Game Mechanics

Advanced Game Design

Ernest Adams
Joris Dormans

New Riders

Game Mechanics

Advanced Game Design

Ernest Adams
Joris Dormans

New Riders

Game Mechanics: Advanced Game Design
Ernest Adams and Joris Dormans

New Riders Games
1249 Eighth Street
Berkeley, CA 94710
(510) 524-2178
Fax: (510) 524-2221

Find us on the Web at www.newriders.com
To report errors, please send a note to errata@peachpit.com
New Riders Games is an imprint of Peachpit, a division of Pearson Education

Senior Editor: Karyn Johnson
Developmental Editor: Robyn Thomas
Technical Editor: Tobi Saulnier
Copy Editor: Kim Wimpsett
Production Editor: Cory Borman
Composition: WolfsonDesign
Proofreader: Bethany Stough
Indexer: Valerie Perry
Interior Design: Charlene Will, WolfsonDesign
Cover Design: Peachpit Press/Charlene Will

ISBN-13: 978-0-321-82027-3
ISBN-10: 0-321-82027-4

9 8 7 6 5 4 3 2 1

Printed and bound in the United States of America

Respectfully dedicated to the memory of Mabel Addis Mergardt, principal designer of *The Sumerian Game* (later made famous as *HAMURABI*), the first game with an internal economy that I ever played.

—Ernest W. Adams

To Marije van Dodeweerd for love.

—Joris Dormans

Acknowledgments

The genesis of this book was a late-night meeting between the two of us during the G-Ameland student game jam festival on a small island off the north coast of the Netherlands. Joris Dormans showed the Machinations framework to Ernest Adams, and Ernest Adams promptly said, “We should write a book about game mechanics.” But it took nearly two years and the advice and assistance of many other people before we were done. Now it is time to thank them.

Our deepest appreciation goes to Mary Ellen Foley and Marije van Dodeweerd, our beloved mates, who patiently tolerated very late nights, missed holidays and weekends, and the occasional rant about the vagaries of the writing process. We’ll make it up to you if we can!

Stéphane Bura suggested that Joris should create an interactive tool when he saw the original, static version of the Machinations diagrams.

Jesper Juul made the invaluable distinction between games of emergence and games of progression that informs the entire book.

Remko Scha had a big impact on the formal scrutiny of the Machinations framework in his capacity as Joris Dormans’s PhD supervisor.

Mary Ellen Foley kindly checked and corrected all our references.

The colleagues and students at the Amsterdam University of Applied Sciences always have been willing test subjects for much of the material that ended up in this book.

We must also thank a number of people for permission to reproduce images: Alexandre Duret-Lutz, for his photo of *The Settlers of Catan*; Andrew Holmes, for his photo of *Kriegsspiel;* Jason Lander, for his photo of *Power Grid*; Johan Bichel Lindegaard, for his photo of *Johan Sebastian Joust*; Wikimedia Commons contributor popperipopp, for his or her photo of the game *Connect Four.* We are also grateful to the Giant Bomb website (*www.giantbomb.com*), for permission to reproduce screen shots from their collection.

Thanks to Mika Palmu, Philippe Elsass, and all other contributors to *FlashDevelop*, for creating the open source development tool that was used to program the Machinations Tool.

We are extremely grateful to the many anonymous people who have helped to build *Inkscape,* the open source Scalable Vector Graphics editor, without which it would have been much more difficult to produce our illustrations.

As Elrond said, the last place is the place of honor. We thank Margot Hutchison, Ernest Adams's agent, for assistance with the contract. Tobi Saulnier was our wise and sharp-eyed technical editor. Her suggestions are present but invisible throughout the book, and we're deeply grateful that the CEO of a game company would be willing to take the time to help us. Robyn G. Thomas, our tireless (and seemingly sleepless) development editor, pleaded, cajoled, threatened, and oversaw the whole process with her usual flair and attention to detail. And finally, special thanks to Karyn Johnson, senior editor at Peachpit Press, for having the faith in us to let us write the book in the first place.

We hasten to add that the blame for any errors or omissions belongs entirely to us and not to any of the foregoing.

We welcome all comments, questions, and criticism; please write to Joris Dormans at *jd@jorisdormans.nl* and to Ernest W. Adams at *ewadams@designersnotebook.com*.

About the Authors

Ernest W. Adams is an American game design consultant and teacher residing in England. In addition to his consulting work, he gives game design workshops and is a popular speaker at conferences and on college campuses. Mr. Adams has worked in the interactive entertainment industry since 1989 and founded the International Game Developers' Association in 1994. He was most recently employed as a lead designer at Bullfrog Productions, and for several years before that, he was the audio/video producer on the *Madden NFL* line of football games at Electronic Arts. In his early career, he was a software engineer, and he has developed online, computer, and console games for machines from the IBM 360 mainframe to the present day. Mr. Adams is the author of four other books, including *Fundamentals of Game Design,* the companion volume to this book. He also writes the Designer's Notebook series of columns on the *Gamasutra* game developers' webzine. His professional website is at *www.designersnotebook.com*.

