Designing Game Mechanics

Game mechanics are the rules, processes, and data at the heart of a game. They define how play progresses, what happens when, and what conditions determine victory or defeat. In this chapter, we’ll introduce five types of game mechanics and show how they’re used in some of the more common video game genres. We’ll also discuss at what stage during the game design process you should design and prototype mechanics, and we’ll describe three kinds of prototyping, addressing the strengths and weaknesses of each. By the end of the chapter, you should have a clear understanding of what game mechanics are for and how to think about designing them.

Rules Define Games

There are many different definitions of what a game is, but most of them agree that rules are an essential feature of games. For example, in Fundamentals of Game Design, Ernest Adams defines games as follows:

A game is a type of play activity, conducted in the context of a pretended reality, in which the participants try to achieve at least one arbitrary, nontrivial goal by acting in accordance with rules.

In Rules of Play, Katie Salen and Eric Zimmerman write the following:

A game is a system in which players engage in artificial conflict, defined by rules, that results in a quantifiable outcome.

In Half-Real, Jesper Juul says this:

A game is a rule-based system with a variable and quantifiable outcome, where different outcomes are assigned different values, the player exerts effort in order to influence the outcome, the player feels emotionally attached to the outcome, and the consequences of the activity are negotiable.

(Emphasis added in all cases.) We don’t intend to compare these different definitions or to claim that one of them is the best. The point is that they all refer to rules. In games, rules determine what players can do and how the game will react.
GAMES AS STATE MACHINES

Many games, and game components, can be understood as state machines (see, for example, Järvinen 2003; Grünvogel 2005; Björk & Holopainen 2005). A state machine is a hypothetical machine that can exist in a certain number of different states, each state having rules that control the machine’s transition from that state into other states. Think of a DVD player: When a DVD is playing, the machine is in the play state. Pressing the pause button changes it to the paused state, while pressing the stop button causes it to return to the DVD menu—a different state. Pressing the play button does nothing at all—the player remains in the play state.

A game begins in an initial state, and the actions of the player (and often the mechanics, too) bring about new states until an end state is reached. In the case of many single-player video games, the player either wins, loses, or quits. The game’s state usually reflects the player’s location; the location of other players, allies, and enemies; and the current distribution of vital game resources. By looking at games as state machines, researchers can determine which rules cause the game to progress from one state to another. Several successful methods allow computer scientists to design, model, and implement state machines with a limited (finite) number of states. However, in contrast to DVD players, games can exist in a vast number of states, far too many to document.

Finite state machines are sometimes used in practice to define the behavior of simple artificially intelligent non-player characters. Units in a war game often have states such as attacking, defending, and patrolling. However, because this is not a book about artificial intelligence, we won’t be addressing those techniques here. State machine theory is not useful for studying the kinds of complex mechanics that this book is about.

Games Are Unpredictable

A game’s outcome should not be clear from the start: To a certain extent, games should be unpredictable. A game that is predictable is usually not much fun. A simple way of creating unpredictable outcomes is to include an element of chance, such as a throw of the dice or the twirl of a spinner in a board game. Short games such as blackjack or Klondike (the most familiar form of solitaire played with cards) depend almost entirely on chance. In longer games, however, players want their skills and their strategic decisions to make more of a difference. When players feel that their decisions and game-playing skills do not matter, they quickly become frustrated. Pure games of chance have their place in a casino, but for most other games, skill should also contribute to victory. The longer the game is, the more true this is.

In addition to chance, there are two other, and more sophisticated, ways to make games unpredictable: choices made by players and complex gameplay created by the game’s rules.

**NOTE** In games and simulations, processes that include elements of chance (such as moving a certain distance based upon a die roll) are called **stochastic processes**. Processes that do not include chance, and whose outcome can be determined from their initial state, are called **deterministic processes**.
A simple game such as rock-paper-scissors (or roshambo/rochambeau) is unpredictable because its outcome depends on the decisions made by the players. The rules do not favor one choice or another; they do not suggest a particular strategy. Trying to second-guess or influence the choice of your opponent might involve empathy or reverse psychology, but it remains largely outside the individual player's control. The classic board game Diplomacy uses a similar mechanism. In this game, players control only a handful of armies and fleets. Victory in battle simply goes to the side that committed the largest number of units to a battle. However, because all the players write down their moves secretly and resolve their turns simultaneously, the players must use their social skills to find out where their opponent will strike and to convince their allies to support their offensive and defensive maneuvers.

When the rules of a game are complex, they can also make a game unpredictable, at least to human beings. Complex systems usually have many interacting parts. The behavior of individual parts might be easy to understand; their rules might be simple. However, the behavior of all the parts combined can be quite surprising and difficult to foresee. The game of chess is a classic example of this effect. The movement rules of the 16 chess pieces are simple, but those simple rules produce a game of great complexity. Whole libraries have been written about chess strategies. Expert players try to lure opponents into traps involving many pieces that might take multiple turns to execute. In this type of game, the ability to read a game's current state and understand its strategic complexities is the most important game-playing skill.

