Chapter 9. Creative and Expressive Play

Playing any game involves an element of self-expression because the decisions a player makes reflect his play style: cautious or reckless, aggressive or defensive, and so on. Video games can let players express themselves in the ways traditional games always have and in a variety of other ways as well. This chapter examines several types of creative and expressive play that you can build into a game: self-defining play, in which players modify the avatar that represents them in the game; constrained creative play, in which players may exercise their creativity but only within certain limits; freeform, or unconstrained, creative play; and storytelling and role-playing, in which players interact with other players in a dramatic context. We end the chapter by briefly discussing some features you may wish to include that allow players to modify your game for their own entertainment: level editors, mods, and bots.

Self-Defining Play

When a player selects a token to represent herself in *Monopoly*, she chooses an avatar and so engages in an act of self-definition. Many games allow the player to

choose an avatar from a number of different ones available and to customize the avatar in various ways. Because the avatar represents the player in the game world, these activities are called **self-defining play**. Players greatly enjoy defining themselves, choosing an avatar that either resembles them physically (if it's a human character) or that is a fantasy figure with whom they identify. It isn't just a question of choosing an avatar, however; players also enjoy customizing their avatar as well.

Forms of Personality Expression

Self-defining play gives the player an opportunity to project his personality into the game world, and explore alternate identities, by means other than gameplay choices. It takes several forms:

- **Avatar selection** allows the player to choose from a number of predefined avatars, usually at the beginning of the game. These avatars are most often humanoid characters, but in driving and flying games, they're vehicles. Many driving games start the player with a small selection of cars, motorcycles, or whatever vehicles are involved and make new choices available as the player's performance improves. You can let the player purchase a new car with winnings earned in previous races, for example. The right to choose a new and more powerful avatar serves as a reward to some

players.

- **Avatar customization** allows the player to modify the appearance or abilities of an avatar that the game supplies by modifying its features. In role-playing games (RPGs), this often takes the form of giving the avatar new skills, clothing, weapons, and armor. In driving games, the customizable features may include the paint color of the car and its engine, transmission, tires, and brakes. Customization can occur both at the beginning of the game and through upgrades awarded or purchased as the game goes on. In this way, a player creates a unique character of her own design. Customization can be purely cosmetic or visual, as with the Nintendo Mii characters, or it can include choices about the character's attributes that may have an effect on gameplay. Younger children pay more attention to visuals than to attributes, as they are not yet used to thinking about games as systems.

- **Avatar construction** gives the player the greatest freedom of all; he can construct his avatar from the ground up, choosing every detail from a set of available options. Usually offered in RPGs, avatar construction allows the player to choose such features as the sex, body type, skin color, and clothing of the avatar, as well as the avatar's strength, intelligence, dexterity, and other functional qualities. The online RPG *Lord of the Rings Online* offers a particularly extensive avatar

the Ringo Online offers a particularly extensive avatar construction feature, as does the single-player RPG *The Elder Scrolls IV: Oblivion* for the PC. Some, such as *Second Life* and *Minecraft*, even let the player import his own graphics for avatars or clothing.

Understanding Attributes

The qualities that a player modifies when constructing or customizing an avatar are called **attributes**. **Chapter 14**, "**Core Mechanics**," discusses attributes in more detail, but for now, it's enough to know that an attribute is any quality that helps to describe something else. *Hair color* is an attribute of a person. *Maximum airspeed* is an attribute of an aircraft. The computer can represent an attribute as a numeric value (such as maximum airspeed) or a symbolic value (such as hair color). All attributes in a video game must be characterized in one of these two ways. Even if you create an attribute intended to describe something that we normally think of as unquantifiable, like smell, ultimately it will come down to either a numeric or a symbolic value.

You can divide attributes in a game into those that affect the gameplay, which are called **functional attributes**, and those that don't affect the gameplay, which are called *cosmetic attributes*. (Some designers prefer the term *aesthetic attributes*, but the meaning is the same.) The next two sections examine these types

more closely.