Joris Dormans (PhD) is a Dutch lecturer, researcher, and gameplay engineer based in Amsterdam, the Netherlands, working in industry and higher education since 2005. For the past four years, he has been researching formal tools and methods to design game mechanics. His other area of research focuses on how to leverage formal design methods to generate games procedurally. Dr. Dormans has presented papers and hosted workshops on game design on many academic and industry conferences. As an independent freelance game designer, he published and worked on several video and board games. Among these are story-driven adventure games, physical platform games, and a satirical political card game. He has also participated in all Global Game Jams to date. His professional website is at *www.jorisdormans.nl*.

About the Technical Editor

Tobi Saulnier is founder and CEO of 1st Playable Productions, a game development studio that specializes in design and development of games tailored to specific audiences. Games developed by 1st Playable span numerous genres to appeal to play styles and preferences of each group and include games for young children, girls, middle schoolers, young adults, and some that appeal to broad audiences. The studio also creates games for education. Before joining the game industry in 2000, Tobi managed R&D in embedded and distributed systems at General Electric Research and Development, where she also led initiatives in new product development, software quality, business strategy, and outsourcing. She earned her BS, MS, and PhD in Electrical Engineering from Rensselaer Polytechnic Institute.

Contents

Introduction

This is a book about games at their deepest level. No matter how good a game looks, it won't be fun if its mechanics are boring or unbalanced. Game mechanics create gameplay, and to build a great game, you must understand how this happens.

Game Mechanics will show you how to design, test, and tune the core mechanics of a game—any game, from a huge role-playing game to a casual mobile phone game to a board game. Along the way, we'll use many examples from real games that you may know: *Pac-Man, Monopoly, Civilization, StarCraft II,* and others.

This book isn't about building Unreal mods or cloning somebody else's app that's trending right now. It's called *Advanced Game Design* for a reason. We wrote *Game Mechanics* to teach you the timeless principles and practice of mechanics design and, above all, to give you the tools to help you do it—for a class, for a career, for a lifetime.

We also provide you with two unique features that you won't find in any other textbook on game design. One is a new tool called *Machinations* that you can use to visualize and simulate game mechanics on your own computer, without writing any code or using a spreadsheet. Machinations allows you to actually *see* what's going on inside your mechanics as they run and to collect statistical data. Not sure if your internal economy is balanced correctly? Machinations will let you perform 1,000 runs in a few seconds to see what happens and put all the data at your fingertips. Machinations was created by Joris Dormans and is easy to use on any computer that has Adobe Flash Player installed in its web browser. You don't have to use the Machinations Tool to benefit from the book, though. It's simply there to help reinforce the concepts.

The other unique feature of *Game Mechanics* is our *design pattern library*. Other authors have tried to document game design patterns before, but ours is the first to distill mechanics design to its essence: the deep structures of game economies that generate challenge and the many kinds of feedback loops. We have assembled a collection of classic patterns in various categories: engines of growth, friction, and escalation, plus additional mechanisms that create stability cycles, arms races, trading systems, and many more. We've made these general enough that you can apply them to any game you build, yet they're practical enough that you can load them in the Machinations Tool and see how they work.

Game mechanics lie at the heart of all game design. They implement the living world of the game; they generate active challenges for players to solve in the game world, and they determine the effects of the players' actions on that world. It is the game designer's job to craft mechanics that generate challenging, enjoyable, and well-balanced gameplay.

We wrote this book to help you do that.

Who Is This Book For?

Game Mechanics is aimed at game design students and industry professionals who want to improve their understanding of how to design, build, and test the mechanics of a game. Although we have tried to be as clear as we can, it is not an introductory work. Our book expands on the ideas in another book by Ernest Adams called *Fundamentals of Game Design* (New Riders). We refer to it from time to time, and if you lack a grounding in the basics of game design, you might find it helpful to read the current edition of *Fundamentals of Game Design* first.

The chapters in *Game Mechanics* end with exercises that let you practice the principles we teach. Unlike the exercises in *Fundamentals of Game Design,* many of them require a computer to complete.

How Is This Book Organized?

Game Mechanics is divided into 12 chapters and 2 appendixes that contain valuable reference information. There is also a quick reference guide to Machinations in Appendix A.

Chapter 1, "Designing Game Mechanics," establishes key ideas and defines terms that we use in the book, and it discusses when and how to go about designing game mechanics. It also lists several forms of prototyping.

Chapter 2, "Emergence and Progression," introduces and contrasts the important concepts of emergence and progression.

Chapter 3, "Complex Systems and the Structure of Emergence," describes the nature of complexity and explains how complexity creates emergent, unpredictable game systems.

Chapter 4, "Internal Economy," offers an overview of internal economies. We show how the structure of an economy creates a game *shape* and produces different phases of gameplay.

Chapter 5, "Machinations," introduces the Machinations visual design language and the Machinations Tool for building and simulating mechanics. It includes an extensive example using *Pac-Man* as a model.

Chapter 6, "Common Mechanisms," describes a few of the more advanced features of Machinations and shows how to use it to build and simulate a wide variety of common mechanisms, with examples from many popular game genres.