Most games mix these three sources of unpredictability. They include an element of chance, player choices, and complex rules. Different players prefer different combinations of these techniques. Some like games that involve many random factors, while others prefer games where complexity and strategy are key. Of these three options, chance is the easiest to implement but not always the best source of unpredictability. On the other hand, complex rule systems that offer many player choices are difficult to design well. This book will help you with that task. We devote most of the chapters to designing rule systems that create, among other things, interesting choices for players. In Chapter 6, “Common Mechanisms,” we cover random number generators (the software equivalent of dice) and discuss them at several other points as well, but we feel that chance serves a supporting, rather than a central, role in mechanics design.

From Rules to Mechanics

The video game design community usually prefers the term game mechanics to game rules because rules are considered printed instructions that the player is aware of, while the mechanics of video games are hidden from the player, that is, implemented in software for which the player is given no direct user interface. Video game players don’t have to know what the game’s rules are when they begin; unlike board and card games, the video game teaches them as they play. Rules and mechanics are related concepts, but mechanics are more detailed and concrete. For example, the rules of Monopoly consist of only a few pages, but the mechanics of Monopoly include the
prices of all the properties and the text of all the Chance and Community Chest cards—in other words, everything that affects the operation of the game. Mechanics need to be detailed enough for game programmers to turn them into code without confusion; mechanics specify all the required details.

The term *core mechanics* is often used to indicate mechanics that are the most influential, affecting many aspects of a game and interacting with mechanics of lesser importance, such as those that control only a single game element. For example, the mechanics that implement gravity in a platform game are core mechanics; they affect almost all moving objects in the game and interact with mechanics for jumping or the mechanics that control damage to falling characters. On the other hand, a mechanic that merely enables players to move items around in their inventories would not be a core mechanic. The artificial intelligence routines that control the behavior of autonomous non-player characters are also considered not core mechanics.

In video games, the core mechanics are mostly hidden, but players will learn to understand them while playing. Expert players will deduce what the core mechanics must be by watching the behavior of the game many times; they will learn how to use a game’s core mechanics to their advantage. The distinction between core mechanics and non-core mechanics is not clear-cut; even for the same game, interpretation of what is core and what is not can vary between designers or even between different contexts within the game.

**MECHANIC OR MECHANISM**

Game designers are perfectly comfortable talking about a *game mechanic* in the singular form. They don’t mean a person who repairs game engines! Instead, they are referring to a single game *mechanism* that governs a certain game element. In this book, we prefer to use *mechanism* as the singular form, indicating a single set of game rules associated with a single game element or interaction. One such mechanism might include several rules. For example, the mechanic of a moving platform in a side-scrolling platform game might include the speed of the platform’s movement, the fact that creatures can stand on it, the fact that they are moved along with it when they do, and the fact that the platform’s velocity is reversed when it bounces into other game elements or perhaps after it has traveled a particular distance.

**Mechanics Are Media-Independent**

The mechanics of a game can be implemented through many different media. In the case of a board game, the mechanics are implemented through the medium of the game’s paraphernalia: board, counters, playing pieces, spinners, and so on. The same game can also be published as a video game. In that case, the same mechanics will be implemented in software, which is a different medium.
Because mechanics are media-independent, most game scholars do not distinguish between video games, board games, and even physical games. The relationships between different entities in the game is much the same whether implemented on a board, with pieces you move by hand, or on a computer screen, with images moved for you by software. Not only can the same game be played in different media, sometimes a single game can use more than one medium. Today more and more games are hybrids: board games that include simple computers, or physical games facilitated by clever devices hooked up to remote computers.

In addition, the media independence of game mechanics allows designers to create mechanics for one game but then implement that game in several different media; this cuts down on development time, since the design work is done only once.

**HYBRID GAME EXAMPLE**

The game *Johann Sebastian Joust*, developed by the Copenhagen Games Collective, is an excellent example of hybrid game design. The game uses no screen, only speakers, and takes place in an open area in which each player holds a PlayStation Move controller (Figure 1.1). Players who move their controller too fast are eliminated from the game, so players try to eliminate each other by shoving other players’ controllers, while maneuvering carefully to protect their own controllers, all in slow motion. Occasionally the tempo of the background music speeds up, indicating the speed at which the player can move safely. *Johann Sebastian Joust* is a hybrid multiplayer game that blends physical performance with simple computer-implemented mechanics to create a satisfying player experience.

![Figure 1.1](image.png)

*Figure 1.1* *Johann Sebastian Joust* in full swing.

Image courtesy of Johan Bichel Lindegaard under a Creative Commons (CC BY 3.0) license.
Using different media can help when creating prototypes. Programming software usually takes much more work than simply writing down mechanics as rules for a board game. If the same game can be played in a board game or physical game form, it’s a good idea to try the rules/mechanics in one of those forms before going to the trouble and expense of implementing them on a computer. As you’ll see in the next section, efficient prototyping techniques are important tools in the game designer’s toolbox.