Functional Attributes

Functional attributes influence the gameplay through interactions with the core mechanics. Functional attributes can be further divided into **characterization attributes**, which define fundamental aspects of a character and change slowly or not at all, and **status attributes**, which give the current status of the character and may change frequently. For example, *maximum airspeed* is a characterization attribute of an aircraft, while *current airspeed* is a status attribute. For the purposes of creative play, we're interested in the characterization attributes.

You have probably heard of the six characterization attributes used in *Dungeons & Dragons:* strength, dexterity, intelligence, wisdom, charisma, and constitution. Each of these attributes affects a character's ability to perform certain actions in the game: fight, cast magic spells, charm others, withstand poisons, and many other tasks. When a *Dungeons & Dragons* player creates a character, she receives a certain number of points (usually obtained by rolling dice) to distribute among these attributes. How she distributes them—giving more to dexterity and less to intelligence, for instance—establishes the character's strengths and weaknesses. These strengths and weaknesses, in turn,

determine how the player must play with the character to be successful in the game: taking advantage of the strengths and avoiding situations in which the weaknesses render the player vulnerable.

NOTE

If the player's choice of avatar or attribute settings will have an effect on the gameplay, you must make the consequences of those choices reasonably clear to the player. If you require the player to make this decision before play begins, either all choices must provide a reasonable chance of winning (even if the fastest way to win varies from one choice to another), or you should clearly mark the choices that make the gameplay easier or harder. Don't force your players to choose an avatar or set its attributes without telling them how those decisions will affect their chances of winning.

When a player sets the characterization attributes of her character, the player defines herself in a creative way. Hardcore players, whose main interest is in winning, tend to look for the setting that gives them the greatest advantage in the game—that is, to optimize the attributes' influence on the core mechanics. Casual players either don't worry about the assignments

much, or they select settings that allow for interesting role-playing. A character who is highly charismatic but physically weak, for example, has to be played quite differently from a conventional warrior.

If you allow players to assign any legitimate value to their functional attributes, some players will set up their attributes in the best possible configuration, and the game will be very easy for them. You may want to prevent this to keep the game challenging. Consider the following approaches:

- Give players a fixed or random number of points to assign among all their attributes, as in *Dungeons & Dragons*. This allows them to make interesting choices and create an avatar who reflects their own personality or fantasies without unbalancing the game. If you generate a random number of points for the player, use a nonuniform distribution as *Dungeons & Dragons* does in order to avoid producing unusually strong or weak characters. See “Random Numbers and the Gaussian Curve” in **Chapter 14**.

- Include a set of default, or recommended, settings so players who want to get started quickly can do so without spending a lot of time setting attributes. This is especially valuable for players who don’t understand how the attributes affect the gameplay anyway. They will find it frustrating to be required to set attributes

when all they want to do is get into the game and start playing. They will appreciate being given a reasonable default. This is sometimes called a *quick start* mode.

- Allow players to *earn* the right to set their character's functional attributes any way they like by completing the game with constrained attributes first. You can also offer this right explicitly as a cheat feature of the game, so players will know they're getting an unusual advantage.

Dungeons & Dragons provides one of the most familiar examples of player-adjustable functional attributes, but many, many games use them. First-person shooters typically give the player a choice of weapons, and when a player chooses a sniper rifle over a submachine gun, she is saying something important about the way she will play the game.

Cosmetic Attributes

Cosmetic attributes don't have any effect on the player's ability to perform actions or overcome challenges; that is, they're not part of the core mechanics of the game. Cosmetic attributes exist to let the player define himself in the game world, to bring his own personal style to the avatar. The paint color of a racing car has no effect on the car's performance characteristics, but the player is apt to enjoy the game more if he can

choose a color that he likes. One cosmetic attribute—shape—differentiates the tokens in *Monopoly*.