Chapter 7, "Design Patterns," provides an overview of the design patterns in our design pattern library and offers suggestions about how to use them to brainstorm new ideas for your designs.

Chapter 8, “Simulating and Balancing Games,” explains how to use Machinations to simulate and balance games, with case studies from *Monopoly* and Will Wright’s SimWar.

Chapter 9, “Building Economies,” explores economy-building games, using *Caesar III* as an example, and takes you through the design and refinement process for a new game of our own, *Lunar Colony*.

Chapter 10, “Integrating Level Design and Mechanics,” moves into new territory, looking at how game mechanics integrate with level design and how properly sequenced challenges help the player learn to play.

Chapter 11, “Progression Mechanisms,” discusses two kinds of progression. We start with traditional lock-and-key mechanics and then consider emergent progression systems in which progress is treated a resource within the economy of the game.

Chapter 12, “Meaningful Mechanics,” concludes the book with an exploration of the role of mechanics in transmitting meaning in games that have a real-world message to send. This topic is increasingly important now that game developers are making more *serious games:* games for health care, education, charity, and other purposes.

Appendix A, “Machinations Quick Reference,” lists the most commonly used elements of the Machinations Tool.

Appendix B, “Design Pattern Library,” contains several patterns from our design pattern library. You can find the completed design pattern library in the online Appendix B at *www.peachpit.com/gamemechanics* and a much more extensive discussion of each design pattern in Chapter 7.

Appendix C, “Getting Started with Machinations,” is available online at *www.peachpit.com/gamemechanics* and provides a tutorial for using the Machinations Tool.

Companion Website

At *www.peachpit.com/gamemechanics* you’ll find material for instructors, digital copies of many of the Machinations diagrams used in this book, more design patterns, and a step-by-step tutorial to get you started with Machinations. To get access to this bonus material, all you need to do is register yourself as a Peachpit reader. The material on the website may be updated from time to time, so make sure you have the latest versions.

This page intentionally left blank

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.

NOTE We use a very broad definition of the word *economy*. It's not just about money! In an information economy, there are data producers, data processors, and data consumers. Political economy studies the way that political forces influence government policies. Economies about money are called market economies. But we use the term in a more abstract way to refer to any kind of system in which resources—of any type—can be produced, exchanged, and consumed.

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.

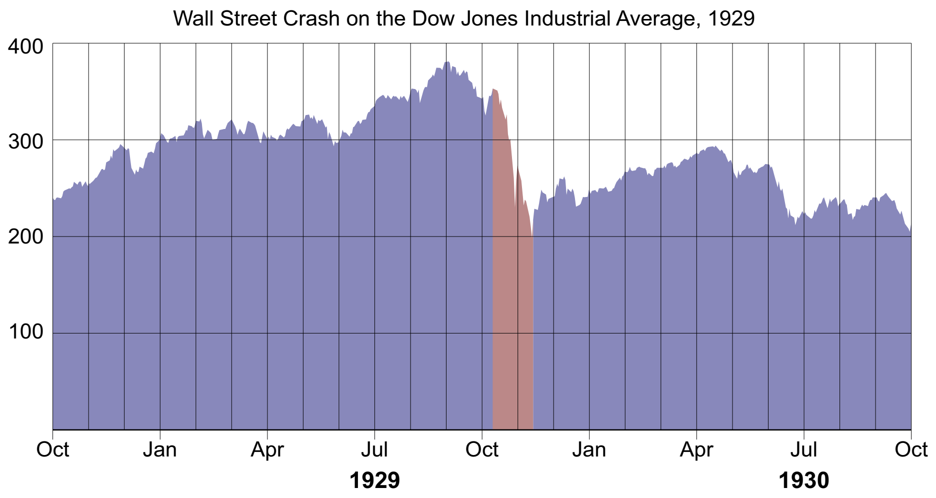


FIGURE 4.1
Graph of the stock market crash leading to the Great Depression. Most movement is chaotic, but the crash is clearly visible.

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**).

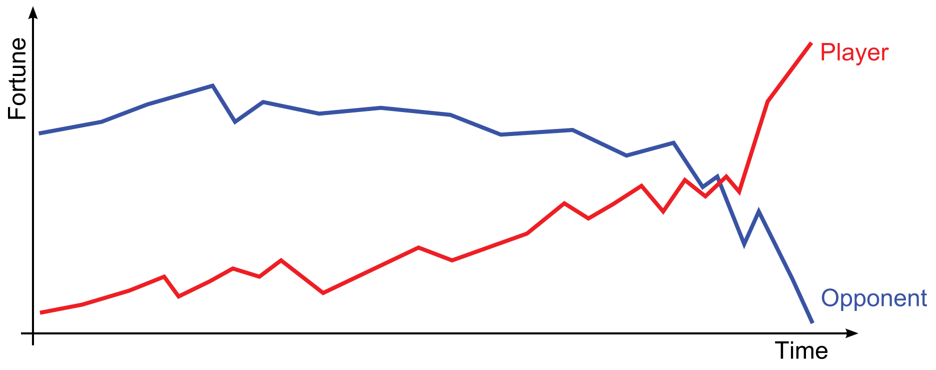


FIGURE 4.2
A long game in which the player triumphs after an extended struggle against a powerful opponent