Five Different Types of Mechanics

The term mechanics has come to indicate many different types of underlying relationships between entities in games. Here are five different types of mechanics that you might expect to find in a game:

- **Physics.** Game mechanics sometimes define physics—the science of motion and force—in the game world (which can be different from the physics of the real world). In games, characters commonly move from place to place, jump up and down, or drive vehicles. Computing a game element’s position, the direction in which it is moving, and whether it intersects or collides with other elements makes up the bulk of the calculations in many games. Physics plays a large role in many modern games, from ultrarealistic first-person shooters to the popular physics-puzzle games such as *Angry Birds*. The implementation is seldom strict; however, games with so-called *cartoon physics* use a modified version of Newtonian mechanics so that characters can do non-Newtonian things such as change direction while in midair. (We also consider such things as timing and rhythm challenges to be part of a game’s physics.)

- **Internal economy.** The mechanics of transactions involving game elements that are collected, consumed, and traded constitute a game’s *internal economy*. The internal economy of a game typically encompasses items easily identified as *resources*: money, energy, ammunition, and so on. However, a game’s economy is not limited to concrete, tangible items; it can also include abstractions such as health, popularity, and magical power. In any Zelda game, Link’s hearts—a visible measure of his life energy—are part of the internal economy. Skill points and other quantified abilities in many role-playing games also qualify; these games have very complex internal economies.

- **Progression mechanisms.** In many games, level design dictates how a player can move through the game world. Traditionally, the player’s avatar needs to get to a particular place to rescue someone or to defeat the main evil-doer and complete the level. In this type of game, the progress of the player is tightly controlled by a number of mechanisms that block or unlock access to certain areas. Levers, switches, and magical swords that allow you to destroy certain doors are typical examples of such progression mechanisms.
Tactical maneuvering. Games can have mechanics that deal with the placement of game units on a map for offensive or defensive advantages. Tactical maneuvering is critical in most strategy games but also features in some role-playing and simulation games. The mechanics that govern tactical maneuvering typically specify what strategic advantages each type of unit may gain from being in each possible location. Many games restrict the location of units to discrete tiles, as is the case for a classic board game like chess. Even modern strategy games played on the computer often implement tiles, although they do a good job of hiding them behind a detailed visual layer. Tactical maneuvering appears in many board games such as chess and Go but also in computer strategy games such as StarCraft or Command & Conquer: Red Alert.

Social interaction. Until recently, most video games did not govern social interaction among the players, apart from prohibiting collusion or requiring that players keep certain knowledge secret. Now, however, many online games include mechanics that reward giving gifts, inviting new friends to join, and participating in other social interactions. In addition, role-playing games might have rules that govern the play-acting of a character, and a strategy game might include rules that govern the forming and breaking of alliances between players. Board games and folk games played by children have a longer history of game mechanisms that guide the interactions among players.

Mechanics and Game Genres

The game industry categorizes games into genres based on the type of gameplay the game offers. Some games derive their gameplay mostly from their economy, others from physics, level progression, tactical maneuvering, or social dynamics. Because the gameplay is generated by the mechanics, it follows that the genre of a game has a significant effect on the kinds of rules it implements. Table 1.1 shows a typical game classification scheme and how these genres and their associated gameplay relate to different types of mechanics. The game genres in the table are taken from Fundamentals of Game Design, Second Edition and correlate to the five different types of game rules or structures. The thickness of the outlines indicates relative importance of those types of rules for most games in that genre.
<table>
<thead>
<tr>
<th>Game Genres</th>
<th>Physics</th>
<th>Economy</th>
<th>Progression</th>
<th>Tactical Manoeuvring</th>
<th>Social Interaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>Detailed physics for movement, shooting, jumping, etc.</td>
<td>Power-ups, collectables, points and lives</td>
<td>Predesigned levels with increasingly difficult tasks, storyline to set player goals</td>
<td>Positioning of units to gain offensive or defensive advantages</td>
<td>Coordinated actions, alliances and competition between players</td>
</tr>
<tr>
<td>Strategy</td>
<td>Simple physics for movement and fighting</td>
<td>Unit building, resource harvesting, unit upgrading, risking units in combat</td>
<td>Scenarios to provide new sets of challenges</td>
<td>Story-line and quests to give player a purpose and goal</td>
<td>Party tactics</td>
</tr>
<tr>
<td>Role-Playing</td>
<td>Relatively simple physics to resolve movement and conflict, often turn based</td>
<td>Equipment and experience to customize a character or party</td>
<td>Party tactics for managing the game</td>
<td>Player acting</td>
<td></td>
</tr>
<tr>
<td>Sports</td>
<td>Detailed simulation</td>
<td>Team management</td>
<td>Seasone, competitions, tournaments</td>
<td>Team tactics</td>
<td>Play acting</td>
</tr>
<tr>
<td>Vehicle Simulation</td>
<td>Detailed simulation</td>
<td>Vehicle tuning between missions</td>
<td>Missions, races, challenges, competitions, tournaments</td>
<td>Managing of resources, economy building</td>
<td>Coordinated actions, alliances and competition between players</td>
</tr>
<tr>
<td>Management Simulation</td>
<td>Managing of resources, economy building</td>
<td>Scenarios to provide new sets of challenges</td>
<td>Managing of resources, economy building</td>
<td>Coordinated actions, alliances and competition between players</td>
<td></td>
</tr>
<tr>
<td>Adventure</td>
<td>Managing a player's inventory</td>
<td>Story to drive game,locks and key to control player progress</td>
<td>Managing a player's inventory</td>
<td>Story to drive game, locks and key to control player progress</td>
<td></td>
</tr>
<tr>
<td>Puzzle</td>
<td>Simple, often non-realistic and discrete, physics generate challenges</td>
<td>Resource harvesting and unit building, resources spend on personalized content</td>
<td>Short levels providing increasingly more difficult challenges</td>
<td>Quests and challenges to give player a purpose and a goal</td>
<td>Players exchange in-game resources, mechanics encourage player cooperation or conflict</td>
</tr>
<tr>
<td>Social Games</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Discrete Mechanics vs. Continuous Mechanics

