A SYSTEMS APPROACH

ADVANCED Game Design

Michael **SELLERS**

Contents

CHAPTER 5

WORKING AS A SYSTEMIC GAME DESIGNER

In this chapter, we move from the foundational theory to the practice of designing games. Here we look at different aspects of the game design process and how to get started in each as a systemic game designer.

This is an overview that will be supplemented by Chapters 6, 7, and 8, where we go into more depth on designing the game as a unified whole, then its loops, and finally its parts.

How Do You Even Start?

Lots of people want to design games. They dream about it and talk about it but somehow never manage to actually get started. This is common, and most people who say they have a burning desire to design games never actually do it. Few manage to gather their courage and begin the journey of wading into the dark waters of game design. Rarer still are those who emerge on the far side, dragging their game kicking and screaming from the inchoate sea of design ideas. (That may seem like an overwrought metaphor, but when you complete your first game, you may no longer think so.)

One of the first questions people commonly ask when contemplating doing game design as more than a hobby, more than a "wouldn't it be cool if" activity, is along the lines of "How do I even start?" Designing a game can seems like an impossible problem with no easy handles, no obvious way in. The sheer complexity and impenetrability of the problem can make it seem like the best you can do is leap in with both feet and hope for the best. That is, in fact, what generations of game designers up to now have done. At some point, those of us who have been designing games for decades just sort of made that first leap. For many the first few attempts are utter failures. Rovio went through 51 attempts before hitting it big with *Angry Birds*—and even this attempt looked like a flop at first (Cheshire 2011).

Failure itself isn't a bad thing; anytime you try something new (which is most of the time in game design), you are going to fail a lot. However, you can reduce the amount and duration of failure by approaching game design systemically. Seeing a game as a system (containing other systems) is a good way to crack the problem of where to start in the otherwise overwhelming process.

From Wholes to Parts or Parts to Wholes

One key to knowing how to start is figuring out whether to begin with the parts, the loops, or the whole of your design. Opinions run high on this question. Many designers are firmly in one camp or another, and what they do works for them. Some designers will declare that any game design must start with "the nouns and verbs"—that is, the parts that will form the systems—while others begin with a more intuitive feeling of the kind of experience they want to create. Occasionally some will even start with Ellenor's (2014) idea of "a machine that does x" and then work out what parts make it go and what sort of gameplay experience emerges from it. Differences of opinion on the "right" way to approach game design can make for miscommunication and talking past each other.[1]

1. I had this experience while working with Will Wright of SimCity fame. He is firmly a "nouns and verbs" kind of guy, while I often approach designs from a more holistic-experiential point of view. It took a while before we were able to understand each other's perspectives.

Despite strong opinions from some designers, there is no single "right" way to approach game design. Our systemic view should make this clear: in designing a game, you need to get to the point where you have fully defined the parts, the loops, and the whole of your design. As a game designer, you need to be able to move up and down the organizational levels with ease, shifting your focus between the parts, the loops, and the whole as needed. As a result, you can start the design process with whichever of these makes the most sense and bounce between them as needed.

Know Your Strengths, Work to Your Weaknesses

When you begin thinking about making a game, where do your thoughts lead you? Do you think about things like having a game where players are sharks or superheroes, or where each is a kite in the sky? Or are you more likely to approach a game as a simulation or modeling problem? If it's a game about a little one-celled organism, do you start by listing all the parts of the cell? Or do you maybe start thinking about a game where the player is the manager of a remote trading post by jotting down how buying and selling would work?

Every game designer has their strengths; everyone has their "home place" where they start—and then retreat to when making the design becomes difficult. You need to find out where your game design home is and then work out ways to not give in to the temptation to stay there; you also need to figure out how to work with others who approach game design differently from you.

The *doing* of game design is the best way to figure out which parts of the process come most naturally to you. Still, it is worth considering where you think it should start and working from there.

Storytellers

Game designers who tend to start with the whole experience often paint an evocative picture of the player's journey through a game: how the player feels, what they encounter, and what sort of changes they go through. Game designers like these can sometimes seem like expert storytellers. They're able to give you the grand sweep of the world…but they can run into trouble. Games aren't stories. "Telling" a game like a story can be a satisfying first pass at building the world that the players inhabit, but ultimately the game has to be much more than that.

