Chapter 21

Online Games

Chapter Objectives

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

- Understand the advantages and disadvantages of online play as compared with single-player play.
- Understand key design issues for online games, including handling arriving and disappearing players, real-time and turn-based play, chat mechanisms, and designing to prevent player collusion.
- Be familiar with some of the technical security problems of online games and some solutions.
- Know how persistent worlds differ from conventional games and what this implies for storytelling, avatar creation and death, and the internal economics of the game world.
- Be familiar with the issues surrounding player-versus-player combat in online games.

Introduction

Over the last ten years, online gaming has grown from a tiny fraction of the interactive entertainment business into a major market in its own right. Online gaming has many features and design challenges that set it apart from the more traditional single-player or multiplayer local games, and we examine some of them in this chapter. Online gaming is a technology rather than a genre; a mechanism for connecting players together rather than a particular pattern of gameplay. Therefore, we don't look for design commonalities in this chapter as we did in the chapters on game genres. Instead, we address some of the design considerations peculiar to online games no matter what genre those games belong to. It's a huge topic, however, and we can cover only the highlights here.

We devote the second half of the chapter to online games that are *persistent worlds,* also known as massively multiplayer online role-playing games, or MMORPGs (although persistent worlds need not necessarily be about role playing). We're particularly indebted to Raph Koster, lead designer of both *Ultima Online* and *Star Wars Galaxies,* and Tess Snider, a long-time online role-player and member of the WorldForge Project, for their help with this material. In his many years of working with online games, Koster has assembled a valuable collection of laws and observations about this particular form of interactive entertainment, some his own and some credited to his colleagues. With his permission, we reprint many of them here. You'll see them scattered throughout the chapter like this:

> Ola's Law about Laws: *Any general law about virtual worlds should be read as a challenge rather than as a guideline. You'll learn more from attacking it than from accepting it.*

You can read the complete set of laws at Koster's Web site at **www.raphkoster.com.**

What Are Online Games?

For the purposes of this chapter, we use the term *online games* to refer to multiplayer distributed games, in which the players' machines are connected by a network. (This is as opposed to multiplayer local games, in which all the players play on one machine and look at the same screen). While *online games* can, in principle, include solitaire games that happen to be provided via the Internet, such as *Bejeweled,* the online aspect of solitaire games is incidental rather than essential to the experience. We could simply classify *Bejeweled* as a puzzle game. Online games need not be distributed over the Internet; we consider games played over a local area network (LAN) to be identical, from a design standpoint, to those played over the Internet.

Advantages of Online Games

> J.C. Lawrence on utopias: *Don't strive for perfection, strive for expressive fertility. You can't create utopia, and if you did nobody would want to live there.*

Some features of online games offer advantages to us as game developers, and some offer advantages to players, attracting people who might not otherwise play computer games.

Player Socializing

Online games offer opportunities for social interaction. Players can't talk among themselves as well as they could if they were in the same room, and they can't give each other high fives, but online games still offer more social stimulation than most PC or console games. The social aspect enhances the players' enjoyment of the experience. Girls and women have traditionally shown less interest in interactive entertainment, especially games for personal computers, in part because these tend to offer solitary activity. Women represent a much greater proportion of the online game market than they do the single-player game market, partially because they enjoy interacting with others.

At the moment, most games offer only limited social interaction with conversation restricted to typing text (*chatting*), which is awkward while trying to play a fast-paced game, but more and more games include voice communication. When enough people get broadband access, online games could include video as well. A time might come when we see players dressing appropriately for their roles in the game so that they'll look cool on camera.

The social element gives online games something of the atmosphere of clubs, cafés, or casinos—places where people get together for fun. As the creator of such spaces, you're more than just a game designer; you must also be a social architect. This is actually your toughest challenge, far more difficult than designing the core mechanics of a single-player game. An online game isn't an experience that you lead a player through; it's a Petri dish for growing social situations, and it's nearly impossible to predict in advance what will happen there. For further discussion of this topic, please read the excellent and insightful *Community Building on the Web* by Amy Jo Kim (Kim, 2000).

Human Intelligence Instead of Artificial Intelligence

In single-player games, the player competes against the computer, so the computer has to have enough artificial intelligence to be a good opponent; building the AI for a complex game presents a huge programming task and one that is difficult to get right. But if the players compete against each other, as they do in most online games, you don't usually need as much AI. The players provide all the intelligence required in many situations.

You can use AI in an online game if you want to: You might include nonplayer characters (NPCs) who need to behave intelligently, or you might design a game in which all the players play cooperatively against artificial opponents. Several popular games have limited NPCs but have some large opponents that the online players must work together to combat. *Guildwars,* for example, encourages this type of play. The AI-controlled enemies are challenging to beat with a team of online friends and impossible for an individual. But many online games rely on their players to provide most of the intelligence in the game, and this can make the game easier to develop in that respect. A real-time strategy game, for example, still needs AI for its

individual units when played online, but the strategic and tactical thinking is supplied by the players.

Online Gameplay Versus Local Multiplayer Gameplay

Multiplayer gameplay, whether online or local, offers great flexibility to the game designer, allowing purely competitive ("everyone for himself"), purely cooperative ("us against the machine"), or team-based ("us versus them") play. In online play, a network links the players, who occupy (generally, but not necessarily) separate locations. Local play can be broken into two categories: local area network (LAN) play or physically local play. LAN play is virtually identical to online play except that Internet access is not required—a technical distinction that has little or no impact on game design. In physically local play (hereafter referred to simply as local play), all the players sit in the same room, playing the game on the same machine and, most important, looking at the same screen.

For the last 30 years, local play has been the standard mode of interaction for multiplayer console games: Each player holds a controller, and all players look at the TV. That could change as the new generation of consoles introduces network capability, but it's likely to remain the most common way people use multiplayer games because such play incurs no network charges and lets friends play together in groups. Computers offer a similar mode of play, but with the drawbacks of smaller monitor sizes, getting additional controllers to work, or playing on the same keyboard, most people currently play with a console machine and a TV and from the comfort of a couch.

Problems with Local Play Local play as described above presents the game designer with serious difficulties. For one thing, because all the players share the same TV, any user interface elements displayed must be duplicated for each player, taking up valuable screen space. If the game maintains a separate point of view for each player, you must subdivide the screen into little windows. Each player will find it harder to see a small individual window than the full screen image, and activity in the other players' windows will distract him.

More important, however, local play using a single display device leaves you with no way to hide information. Each player can see everything the others do. This works well for fighting games, but not as well for any game in which players might want to keep their activities secret—war games, for instance.

(The Nintendo GameCube does allow players to plug in a Game Boy Advance and use it as a controller; the designer may take advantage of the Game Boy's screen to display secret information only to the player who should see it. However, this feature requires players to own both devices as well as the necessary cable. Although this hardware solves the problem in theory, in practice any commercial game that requires that hardware setup as a gameplay mechanism limits its market to those players who actually own all the gear.)

Finally, local play necessarily imposes limits on the number of people who can participate at one time. Consoles seldom support more than four players; PCs support even fewer. Even if you could add players indefinitely, the screen would become crowded with characters and other data, and the machine itself would bog down as the computing tasks grew.

Benefits of Networked Play Online gaming solves all these problems. Each player uses her own screen, and the entire display supports only her gaming experience. The game can present her with her own unique perspective, including exactly as much information as the designer wants her to have and no more. And online games can support large numbers of people (although games requiring a central server may find the server capacity limiting); it's not uncommon for some games to support tens of thousands of players online at a time. With an online game, players can always find other people to play with at any hour of the day or night.

Disadvantages of Online Games

Playing games over a network, especially the Internet, presents the designer with certain disadvantages, as well. This section discusses some of these technical challenges, as well as the ongoing responsibilities of providing new content and customer service. You should also be aware that strangers playing your game anonymously over a network can cause social friction and that this can range from minor misbehavior to serious criminal offenses.