In multiplayer video games, cosmetic attributes can play a more important role because other players rely on visual appearances to make decisions. A few years ago, some bright player in a first-person shooter game got the idea to design an avatar that looked exactly like a crate. The other players assumed that they were looking at an actual crate, so they ignored it and then were surprised when they were shot by someone in a room that apparently contained only a crate. In online RPGs, players also use cosmetic attributes to identify themselves as members of a particular clan or group.

Cosmetic attributes make a game more fun at a low implementation cost. Because they don't affect the gameplay, they don't have to be tested and balanced as thoroughly as a functional attribute. Just be sure that your cosmetic attributes really *are* cosmetic. Avatar body size may sound like a cosmetic attribute, but if you later decide to take it into account when performing combat calculations (bigger people make bigger targets, for instance), then size becomes a functional attribute after all.

Typical cosmetic attributes for human characters include headgear, clothing, shoes, jewelry, hair color, eye color, skin color, and body type or size. Players typically customize paint color and decals or insignia of ve

cally customize paint color and decals or insignia of vehicles.

SHOULD SEX BE A FUNCTIONAL ATTRIBUTE OR A COSMETIC ATTRIBUTE?

Should the sex of an avatar have an effect on gameplay? Because men generally have more upper-body strength than women do and women are generally more dexterous than men are, you may be tempted to build these qualities directly into your core mechanics: to restrict the strength of female avatars and to restrict the dexterity of male ones.

However, unless you're making an extremely realistic simulation game, it's better not to associate bonuses or penalties with one sex or the other. First, although men as a group are *generally* stronger than women, it is not true that all men are stronger than all women. Women who exercise are often stronger than men who don't, and men who play the piano are usually more dexterous than women who don't. There are always exceptions and overlaps.

Second, video games provide a form of escapism. Players like to imagine themselves doing things that they can't do in the real world. If you impose real-world rules on what is meant to be their fantasy experience, you take some of the fun out of it.

It's better to allow the players to construct their avatars to suit their own styles of play rather than to

avatars to suit their own styles of play rather than to establish an arbitrary standard connected to sex. Leave sex as a cosmetic attribute and let the players adjust their functional attributes, such as strength and dexterity, independently.

Creative Play

Many games offer the player the chance to design or build something. In the *Caesar* series, it's a Roman city; in *Spore*, it's a creature; in *Minecraft*, it's a landscape. People enjoy designing and building things, and this kind of play is the main point of construction and management simulations.

If you offer creative play, you should allow players to save their creations at any time and reload them to continue working on them. You should also let players print out their creations, take screenshots, copy them to other players' machines, and upload them to websites. Sharing creations contributes to the fun.

Computerized creative play falls into two categories, constrained creative play and freeform creative play. A computerized game necessarily restricts creative play to whatever domain the game supports—painting, composing music, animation, and so on. In freeform creative play, few or no rules limit what the player can do within the confines of the game world, although

play remains constrained by the domain, the set of actions that the user interface offers, and the machine's physical limitations.

Constrained Creative Play

If the player may create only within artificial constraints imposed by the rules, her activity is called **constrained creative play**. Constraining creativity may sound undesirable, but it really just provides a structure for the player's creativity. This type of gameplay grows out of some familiar ideas: the expressive power offered by creativity tools; the growth in the number of actions available to a player as games progress; and the fact that players must overcome challenges in order to succeed. These may be combined in various ways, as the next two sections discuss.

Play Limited By An Economy

In *SimCity*, the player can't build a whole city immediately; it costs money to zone each empty plot of land, and he can use only the money he has available. As his city prospers, he earns more money and so can establish new neighborhoods. Once he gets enough money, new features such as stadiums and airports, which were too expensive in the early stages of the game, become available to him. As long as the player continues to produce economic growth, he can make his city ever larger and add more and more facilities.