A storyteller needs to hang on to their talent for painting a mental picture of the experience of a world but not get stuck there. If you are a storyteller, you need to build your talents for creating working systems that have their own tokens, rules, and dynamic elements. You likely have the thematic part in hand, but you need to support it with the structure of the underlying game—and work with others who can help you do so.

Inventors

Many game designers are enamored of inventing complex mechanisms—things like clocks with lots of gears, marble-run sculptures, and so on. These can be mesmerizing displays of systems in action. Similarly, sometimes game designers come up with ideas for new kinds of ecological or economic mechanisms and spend time playing with them. For example, the early prototypes for the game *Spore* included lots of different simulation mechanisms, including one that (with a bit of help from the player) simulated the formation of a star system from an interstellar cloud of gas and dust.

But as fascinating as these inventions can be, they aren't games. As with telling a story about a game, designers will sometimes build a mechanism that scratches the "watch it go" itch, only to realize that they left out the need for a human player. The designer may toss the player a few scraps of things to do, but it's clear that the mechanism or simulation remains in the spotlight. If you are an inventor, you can do a lot to build fascinating dynamic systems—but don't forget that games must have human involvement as an integral part of the system and that players need to have long-term goals and reasons to play the game (the whole of the game), or it will be uninteresting to them.

Toymakers

Finally, some game designers are first and foremost toymakers. They love to make little pieces or mechanisms that don't really *do* anything but are still attractive and engaging, at least for a minute or so. Or they might be among those with highly specific domain knowledge—things like the climbing rate and ammunition capacity for a Sopwith Camel or the relative merits of different sorts of swords in medieval (or at least fantasy) combat, or the types of coral on a typical reef—or may just love digging in to find this kind of information.

Many game designers who start with the "nouns and verbs" of their design fit into the toymaker category. Maybe you want to make a game about cells in the immune system attacking invading viruses, and so you start with what you know (or anything you can find) about how a T-cell works. What the player does and why this is engaging or fun are questions that you may not think about right away or that you may have difficulty finding answers for. Having the ability to ground your design in specific parts and behaviors—tokens and rules, nouns and verbs—helps you create prototypes quickly. However, to make it into a game, you need to find ways to build interactive systems and find some goals for the player to pursue and experience.

Working Together to Find the Fun

The good news about these different views of game design is that once you find your starting point as a designer, you can extend your abilities into the other areas. Any one of these is great as a starting point, as long as you don't end there, too. The better news is that you can also find others who have different game design talents and work with them. It can be difficult and even frustrating for game designers with different design styles to work together, but the result is almost always far better and more engaging for the player as a result.

No matter which part of the game design process you prefer, you will need to extend yourself into the other areas and learn to listen to and work with those who see the game design process differently from you. A lot of game design comes down to being able to communicate your ideas, hear other people's ideas, and generally work together with those who have strengths that are different from yours. Understanding game design as systemic design helps illuminate these different views on games as systems and on game designers as system designers. That understanding should help you refine your skills and look for others who complement them.

A large part of *doing* game design is in the oft-repeated phrase "find the fun." You may start with a cool toy, an intriguing mechanisms, or a compelling experience—the parts, loops, and whole of a game—but you will need all three elements plus engaging interactivity to build a fun game. To do that, you need to apply your knowledge of systems to creating game systems and games *as* systems.

Designing Systemic Games

As a way to approach designing games as systems, we can look at the properties of effective systems in games and how they affect the process of game design.