We've listed five types of mechanics, but there's another important distinction to be made: Mechanics can be \textit{discrete} or \textit{continuous}. Modern games tend to simulate physics (including timing and rhythm) with precise mechanics that create a smooth, continuous flow of play. A game object might be positioned half a pixel more to the left or right, and this can have a huge effect on the result of a jump. For maximum accuracy, physical behaviors need to be computed with high-precision fractional values; this is what we mean by \textit{continuous mechanics}. In contrast, the rules of an internal economy tend to be discrete and represented with integer (whole-number) values. In an internal economy, game elements and actions often belong to a finite set that does not allow any gradual transitions: In a game you usually cannot pick up half a power-up. These are \textit{discrete mechanics}. This difference between game physics and game economies affects a game's level of dependence on its medium, the nature of the player interaction, and even the designer's opportunities for innovation.

Understanding the Mechanics of Physics

Accurate physics computations, especially in real time, require a lot of high-speed mathematical operations. This tends to mean that physics-based games must be implemented on a computer. Creating a board game for \textit{Super Mario Bros.}, in which the gameplay requires moving and jumping from platform to platform, would be difficult. In platform games, physical dexterity matters, just as it does in playing real-life football; those skills would be lost in a board game. \textit{Super Mario Bros.} is probably better mediated as a physical course testing players' real running and jumping abilities. The point is, a rule that states that you can jump twice as high after picking up a certain item can be easily translated between different media, but actually implementing that jump cannot. The continuous, physical mechanics of a game need computing power more than the discrete rules that govern a game's economy.

Interestingly, when you look back at the early history of platform games and other early arcade games, the physics calculations were more discrete than they are today. The moves in \textit{Donkey Kong} were much less continuous than they were in \textit{Super Mario Bros}. In \textit{Boulder Dash}, gravity is simulated by moving boulders down at a constant speed of one tile every frame. It might play slowly, but it is possible to create a board game for \textit{Boulder Dash}. In those days, the rules that created the game's physical mechanics were not that different from other types of game rules. The early game computers did not have any floating-point arithmetic instructions, so the game physics had to be simple. But times have changed. Today the physics in a platform game have grown so accurate and detailed that they have become impossible, or at least inconvenient, to represent with a board game.
Mixing Physical Mechanics with Strategic Gameplay

With discrete rules, it is possible to look ahead, to plan moves, and to create and execute complex strategies. Although this isn’t always easy, it is possible, and many players enjoy doing it. Players interact with discrete mechanics on a mental, strategic level. Once players grasp the physics of a game, they can intuitively predict movements and results, but with less certainty. Skill and dexterity become a more important aspect of the interaction. This difference is crucial for gameplay and can be seen in a comparison between *Angry Birds* and *World of Goo*, two games that mix physical mechanics with strategic gameplay.

In *Angry Birds*, players shoot birds from a catapult at defensive structures protecting pigs (**Figure 1.2**). The catapult is operated with a touch device, and because the physical simulation is so precise, a small difference in launch speed or angle can have a completely different effect on the structural damage the player causes. Catapulting the birds is mostly a matter of physical skill. The strategy in *Angry Birds* involves those aspects of the game that are governed by discrete rules. Players have to plan to attack the pigs’ defenses most effectively using the number and types of birds available in the level. This requires identifying weak spots and formulating a plan of attack, but the execution itself is based on hand-eye coordination, and the effects can never be foreseen in great detail.