Technical Issues

Because this is a book about design, we won't go into the technical issues in detail, but it's worth listing a few such issues just to let you know what you're up against. These technical problems probably won't affect your design work; they're just something to be aware of.

> Murphy's Law: *Servers crash—and don't restart—only when you go out of town.*

Communication Models Your programming team will need to choose a communication model from the two currently in use in networked gaming. In the first, *client/server,* each player runs a program, called the *client,* on his computer, that communicates with a central program, the *server,* on a computer owned by a company providing the game service. In the client/server model, the server runs the game engine, sending packets of information to the various clients, and the clients merely present that information to the players. The other model, *peer-to-peer,* involves direct communication between the players' computers.

Implementation of peer-to-peer (sometimes abbreviated P2P) communication is quite straightforward for two-player games but becomes more complicated as more players are involved. The players' systems must decide which machine to designate as the *host*—that is, which will control the game while the others become *guests.* If the host logs out of the network, one of the guests' computers must take over and become the new host—preferably automatically and without anyone's noticing (known as *automated host migration,* a feature already supplied by Microsoft's DirectPlay facilities). Some companies also operate *matchmaking* services in which the company's server functions only to allow players to find one another and connect together in peer-to-peer networks. All of this is programming work that offline games don't have to bother with.

Transmission Delay Times The Internet, being designed for redundancy rather than speed, doesn't make any guarantees about how long a given packet of data will take to get from one point to another. In many games, a faster connection translates into a gaming advantage, making players with high-speed connections more likely to win the game. You can design around this by making your game turn-based or trying to match up opponents on the basis of their connection speeds. At the moment, there is no one satisfactory answer.

Dropped and Garbled Packets What happens to your game if it doesn't get some of the information it needs because of a glitch in the network? Your system will require a mechanism for detecting a missing packet or one containing bad data, and requesting that the packet be resent from the server. Packets can also arrive out of order, which can cause confusion if your client receives information that a racecar is about to cross the finish line, but the next packet indicates that the car is 100 yards back on the track instead. Every packet must have a unique serial number, in sequence, so that you can tell if one is missing or if packets are arriving in the wrong order. Fortunately, middleware companies are starting to offer software packages to help manage this problem.

It's Harder to Suspend Disbelief

For some players, gaming is a form of escapism that takes them away to a magical place, and they want it to stay magical while they're there. To them, it's particularly important that nothing occur in the game to break their suspension of disbelief, but in online games there will always be players who won't stay in character or who will talk about real-world issues and events while they're in the game. Unless there's a strong (and enforced) ethos of in-character role-playing, people who play in an online game have to accept that their imaginary world includes a lot of entirely real people.

> Koster's Law: *The quality of role-playing is inversely proportional to the number of people playing.*

> Enforcing role-playing: *A role-play–mandated world is essentially going to have to be a Fascist state. Whether or not this accords with your goals in creating such a world is a decision you will have to make.*

The persistent world *World of Warcraft* offers players multiple versions of its game world with different versions imposing different rules on the player. In some versions, in-character role-playing is mandatory.

Misbehavior

Unfortunately, playing with strangers—particularly anonymous strangers—creates opportunities for a variety of types of misbehavior that can ruin the game for others. These range from simple rudeness to harassment, cheating in various forms, and outright fraud. Rudeness might not sound very serious, but it drives away other customers. (Raph Koster said, only partly joking, that the sole rule in *Star Wars Galaxies* is, "Any behavior that hurts business is bad behavior.") Furthermore, if you want children to play your game, it is particularly important to make sure you offer a safe environment—you may even have a legal obligation to make sure adults don't use your game environment to abuse children—and that means hiring customer service people to monitor the players. Self-contained networks such as America Online have some tools at their disposal to manage these problems, but on open networks such as the Internet, it's much harder. We address some of these issues in the *Design Issues for Online Gaming* section, below.

> John Hanke's Law: *In every aggregation of people online, there is an irreducible proportion of [jerks].*

The Need to Produce Content

When you're building a single-player game to be sold in retail stores, your job generally finishes when the gold master disc goes off to manufacturing. The players buy the game and you can go off to work on another project.

Online games don't work this way; they earn money either through advertising revenue or subscriptions or both. To keep people interested, you have to change things, and that means producing new content on an ongoing basis. This is expensive for the service provider and ties up skilled development staff. The problem is most obvious with persistent worlds, but even simple games need to be kept fresh. America Online used to run a suite of games called *RabbitJack's Casino* (casino-style games in which no real money changed hands). Although the graphics didn't change, the game's managers held theme nights and gave away additional prizes when rare events occurred, such as a player getting four aces in poker.

Persistent means it never goes away: *When you open your online world, expect to keep your team on it indefinitely. Some of these games have never closed. And closing one prematurely could cost you the faith of your customers, damaging the prospects for other games in the same genre.*

Customer Service

All game companies require customer service staff to help players with problems, but online games need far, far more of them. With offline games, players mostly need help with technical difficulties; for gameplay problems, they can buy strategy guides or find hints on the Internet. But in a live, online environment, players expect to get help immediately, and they demand help for a much larger range of issues than they do in offline games. Players expect customer service people not only to solve technical problems but also to explain the user interface, answer questions about game content, and enforce justice by investigating and punishing misbehavior by other players. With thousands of players logged on at any one time, providing these services can become very expensive.

Anonymity and in-game administrators: *The in-game administrator faces a bizarre problem. He is exercising power that the ordinary virtual citizen cannot, and he is looked to in many ways to provide a certain atmosphere and level of civility in the environment. Yet the fact remains that no matter how scrupulously honest he is, no matter how just he shows himself to be, and no matter how committed to the welfare of the virtual space he might prove himself, people will hate his guts. They will mistrust him precisely because he has power and they can never know him.*

Design Issues for Online Gaming

In this section, we address some design issues peculiar to online games: the problems presented by players arriving or disappearing during play, the pros and cons of real-time versus turn-based play, things to consider when providing a chat feature, and a variety of issues regarding security and the prevention of cheating.

Arriving Players

Players can log on wanting to play your game at any time, and the game must be capable of dealing with them intelligently. In most noncomputer games, all the players must be present at the beginning of the match or it won't be

fair. In *Monopoly,* for example, anyone who entered the game late would be at a significant disadvantage—the others would have already grabbed the best properties, and the game's built-in inflation would swiftly bankrupt newcomers.

The usual solution is to start new matches at frequent intervals and to have a waiting area, or *lounge,* where the players can hang around while they wait for a new match to begin. In a game that can be played with any number of players, such as bingo, you can simply start a new match, say, every three minutes, and whoever is waiting may play. In games requiring a fixed number of players, such as bridge, you will need to establish a matchmaking service that allows them to form groups and to wait (more or less patiently) for enough players to join a particular group; the game begins as soon as the required number of players arrives. The number of players needed for a game should be small, however, to minimize waiting times. Any game that requires more than about eight players risks alienating players who do not want to wait.

In some games, players can join almost immediately without any disadvantage—poker, for instance. Each hand takes little time, and new players can join at the end of the current hand. Tournament play, of course, has a definite start, and players cannot join after the game begins. For games of indefinite duration, such as persistent worlds, nothing can be done about the fact that some players possess advantages other players don't. The players who began the earliest and who devote the most time to play will always have an advantage (unless you allow players to purchase prebuilt characters for real money on eBay, but that just shifts the advantage from players who have the most time to players who have the most money). You can, however, prevent those advantages from spoiling the game for other players:

- *Get rid of the victory condition.* Without winners and losers, an online entertainment ceases to be a game per se and becomes a different kind of amusement. The player focuses on her own achievements rather than on defeating all the other players. In this case, the old cliché becomes apt: It's not whether you win or lose, but how you play the game. Persistent worlds, which we address later in the chapter, work on that basis.
- *Discourage competition between old-timers and newcomers.* You can measure the progress of your players and see to it that only those who are fairly matched come into direct conflict. Tournament chess uses a ranking system to do just that. An old-timer who beats a newcomer gets little or no reward for it. The cliché "pick on someone your own size" describes this solution.
- *Be sure that direct competition is consensual.* If old-timers do get the chance to compete directly with newcomers, you should give the newcomers the option to refuse to play. No one should be forced to take part in an unfair competition.