Qualities of Game Systems

Achterman (2011) has provided helpful guidelines for building game systems. In his view, five qualities are the hallmarks of effective game systems:

- **Comprehensible:** As a designer, you have to understand your game as a system and the systems within it. Of course, your players have to be able to comprehend it, too. This is why both design documentation (for you) and presenting the game in such a way that players can build a mental model of it are so important.
- **Consistent:** Achterman points out the importance of having "rules and content [that] function the same in all areas of your game." It can be tempting to add an exception or a special case to fix a problem, but doing so tends to decrease the resilience of the system (which sets up the game for later problems) and makes it more difficult to learn. (This is similar to the discussion in Chapter 3, "Foundations of Games and Game Design," on elegance.)
- **Predictable:** Game systems should have predictable outputs for given inputs. While making games predictable helps players build mental models of the games, it can also be somewhat at odds with designing systems for emergence. Being predictable should not be taken as meaning that game systems should be obviously or boringly mechanistic. However, neither should your systems produce wildly different results for similar inputs, much less become brittle and break down due to unforeseen circumstances. You should at

least be able to know that you have accounted for any edge cases that might hurt a player's experience or provide them with a gap in the system to exploit to their advantage.

- **Extensible:** Building games systemically typically makes them highly extensible. Rather than depend on custom-created content "set pieces" (e.g., expensive hand-created levels), as much as possible you should create game systems such that content can be reused in new ways or created procedurally. You want to create parts and loops that can be used in multiple ways, not a single-use arc that makes for a complicated rather than complex set of relationships. While in a loop the parts affect each other cyclically, as veteran game designer Daniel Cook said, "An arc is a broken loop that you exit immediately" (Cook 2012). Designing in terms of loops rather than arcs also makes it easier to take a system and add it to a new game or put it in a new context, where it acts as a part in a new larger system. For example, you may decide that you want to add a whole new class of buildings for players to construct; if you have a general "building construction" system in the game, this is much easier to do than if you have to hand-craft another one. By designing game systems carefully, with only the needed parts and sufficient loops between them, you will be able to extend the systems internally or extend their use externally far more easily than if you rely on more static content or fractured, separated systems in the game.
- **Elegant:** As discussed in earlier chapters, elegance is often a hallmark of systems. This quality sums up the ones above. It goes beyond but is related to the quality of consistency discussed above. The following are some examples of elegance:
  - Creating a diverse space for players to explore based on only a few rules (Again, *Go* is the archetypal example of this.)
  - Having systemic rules with few exceptions that are easy to learn, where both predictable and emergent behaviors are possible
  - Enabling the system to be used within multiple contexts or to have new parts added within it

Tabletop and Digital Games

This book uses examples from both tabletop games—also called analog games, board games, physical games, and so on—and digital games—those played on a computer, console, tablet, or phone. From a game design point of view, there is a great deal of commonality between these types of games, no matter their genre or other differentiating attributes.

There is a great deal to be learned from studying tabletop games, even if you never plan to design one. Designing for situations in which the only "computing power" is in the players' heads and where all interaction must happen using tokens the players can physically manipulate presents a significant challenge. It constrains what you as a designer can do to bring a game concept to life and highlights the relationships between the game's tokens

and rules, loops, and overall experience. Digital games can hide a lot of game-designer laziness behind flashy graphics and narrative cut-scenes; tabletop games do not have that luxury.

In speaking to university theatre students, actor Terrence Mann said, "Movies make you famous, television will make you rich; but theatre will make you good" (Gilbert 2017). There is an analogy here to game design (not that any particular type of game design will necessarily make you rich or famous): designing tabletop games has the same sort of relationship to designing digital games that acting in theatre does to acting in movies. Like theatre, tabletop games are closer to the audience; you as a game designer can hide less, and must hone your craft in designing for this environment.

This is not to say that all game designers must design board or tabletop games, though it is good practice. But if at times you wonder why so many board games are used as examples when "modern" games are typically played on computer, this is the reason. Tabletop games have undergone every bit as much of a renaissance in the early 21st century as have digital games. As a systemic game designer, you can learn from both, and you may well find that designing tabletop games challenges your skills in ways that designing for games run on the computer does not.

The Process of Designing Games as Systems

Stepping down a bit from the abstract qualities we hope to find in game systems, we can look at the overall design process common to systemic game design (whether tabletop or digital).