**FIGURE 1.2**
*Angry Birds*
Compare the mix of strategy and skill in *Angry Birds* with a similar mix in *World of Goo* (Figure 1.3). In *World of Goo*, players build constructions from a limited supply of goo balls. The game includes a detailed physical simulation that controls the player-built constructions. Physical phenomena such as gravity, momentum, and center of mass play an important role in the mechanics of the game. Indeed, players can form an intuitive understanding of these notions from playing *World of Goo*. But more importantly, players learn how to manage their most important (and discrete) resource, goo balls, and use them to build successful constructions. The difference between *Angry Birds* and *World of Goo* becomes very clear when you consider the respective effects of both games’ continuous, pixel-precise physics. In *Angry Birds*, the difference of a single pixel can translate into a critical hit or complete miss. *World of Goo* is more forgiving. In that game, releasing a goo ball a little more to the left or right usually does not matter, because the resulting construction is the same, and spring forces push the ball into the same place. The game even shows what connections will be made before the player releases a ball (as shown in Figure 1.3). You can see that the gameplay is more strategic in *World of Goo* than it is in *Angry Birds*. *World of Goo* depends more on its discrete mechanics than on its continuous mechanics to create the player’s experience.
CHAPTER 4

Internal Economy

In Chapter 1, we listed five types of mechanics that you might find in a game: physics, internal economy, progression mechanisms, tactical maneuvering, and social interaction. In this chapter, we’ll focus on the internal economy.

In real life, an economy is a system in which resources are produced, consumed, and exchanged in quantifiable amounts. Many games also include an economy, consisting of the resources the game manipulates and the rules about how they are produced and consumed. However, in games, the internal economy can include all sorts of resources that are not part of a real-life economy. In games, things like health, experience, and skill can be part of the economy just as easily as money, goods, and services. You might not have money in Doom, but you do have weapons, ammunition, health, and armor points. In the board game Risk, your armies are a vital resource that you must use and risk in a gambit to conquer countries. In Mario Galaxy, you collect stars and power-ups to gain extra lives and to get ahead in the game. Almost all genres of games have an internal economy (see Table 1.1 in Chapter 1 for some more examples), even if it does not resemble a real-world economy.

To understand a game’s gameplay, it is essential to understand its economy. The economies of some games are small and simple, but no matter how big or small the economy is, creating it is an important design task. It is also one of the few tasks that belongs exclusively to the designer and no one else. To get game physics right, you need to work closely with the programmers; to get a level right, you need to work closely with the story writers and level designers; but you must design the economy on your own. This is the core of the game designer’s trade: You craft mechanics to create a game system that is fun and challenging to interact with.

In Fundamentals of Game Design, Ernest Adams discussed the internal economy of games. The discussion in this book repeats some of those points and expands the notion of internal economy.

Elements of Internal Economies

In this section, we briefly introduce the basic elements of game economies: resources, entities, and the four mechanics that allow the resources to be produced, exchanged, and consumed. This is only a summary; if you need a more in-depth introduction, please see Chapter 10, “Core Mechanics,” in Fundamentals of Game Design.
Resources

All economies revolve around the flow of resources. Resources refer to any concept that can be measured numerically. Almost anything in a game can function as a resource: money, energy, time, or units under the player’s control all are examples of resources, as are items, power-ups, and enemies that oppose the player. Anything the player can produce, gather, collect, or destroy is probably a resource of some sort, but not all resources are under the player’s control. Time is a resource that normally disappears by itself, and the player usually cannot change that. Speed is also a resource, although it is generally used as part of a physics engine rather than part of an internal economy. However, not everything in a game is a resource: platforms, walls, and any other type of inactive or fixed-level features are not resources.

Resources can be tangible or intangible. *Tangible resources* have physical properties in the game world. They exist in a particular location and often have to be moved somewhere else. Examples include items the avatar carries around in an inventory or trees that can be harvested in *Warcraft*. In a strategy game, the player’s units are also tangible resources that must be directed through the world.

*Intangible resources* have no physical properties in the game world—they do not occupy space or exist in a particular location. For example, once the trees in *Warcraft* have been harvested, they are changed into lumber, which is intangible. Lumber is just a number—it doesn’t exist in a location. The player doesn’t need to physically direct lumber to a site to build a new building. Simply having the right amount of lumber is enough to start building, even if the building is constructed far away from the location where the lumber was harvested. *Warcraft*’s handling of trees and lumber is a good example of how games can switch between tangible and intangible treatments of resources. Medical kits (tangible) and health points (intangible) in shooter games are another example.

Sometimes it is useful to identify resources as either *abstract* or *concrete*. Abstract resources do not really exist in the game but are computed from the current state of the game. For example, in chess you might sacrifice a piece to gain a strategic advantage over your opponent. In this case, “strategic advantage” can be treated as an abstract resource. (Abstract resources are intangible too—obviously, “strategic advantage” is not a thing stored in a location.) Similarly, the altitude of your avatar or units can be advantageous in a platform or strategy game; in this case, it might make sense to treat altitude as a resource, if only as a way of factoring it into the equation for the strategic value of capturing particular positions. The game normally does not explicitly tell the player about abstract resources; they are used only for internal computation.
Note that in video games some resources that might appear to be abstract are in fact quite concrete. For example, experience points are not an abstract resource in a role-playing game. Instead, they are an intangible, but real, commodity that must be earned and (sometimes) spent like money. Happiness and reputation are two more resources used by many games that, although they are intangible, are nevertheless concrete parts of the game.