Disappearing Players

Just as players can appear at any time, they can log off at any time, or technical difficulties can occur that cause them to disappear. If possible, your game should deal with this neatly and with minimal disruption to other players. In many games, such as racing games, players compete against one another in a free-for-all. The disappearance of one player doesn't make that much difference—his car vanishes from the track, and that's that. It's as if he pulled out with mechanical trouble. In effect, the player forfeits the race and the others continue. On the other hand, if the game requires players to work in teams, the disappearance of one player could put his team at a serious disadvantage. In games that require a fixed number of participants, your only options are to give the person a chance to reconnect, assuming the disappearance was a mistake, or to include an AI element that can take over for the missing player or to shut down the game.

Tournaments require special consideration. If players compete to get the best win-loss ratio, one might deliberately choose to log out rather than lose the game—which can deny the other person victory. Should the vanishing player be forced to forfeit? What if the disconnection was an accident, caused by a bad line? Unfortunately, there's no sure way to tell if it was.

You may find that one of the following suggestions solves the problem of vanishing players for your game:

- *The vanishing player forfeits*. This solution may unfairly penalize players who are disconnected by accident. Unless your network offers extremely reliable connections, unlikely to break (such as in a game played on a local area network), you have to consider the possibility of accidental disconnection if anything of tangible value depends upon the outcome of the game; that is, in tournament play or when prizes are at stake.
- *Institute a penalty for disconnections that is less severe than forfeiture*. If a player disconnects in the middle of combat during an *EverQuest* session, the avatar remains in the game for a minute, taking additional damage. Unfortunately, the avatar doesn't fight very well by itself. On the MSN network, players who get disconnected once have ten minutes to reconnect and resume the game; if they fail to do so, they forfeit or, in some games, an artificial player managed by the server takes over for them. If they get disconnected twice, they forfeit automatically. In many games, the game tries to reconnect to the player for a limited amount of item. In a turn-based game, such as poker, this has a minimal impact on the other players who have to wait for their turn anyway. Ultimately, the player is assumed to be away from his computer, and play continues without him until he reconnects.
- *Award victory to whomever is ahead in the game at the time of the disconnection*. This solution seems fair but means that the moment

someone goes ahead, she can disconnect to deny her opponent a chance to catch up. Again, you should consider this only in circumstances in which it is difficult or impossible to disconnect intentionally.

- *Record it as a tie*. While this solution might motivate a losing player to disconnect intentionally, it still makes a fairly neutral solution.
- *Record it as a "disconnected game."* You then have to decide exactly what this means in the context of a tournament. If other players can view the records, they can tell when someone racks up a suspiciously high number of disconnections and avoid playing with that person. Or the server can determine that a player is being disconnected too often and prevent him from playing for a period of time.
- *Abandon the game entirely*. This is the fairest solution in the case of accidental disconnections, but it is unfair to whomever is leading if the player who is behind pulls the plug.
- *Use referees*. The World Cyber Games, a large gaming tournament, keeps a log file during play, and in the event of disconnection, a referee can examine the file to adjudicate victory. If the players agree, they can also restart the match. This requires a human referee to be available, however, which adds to the operating costs.

There's no one right answer to this problem; it depends too much on the nature of the individual game. It's up to you as the designer to think about the problem and try to decide what's fair.

Real-Time Versus Turn-Based Games

Many online games take place in real time with each player acting simultaneously. This offers players maximum freedom; they always have something to do and can order their activities any way they like. It's also more immersive than turn-based gaming. Waiting your turn while other players act harms suspension of disbelief. Unfortunately, real-time gaming tends to make a strategy game into an action game. Whichever player moves his pieces fastest has the advantage. In games such as *Command & Conquer,* victory becomes a matter of establishing an efficient weapons-production system as quickly as possible.

Turn-based games seem rather old-fashioned nowadays, but there is still a demand for them. Many simpler online games are automated versions of non-computerized card games and the like, and they still require players to take turns. For this to work smoothly, you must include certain features:

- *Limit the number of players in one game.* Four or five is a good maximum. With more than this, players will have to wait too long between turns and will grow impatient.

- *Set a time limit on the length of a player's turn.* A slow player or one who has left to answer the phone mustn't be allowed to hold up the game. Both the player whose turn it is *and* all the other players should be able to see a countdown timer. The length of time will naturally vary depending on the sort of game; for a card game such as hearts, 10 seconds should be plenty.
- *Determine a reasonable default action if the player runs out of time.* In games in which it's possible to pass, the best default might simply be to pass without acting, but in a game such as checkers, in which a move is required, the game will have to choose a move. It doesn't have to be a very smart move, however. It's up to the player to supply the intelligence; if he doesn't, it's his own fault.
- *Let players do other things while waiting their turn.* They should definitely be allowed to chat with one another, study the battlefield, organize their units, or do anything else that doesn't actually influence the gameplay.

A few games, such as *Age of Wonders II* or *Civilization IV,* allow all the players to take their turns simultaneously—that is, they each choose their next move at the same time, without knowing what the others are doing. Once they have all chosen (or a timer runs out), the turn ends, and the computer processes and displays the results of all of the moves.

Note that some turn-based games permit very long turns in which players make only one move every 24 hours or exchange their moves by e-mail. This can allow novices to compete against more advanced players. However, such games are so rare that we don't address them here; our concern is with players who are online in real time, even if the game itself is turn-based.

Chat

Every multiplayer game for machines that use keyboards (or headsets) should include a chat feature—a mechanism that enables players to send messages to one another. Depending on the nature of the game, players should be able to send private messages to one other individual, messages only to members of their own team (if any), or general broadcast messages to all other players who might reasonably be interested. In a game played by thousands of players, any one player should be able to broadcast messages only to those in his vicinity or on his team, whatever that might mean in the context of the game—the players at his table, the players in the same room of a dungeon, etc.

Unfortunately, chat brings a new set of problems: the potential for rude, abusive, or harassing behavior. People who pay to play your game expect that others will meet certain minimum standards of civility. In a sporting event, the referee enforces rules that maintain these standards, or if there is no referee, then the collective authority of the other players must suffice. Online, it's much more difficult to police players' behavior.

> Psychological disinhibition: *People act like jerks more easily online because anonymity is intoxicating. It is easier to objectify other people and, therefore, to treat them badly. The only way to combat this is to get them to empathize more with other players.*

Profanity Filters Designers have tried profanity filters, but they aren't fully reliable, and they sometimes produce laughable results. Words such as *damn* and *hell* are perfectly legitimate when talking about religion, even if they're considered swearing in another context—and don't think that people won't talk about religion when they're in your dungeon; they'll talk about everything under the sun. In any case, people can easily get around such filters by misspelling the words. A profanity filter should always be backed up by other means, such as online customer service representatives to whom players can report bad behavior.

Complaint and Warning Systems Some chat systems include complaint mechanisms designed to discourage online rudeness. Some online games give players a *report* button they can push whenever they receive an offensive message. The offending message is automatically forwarded to someone with authority over the game (usually online customer service staff), who can then investigate and take appropriate action: warn the offender to mend his ways, kick him offline, or even ban his account.

The America Online Instant Messenger includes a fully automated system that allows users to warn each other, either anonymously or openly, when one participant behaves badly. A user can be warned once per message that he sends; the more warnings he receives, the less frequently the system permits him to send messages. If he receives enough complaints, he may be unable to send any further messages for several hours. The record of the complaints is deleted over time, so a user's bad behavior is not held against him permanently. If he has behaved himself for a while, he can resume sending messages.