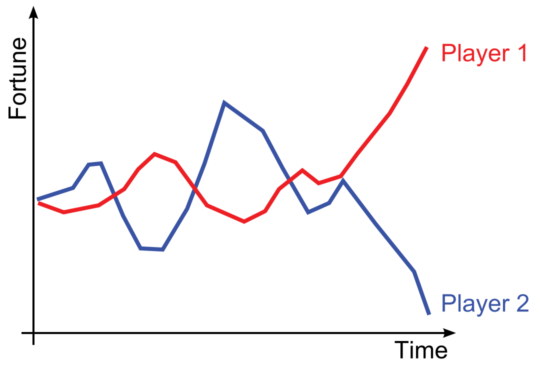


FIGURE 4.3 A short game with quick reversals of fortune

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.

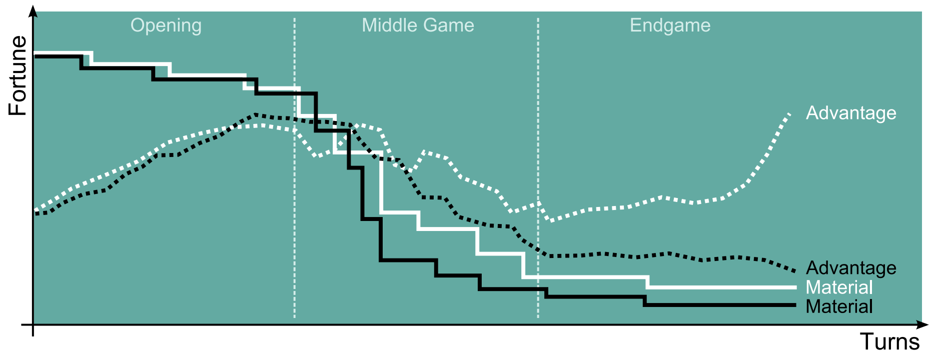


FIGURE 4.4 The course of a particular game of chess. The color of a line indicates the color of the player it refers to.

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.

NOTE This analysis of chess is a high-level abstraction to illustrate an economic principle using a familiar game. Classic texts on the theory of chess do not treat it in economic terms, because chess is about checkmating the king, not taking the most pieces. However, our illustration shows that gameplay and game progress can be understood in economic terms even if the game itself is not about economy.

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.

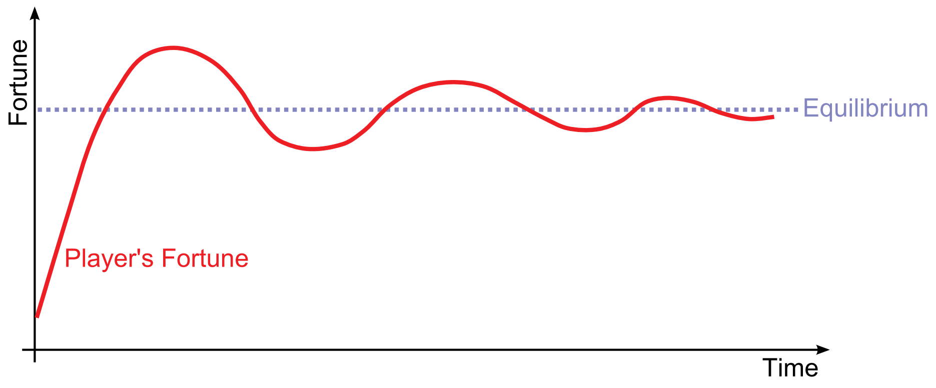
From Mechanics to Shapes

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

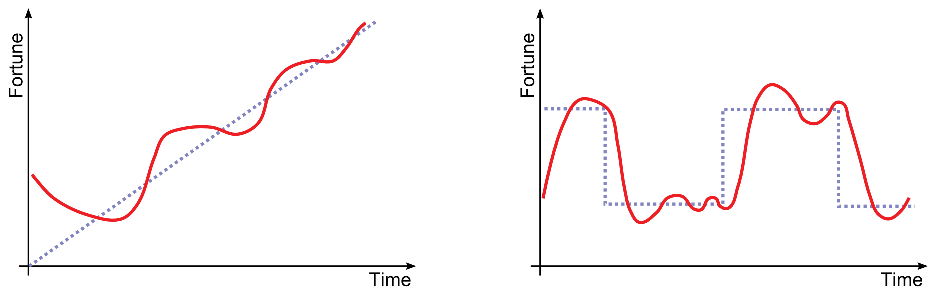
Negative feedback (as discussed in Chapter 3, "Complex Systems and the Structure of Emergence") is used to create stability in dynamic systems. Negative feedback makes a system resistant to changes: The temperature of your refrigerator is kept constant even if the temperature outside the refrigerator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

FIGURE 4.5 The effect 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 4.6 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.