To design a game’s internal economy or to study the internal economy of an existing game, it is most useful to start identifying the main resources and only then describe the mechanisms that govern the relationships between them and how they are produced or consumed.

**Entities**

Specific quantities of a resource are stored in entities. (If you are a programmer, an entity is essentially a variable.) A resource is a general concept, but an entity stores a specific amount of a resource. An entity named “Timer,” for example, stores the resource time—probably the number of seconds remaining before the end of the game. In Monopoly, each player has an entity that stores available cash resources. As the player buys and sells, pays rent and fines, and so on, the amount of cash in the entity changes. When a player pays rent to another player, cash flows from the first player’s entity to the second player’s entity.

Entities that store one value are called simple entities. Compound entities are groups of related simple entities, so a compound entity can contain more than one value. For example, a unit in a strategy game normally includes many simple entities that describe its health, damage capability, maximum speed, and so on. Collectively, these make up a compound entity, and the simple entities that make it up are known as its attributes. Thus, a unit’s health is an attribute of the unit.

**Four Economic Functions**

Economies commonly include four functions that affect resources and move them around. These are mechanics called sources, drains, converters, and traders. We describe them here. Again, this is a summary; for further details, see Chapter 10 of Fundamentals of Game Design.

- **Sources** are mechanics that create new resources out of nothing. At a certain time, or upon certain conditions, a source will generate a new resource and store it in an entity somewhere. Sources may be triggered by events in the game, or they may operate continuously, producing resources at a certain production rate. They may also be switched on and off. In simulation games, money is often generated by a source at intervals, with the amount of money created proportional to the population. As another example, some games that involve combat automatically regenerate health over time.
■ **Drains** are the opposite of sources: They take resources out of the game, reducing the amount stored in an entity and removing them permanently. In simulation games in which it is necessary to feed a population, the food is drained at a rate proportional to the population. It does not go anywhere or turn into anything else; it simply disappears. In shooter games, ammunition is drained by firing weapons.

■ **Converters** turn resources of one kind into another. As we mentioned, in *Warcraft*, trees (a tangible resource) turn into lumber (an intangible one) when the trees are harvested. The act of harvesting is a converter mechanic that converts trees into lumber at a specific rate: A given number of trees will produce a given amount of lumber. Many simulation games include technology upgrades that enable players to improve the efficiency of the converter mechanics in the game, causing them to produce more of the new resource from the old one.

■ **Traders** are mechanics that move a resource from one entity to another, and another resource back in the opposite direction, according to an exchange rule. If a player buys a shield from a blacksmith for three gold pieces, the trader mechanic transfers the gold from the player’s cash entity to the blacksmith’s and transfers the shield from the blacksmith’s inventory to the player’s. Traders are not the same as converters. Nothing is created or destroyed; things are just exchanged.

### Economic Structure

It is not particularly difficult to identify the entities and the resources that comprise an economy, but it is harder to get a good perspective on the system as a whole. If you were to make graphs of the elements in your economy, what shapes would the graphs reveal? Is the amount of a given resource increasing over time? How does the distribution of resources change? Do resources tend to accumulate in the hands of a particular player, or does the system tend to spread them out? Understanding the structure of your economy will help you find the answers.

### Economic Shapes

In the real world, people represent features of an economy with charts and figures (Figure 4.1). These graphs have a few interesting properties. At the small scale, their lines move chaotically, but at larger scales, patterns become visible. It is easy to see whether a line is going up or down in the long run and to identify good and bad periods. In other words, we can recognize and identify distinctive shapes and patterns from these types of charts.
We can draw similar charts displaying the fortunes of players in a game. As you will see, distinctive shapes and patterns emerge from the internal economy of a game. However, there is no one shape that identifies quality gameplay. What constitutes good gameplay depends on the goals you set for your game and the context that surrounds it. For example, in one game you might want the player to struggle for a long time before managing to come out on top (Figure 4.2). In another, you might aim for quick reversals in fortune and a much shorter play-through (Figure 4.3).
The Shape of a Game of Chess

We can take the development of players’ fortunes in a game of chess as a basis for studying shapes in game economies. In chess, the important resources are the players’ pieces. Chess players (and computer chess programs) assign a point value to each piece depending on what kind it is. For example, in one system, pawns are worth one point, rooks five, and the queen nine. Adding up the value of all the pieces one player has on the board produces a number called material. Players use their pieces to maneuver on the board to gain strategic positions. Strategic advantage can be measured as an abstract resource in the game. Figure 4.4 depicts what might be the course of play between two players in a game of chess.
You can discover a few important patterns in this chart. To start with, the long-term trend of both players’ main resource (material) is downward. As play progresses, players will lose and sacrifice pieces. Gaining material is very difficult. In chess, the only way to gain a piece is to bring a pawn to the other side of the board to be promoted to another, stronger piece, which would lead to an increase of material. This is a rare event that usually initiates a dramatic change of fortune for the players. If we consider only the material, chess appears to be a battle of attrition: Players who can make their material last longest will probably come out on top.