Ignoring Other Players Some chat mechanisms allow a player to hide messages from other individuals whose behavior they find offensive, a practice called *ignoring*. The player simply selects the name of a person he wants to ignore, and he no longer receives chat messages from that person, no matter what the person writes. You can permit this to take place silently (the other player doesn't know she's being ignored) or automatically send a message telling her that she has been ignored—the online equivalent of deliberately turning your back on someone. This mechanism is both effective—the users have full control over whom they hear from and whom they don't—and inexpensive because it doesn't require staff intervention. You should also include the ability to turn chat off entirely if players simply don't want to hear from anyone else.

Moderated Chat Spaces The most effective, but also the most expensive, way of keeping order in a chat space is to give one person authority to discipline the others at all times. Internet Relay Chat uses this method; the creator of a chat room exercises the authority to kick people out. Because the enforcer also participates in the game, this method is, unfortunately, subject to abuse. When people pay to play your game, you can't safely hand over that authority to one of the players; the moderator must be an impartial representative of the game's provider—a customer service agent, in effect, whether paid or unpaid.

Collusion

Most designers of games—true games, in which players or teams compete for victory—work on the assumption that the players will try to beat one another and that there is never any reason for them to help an opponent. Often the rules formalize this assumption; in *Monopoly,* one of the rules states, "No player may borrow from or lend money to another player."

Collusion is a form of cheating in which players who are supposed to be opponents work together in violation of the rules. In *Monopoly,* the players' agreement to abide by the rules, the potential argument or even damage to their friendship if they don't, and the fact that they're all watching each other so it's hard to get away with anything, all work to prevent collusion. Unfortunately, you can't count on any of these factors in an online game. Some players will join a game with a deliberate, even avowed, intent to cheat. Because they're playing with strangers, there's no social relationship at stake, and because they're physically miles apart, no one can see them do it. If you cheat at a board game, you'll most likely be thrown out of the game and your friends won't ever play with you again. Unfortunately, these kinds of penalties are difficult to enforce in online games.

Examples of Collusion Computer games seldom have written rules because the designers assume that the game will enforce the rules automatically: The players simply can't make illegal moves, in most cases. However, software can't detect certain kinds of collusion between players.

Consider an online multiple-choice trivia game with three possible answers for each question. Each player receives the same question from the server and has a fixed length of time in which to enter an answer. When a player enters his answer, he immediately learns whether he was right or wrong. Correct answers earn points, and the player with the largest number of points at the end of the game wins.

Four players can easily collude at this game to guarantee that one of them will win. They all play on different machines in the same physical location—an Internet café, for instance. When a question appears, three of the players each immediately enter a different response—A, B, or C—and the fourth one waits. When the software informs one of three players that she is

correct, she immediately calls out her letter, and the fourth player enters it before the time runs out. This way the fourth player always enters the correct answer. Even with fewer than four players colluding this way, they can greatly increase the odds of winning.

You can easily defeat this form of collusion: You simply don't reveal the correct answer until the time for entering answers runs out. Players who enter an answer early simply have to wait to find out whether they answered correctly. But other forms of collusion can be more insidious. Online poker, for example, can involve players sharing information about their cards via instant messaging or some form of physical communication. There is no way for the system to account for external means of communication. If you offer a prize for the player who wins the greatest number of chess games in a certain length of time, for example, two players can collude to play each other, with one always trying to lose to the other as quickly as possible.

Designing to Reduce Collusion Part of your job as a designer of an online game is to try to anticipate collusion as much as possible. Unfortunately, experience shows this to be extremely difficult. There are no limits on players' ingenuity or the lengths to which some will go, and players outnumber you thousands to one. Even if your game doesn't offer a chat feature, players can play from two machines in the same room, call each other on mobile phones, or use any online chat facility to collude.

You can't prevent players from colluding, but you can design the game to minimize the effects of cheating. You should consider in what ways the following types of collusion might affect your game:

- *Sharing secret knowledge.* Does the player ever have secret knowledge that she can share to someone else's benefit? In the trivia game described previously, some players receive the correct answer before the time runs out. Withholding this information makes collusion pointless.
- *Passing cards (or anything else) under the table.* Does the game include mechanisms to transfer assets from one player to another? Is there any way to abuse these mechanisms?
- *Taking a dive.* What are the consequences if one player deliberately plays to lose? If you allow gambling on matches (even if only with play money or points), you should look out for this.

If you're designing a game in which the competition mode is supposed to be every player for herself, try imagining what would happen if you made it a team game in which you encouraged players to collaborate. If it's already a team game, try to imagine what would happen if one player on the team spied for the other team.

Technical Security

People feel a strong impulse to test the limits of computer software—to see what it will do with nonsensical data, for example. In playing a war game, it is a rare player who will not at some point try to order his troops to fire on his own side, just to see what will happen. Similarly, players often think of ways to do things in a manner that the designers never intended or expected. Sometimes these unanticipated maneuvers, such as using the rocket launcher to propel the player upward in *Quake,* even become standard tactics.

Making unexpected but legal moves is not cheating; one can argue that designers should anticipate these tactics or that testers should discover them.

Other forms of cheating, such as hacking the game's software or data files, are clearly unfair. In a single-player game, it doesn't really matter—it's like cheating at solitaire—but cheating in multiplayer games constitutes a considerably more serious issue. People who wouldn't dream of cheating their close friends in person—say, playing poker around the living room table—happily cheat strangers when protected by the distance and anonymity that an online game offers. There's still that temptation to try to break the software, too, especially because players become used to the notion that computer games will enforce the rules for them.

This presents a grave problem. Players have a moral right to expect a fair game when they're playing against other people, and they have a legal right to a fair game as well if they're paying money for the privilege. This becomes even more crucial if they're playing for prizes. Although all game software comes with a disclaimer that the publisher sells the software as is and without any warranty, the moment you start to give out prizes with monetary value, you must be very careful to ensure the fairness of the game if you don't want to end up in court.

Use a Secure Telecommunications Protocol It takes an extremely dedicated hacker to tamper with the data stream between the client software and the server, but it takes only one. If the stakes are high enough, someone will decide the reward is worth the time spent. To foil hackers, your software must use a secure telecommunications protocol. Designing such a thing is a programming problem and is beyond the scope of this book, but we include a few suggestions so you'll be better prepared to determine whether a telecommunications protocol under consideration will adequately protect your game.

- First, all data should be encrypted to prevent users from understanding it outright. Each packet of data should be sent with a suitable checksum, which will enable the software to detect whether the data has lost integrity in transmission.
- Second, you might want to consider a *heartbeat* mechanism in which your client software sends a short packet to your server at regular intervals, even when it doesn't need to transmit data, simply to tell the server

that the client is still present. This enables you to detect disconnections. If the client can remain silent indefinitely, the server doesn't know if the client has disconnected or the player is just thinking.

- As described earlier, each packet should include a unique serial number, with packets and serial numbers in sequence, to indicate the correct order of packets and to prevent spurious packets from being inserted by unauthorized means.

Don't Store Sensitive Data on the Player's Computer Typically two kinds of data about a player exist in a game. Your game needs to keep settings or preferences about the way the player appears and likes to play, as well as information that's actually relevant to the game state: the player's position, score, possessions, and so on. In *Monopoly,* for instance, the player's playing piece (hat, shoe, car, and so on) belongs in the former category; it doesn't matter to the state of the game which token the player uses. However, the player's properties, cash, and position on the board belong in the latter category; changes to those attributes affect the player's status in the game.

This second kind of information shouldn't be stored on the player's own computer. Even with encryption techniques, you have to assume that someone will tamper with any data kept on the player's machine to give that player an unfair advantage. If your game truly generates too much sensitive data about each character to store it all on the server, at least store a checksum over the data when the player logs out so that when he logs back in again, you can check his data and determine whether it has been improperly modified in the meantime.