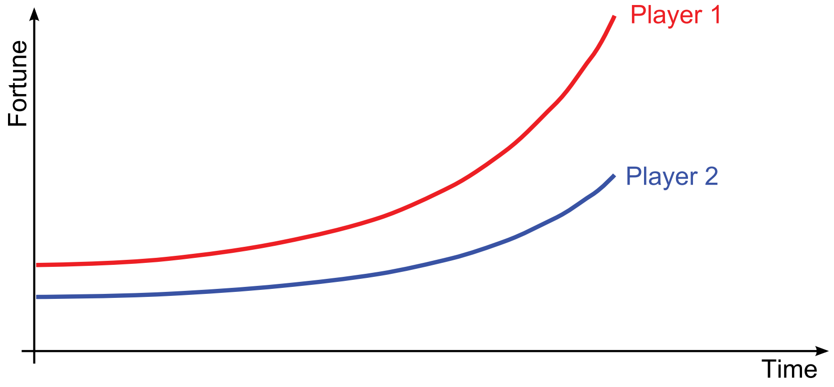


FIGURE 4.7 Positive feedback creates exponential curves.

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.

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.

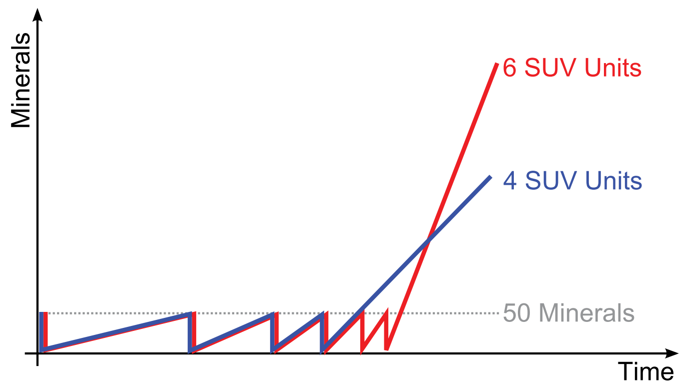


FIGURE 4.8
A race of accumulation

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.

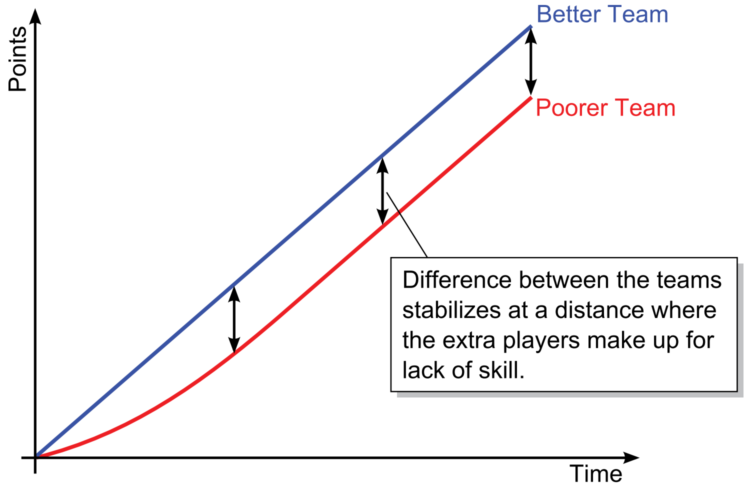
VARIATION FROM PLAYER PERFORMANCE AND RESOURCE DISTRIBUTION

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 counterintuitive 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**).

FIGURE 4.9 Score graph of negative feedback basketball



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.

When two teams are playing positive feedback basketball, the differences in skills are aggravated. When one side is better than the other, this will result in a very one-sided match. However, when both sides are closely matched, a different pattern emerges: The game will probably remain close, until one side manages to take a decisive lead after which the match becomes very one-sided again. In this latter case, a small difference in skill, an extra effort, or sheer luck can become the decisive factor.

In Chapter 6, we explore the gameplay effects of positive and negative feedback on basketball in more detail.

RUBBERBANDING IS NEGATIVE FEEDBACK ON RELATIVE POSITION

Racing games frequently use negative feedback based on the players' position in the field to keep the race tight and exciting. This mechanism is often referred to as *rubberbanding*, because it seems to players as if the other cars are attached to theirs by a rubber band—they never get too far ahead or too far behind. Some games implement rubberbanding by simply slowing leading cars down and speeding trailing cars up. Other games use more subtle negative feedback mechanics to reach similar effects. In *MarioKart*, players are awarded with a random power after picking up a power-up. However, trailing players have a better chance of picking up a more powerful power-up than leading ones do. In addition, because most weapon power-ups in *MarioKart* are used on opponents in front of the player, the leader of the field is a target more often than the player in the last position. This causes the lead to change hands frequently and increases the excitement of the game, increasing the likelihood of a last-minute surge past the leader.

Uses for Internal Economies in Games

In the previous sections, we discussed the elements and common structures of internal game economies. In this section, we will discuss how game economies are typically used in games of different genres. Table 1.1 provided a quick overview of some mechanics that are typically part of that economy. Now, we will discuss the typical economic structures found across game genres in more detail.