Strategic advantage is more dynamic in the game; it is gained and lost over the course of play. Players use their material to gain strategic advantage or reduce the strategic advantage of their opponents. There is an indirect relationship between the different amounts of material the players have and their ability to gain strategic advantage: If a player has more material, then gaining strategic advantage becomes easier. In turn, strategic advantage might be leveraged to take more pieces of an opponent and reduce that player’s material. Sometimes it is possible to sacrifice one of your pieces to gain strategic advantage or to lure your opponent into losing strategic advantage.

A game of chess generally progresses through three different stages: the **opening**, the **middle game**, and the **endgame**. Each stage plays a particular role in the game and is analyzed differently. The opening usually consists of a sequence of prepared and well-studied moves. During the opening, players try to maneuver themselves into a position of advantage. The endgame starts when there are relatively few pieces left, and it becomes safer to involve the king in the game. The middle game falls somewhere between the opening and the endgame, but the boundaries between the stages are not clear. These three stages can also be identified from the economic analysis in Figure 4.4. During the opening, the number of pieces decreases only slowly, while both players build up strategic advantage. The middle game starts when players are exploiting their strategic advantage to take their opponents’ pieces; it is characterized by a sharper decline of material. During the endgame, the material stabilizes again as the players focus on their final attempts to push the strategic advantage to a win.

**From Mechanics to Shapes**

To produce a particular economic shape, you need to know what type of mechanical structures create what shapes. Fortunately, there is a direct relationship between shapes in a game’s economy and the structure of its mechanics. In the next sections, we discuss and illustrate the most important building blocks of economic shapes.

**NEGATIVE FEEDBACK CREATES AN EQUILIBRIUM**

Negative feedback (as discussed in Chapter 3, “Complex Systems and the Structure of Emergence”) is used to create stability in dynamic systems. Negative feedback makes a system resistant to changes: The temperature of your refrigerator is kept constant.
even if the temperature outside the refrigerator changes. The point at which the system stabilizes is called the equilibrium. Figure 4.5 displays the effects of negative feedback.

The simplest shape of the equilibrium is a straight horizontal line, but some systems might have different equilibriums. An equilibrium might change steadily over time or be periodical (Figure 4.6). Changing equilibriums requires a dynamic factor that changes more or less independently of the negative feedback mechanism. The outside temperature throughout the year is an example of a periodical equilibrium that is caused by the periodic waxing and waning of the available hours of daylight and the relative strength of the sun.

**FIGURE 4.5**
The effect of negative feedback

**FIGURE 4.6**
Negative feedback on changing equilibriums. On the left, a rising equilibrium; on the right, a periodically changing equilibrium.

**POSITIVE FEEDBACK CREATES AN ARMS RACE**

Positive feedback creates an exponential curve (Figure 4.7). Collecting interest on your savings account is a classic example of such a curve. If the interest is the only source of money going into your savings account, the money will spiral upward, gaining speed as the accumulated sum creates more and more interest over time. In games, this type of positive feedback is often used to create an arms race between multiple players. A good example is the harvesting of raw materials in *StarCraft* (or similar constructions in many other RTS games). In *StarCraft*, you can spend 50
minerals to build a mining unit (called an SCV, for Space Construction Vehicle) that can be used to collect new minerals. If StarCraft players set aside a certain portion of their mineral income to build new SCVs, they get the same curve as money in a savings account.

Obviously, StarCraft players do not spend their resources only on SCV units. They also need to spend resources to build military units, to expand their bases, and to develop new technology. However, the economic growth potential of a base in StarCraft is vital in the long run. Many players build up their defenses first and harvest many resources before pushing to destroy their enemy with a superior capacity to produce military units.

FIGURE 4.7
Positive feedback creates exponential curves.

DEADLOCKS AND MUTUAL DEPENDENCIES

Positive feedback mechanisms can create deadlocks and mutual dependencies. In StarCraft, to get minerals, you need SCV units, and to get SCV units, you need minerals. These two resources are mutually dependent, and this dependency can lead to a deadlock situation: If you are left without minerals and SCV units, you can never get production started. In fact, you need enough minerals and at least one SCV unit to be able to build a headquarters, a third resource that enables this feedback loop. This deadlock situation is a potential threat. An enemy player might destroy all your SCV units. If this happens when you have spent all your minerals on military units, you are in trouble. It can also be used as a basis for level design. Perhaps you start a mission with military units, some minerals, but no SCV units or headquarters. In this case, you must find and rescue SCV units. Deadlocks and mutual dependencies are characteristics of particular structures in mechanics.
One of the most useful applications of positive feedback in games is that it can be used to make players win quickly once a critical difference is created. As should become clear from Figure 4.7, positive feedback works to amplify small differences: The difference between the balances of two bank accounts with equal interest rates but different initial deposits will only grow over time. This effect of positive feedback can be used to drive a game toward a conclusion after the critical difference has been made. After all, nobody likes to keep playing for long once it has become clear who will win the game.