Don't Send the Player Data He Isn't Supposed to Have A common characteristic of real-time strategy games is the *fog of war,* in which unexplored areas of the map appear dark and the player cannot detect movements of the enemy unless a friendly unit nearby can plausibly see them. Single-player games store all of this information in the player's computer; it's just not visible to the player. Online games should not send any such hidden information to the player. If the player hacks the game to lift the fog of war, he can see unexplored areas and watch the movements of enemy units, giving him a significant advantage over his opponents.

Don't Let the Client Perform Sensitive Operations In designing a client/server game, you must always strike a balance between the amount of processing that the server does and the amount that the client does. It saves CPU time on the server for you to offload as much of the processing work onto the client as you can, but it isn't always safe. Suppose, for example, that you're designing a very simple role-playing game in which the player occasionally encounters monsters and must fight them. It reduces the load on the server if the server sends the client some information about the current monster and lets the

fight take place entirely on the player's computer. After the fight, the client sends a message back to the server reporting whether the player won, lost, or ran away, but this presents a danger: If the player hacks the client, he can program it to report that he wins every fight. In fact, the server, not the client, should perform the computations for the fight and determine whether the player won or lost.

> Never trust the client: *Never put* anything *on the client. The client is in the hands of the enemy. Never ever forget this.*

The legitimate players aren't the enemy, of course, but the handful of cheaters are. We lock our doors at night not to protect ourselves from the honest majority of the population but to protect ourselves from the dishonest minority. You will have to design your game with the same consideration in mind.

Persistent Worlds

A game, as we defined it earlier, is a contest with rules and a winner. However, a good many online games are not really games at all by that definition. Persistent worlds such as *World of Warcraft, EverQuest, Anarchy Online,* and so on constitute permanent environments in which players can play, retaining the state of their avatar from one session to another. Persistent worlds present a number of special problems and design requirements, which we discuss here. For a more in-depth discussion, we suggest that you read *Designing Virtual Worlds* by Richard Bartle (Bartle, 2003), and *Developing Online Games: An Insider's Guide* by Jessica Mulligan and Bridgette Patrovsky (Mulligan and Patrovsky, 2003).

The Origins of Persistent-World Gaming

Persistent worlds significantly predate today's popular graphical MMORPGs. Since 1978, a small but dedicated community of developers has been building, playing, and studying text-based persistent worlds called *MUDs* (*multiuser dungeons* or *domains,* depending on who you talk to) that could be played by groups of people over the Internet. In these worlds, in which players accomplish all actions by typing commands, a rich culture of online role-playing evolved. MUDs later fragmented into subcategories: MUSHes (*multiuser shared hallucinations,* dedicated primarily to personal interaction rather than combat) and MOOs (*MUDs, object-oriented,* in which players can use a scripting language to design their own objects for use in the environment), and others even more esoteric.

We won't go into MUD design in any detail here; there is no commercial market for MUDs, and you can already find a vast amount of literature about the subject on the Internet. Many of the design problems of today's MMORPGs, particularly those relating to social interactions among players, were solved—or at least studied—long ago in the MUD community.

How Persistent Worlds Differ from Ordinary Games

Part of the appeal of computer games is the environment in which the player finds herself: a fantasy world where magic really works or behind enemy lines in World War II. Another part is the role the gamer will play in the game: detective or pilot or knight-errant. Yet another is the gameplay itself, the nature of the challenges the player faces and the actions she may take to overcome them. And, of course, there is the goal of the game, its victory condition: to halt the enemy invasion or find the magic ring. The victory is usually the conclusion of a story that is told partly through narrative supplied by you, the designer, and partly through the player's own actions.

Persistent worlds offer some of these things but not all of them, and there are significant differences between the kinds of experiences that persistent worlds offer and those that conventional games offer.

Story Because persistent worlds have so many players, and because they are intended to continue indefinitely, the traditional narrative arc of a single-player game doesn't apply. Persistent worlds may offer storylike quests, but they always return to the world eventually; you can't have a once-and-for-all ending in the sense that a story does.

The setting of a persistent world consists of the environment itself and the overall conditions of life there. It can be a dangerous place or a safe one, a rich place or a poor one, a place of tyranny or a place of democracy. You can challenge players to respond to problems in the world as it is or to problems that you introduce, whether slowly or suddenly.

> Elmqvist's Law: *In an online game, players find it rewarding to save the world. They find it more rewarding to save the world together, with lots of other people.*

The goal is a quest or errand that the player undertakes as an individual or with others. Goals can be small-scale (eliminate the pack of wild dogs that has been marauding through the sheep flocks) or large-scale (everyone in the town gets together to rebuild the defenses in anticipation of an invasion). Most persistent worlds offer large numbers of quests from which players may choose.

As a designer, you would probably want players to feel as if they were the first ones ever to undertake a particular quest, or to explore an area of the game world. Ordinary computer games allow you to evoke that feeling, because the game world is created afresh when the player starts up the game program. In a persistent world, on the other hand, only those who logged in on its first day of operation are the first to experience a quest or explore a new area. Furthermore, those who went before will always tell those who arrive later what to expect. In short, it's impossible to keep anything secret about a persistent world. As soon as a few players know it, they'll tell the other players.

In Chapter 7, "Storytelling and Narrative," we introduced the idea of *emergent narrative*: stories that emerge from the core mechanics of a game. In a persistent world, stories emerge not so much from the core mechanics as from interactions among the players. The best emergent stories (those that make the player feel as if he's participating in a story created by a great writer) occur in purely role-playing environments with almost no gamelike elements, such as in MUSHes. In effect, the story experience in a persistent world comes about when the players are excellent role-players: good at acting and improvisational theater. As a designer, you cannot force good stories to emerge; it depends too much on the imagination and talent of the participants.

The Player's Role In a single-player, plot-driven game, the player's role is defined by the actions he is allowed to take and is constrained by the requirements of the story. In a persistent world, the player doesn't follow a single storyline, so he may, in theory, choose from a larger variety of things to do and has more opportunities to define his own role. The early persistent worlds offered only a limited number of roles, but modern ones are increasingly rich and varied.

As the designer, you must supply an assortment of possible roles the player may take on and make those roles meaningful in your world. You should also give the player the freedom to change her role (though not always easily or immediately) as she sees fit. Because the world continues indefinitely without coming to a narrative conclusion, you can't expect the player to want to play the same way forever. Just as people change careers and hobbies over time, players need to be able to change roles.

Gameplay Finally, there's the question of the gameplay. Without a victory condition, you can't simply offer the player a predefined sequence of challenges and achievements as her ultimate objective. In the familiar persistent worlds designed like role-playing games, the player's objective is to advance her character. She (usually) accomplishes this by fighting AI-controlled opponents, such as monsters, although she could also attain many other things as well: wealth, political power, fame (or notoriety), and so on.

FYI *If Your Game Is Narrow, It Will Fail*

Your game design must be expansive. Even the coolest game mechanic becomes tiresome after a time. You have to supply alternate ways of playing or alternate ways of experiencing the world. Otherwise, the players will go to another world where they can have new experiences. This means new additions or, better yet, completely different subgames embedded in the actual game.

In a single-player game, the player tries to read the designer's mind to some extent, to figure out what you want him to do, and then do it. His play is often reactive, a response to challenges thrown at him. In a persistent world, the player decides for himself what he wants to do. He seeks out challenges if he feels like it, but he can spend all his time socializing if he prefers. His gameplay—and, indeed, the entire nature of the experience—is *expressive* and active rather than reactive. Role-playing, especially when it's taken seriously by the players, shares characteristics of live theater and improvisation. This quality of persistent-world play has profound effects on the design of such worlds, as you will see later in this section.