This is necessarily an iterative process between designing the parts, the loops, and the whole. At first, this process may be iterative in your head, on a whiteboard, and on scraps of paper and then in documents and spreadsheets. Once the game begins to take shape, the iterative cycle of prototyping and playtesting discussed briefly below (and in more detail in Chapter 12, "Making Your Game Real") becomes important: it is far better to prototype fast and playtest early than to hope the idea you have in your head will spring forth fully formed like Athena from Zeus's skull. (They never do.) This process is the game designer's loop shown in Figure 5.1 (which is the same as Figure 4.3).

As stated earlier, it is possible to begin at any point in the systemic structure: with parts, loops, or the whole experience—as long as, having started with one, you move to the others so that they mutually support each other. With that reminder, for convenience here we will start with the whole, the architectural and thematic elements, and then move to the functional looping aspects, and finally move to the parts.

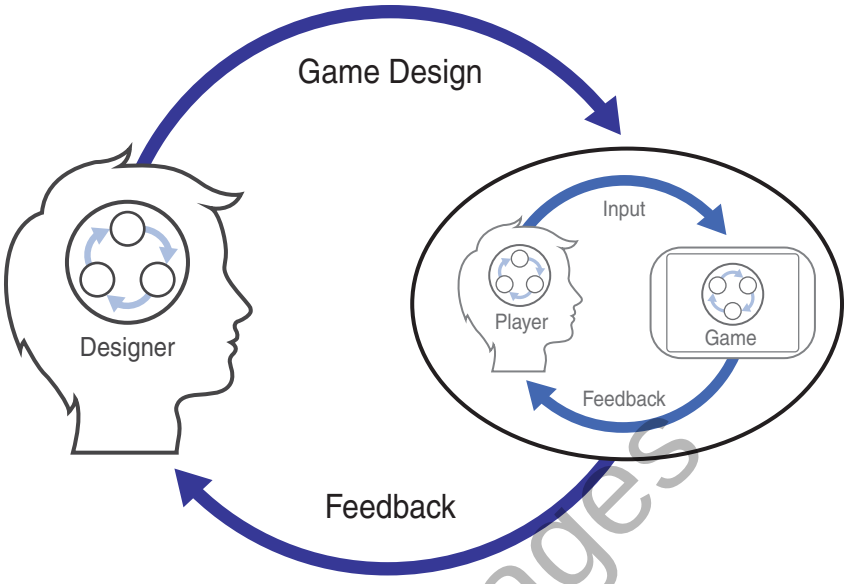

Figure 5.1 The game designer's loop enables you to iteratively design and test your designs

The Whole Experience: Thematic Architecture

As discussed in Chapter 3, the high-level design of a game has to do with the player's overall experience. We can separate this into architectural and thematic elements—the technical aspects of the *user experience* (how the game looks and feels) and the more ethereal, sometimes tacit qualities that define *what the game is about*. Understanding the whole of the game answers the question What is the point of the game (or a system within the game)?

As one example, in a recent conversation, Jason VandenBerghe, creative director on the game *For Honor*, said, "I believe that combat is an art form. The game sprung from that belief" (personal communication, December 2016). His desire was for the player to experience hand-to-hand combat as a lethal, dance-like form of art. While that desire is not enough on its own to support the game design, it is a compelling vision, a star to guide the game's developers and from which all the interactions and details of the game eventually arise.

Many times, game designers or entire development teams will launch themselves into the game development process without stopping to entirely clarify what the "whole experience" is that they want in their game. Questions of theme and vision seem frivolous; the team wants to get to making the game! However, as you will see in Chapter 11, "Working as a Team," having a shared, coherent vision of the game your team is making is the single most important indicator of success.

There are multiple aspects of any overarching vision, as discussed in the following sections. These aspects represent and point to more detailed elements that have to be articulated to get an idea of what the game will be.

The Game's World and History

To begin with, what is the world, and what is the player's point of view within it? You may be thinking of a gritty, cold-hearted world of spies and double-dealing—but is the player a spy working their way up in this world? A spy-master overseeing and pulling the strings on a sometimes wayward team of spies? Or possibly an old spy coming out of retirement for one last vengeful mission? Each of these paints a different picture and will take your game design in a different direction.