Use an Internal Economy to Complement Physics

Obviously, physics make up the largest part of action games' core mechanics. Physics are used to test the player's dexterity, timing, and accuracy. Still, most action games add an internal economy to create an integral reward system or to establish a system of power-ups that requires resources. In a way, the simple use of a scoring system adds economic mechanics to many action games. If you collect points for taking out enemies, players will have to consider how much they will invest to take out that enemy. Will they put their avatars at risk, or will they waste ammunition or some sort of energy that cannot easily be regained?

Super Mario Brothers and many other similar platform games use a simple economy to create a reward system. In *Super Mario Brothers*, you can collect coins to gain extra lives. Because you need to collect quite a few coins, the designer can place them liberally throughout a level and add or remove them during play-testing without affecting the economy significantly. In this way, coins can be used to guide a player through a level. (Collectible objects that are used to guide players are often called *breadcrumbs*.) It is safe to assume that you are able to reach all coins, so if you spot a coin, there must be a way to reach it. This creates the opportunity to reward skillful players for reaching difficult places in the game. Used in this way, the internal economy of the game can be very simple. However, even a simple economy like this already involves a feedback loop. If players go out of their way to collect many coins, they will gain more lives, thus allowing them to take more risks to collect more coins.

When setting up a system like this, you must be careful to balance the risks and rewards. If you lure players into deadly traps with just a single coin, you are inviting them to risk a life to gain a single coin. That simply isn't fair, and the player will probably feel cheated. As a designer, you have a responsibility to match the risks and rewards, especially when they are placed close to the path novice players will take. (Creating a reward that the player can see but *never* reach is even worse—it causes players to take risks for rewards they can never obtain.)

Power-ups, including weapons and ammunition in first-person shooters, create a similar economy. Power-ups and ammo can be rewards in themselves, challenging the player to try to eliminate all enemies in a level. As a game designer, you have to make sure that the balance is right. In some games, it is perfectly all right if killing enemies will, on average, cost more bullets than the players can loot from their remains. However, if this leads to a situation in which the player is eventually short on the proper ammo for the big confrontation with a boss character, you risk penalizing players for making an effort in the game. In survival-oriented first-person shooters, creating a scarce economy of weapons and ammo is generally a good thing because it adds to the tension and the drama, but it is a difficult balance to create. If your shooter is more action-oriented, then it is probably best to make sure there is plenty of ammo for the player, and you should make sure that taking out extra enemies is properly rewarded.

Use an Internal Economy to Influence Progression

The internal economy of a game can also be used to influence progression through a game that involves movement. For example, power-ups and unique weapons can play a special role in an action game's economy. They can be used to gain access to new locations. A double-jump ability in a platform game will allow the player to reach higher platforms that were initially unreachable. In economic terms, you can think of these abilities as new resources to produce the abstract resource *access*.

Access can be used to gain more rewards or can be required to progress through the game.

In both cases, as a designer, you should be wary of a deadlock situation. For example, you might have a special enemy guard the exit of a level. Somewhere in the same level there is a unique weapon that is required to kill that enemy with a single shot. The weapon is usable throughout the level. When the player finds the weapon, it is loaded with ten bullets, and there are no more until the next level—but the player doesn't know this the first time playing. Now, a first-time player finds the weapon, fires a couple of shots to experiment with it, uses it on a couple of other enemies, and finds himself at the exit with one bullet left. The player fires and misses. You have just created a deadlock situation. The player needs access to the next level to gain bullets but needs bullets to gain access.

DEADLOCK RESOLUTION IN ZELDA

In many Zelda games, players frequently must use consumable items—arrows or bombs—to gain access to new areas. This creates a risk of deadlocks, if the player runs out of the items needed. The designers of Zelda games prevent these no-win situations by making sure there are plenty of renewable sources for the required resources. Dungeons are littered with useful pots that yield these resources if the player destroys them (**Figure 4.10**). Broken pots are mysteriously restored as the player moves from room to room, creating a source that is replenished from time to time. Because the pots can contain anything, as a designer you can use a mechanism like this to provide the player with any resource required. You can even use it as a way of providing gameplay hints: If players are finding a lot of arrows, they are probably going to need a bow soon.

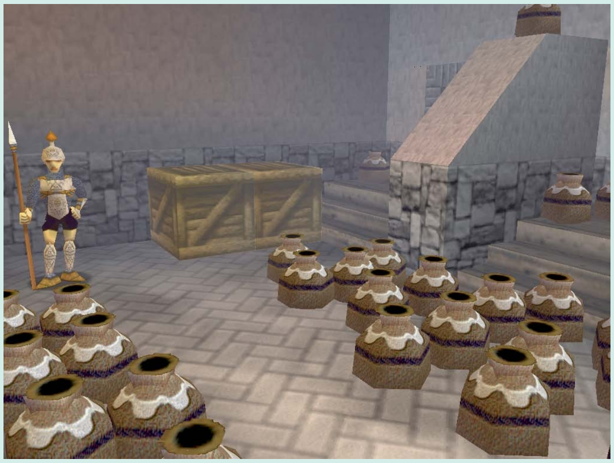


FIGURE 4.10 Pottery is a useful source in Zelda games.