Construction and management simulations routinely implement this system of structuring the player's creativity. The player must successfully manage an economy to construct larger creations and also to get additional creative power. This is a system closely related to that found in RPGs, in which players must gain experience to learn new magic spells, and to that found in strategy games, in which players must harvest resources to perform the research necessary to get better weapons. In those genres, the economy of the game limits the player's ability to have adventures and fight wars; in creativity games, the economy limits the player's ability to create. The primary challenge in such games is successful economic management, with creative power serving as the reward for success.

This system rewards skill, granting players more exciting and powerful tools once they master the tools they already have. Educational software also uses this mechanism.

Creating To Physical Standards

Another approach to constrained creative play gives players all the tools and resources they would like but requires them to construct an object that meets certain requirements, usually having to do with making the object perform a function. For example, *Spore* from

Maxis lets players design and build virtual creatures that then interact with creatures created by other players. The player gets a set of standard parts including arms and legs, eyes and ears, and weapons such as claws. Once the player constructs a creature, she turns it loose in the game world to fight or socialize with other creatures, and she can add additional features as she earns "DNA points." Although *Spore* can animate almost anything no matter how odd-looking, it does impose some physical standards: Every creature must have a backbone and be a land animal. The game offers no way to create a creature with an exoskeleton, like an insect, or no skeleton at all, like an octopus.

The *RollerCoaster Tycoon* series requires the player to construct roller coasters in a theme park. The roller coaster must be designed in such a way that it doesn't crash or make the (virtual) riders sick but is still exciting to ride. *RollerCoaster Tycoon* combines the challenge of meeting physical standards with an economic challenge: Each element of the theme park costs money, and the player must stay within a budget.

Whenever you require the player to build to a standard and test his construction, when he fails he needs to know why—otherwise he can't learn the principles upon which you based the standards. *RollerCoaster Tycoon 2* includes a feature to show the player how high and how steep the different segments of the

coaster are, so he can figure it out with a little experimentation. See **Figure 9.1**.