**POSITIVE FEEDBACK ON DESTRUCTIVE MECHANISMS**

Positive feedback does not always work to make a player win; it can also make a player lose. For example, losing pieces in a game of chess weakens your position and increases the chance that you will lose more pieces; this is the result of a positive feedback loop. Positive feedback can be applied to a destructive mechanism (as is the case with losing material in chess). In this case, it is sometimes called a *downward spiral*. It is important to understand that positive feedback on a destructive mechanism is not the same as negative feedback—negative feedback tends to damp out effects and produce equilibrium. You can also have negative feedback attached to a destructive mechanism. The shooter game *Half-Life* starts spawning more health packs when a player is low on hit points.

**LONG-TERM INVESTMENTS VS. SHORT-TERM GAINS**

If *StarCraft* were a race to collect as many minerals as possible without any other considerations, would the best strategy be to build a new SCV unit every time you’ve collected enough minerals? No, not exactly. If you keep spending all your income on new SCVs, you would never save any minerals, which is what you need to win the game. To collect minerals, at some point you need to stop producing SCVs and start stockpiling. The best moment to do this depends on the goals and the constraints of the game—and what the other players do. If the goal is to accumulate the biggest pile of minerals in a limited amount of time or to accumulate a specific number of minerals as quickly as possible, there is an ideal number of SCV units you should produce.

To understand this effect, look at Figure 4.8. It shows that as long as you’re investing in new SCVs, your minerals do not accumulate. However, as soon as you stop investing, the minerals increase at a steady pace. This pace depends on the number of SCV units you have. The more you have, the faster your minerals will increase. The longer you keep investing, the later you will start accumulating minerals, but you will eventually catch up and overtake anybody who started accumulating before you did. Depending on the target goal, one of those lines is the most effective.
It is a good thing *StarCraft* is about more than just collecting minerals. Spending all your minerals on SCV units is a poor strategy because eventually you will be attacked. You have to balance your long-term goals with short-term requirements such as the protection of your base. In addition, some players favor a tactic in which they build up an offensive force quickly in a gambit to overwhelm their opponent before they can build up their defenses—the “tank rush,” which was first made famous in *Command & Conquer: Red Alert*. On some maps, initial access to resources is limited, and you must move around the map quickly to consolidate your access to future resources. Investing in SCV units is a good strategy in the long run, but it requires you take some risk in the beginning, possibly giving up on quick military gains via the tank rush.

**VARIATION FROM PLAYER PERFORMANCE AND RESOURCE DISTRIBUTION**

In *StarCraft*, it is not only the number of SCV units that determines the pace at which you harvest minerals. Minerals come from deposits of crystals, which have a particular location on the map. Finding the best location for your base, and micro-managing your SCV units to harvest minerals from crystals effectively, is a skill in itself. These are good examples of how player skill and game world terrain can produce input variation that affects the economic behavior of your game. Of course, the players’ inputs must influence the economy, but it is best if the player’s inputs occur frequently but no one input has too large an effect.
FEEDBACK BASED ON RELATIVE SCORES

During Marc LeBlanc’s talk on feedback mechanisms in games at the Game Developers Conference in 1999, he described two alternate versions of basketball. In “negative feedback basketball,” for every five points that the leading team is ahead, the trailing team is allowed to field one extra player. In “positive feedback basketball,” this effect is reversed: The leading team is allowed to field one extra player for every five points they are ahead. The effects of using the difference between two players to create a feedback mechanism are slightly different from using absolute values to feed this mechanism: The effects of the feedback mechanisms affect the difference between the players, not their absolute resources. This can produce some counter-intuitive effects. The economic chart of negative feedback basketball, for example, shows the lead of the better team settling on a stable distance at which the lack of the skill of the trailing team is offset by the extra players they can field (Figure 4.9).

DYNAMIC EQUILIBRIUM

The equilibrium that is created by a negative feedback mechanism that is fed by the difference in resources between two players is a dynamic equilibrium: It is not set to a fixed value but is dependent on other, changing factors in the game. You will find that most interesting applications of negative feedback in games are dynamic in this way. Making the equilibrium of a negative feedback loop dynamic by making it dependent on the relative fortune of multiple players, or other factors in the game, is a good way to move away from a too predictable balance created by a nondynamic equilibrium. With experience, knowledge, and skill, you will be able to combine several factors to compose dynamic equilibriums that are periodic, are progressive, or follow another desired shape.