To fill in the world somewhat, what are the major events in its history—those that are applicable to the players? If you're a storyteller, you may have to resist the urge to write 100 pages of world lore. If you have the time and money, and especially the experience to know what's useful and what's not, then you can indulge yourself in this; you will likely add important details to the game world that make it come to life all the more vividly. But if you have any time or budget constraints, or if you're just starting out, you should avoid the siren song of diving too deeply into the backstory. You need to know what the world is and what it's about, but to start with, you can do this in a page or two of text. You shouldn't write any more than you need to support the rest of the design. Later, as the game is beginning to come together, you can flesh out the deep, tragic history of the city where the streets hold a million secrets.

Narrative, Progression, and Key Moments

The game world's history is its past. Its present and future are contained in the game narrative. Does your game have a predefined story the player has to work within? Are there larger events happening around the player that grow out of the large-scale history but that leave room for the player to make their own decisions? Or is the game's history a jumping-off point for the player, where what's past is prologue, and there is little in the way of continuing narrative to guide the player's actions?

Understanding your game's world and (some) of its history will also help you begin to define major events that happen in the game, the player's goals and progression through it, and "key moments"—short moments or stories that you can tell that help communicate meaningful, climactic points for the player.

Art, Monetization, and Other Whole-Experience Concerns

There are a variety of questions to work through at the level of the whole-game experience: Will the game's art style be 2D or 3D? Painterly, cel-shaded, or super-realistic? How does your choice reflect the game's heart and theme to the player? Closely aligned with this is the way the player interacts with the game—the user interface and user experience, often referred to as UI/UX. Even monetization design—how your game makes money—is something you have to consider at this stage.

In Chapter 6 we will look in more detail at the process of designing and documenting the gameplay experience as a whole. For now, keep in mind that it doesn't matter so much whether you start with a high-level, blue-sky creative vision that you then support with underlying

loops and parts or whether you arrive here after first nailing down those dynamic and specific aspects; either way, you will iterate back and forth between them as you refine your ideas. What matters is that before you begin developing your game—before you assure yourself that you know what the game *is*—you have this theme and vision, the whole of the player's experience, clearly articulated and shared by your team.

Systemic Loops and Creating a Space for Play

Chapters 3 and 4, "Interactivity and Fun," discuss the game's loops: the game's dynamic model of its world, the player's mental model of the game, and the interactions that happen between the player and the game. Designing and building these loops and the structures that support them is the heart of being what is often loosely referred to as a "systems designer." In addition to the overview here, this topic is explored in detail in Chapter 7.

In creating a space for the player to explore and inhabit—rather than a singular path for them to follow through the game—you need to define the game's systems. These systems need to support the theme and desired player experience, and they must work interactively between the game and the player. You need to specify and create (via iterative prototyping and playtesting) the player's core loops, explicit goals, and the way they progress through the game.

Creating systems like this may be the most difficult part of game design: it requires that you envision the system as it uses the game's tokens and rules to create an experience that is hard to see clearly in advance. Of course, you don't have to do this all at once—which is why prototyping and playtesting are so important—but being able to imagine multiple looping systems well enough to record their designs and implement them is nevertheless a daunting task. For example, in many games, the systems controlling resource production, crafting, wealth production, and combat all have their own internal workings, and all interact with each other and the player to create the player's experience. Getting all these to work on their own and contribute to a systemic whole requires skill, patience, and resilience in the face of repeated attempts when something just doesn't quite work.

Balancing Game Systems

Part of making game systems is ensuring that all parts defined by the game are used and balanced against each other and that every system in the game has a clear purpose. If you add a quest system to your game and players ignore it, you need to understand why it isn't contributing to their experience and determine whether to remove it or fix it so that it does. Chapter 9, "Game Balance Methods," and Chapter 10, "Game Balance Practice," go into this process in detail.

The Structural Parts: Tokens, Values, and Rules

It may be that you started the game design process with an idea for a fun looping mechanism or interaction. Or maybe you started with the kind of experience and feeling you want the players to have, and so you're defining the game and interactive loops. Or in some cases you may start with an idea for the building blocks out of which you want to construct your game. In any

case, before the game is really a game, you need to situate the game's functional loops into the context of the whole—the game experience—and also create the structural parts of the game's systems.