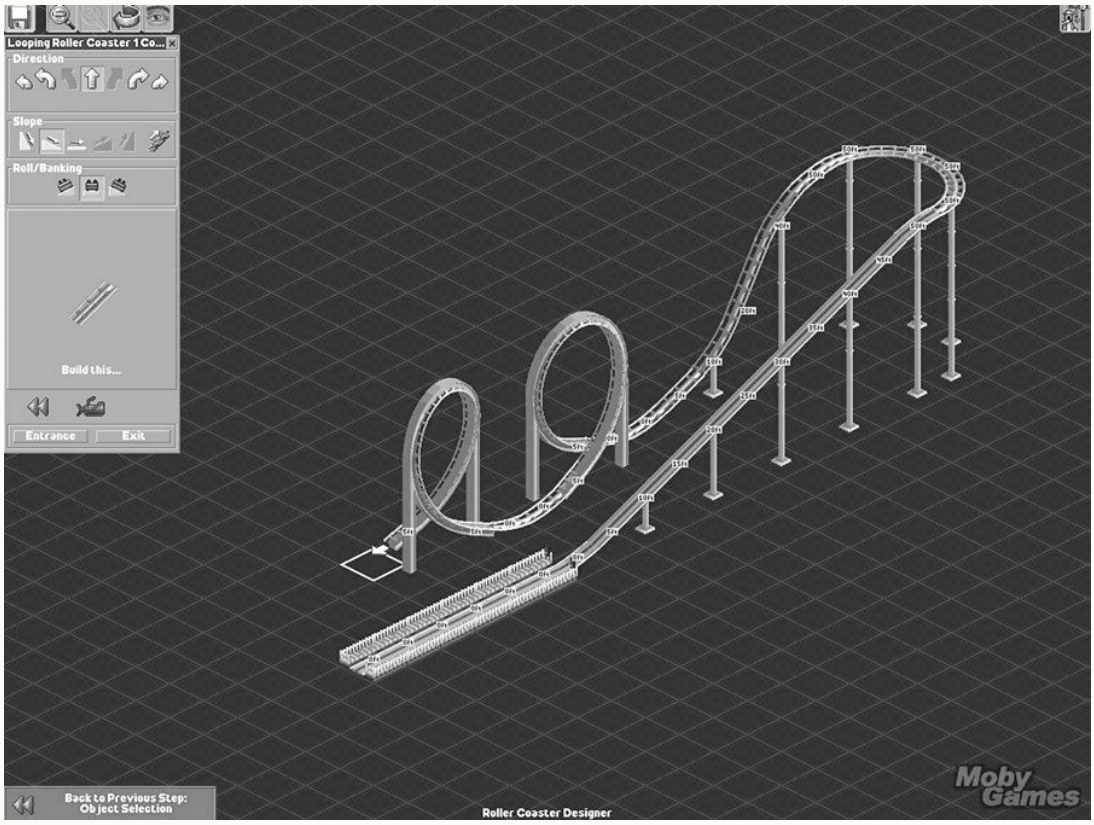

Figure 9.1 *RollerCoaster Tycoon 2*'s coaster-design screen.

The physical standards of a game do not need to conform to the conventional laws of physics, however. The physics of *Minecraft* are so rich that players can build incredible things in the game world, up to and including working computers. But *Minecraft*'s physical laws are quite different from those of the real world.

Creating To Aesthetic Standards

With an adequate physics simulation, any game can test a player's creativity against a physical standard. An architecture game can test a building both for structural integrity and also for usability—rooms with

no way to get into them are useless.

Aesthetics present a much larger problem, because they don't consist of a set of universal laws in the way that physics does. To test the aesthetic quality of a player's creations, you have to set some standards of your own. Consider some of the following options:

- **Test against a fixed (but hidden) set of rules that you establish.** A game about clothing design could include well-known rules about color combinations, not mixing stripes with polka dots, and using fabric textures that harmonize and complement each other. This system rewards players who know the rules and conform to them. It's good for teaching the basics, but it doesn't encourage brilliant but unconventional combinations. Nintendo's *Style Savvy* for the 3DS improves on this by giving the player non-player character (NPC) customers to make clothes for, but instead of using a single set of rules, each customer has personal preferences within certain broad categories (princess, punk, Goth, contemporary, and so on). The customer explains in general terms what she wants and the player proposes an outfit. If the outfit isn't acceptable to the customer, she gives hints about what she might like better. In effect, each customer is a puzzle to be solved. Better solutions (happier customers) produce repeat business, so the player comes to know the customers well.

NOTE

Ubisoft's *Imagine: Fashion Designer* implements the fixed-rule mechanic but does so badly. The player designs clothing for a NPC "client." Unfortunately, the game offers no way for the player to find out exactly what the client wants. If the client refuses to accept a design, he doesn't say why, so the player has to find out by trial and error—a serious design flaw.

- **Create a system of trends that the player can research.** If you want to make a game in which creative challenges change over time, such as the way fashion trends change from year to year, design a system in which the standard against which you measure the player's work fluctuates. Each attribute of the player's product could be tested against a trend with its own rate of variation, so—using the clothing example again—hemlines might move up and down over a 10-year period, and preferred fabrics might change from synthetics to natural fibers over a 20-year period. The periodicity should never be completely regular or predictable, however. The trend information should be hidden from the player but partially accessible via a research process. When I ran a series of game design workshops on this theme, participants suggested sev-

eral options for doing this research. The player, in the role of a fashion designer, could attend parties within the game and listen to computer-generated gossip, some of which would include clues about current trends; he could read automatically created fashion magazines and newspapers for clues; or he could even break into other fashion designers' workshops to find out about their works in progress.

- **Allow the public to vote online.** You can let the players upload their creations to a website and let the community vote on them. For example, *The Sims 2: H&M Fashion Runway* allows players to vote on clothing created in *The Sims 2*. This system relieves the computer of the responsibility for determining the aesthetic quality of the player's creations, but it significantly lengthens the time scale of the game—the player may have to wait hours or days until the votes come in, unless the game has a large player base. You will also have to build a secure system that rewards players for voting and prevents vote rigging.