> Schubert's Law of Player Expectations: *A new player's expectations of a virtual world are driven by his expectations of single-player games. In particular, he expects a narrow, predictable plot line with well-defined quests, carefully sculpted for himself as the hero. He also expects no interference or disruption from other players. These are difficult and sometimes impossible expectations for a virtual world to actually meet.*

The Four Types of Players

In 1997, MUD developer Richard Bartle wrote a seminal article called "Hearts, Clubs, Diamonds, Spades: Players Who Suit MUDs" for the first issue of the *Journal of Virtual Environments* (Bartle, 1997). He proposed that MUD players fall into four categories depending on whether they enjoy either *acting on* (manipulating, exploiting, or controlling) or *interacting with* (learning about and communicating with) either the world or the other players (see Figure 21.1). Those who enjoy acting on other players he dubbed Killers, or clubs; those who enjoy interacting with other players he called Socializers, or hearts. Those who enjoy acting on the world he described as Achievers, or diamonds; those who enjoy interacting with the world he referred to as Explorers, or spades.

Bartle went on to claim that a healthy MUD community required a certain proportion of each of these types of players and that adjusting the game design

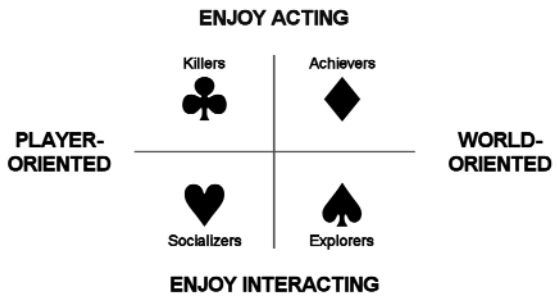

FIGURE 21.1 Bartle's four types of players.

to attract or discourage any given type of player would tend to influence the numbers of others as well. In effect, a persistent world is a sort of ecology in which the players' styles of play influence the balance of the population. Bartle drew his data from personal observation rather than rigorous statistical analysis, so his conclusions can certainly be questioned. Nevertheless, his grouping of player types proved to be useful not only in the design of MUDs, but of graphical persistent worlds as well.

Creating an Avatar

As we said earlier, playing in persistent worlds is more than merely a form of gameplay; it's also a form of expression. The first thing a player does in joining a persistent world is to create an avatar, or character who represents her in the game, one of the most expressive things she can do. We discuss avatar creation at greater length in Chapter 5, "Creative and Expressive Play."

If you're making an online role-playing game that includes traditional avatar attributes such as speed, strength, and so on, consult Chapter 15, "Role-Playing Games," for more information.

Players like to maintain a profile listing some of their intangible attributes in order to identify and describe their avatars to other players. Profiles can include such things as:

- *Unique name or handle.* Unless your game allows totally anonymous play, people will need some way of identifying their avatars by name. That way, a player's name can appear in documents, on leader boards, in chat rooms and bulletin boards, and so on.
- *Physical appearance.* People clearly need to be able to tell one character from another on the screen. The physical appearance of avatars should be as customizable as you can afford to make it. Even if appearance does not affect gameplay, players identify with and respond to physical appearances.
- *History or experience.* Players like to record their characters' achievements for others to see. Records can include experience levels, quests undertaken, kills in battle, or any other accomplishments the player might be proud of. You'll have to decide whether some players will want to keep some of these things private and, if so, whether they should be allowed to.
- *Reputation,* which is stored as a number or symbol computed by the system based on the player's play or on complaints or praise received about the player. (The eBay auction Web site includes a simple reputation system.) Some games use the reputation mechanism as a way of automatically tagging players who frequently take advantage of others. The reputation attribute warns other players, "This person is dangerous" or

"This person is trustworthy." Beware, however: An automated system is subject to abuse through collusion if you don't place limits on it. If you offer a player the opportunity to repair a bad reputation through some apparently virtuous action such as donating money to another character, he can simply donate money repeatedly to a friend, who promptly donates it all back to him.

- *Player autobiography.* It's fun for a player to make up a history for his character, a background that will introduce that character to others in the world. It's another form of self-expression. If children play in your world, you will need to have a real person approve autobiographies for suitability.

You might or might not want to include important gameplay attributes in the player's profile; it depends on how making such information public affects the gameplay. Does allowing a player to hide attributes from the world constitute a legitimate part of gameplay or an unfair advantage? (Consider *Monopoly,* which does not allow players to hide their property cards under the table but allows everyone to see what all players own.)

World Models

If you plan to offer more than just a chat room, you must give players something to do. The types of things that you give them to do and the rewards they earn for doing those things constitute the *world model*. Raph Koster identified five classic world models, although you can undoubtedly devise more. Yours may include elements from more than one of Koster's original five, listed here:

- *Scavenger model.* Players collect things and return them to places of safety. The game is primarily a large treasure hunt, and players don't risk losing anything they've collected.
- *Social model.* The world exists primarily to provide an expressive space. The fun comes from role-playing in character; most goals represent social achievement (political power, adulation, notoriety, and so on). Players use their characters' attributes as a basis for role-playing rather than computer-managed combat.
- *Dungeons & Dragons model.* In games based on this, the best-known model, the player is primarily in conflict with the environment, fighting NPCs for advancement and doing some scavenging along the way. Such games rely heavily on the functional attributes of the avatar for gameplay and include feedback mechanisms: Defeating enemies advances the character, which requires the game to offer tougher enemies next time. Such worlds tend to include quests as a form of narrative and a way of offering challenges to the players.

- *Player-versus-Player (PvP) model.* In this sort of world, players advance by defeating one another at contests, often characterized as combat. Players advance through a combination of their natural skill and rewards from winning battles. For this to work successfully, they need to be reasonably evenly matched; you can't have the old-timers beating up the newcomers all the time.
- *Builder model.* This somewhat rare sort of world enables players to construct things and actually modify the world in which they play. It's a highly expressive form of entertainment. People get kudos not for their fighting skills, but for their aesthetic and architectural ones, both intangible qualities.

Avatar Death

In any persistent world that includes combat, you must decide whether it's possible for the player's avatar to die and what will happen if it dies. As in other games, avatar death must be accompanied by a disincentive of some kind or combat will not be a meaningful part of gameplay. The trick is to find a disincentive that is proportional to the likelihood of the avatar's death. It is a question of balance: If the avatar can easily be killed through no fault of the player (such as through ignorance or bad luck), then the cost of dying—the disincentive—should be low, but if the player really has to be stupid to get his avatar killed, the cost should be high. Some examples follow.

Permanent Death In the most extreme case, the avatar dies and cannot be resurrected. The player loses all property that he owns (in which case you must decide what happens to that property) and must start over from scratch with a new avatar. This makes sense in games of short duration, but seldom in persistent worlds. Players in persistent worlds put too much time and effort into building up their avatars for you to ask them to start over.

Resurrection with Reduced Attributes Designers commonly penalize the player for letting her avatar die by bringing the avatar back to life with reduced functional attributes—with less strength or perhaps fewer skills. In effect, you give the player a bit of a setback in her quest to grow a powerful avatar. Players find this irritating, so fear of incurring such a setback discourages risky play, but the penalty makes a certain amount of sense. If a gang of club-wielding trolls beats the avatar to death, the avatar ought to feel pretty lousy for a while when it comes back to life!

Resurrection with Some Property Missing Another classic disincentive for dying involves loss of money, gear, clothes, and other items in the avatar's inventory at the time. How much of his property he loses and what

becomes of it can vary considerably from game to game. You can also allow players to have a vault in the game in which they can keep items that they're not carrying with them, and these items can remain safely in the vault for use by their resurrected avatar. You might as well include this feature because, if you don't, the players will create a second character that they never play with, known in MMORPG parlance as a *mule,* to hold their primary avatar's things for them.

The Player-Killer (PK) Problem

No aspect of the design of persistent worlds has been debated more than this one simple question: Should players' avatars be allowed to kill one another? We won't offer a definitive answer here, but instead we'll summarize some of the issues so that you can make an informed decision for your own game.