You first read about the tokens, values, and rules in a game in Chapter 3. You will see them again in detail in Chapter 8. For now, in terms of working as a systemic designer, you should understand that the process of nailing down exactly what is going on in a game—getting past the hand-waving descriptive stage and being able to implement the game—is vital. You don't have a game without it.

This aspect of game design is sometimes called "detailed design," and it is where the game design becomes entirely specific. Does that sword have a weight of 3 or 4? A cost of 10 or 12? How many types of troops, or horses, or flower petals are there in the game, and what differences do these numbers make to the overall gameplay? Tracking and specifying these structural parts of the game has been called the "spreadsheet-specific" part of game design. This is a crucial part of systemic design; it is in many ways how the game becomes real. Such specific design is needed for balancing the different tokenized parts against each other to make the game a cohesive whole rather than allow it to become separate systems that can fly apart.

The issues you need to think about here are how to specify tokens that represent the objects in the game—the player, other people, nations, creatures, spaceships, or whatever the operative units are within your game—and give each of them sufficient attributes, values, and behaviors to define them. One way to think of this is to answer the question What is the smallest number of attributes, states, and behaviors you can use to support the game's systems and provide the overall gameplay experience you want?

Related to this are the issues of how to make obvious to the player what the tokens in the game are, what they do, and how the player can affect them. This in turn feeds into the game's UI/UX—how the board or screen is laid out to present the necessary information about the game to the player. This cannot be specified until you know what the necessary information is. At the same time, approaching this issue by asking what sorts of information you think the player needs to know to play the game can itself help clarify the tokenizing process.

Chapter 8 talks about this process in more detail, including how to create complex objects, game pieces, or tokens by having a small number of general attributes that interact with each other to create their own subsystems within the larger game systems. Chapter 8 also discusses the importance of inter-object behaviors and how to avoid "easy win" or other gameplay-killing tokens in your games.

Revisiting the Systemic Design Process

As a systemic game designer, your loop—the designer's loop—involves cycling between seeing the game as a whole, as systems, or as individuated parts (see Figure 5.2). You need to be able to see them all at the same time and how they affect each other. You also need to

be able to dive into any one in detail, depending on what's needed by the game design. It's important that you not focus on any one level to the exclusion of the others; you also don't want to continue to work ineffectively on any one of them. When you find yourself pushing on one level without any real effect, it can often help to switch and work from the point of view of the other levels to help reveal what you need in another. If you can't quite get the experience down, explore the tokens and how they work; see how they inform the experience. Or if you have the experience clearly in mind but can't quite specify the tokens, see what the systems tell you about how those have to work. At the same time, don't let yourself avoid tokenizing your systems, making sure there are interesting interactions, or ensuring a cohesive theme because one or more of these aren't in your comfort zone as a game designer. All of these are necessary for any working game, and all are necessary activities for a systemic game designer.

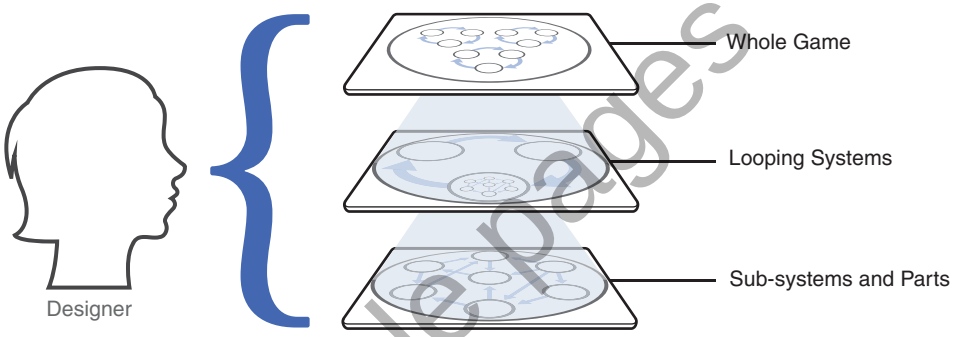

Figure 5.2 As a game designer, you need to be able to see the parts, the loops, and the game's whole experience all at the same time and zoom in on any one of them as needed

Analyzing Games from a Systems View

Working as a game designer doesn't just mean designing your own games; it also means playing and analyzing a lot of other people's games. It's important to be able to understand what makes other games work—or not work in particular areas.

