a presentation layer. A game is designed by creating a concept and identifying an audience in the concept stage, fleshing out the details and turning abstract ideas into concrete plans in the elaboration stage, and adjusting the fine points in the tuning stage. All video games have a structure, made up of gameplay modes and shell menus, that you must document so your teams know what they are building and how it fits together. In the course of this process, you will use a wide variety of skills to create a wide variety of documents for your team. And at all times, you should seek to create an integrated, coherent experience for your player that meets your most important obligation: to entertain her.

# Test Your Skills

## MULTIPLE CHOICE QUESTIONS

1. Which of the following is *not* an essential skill for a game designer?
   A. Technical writing.
   B. Research.
   C. Programming.
   D. Fiction writing.

2. Game design is which of the following?
   A. An art.
   B. A form of engineering.
   C. A science.
   D. A craft.

3. Which are the two duties of a designer using the player-centric approach?
   A. To empathize and to entertain.
   B. To impress and to make money.
   C. To be imaginative and to be efficient.
   D. To be accurate and under budget.

4. A gameplay mode is
   A. any kind of menu that appears in a game.
   B. the subset of a game's gameplay that is available at one time, plus the user interface that presents it.
   C. the state of the core mechanics at any given point in the game.
   D. a document for defining the gameplay.

5. A flowboard is a means of documenting a game's
   A. programming.
   B. artwork.
   C. story.
   D. structure.

6. Which of the following is *not* an interaction model?
   A. Avatar-based.
   B. Multipresence.
   C. Third-person.
   D. Contestant.

7. Which of the following design tasks belongs in the concept stage?
   A. Level design.
   B. Designing the core mechanics.
   C. Designing the game world.
   D. Identifying an audience.

8. Which of the following must you *not* do after leaving the concept stage?
   A. Redefine the game concept.
   B. Modify the core mechanics.
   C. Change the game's story.
   D. Add any additional gameplay modes.

9. In the tuning stage, the designer's function is restricted to
   A. testing the game.
   B. refining details without adding features.
   C. adding only minor gameplay modes.
   D. writing and recording dialog.

10. On a large design team, the lead designer's job is primarily to
    A. work with the marketing team to devise the best way to sell the game.
    B. create and document the core mechanics.
    C. oversee the work of the other designers.
    D. negotiate with the publisher.

11. Which of the following design documents is primarily a sales tool rather than a design tool?
    A. The game script.
    B. The flowboard.
    C. The game treatment.
    D. The world design.
12. Which document is one that a game designer does not create?
    A. The high concept.
    B. The technical design.
    C. The story and level progression.
    D. The flowboard.

## EXERCISES


In these exercises, you will document existing games for practice. Your instructor may require that you do so with certain particular games that he is familiar with and will set the expected scope of the work—the amount of material he wants to see. Use a document template from the Companion Website or one supplied by your instructor.

1. Document the primary gameplay mode of a reasonably simple game that you like. Be sure to include a sketch of the screen, including perspective and user interface; a list of all the buttons and menu items available in that mode; and a list of the other modes that this mode can switch to. Describe the challenges and actions that make up the mode.
2. Take a classic, well-known arcade game with a small number of gameplay modes and shell menus, and create a flowboard for it. (As you cannot hand in an entire wall of pages, create a miniature version with two or three gameplay modes per sheet.)
3. Document the level progression of a well-known game that you have played all the way through, such as *StarCraft* or *Doom 3.* (If the game does not have explicit levels, as in *Half-Life,* document major areas or sections of the game.) Describe the game world in each level and the types of challenges that it presents. If the levels are integrated into a story, explain how each level supports or relates to the story.
4. Research one of the following art or design styles from history and write a short paper to explain how it might help to establish an emotional tone in a video game: Impressionism, Constructivism, Symbolism, Pop Art, Art Deco, Art Nouveau. Back up your argument with examples from famous paintings or other works in that style.

## DISCUSSION QUESTIONS

1. What are the strengths and weaknesses of player-centric game design? In what ways might it conflict with other requirements imposed on the game or the desires of the game designer?
2. Do you feel that our list of qualities in an ideal game designer is complete? What other qualities might you add? Are there any that you would remove?
3. Our list of members of a design team does not include a lead programmer. Should it? Suggest some arguments for and against.