Most designers offer persistent worlds resembling role-playing games in that players advance in skill and power through combat. It's generally more interesting if this combat occurs against another player rather than against an NPC, for several reasons. First, another player's avatar is more likely to be carrying interesting and valuable objects worth looting (if the game permits looting) than a randomly generated NPC; unlike an NPC, another player will have kept only valuable items and gotten rid of anything not worth keeping. Second, fighting another player is a social experience, which an NPC cannot offer. Finally, a player can use his human intelligence to put up a better fight; an NPC has to use AI, which is seldom as sophisticated.

The *Ultima Online* Experience The designers of *Ultima Online* initially permitted players to kill one another without restraint (except in towns), hoping that players would establish their own justice mechanisms within the game. Unfortunately, the world quickly began to resemble Afghanistan after the Soviets left: unremitting random violence, feuds, continual victimization of the weak by the strong, and petty warlords or gangs of bandits controlling areas of turf. Players engaging in this behavior became known as *player-killers,* or PK players. No satisfactory solution arose from the players, partly because the software did not offer any genuinely painful punishment mechanisms for them to use against offenders. (In real life, we either lock murderers away for a very long time or kill them permanently, neither of which any for-pay persistent world can afford to do.) Designers then tried a variety of different automated mechanisms for encouraging justice, but players found ways of exploiting most such mechanisms. In the end, the developers threw up their hands and divided the world into *shards* (separate, independent versions of the world) with different rules for each. Some allowed player-versus-player (PvP) combat, and others did not. Approximately 80 percent of the players chose to play in non-PvP shards.

Justice Mechanisms Koster offers the following summary of approaches to regulating PvP combat, whether fatal or not:

- *No automated regulation.* Anyone can attack anyone, and administrators or social mechanisms (vigilante justice) deal with rogue players. Koster estimates that as many as 40 percent of the potential audience will avoid this type of game because they don't like PvP.
- *Flagging of criminals.* Player-killing is considered a criminal act within the game's rules: not prevented by the system, but wrong. The server automatically detects criminal behavior and flags the criminals, who become fair game for others to attack. The system can also reduce the attributes of criminals, in effect penalizing them for their behavior. This can be used for thievery and other crimes as well as murder, matching the reduction to the severity of the crime.
- *Reputation systems.* This is similar to flagging, except that players decide when to report someone for criminal behavior and can choose not to do so. In practice, they almost always do, however.
- *PvP switch.* Players indicate their willingness to fight other players by setting a switch (a binary attribute) in their profile, becoming either a PvP player, who can attack and be attacked, or a non-PvP player, who cannot attack or be attacked by others. This switch can also be used to give temporary consent for duels and arena-based combat. Unfortunately, this mechanism creates suspension-of-disbelief problems when players use area-effect weapons such as a magic fireball spell: Three PvP players get roasted by a fireball that leaves a non-PvP player in the same vicinity untouched because he cannot be attacked.
- *Safe games; no PvP allowed.* The least troublesome solution, even this approach has its hazards. Players will still find ways of abusing one another—for example, by luring an unsuspecting newcomer into an area where he will be attacked by a monster. Koster estimates that this approach will cost you up to 20 percent of your potential audience, that 20 percent being those players who like PvP.

You can also divide the world into safe and dangerous geographic zones, but in practice people tend to either stay in the safe zones or play near the edges, hoping to lure a potential victim over the line without his realizing it.

Faction-Based PvP A number of persistent worlds allow players to belong to factions. These can be as small as gangs or as large as entire nations at war. The rules enable players to attack members of enemy factions but not members of their own faction—in effect, it's team play. Different factions control different regions, so players can generally tell safe areas from unsafe areas. For the

most part, this arrangement solves the random violence problem that initially plagued *Ultima Online*.

The Bottom Line on Player Killing You cannot please everybody, so you will benefit from deciding *whom* you want to please and tailoring your environment to them. In a game that allows player-killing, a certain number of players will inevitably abuse the system, exploiting their superior strength to victimize weaker ones without ever putting themselves at risk. You can't please them without also providing them with victims who *won't* be pleased, so it's not worth trying. Ultimately, you need to bear two things in mind:

1. **It's a fantasy-world.** That means it's supposed to be enjoyable, escapist entertainment. People don't fantasize about being harassed, bullied, or abused. A fair contest among consenting players is one thing; perpetual harassment or an ambush by a gang is quite another.
2. **People pay to play.** This makes your world distinctly different from the real world. The real world doesn't owe us anything; we survive as best we can. But as a game provider, you've taken the players' money, so you have obligations to them. Just exactly what those obligations are is open to debate, of course, but if players don't feel that you are meeting your obligations, they will leave.

FYI *Player Expectations*

Players have higher expectations of the virtual world than of the real world. For example, players will expect all labor to result in profit; they will expect life to be fair; they will expect to be protected from aggression before the fact, not just having to seek redress after the fact; they will expect problems to be resolved quickly; and they will expect that their integrity will be assumed to be beyond reproach. In other words, they will expect too much, and you will not be able to supply it all. The trick is to manage their expectations.

The Nature of Time

In a single-player computer game, you maintain a great deal of control over the relationship between game time and real time. Most games run at many times the speed of real time, and a player often experiences a simulated day in a game world in an hour or less of real time. You can also hand over control of the speed of time to the player when you want to; it's not uncommon for players in combat flight simulators to speed up time when flying to and from the combat zones and then slow down to real time again when they get there. Finally, you can skip

time entirely—useful during periods when the player's avatar sleeps. You can blank the screen for a moment and then put up a text message that says, "8 hours later . . . ," and continue with the game.

But you can't use any of these options in multiplayer games. You obviously can't allow some players to move through time at different speeds than others, and you can't skip time unless you force all players to skip that time together. Although game time might be faster than real time, game time must proceed at the same fixed pace for everyone.

As a result, you must be careful about designing time-consuming activities. *EverQuest,* for example, employs a mechanism called *meditation,* in which players simply have to wait around for a while to restore their magic powers. There's no way to speed up this process—it literally does involve waiting. Nor can they log out of the game while meditating and log back in again later when the meditation process finishes. Players can't even switch to a different process on their computers. Verant, the developers, eventually built in a mini-game for players to play while waiting—a patch but not a real solution. If your game contains features so boring that you have to distract the players, you need to rethink the features.

Single-player games can be stopped and restarted at the player's discretion—a key consideration for designers of such games. If the player can save the game, he can essentially reverse time by going back to a previous point in the game and replaying the game from that point. This robs single-player games of the emotional impact of events because anything that happens, good or bad, can be reversed by reloading a saved version of the game. You can design the game to include some inevitable events, but of course, the more of these you include, the less interactive the game is.

In an online game, time is irreversible. Even if you had a convenient way to reverse time, you can't reasonably ask all your players to agree to reverse time to an earlier point (although the managers of some persistent worlds have had to roll back to a saved state when the game got into problems). In the ordinary course of events, when an event occurs in an online game, it's done and can't be undone. A bad situation can be corrected, but it cannot be made as if it had never happened.

Persistent World Economies

If the players in a persistent world can collect and trade things of value, then the world includes an economy. We discussed economies in some detail in Chapter 18, "Construction and Management Simulations." Economies are *much* easier to design and tune in a single-player game than they are in a persistent world. You can control the actions of a single person fairly strictly; in a persistent world, thousands of people interact within your game in ways that you might not have anticipated.

The original *Ultima Online* had a completely self-contained, closed economy with a fixed number of resources flowing around and around. You could

mine iron ore, smelt it into iron, and forge the iron into weapons. Using the weapons would cause them to deteriorate, and when worn, they would return to the pool of raw iron ore available for mining. This last step wasn't strictly realistic, but it did close the loop.

The designers, however, didn't anticipate that players would hoard objects without using them. Because unused objects didn't go back into the pool, the iron ore quickly ran out, and as resources dwindled, inflation ran rampant. The players with hoards of iron had cornered the market and could charge extortionate fees for iron objects. Eventually, *Ultima Online*'s proprietor could do nothing but adopt an open economy in which servers add new resources at intervals (Simpson, 2000).