Freeform Creative Play and Sandbox Mode

If a game lets the player use all the facilities that it offers without any restrictions on the amount of time or resources available (other than those imposed by technological limitations), then it supports **freeform creative play**. Many games that normally offer con-

strained creative play also include a special mode that removes ordinary constraints. This mode is called the **sandbox mode**. Sandbox mode lets the player do whatever she wants but usually doesn't offer the same rewards as the constrained mode—and may not offer any rewards at all. In this mode, the game resembles a tool more than a conventional video game.

The first *RollerCoaster Tycoon* game did not include a sandbox mode, and reviews of the game cited it as a deficiency. The developers added a sandbox mode to the later editions. Generally speaking, if you are making a game that offers creative play, you should include a sandbox mode if you can. Players enjoy them, and because sandbox modes don't interact much (if at all) with the game's core mechanics, they are fairly easy to tune.

One interesting, though very early, form of freeform creative play appeared in *Pinball Construction Set*. This game let players build and play their own virtual pinball machines. A more recent example, and certainly the most successful of all time, is the Creative mode of *Minecraft*. Players have an infinite number of blocks to build with and can fly through the air, so their construction activities are largely unconstrained. *LittleBigPlanet* allows players to create and share whole levels and a great deal of other content besides, and Microsoft's forthcoming *Project Spark* apparently

plans to let players build entire video games.

Creative play isn't restricted to sandbox modes within a complicated world, though. *Draw Something* (**Figure 9.2**) is an enormously popular two-player mobile game in which one player draws a picture that represents a word he has been given by the game, and the other player has to identify the word by looking at the picture. The player doing the drawing can draw anything he likes (although his choices of colors is initially limited).

Figure 9.2 *Draw Something*

Other Forms of Expression

This sections covers forms of creative play that don't involve making something substantive (like building a virtual city or drawing a picture). Two of the major forms of expression commonly found as gameplay are role-playing and storytelling.

Role-Playing

The term *role-playing* is rather overloaded in gaming, because it can mean everything from genuine acting in an improvisational drama to simply playing a video game with *Dungeons & Dragons*-style rules. For the purposes of this chapter, role-playing means controlling an avatar character in such a way that it exhibits some of the behaviors or personality characteristics of real people.

To offer a player opportunities to act as a character, you have to first design the avatar that she will play with and decide to what extent the player can modify it. We already covered that in the section "**Self-Defining Play**" earlier in this chapter, and there's a great deal more in **Chapter 10**, "**Character Development**." In the process of doing this, you also have to decide what means you will give the player to

have to decide what means you will give the player to make the avatar seem like a person and not just a puppet.

Actors portray characters through their voice, movements, and facial expressions. In presentational entertainment, these details are normally worked out in advance with a director, but in video games they are improvised, and of course everything has to be mediated through the game's user interface. Here are some features you might consider implementing:

- **Mood indicators.** At the moment most games don't give the player any way to change the facial expression of an avatar. A mood indicator is an icon that the player can select that allows her to indicate her avatar's state of mind in a crude sort of way. The game engine can detect the mood and have NPCs react appropriately; in a multiplayer game, other players can see it.

- **Emotes.** Massively multiplayer online role-playing games (MMORPGs) usually offer *emotes*, special animations the player can trigger to indicate an emotional state, but of course these don't have the subtlety of real body language. The dances available in *World of Warcraft* are one example. Emotes are usually used to entertain other players rather than to influence the game engine.

Dialogue choices. One of the most common ways of letting players enact a character is to offer them several different ways to say the same thing in a conversation with an NPC. You can give the player aggressive, courteous, or humorous ways of replying to another character. The player's choice affects the character's attitude and may open up, or close off, different lines of conversation; it may also influence the game engine by changing the NPC's attitude toward the player.