Use an Internal Economy to Add Strategic Gameplay

It is surprising how many of the strategic challenges in real-time strategy games are economic in nature. In a typical game of *StarCraft,* you probably spend more time managing the economy than fighting the battle. Including an internal economy is a good way to introduce a strategic dimension to a game that operates on a larger time span than most physical and/or tactical action.

One of the reasons that most real-time strategy games have elaborate internal economies is that these economies allow the games to reward planning and long-term investments. A game about military conflict with little forward planning and no long-term investments would be a game of tactics rather than strategy, because it would probably be more about maneuvering units on the battle field. To sustain a level of strategic interaction, a game's internal economy needs to be more complicated than the internal economies that simply complement the physics of an action game. Economies in strategy games usually involve multiple resources and involve many feedback loops and interrelationships. Setting up an economy like that for the first time is challenging, and finding the right balance is even more difficult. As a designer, you need to understand the elements of the economy and develop a keen sense to judge its dynamic effects. Even if you have years of experience, it is easy to make mistakes: There have been many tweaks to the economy of games like *StarCraft* to retain the right balance after players developed new strategies, even after the game had been long published!

Even without a focus on the economics of production (such as *StarCraft*'s minerals and SCV units), internal economies can add strategic depth to almost any game. In most cases, this involves planning to use the available resources wisely. As already discussed, the economy of chess can be understood in terms of material (playing pieces) and strategic advantage. Chess is not about production, and gaining a piece in chess is unusual. Rather, the game is about using and sometimes sacrificing your material in order to produce as much strategic advantage as possible. In other words, chess is all about getting the most mileage out of your pieces.

You can find something similar in the game *Prince of Persia: The Sands of Time*. In this action-adventure game, the player progresses through many levels filled with dexterity and combat challenges. Early in the game, the player is awarded a magical dagger that allows that player to control time. If anything goes wrong, the player can use sand from the dagger to rewind time and to try again. This power can also be used during combat, for example just after the player has taken a big hit. In addition, the player can use sand as a magical power to freeze time. This helps when battling multiple enemies. The sand is not limitless, however. The player can rewind time only so often, but fortunately, defeating enemies provides the player with new sand. This means that, in additional to the usual action-oriented gameplay, the player has to manage a vital resource. The player must decide when is the best time to invest some sand. Different players will have different ideas about when they should use their sand. Some will use it more often to help out with combat, while others will prefer to save it for challenging jumping puzzles. In this way, the sand is a versatile resource: Players are able to use it to boost their performance where they need it most.

Use an Internal Economy to Create Large Probability Spaces

As internal economies grow more complex, the probability space of your game expands quickly. Games with a large probability space tend to offer more replay value, because players will have more options to explore than is generally achievable with a single play-through. Another benefit is that these games can also create a more personal experience, because the performance of players and their choices directly affect what parts of the probability space open up for exploration.

Games that use an internal economy to govern character development, technology, growth, or vehicle upgrades often use an internal currency to provide options to the player. This is a typical gameplay feature found in role-playing games, in which players spend in-game money to outfit their characters and spend experience points to develop skills and abilities. It is also found in certain racing games that allow players to tune or upgrade their vehicles between (or sometimes even during) races. As long as there are enough options and the options present really different solutions to problems encountered in the game, or are otherwise important to the player, this is a good strategy.

When using an internal economy to customize the gameplay, there are three things you need to watch out for. First, in an online role-playing game, if a particular combination of items and skills is more efficient than others, players will quickly identify and share this information, and the economy will be thrown off-balance. Either players will choose only that option, effectively reducing the probability space and creating a monotonous experience, or they will complain that they cannot keep up with players who did. In games like this, it is important to understand that customization features are best balanced by some sort of negative feedback. Role-playing games usually implement many negative feedback mechanisms for this reason: Every time characters gain a level and improved skills, they need more experience points to get to the next level. This effectively works to reduce the differences in levels and abilities and requires more investment from a player for each level earned.

Second, you have to be sure that the probability space is large enough that players do not end up exploring it entirely in one play session. For example, if in a role-playing game players have a rating between 1 and 5 for the attributes of strength, dexterity, and wisdom, and the player can choose which one to increase from time to time, it is generally a poor design decision to require them to upgrade all these attributes to the maximum in order to finish the game. Similarly, if the player has only limited choice over what order to upgrade her attributes, the consequences of those choices are reduced. A good way to include choices that have real consequences is to create choices that exclude each other. For example, players can generally choose only one class for their character in a role-playing game. Each class should have a unique set of different skills and abilities. In *Deus Ex*, the player is also presented with choices to improve the cyborg character that have gameplay consequences: The player might be forced to choose between installing a module that will render the character invisible for short periods and a special type of subdermal armor that will make the character much more resistant to damage.