You can follow the same systemic structure for analysis as for design. It involves looking at the whole experience, including how you build your mental model of the game; the game's internal and interactive loops; and the rules and tokens that make those up. By carefully identifying and separating these, you can gain insight into the decisions made by the game's designer and improve your own designs as a result.

When beginning to play a game for the first time, examine how you go about building your own mental model of it: Do you understand the setting and theme? What surprises you about it? What concepts about the game did you find to be important, incomplete, or hard to understand as you learned the game? How might the game have increased your engagement early on?

While playing and after playing, think about the whole of the experience you had. What kind of experience and feelings do you think the game designer was trying to elicit in you as a player? Were there particular aspects of the game that supported or detracted from your experience?

What visual and interactive elements of the game support its theme and the desired player experience? What can you infer about the game designer's intent for the game, based on the art style and interactive aspects?

What specific game systems can you identify in the game? Are there systems that operate independently of the players, or do they all rely on the players doing something first? The board game *Power Grid* is a great example of a (nondigital) game that has systems that operate mostly outside player control. For example, in this game there is a simple but highly effective depiction of supply-and-demand economics: as players buy more of any one kind of fuel, the price for it goes up until its supply is replenished on the next turn (see Figure 5.3).

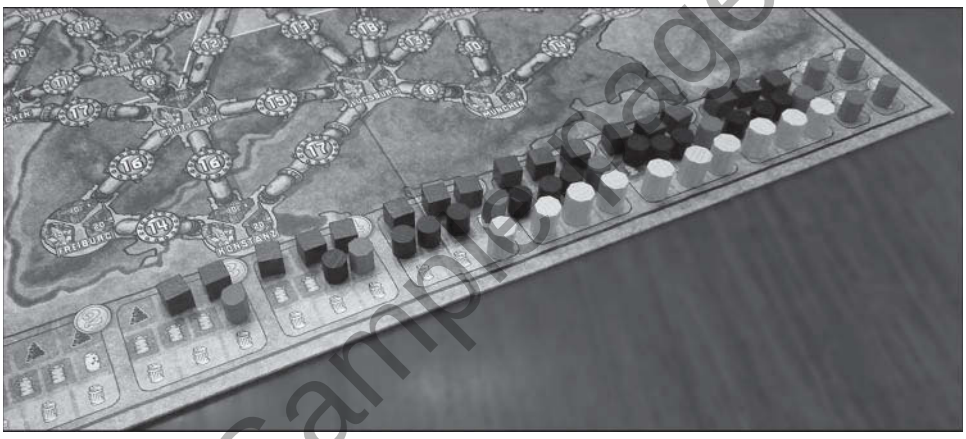

Figure 5.3 The board game *Power Grid*, showing the track representing prices for the resources coal, oil, trash, and nuclear fuel. As players purchase each and supply decreases, its price rises. Supply is replenished each turn, driving prices lower if the fuel is not used

Continuing with the analysis overview, as a player in a game, how do you progress, and what reinforcing loops can you identify? What balancing loops are there that push back against player advancement or that keep one player who outstrips others early on from simply winning the game?

What are the primary forms of interactivity in the game? How does the game allocate its interactivity budget? Is this a game of strategy and socializing, or one of quick thinking and fast action? Do the ways you as a player interact with the game help establish the game's theme, or do they work against it?

Finally, what are the particular tokens and rules—the atomic parts of the game with their values and behaviors? Do they support the desired gameplay experience or get in its way? Having learned one system in the game, can you transfer how that works to another part of the game, or are there lots of rules to learn, each with its own exceptions—so that you have to spend a lot of time thinking about how to play the game?