These are just a few suggestions for ways to let the player express herself beyond choosing physical actions in the game. Role-playing allows the player to feel like she is someone else, a powerful kind of entertainment and personal self-expression.

Storytelling Play

Some players enjoy creating stories of their own, using features provided by a game, which they can then distribute online for others to read. *The Movies*, by Lionhead Software, stands as the most ambitious project of this kind to date. The purpose of the game is to let people make their own movies and share them online. *The Movies* also allows players to export movies so they can edit their films using external tools such as Adobe Premiere Pro. *The Movies* offers more expressive power than any other storytelling game yet made, but it does require a lot of effort from the players. If

you want to make a game with similar features, you will have to work with the programmers to design a system that allows players to set up cameras in the game world, record the images and sounds generated by the game engine, and edit them.

The independent designer Jason Rohrer used another approach in his game *Sleep Is Death.* This game is designed for two players, the storyteller and the story-player. The teller creates a storylike experience in real time for the player and has to react to the player's actions. It has something in common with tabletop role-playing in that respect, but it avoids all the number crunching and the emphasis on quests and magical items. The game is small and the graphics are retro, but *Sleep Is Death* got very good reviews from a number of game publications and is well worth your time to investigate.

But you don't have to build a complex storytelling mechanism for someone to play with. *The Sims* proved to be a huge success with a much simpler system: Players can create characters and construct houses for them to live in, and then initiate events by giving commands to the characters. *The Sims* also lets players capture screen shots from the game, put captions under them, organize them into storyboards, and upload them to a website for others to see. Telling stories this way requires much less complex software than *The*

Movies uses, and the players don't have to know how to edit video.

An even easier solution involves generating a log of the player's activities in text form. She can then edit this log any way she likes, turning her raw game actions and dialogue into narrative form.

Game Modifications

To give your players the utmost creative freedom with your game, you can permit them to modify the game itself—to redesign it themselves. Game modifications, or **mods**, are extremely popular with the dedicated gamer community and almost an obligatory feature of any large multiplayer networked game (though not persistent worlds, because in those games the company must maintain control of the game world).

Providing the player with mod-building tools also makes good business sense. Your game's original content can keep people interested for only a certain amount of time, but if people can build mods that use your game engine (as they can with the *Unreal* and *Half-Life 2* engines), people will continue to buy your game just to be able to play the mods.

Allowing for mods is more of a programmer's problem than a designer's problem, so this book does not discuss them in much detail.

DANGERS OF ALLOWING MODS

Mods bring with them certain risks. When you allow players to modify your game, you risk the possibility that they will create a mod that includes material you would never use yourself: pornographic or racist content, for example. You could find people distributing a highly offensive variant of your game, but one that still displays your company's name and logo when it starts up. The public might be sophisticated enough to realize that a game designer shouldn't be held responsible for the contents of homemade mods, but then again it might not be. The general public doesn't know much about video game development, and the politicians who seek to regulate video games know even less.

There are also questions about the intellectual property ownership of game mods. The game's publisher will probably assume, not unreasonably, that it owns the game and any modification of the game as well; it certainly can't afford to let control of the game pass to someone else. On the other hand, players who mod games extensively may feel as if they are entitled to own the fruits of their own labor. This isn't really a design issue, but if you're planning to allow mods to your game, you may want to discuss it with the lawyers who draw up the game's licensing terms.

Level Editors

A level editor allows players to construct their own levels for a game. Some level editors permit players to define only a new landscape; others allow them to define new characters as well; and a few go so far as to permit rebuilding the entire game. Generally, however, a good level editor lets the player construct a completely new landscape, place challenges in it, and write scripts that the game engine can operate. If you work on a large game for commercial sale, your team will almost certainly include tools programmers who will build a level editor for the level designers to use. To make the level editor available to the players, rather than useful only as an in-house tool, you must make sure it is as robust and well-designed as the game software itself. Two superb level editors that you should study are the 2D *StarCraft* Campaign Editor, which is included with *StarCraft,* and the Hammer 3D editor that comes with *Half-Life 2*. *Minecraft*'s world-building tools are worth a look, too. For further reading about level editors and other design tools, see Richard Rouse's article "Designing Design Tools" in the *Gamasutra* developers' webzine (**Rouse, 2000**).