Third, you should ideally design your levels in such a way that players can use different strategies to complete them. For example, in *Deus Ex*, the player can choose to develop a character in different ways. The player can focus on combat, stealth, or hacking as alternative ways of solving the many challenges in the game. This means that almost every level has multiple solutions. This is not an easy balance to strike. If you estimate that the player has managed to upgrade three options before a certain level, you have to take into account that the player upgraded the combat abilities three times, stealth three times, hacking three times, or perhaps all of them once. In *Deus Ex,* this problem is even more pronounced because all the sources of experience points that you require to upgrade are not renewable: You gain them for progressing and performing certain side quests. Going back to a previous area to harvest some more experience is not an option.

This example illustrates that the levels in games that permit customization must be more flexible, and more general, than in conventional action games, because you don't know exactly what abilities the player's avatar will have. *Deus Ex Human Revolution* contained a flaw: It allowed the players different ways to play the game but only one way to beat the boss characters, which defeated the point of allowing the players to customize their avatars.

Tips for Economy Construction Games

Games in which the player builds an economy, such as construction and management simulations, tend to have large and complex internal economies. *SimCity* is a good example. As players zone areas and build infrastructure, they use these building blocks to craft an economic structure that produces the resources they need to increase it even further. Building a game like this requires the designer to assemble a toolbox of mechanics that the player can combine in many interesting ways. This is even harder than designing a complete, functional, and balanced economy yourself. You have to be aware of all the different ways your economic building blocks combine. When successful, playing the game can be very rewarding, because the economy the players build up through play directly reflects their choices and strategies. This is why no two cities in *SimCity* are alike.

If you are designing an economy construction game, there are three strategies that can help you keep the complexity of your task under control:

- **Don't introduce all the player's building blocks at once.** Construction and management simulations typically allow the player to build something—a farm, factory, or city, for example—out of elementary units, building blocks, that play a role in the economy. (In *SimCity*, these are zoned land and specialized buildings.) It is a good idea to gently introduce players to the different elements in your game, a few at a time. This makes it easier to control the probability space, at least initially. By allowing certain building blocks and disallowing others, you can craft scenarios and create special challenges. If your game has no distinct levels or special scenarios, make sure that not all building options are available from the start. Have players accumulate resources before they can use the more advanced building blocks that unlock new options. *Civilization* is an excellent example of an economy construction game in which most of the building blocks are locked at the beginning of the game and must be unlocked one by one before the players can use them.

- **Be aware of the meta-economic structure.** In an ideal economy construction game, the number of ways of putting the economic building blocks together is endless. However, in most such games, certain approaches are better than others (and in games with a victory condition, some approaches are unwinnable). As a designer, you should be aware of typical constructions that might be called *meta-economic structures*. For example, in *SimCity*, a particular mix of industrial, residential, and commercial zones will prove to be very effective. Players will probably discover these structures quickly and follow them closely. One difficult, but effective, way of dealing with patterns that could become too dominant is to make sure that patterns that are effective early in the game cease to be effective later. For example, a particular layout of zones might be an effective way to grow your population initially but causes a lot of pollution in the long run. Slow-working, destructive positive feedback is a good mechanism to create this sort of effect.

- **Use maps to produce variety and constrain the possibility space.** *SimCity* and *Civilization* wouldn't be nearly as much fun if you could build your city or empire on an ideal piece of land. Part of the challenge of these games is to deal with the limitations of the virtual environment's initial state. As a designer, you can use the design of the map to constrain players or to present opportunities. So, although there might be a best way of building the economy (something that we might call a *dominant* meta-economic structure), it is simply not possible to do so in particular terrain. This forces players to improvise, and rewards players who are more flexible and versatile. In *SimCity*, the disaster scenarios in which players can unleash several natural disasters on their cities challenges their improvisation and flexibility in a similar vein; and of course, *SimCity* also generates disasters at random, setting back the player's progress.

Summary

In this chapter, we introduced the essential elements of an internal economy: resources, entities, and some of the mechanics that manipulate them, including sources, drains, converters, and traders. We examined the concept of economic shapes as seen through graphs and showed how different mechanical structures can produce different shapes. Negative feedback creates equilibrium, while positive feedback creates an arms race among opponents. Implemented another way, positive feedback can produce a downward spiral, because a player finds it harder and harder to grow his economy. Feedback systems based on relationships between two players can produce effects that keep games close or tend to cause the player in the lead to stay in the lead.

Game designers can use internal economics in many ways to make games interesting, enriching both the progression of a game and the strategic choices a player has to make. The internal economy also affects the competitive landscape between diverse or closely matched players in multiplayer games. The chapter ended with specific suggestions about how to build games in which players construct an economy, as in *SimCity*.

Exercises

1. Identify the resources and economic functions in a published game. (Your instructor may specify particular games to study.)

2. Find an example of a game (not referred to in this chapter) that exhibits one of these properties: negative feedback with periodic equilibrium, a downward spiral, a short-term versus long-term investment trade-off, feedback based on players' relative scores, or rubberbanding. Explain which resources are involved, and show how the game's mechanics produce the effect you discovered.

3. Find an example of a game (other than a Zelda game) in which a deadlock may occur. Does the game provide a means of breaking the deadlock? Explain.

INDEX