It's essential in any economy that players not find a way to create something for nothing; that is, to return a resource to the system for more than they paid for it in the first place. Otherwise, they'll find a way to automate this process and generate an unlimited stream of that resource. Koster observes:

> Online game economies are hard: *A faucet/drain economy is one where you spawn new stuff, let it pool in the "sink" that is the game, and then have a concomitant drain. Players will hate having this drain, but if you do not enforce ongoing expenditures, you will have Monty Haul syndrome, infinite accumulation of wealth, an overall rise in the "standard of living" and capabilities of the average player, and thus produce imbalance in the game design and poor game longevity.*

IN PRACTICE: Secrets to Successful Persistent Worlds

The secrets to a really long-lived, goal-oriented, persistent world of wide appeal:

- Have multiple paths of advancement (individual features are nice, but making them ladders is better).
- Make it easy to switch between paths of advancement (ideally, without having to start over).
- Make sure the milestones in the path of advancement are clear and visible and significant (having 600 meaningless milestones doesn't help).
- Ideally, make your game not have a sense of running out of significant milestones (try to make your ladder not feel finite).

CONTINUED ON NEXT PAGE

CONTINUED

Ownership is key: You have to give players a sense of ownership in the game. This is what will make them stay—it is a "barrier to departure." Social bonds are *not* enough because good social bonds extend outside the game. Instead, it is context. If they can build their own buildings, build a character, own possessions, hold down a job, feel a sense of responsibility to something that *cannot* be removed from the game—then you have ownership.

Summary

Multiplayer games are harder to design than single-player ones; online games are harder still, and persistent worlds are the hardest of all. It's a bit like the difference between cooking for yourself and planning a dinner party. When you're cooking for yourself, you decide what you want, make it, and eat it. When you're planning a dinner party, you have to take into account more variables: who likes what food, who gets along with whom, and what entertainment should you offer aside from the food. A dinner party requires more work ahead of time—but it's a lot more fun than eating by yourself, too. The flexibility and power of online gaming enables you to create entertainment experiences that you simply can't produce in other forms.

Test Your Skills

MULTIPLE CHOICE QUESTIONS

1. Which of these makes online games different from multiplayer local games?
   A. Online games allow social interaction.
   B. Online games offer avatar creation facilities.
   C. Online games always have hundreds of players.
   D. Online games support secret information for each player.

2. A game designer for online games also has to design
   A. clubs and casinos.
   B. social environments.
   C. networking protocols.
   D. tournaments and challenges.

3. Which of these is *not* a problem for online games?
   A. It's harder to suspend disbelief.
   B. The company has to provide customer service.
   C. AI for online games has to be much smarter than AI for single-player games.
   D. The game must handle a number of technical issues related to networked communications.

4. What role do other players take in an online game?
   A. Players can perform many of the tasks previously done by AI-controlled NPCs.
   B. They can rewrite the quests.
   C. Players should be allowed to act as game managers.
   D. Enemy design.

5. Playing against human opponents is usually a richer experience because computer software is not very good at
   A. puzzle solving.
   B. pathfinding.
   C. creating believable opponents.
   D. emulating human behavior.

6. Which is the correct way to handle a disappearing player in a game?
   A. Penalize the disappearing player.
   B. Record it as a disconnected game.
   C. Use referees to decide what's fair.
   D. Any of the above, depending on the circumstances.

7. In a turn-based online game, which of the following is *not* a required feature of the design?
   A. The game allows all players to choose their actions simultaneously.
   B. The game sets a time limit on the length of a player's turn.
   C. The game executes a reasonable default action if a player runs out of time.
   D. The game lets players do other things while waiting.

8. What should a game designer for an online card game include to handle players using inappropriate language?
   A. Have a moderator to screen all chat messages.
   B. Never allow chat.
   C. Use a filter to screen for certain words and don't display them.
   D. Make the game harder for the offending player.

9. Though you can't stop all collusion in online games, which of these would you *not* implement in your design of an online multiplayer game?
   A. Share the answer to a puzzle only after all players have submitted.
   B. Randomly change the puzzle with every new attempt.
   C. Don't allow players to trade inventory items.
   D. Allow the game to progress only when all players get the correct answer.

10. One way to thwart software hackers who might try to gain an unfair advantage in an online game is to
    A. not allow chat.
    B. avoid storing any critical or valuable information on the client machine.
    C. allocate serial numbers for all packets in the network.
    D. update the client software from the server daily.

11. Bartle's four types of players are:
    A. Killers, socializers, explorers, achievers.
    B. Socializers, killers, collectors, achievers.
    C. Explorers, achievers, killers, socialists.
    D. Achievers, socializers, explorers, nurturers.

12. Which world model would be the best for an online adventure game with no combat?
    A. *Dungeons & Dragons* Model.
    B. PvP Model.
    C. Scavenger Model.
    D. Builder Model.

13. What is a *mule* in a persistent world?
    A. A pack animal used to carry an avatar's possessions around the game world.
    B. An acronym for Multi-Use Labor Element.
    C. A slang term for a player who repeatedly complains to game administrators about being abused by other players.
    D. A character created solely for the purpose of holding an avatar's property.
14. Which of these is not a justice system used by persistent worlds?
    A. No automated regulation; only social or vigilante systems.
    B. PvP switch.
    C. No PvP play allowed.
    D. Cash fines imposed by charging the player's credit card.
15. How is the nature of time different in persistent worlds from that in single-player games?
    A. In persistent worlds, game time can speed up and slow down for individual players.
    B. In single-player games, the game can skip periods of time in which nothing interesting happens.
    C. In single-player games, game time must always move at a fixed rate.
    D. In persistent worlds, you must design time-consuming activities.

DESIGN QUESTIONS

The following questions deal with basic features of online games generally, not with persistent worlds specifically or any one particular genre.

1. How do you plan to deal with the issue of new players arriving in the middle of a long game? Get rid of the victory condition, or find a way to make sure that players are matched with those of similar ability?
2. What will happen to the gameplay when a player vanishes? How will it affect the other players' experience of the game (what they see and hear)? Does it disrupt the balance of the game? Will it make the challenges easier or harder? Is the game even meaningful anymore?
3. What happens to the game's score when a player vanishes? Is the game still fair?
4. Does your game offer a player an advantage of some kind for intentionally disconnecting himself (whether by preventing himself from losing or by sealing his own victory)? Is there any way to minimize this without penalizing players who are disconnected accidentally?

5. In a turn-based game, what mechanism will you use to prevent a player from stalling play for the other players? Set a time limit? Allow simultaneous turns? Implement a reasonable default if the player does nothing?
6. If you offer a chat mechanism, what features will you implement to keep it civil? Dirty-word filters? A complaint system? An ignore system? Or will your game require moderated chat spaces?
7. Is your game designed to prevent (or alleviate) collusion? Because you can't prevent players from talking to each other on the phone as they play, how will you address this? Or can you design your game in such a way that collusion is part of the gameplay, as in *Diplomacy?*

References

Bartle, Richard. 1997. "Hearts, Clubs, Diamonds, Spades: Players Who Suit MUDs." *Journal of Virtual Environments* (formerly the *Journal of MUD Research*) 1.

Bartle, Richard. 2003. *Designing Virtual Worlds.* Indianapolis: New Riders Games.

Kim, Amy Jo. 2000. *Community Building on the Web: Secret Strategies for Successful Online Communities.* Berkeley, CA: Peachpit Press.

Mulligan, Jessica, and Bridgette Petrovsky. 2003. *Online Games: An Insider's Guide.* Indianapolis: New Riders Games.

Simpson, Zack Booth. 2000. "The In-Game Economics of *Ultima Online*." Lecture delivered at the Game Developers' Conference, San Jose, California, March 2000.