Bots

A **bot** is an artificially intelligent opponent that the player can program for himself. (*Bot* also has a sec-

ondary meaning: a program that help players cheat at multiplayer networked games. This section is about the other kind.) By building bots, players can create tougher and smarter opponents than those that normally ship with the game (usually a first-person shooter). Some players use bots as sparring partners for practice before playing against real people in online tournaments. *Quake III Arena* contains a great deal of support for bots, and a number of third-party tools have been built to help players create them.

Summary

Players love to express themselves and to build things. This chapter looked at options for self-expression through avatar selection, customization, and creation. It also examined both freeform and constrained creative play, and discussed some of the different kinds of constraints that you may impose on a player's creativity to produce challenges. We noted some options for permitting storytelling play and ended with a brief discussion about allowing players to modify your game. With these tools in hand, you should be able to add support for creative play to your game.

Design Practice Exercises

1. Using a game that you are familiar with and that permits avatar customization or construction (or one

that your instructor assigns), identify and document those functional and cosmetic attributes of the avatar that the player may modify. In the case of the functional attributes, indicate how they affect the gameplay.

2. Think of an idea for a game that permits the player to construct something you have not seen in commercial games (no cities, buildings, or vehicles). Assume that you will constrain the player's abilities via an economy but allow him to earn new tools and features for his construction over time. Design a set of elementary parts from which the item may be constructed, and specify a price for each part. Include a number of upgrades—more expensive parts that replace cheaper versions. Write a short paper explaining the domain in which the player will be creating, and supply your list of parts, giving them in the order in which the player will earn the ability to buy them (cheapest to most expensive). Also indicate in a general way how the player can use the item he constructs to earn money. Your instructor will inform you of the scope of the exercise.

3. Choose a domain in which the player must construct something to meet an aesthetic standard for which a known set of aesthetic rules exists in the real world, such as architecture, clothing design, music, interior decoration, landscaping, or a domain that your instructor assigns. Research your chosen domain to learn its

aesthetic rules. (Because many such rules change over time, you may choose a period from history if you can find adequate documentation.) Be careful not to confuse rules about usability with those about aesthetics. Write a short paper explaining your domain, including the range of choices that the player may make in constructing something, and document the aesthetic rules. Provide references to your sources.

Design Practice Questions

1. What features do you want to include to allow the player to define herself in the game: avatar selection, customization, or creation from the ground up? What attributes can she change, if any?

2. How will you make clear to the player the possible consequences of his avatar customization decisions so that he can make informed choices? Where will you provide this information?

3. If you offer creative play, what will its domain be? What limitations will the machine impose?

4. Do you plan to offer constrained creative play? If so, what will be the constraining factor or factors—economics, physics, or aesthetics? Will the game provide a growth path to gradually free the player of constraints? In the case of aesthetics, how will you implement an aesthetic judgment mechanism, and how will

the results of that judgment become clear to the player? Will aesthetically successful (however you determine that) creations earn more money, win prizes, produce points, or gain some other reward?

5. Will you offer freeform creative play? If so, will it be part of ordinary play, or will it be a separate sandbox mode? If you do offer freeform creative play, can the player's creations affect the gameplay?

6. Does your game include features for role-playing or storytelling play? What will they be? How can you seamlessly integrate such features into the rest of the game?

7. Do you plan to allow mods? What will you let players modify? Can they create new levels, bots, or narrative material? What tools will you want to ship to support these activities?

8. How will you create a sense of community between the players and allow them to share their creations with others?