Often the art style of a game is expressed in its individual tokens, sometimes in surprising ways. For example, the tabletop game *Splendor* is about building up your business as a gem merchant, starting with individual mines and ending with courting the favor of various nobles. The physical pieces in the game are like poker chips. They represent individual gems, and each has an unusual amount of heft. Their weight subtly adds to the desired experience of the game, even though, like the rest of the art (and most art in games), it is nonfunctional.

As you analyze games by examining their parts, loops, and wholes, you will begin to see commonalities across them, as well as how each is unique. Understanding the similarities and differences will help you improve your own designs—avoiding the mistakes of others, springboarding off their good ideas, and keeping your game design fresh and engaging.

Prototyping and Playtesting

A final important part of working as a systemic game designer is iteratively getting feedback. Game design is necessarily a process of repeatedly testing and refining game design ideas in the service of an overall vision for the game. Game ideas will not make it from your mind to their final form in front of the player without having gone through many changes first. It's common for almost everything about a game except for its single unifying vision to change multiple times during development.

As an example from a related creative field, making movies, Ed Catmull, president of Pixar, has been open about the many gyrations that films at his studios go through. "All of our movies suck at first," he said when speaking to aspiring movie animators. He clarified that statement by adding, "A lot of people don't believe me when I say that. They think I'm being self-effacing or modest, but I don't mean it in that sense. I mean it in the way that the film sucks." He went on to discuss the many story changes that the movie *Up* went through during its development: it started with a story about a kingdom in the sky with two princes who didn't like each other, who fall to earth and end up meeting a giant bird named Kevin. That version went through a huge number of changes. By the time they completed the movie, he said, "All that was left was the bird and the word 'up'" (Lane 2015).

The same sort of thing happens in games. While your game may not change as drastically as a movie like *Up*, you must be prepared for many iterations—many cycles through the creative process. This means you have to be willing to test your ideas over and over again, learning and changing them as you go. And it means you have to be humble enough to change an idea or throw it out if it isn't working. Iterating and "finding the fun" inevitably means throwing away

a lot of work—drawings, animations, programming, design documents, and so on. You cannot cling to something you have worked on just because you put a lot of time into it. If you do, you will be settling for an idea that is okay (or mediocre) when with a little more work and polish it could have been great.

To iterate effectively on game designs, you need to make them real. The only way to do this is to make early versions—prototypes—and test them. You may start with drawings on a whiteboard or pieces of paper and coins being pushed around on a table—anything to start actually playing with the idea you have. Most of your prototypes will be varying degrees of ugly or unfinished, converging on the full, finished, and polished product at the end. The point is to take your game design out of the realm of ideas and into real implementations that can be played and tested—and to do so as quickly and often as possible.

Playtesting is how you validate your prototypes—or, more often, how you find out where your game design is broken. Developing a game designer's intuition for what will work or not is important, but even for the most experienced designers, it is never a substitute for testing the gameplay on players who have never seen it before. As Daniel Cook has said, without implementation and playtesting a game design remains an "ineffectual paper fantasy" (Cook, 2011b). You will need to test your design ideas with other people early and often to keep your game on track.

We will return to the topics of prototyping and playtesting often in the following chapters, particularly in Chapter 12. For now, understand that a core aspect of working as a game designer is having the humility and creative flexibility to test and refine your game design ideas based on what others think of them. You will need to make fast, often ugly prototypes, and you will need to test them with potential players repeatedly during design and development. The bright shining idea you have in your head will never survive contact with reality without change—most likely a lot of change.

Summary

This brief chapter provides an overview of what it means to work as a systemic game designer. While getting started on a new game design can be truly daunting, by breaking down the game into its parts, loops, and wholes—not necessarily in that order—you can begin to get a handle on defining the game at each of those levels.

The coming chapters add more detail to the topics discussed here. Chapter 6 examines the whole of the game experience in more detail—how you discover it, document it, and set up for creating the underlying systems. Chapter 7 revisits the game's functional loops, this time using the knowledge of systems thinking and game loops to specify the particular loops for your game. Then Chapter 8 looks again at the game's parts and how to create these "spreadsheet-specific" tokens, values, and rules.