Chapter 5

Working with System Dynamics

In the previous two chapters, I discussed games in terms of their formal and dramatic elements. Now let's look at how the elements of games fit together to form playable systems and how designers can work with system properties to balance the dynamic nature of their games.

A system is defined as a set of interacting elements that form an integrated whole with a common goal or purpose. General system theory, the idea that the interaction among elements of systems can be studied across a wide variety of disciplines, was first proposed by the biologist Ludwig von Bertalanffy in the 1940s. Variations of system theory have evolved over time, each focusing on different types of systems. My goal here is not to examine all the various disciplines of system theory but rather to investigate how we can use an understanding of basic system principles to control the quality of interactions within our game systems as well as the growth and change of those systems over time.

Games as Systems

Systems exist throughout the natural and human-made world wherever we see complex behavior emerging from the interaction between discrete elements. Systems can be found in many different forms. They can be mechanical, biological, or social in nature, among other possibilities. A system can be as simple as a stapler or as complex as a government. In each case, when the system is put in motion, its elements interact to produce the desired goal, for example, stapling papers or governing society.

Games are also systems. At the heart of every game is a set of formal elements that, as we have seen, when set in motion, create a dynamic experience in which the players engage. Unlike most systems, however, it is not the goal of a game to create a product, perform a task, or simplify a process. The goal of a game is to entertain its participants. When I discussed formal and dramatic elements, I showed how games do this by creating a structured conflict and providing an entertaining process for players to resolve that conflict. How the interaction of the formal and dramatic elements is structured forms the game's underlying system and determines a great deal about the nature of the game and the experience of the players.

As I mentioned earlier, systems can be simple or complex. Systems can produce precise, predictable results, or they can produce widely varied, unpredictable effects. What type of system is best for your game? Only you can determine this. You might want to create a game in which there is a certain amount of predictability, in which case you might design a system with only one or two possible outcomes. On the other hand, you might want to create a very unpredictable system, in which there are countless

possible outcomes determined by the choices of the players and the interactions of the game elements.

To understand why it is that systems act in such different ways and be able to control the type of system elements that affect the outcome of your own games, we need to first identify the basic elements of systems and look at what factors within these elements determine how a system acts in motion.

The basic elements of systems are objects, properties, behaviors, and relationships. Objects within the system interact with each other according to their properties, behaviors, and relationships, causing changes to the system state. How those changes are manifested depends on the nature of the objects and interactions.

Objects

Objects are the basic building blocks of a system. Systems can be thought of as a group of interrelated pieces called objects, which can be physical, abstract, or both, depending on the nature of the system. Examples of objects in games might be individual game pieces (such as the king or queen in chess), in-game concepts (such as the bank in Monopoly), the players themselves, or representations of the players (such as the avatars in an online environment). Areas or terrain can also be thought of as objects: the squares on a grid board or the yard lines on a playing field. These objects interact with other game objects in the same way that playing pieces do, and they need to be defined with the same amount of consideration.

Objects are defined by their properties and behaviors. They are also defined by their relationships with other objects.

Properties

Properties are qualities or attributes that define physical or conceptual aspects of objects. Generally, these are a set of values that describe an object. For example, the attributes of a bishop include its color (white or black) and its location. The properties of a character in a role-playing game can be much more complex, including variables such as health, strength, dexterity, experience, level, as well as its location in the online environment, and even the artwork or other media associated with that object.

The properties of objects form a block of descriptive data that can be essential to determining interactions of objects in a game system. The simplest types of game objects have very few properties, and those properties do not change based on gameplay. An example of this type of object would be the checkers in a checker game. Checkers have only three properties: color, location, and type. While the location of checkers changes, their color never does. The type of checker can change from "normal" to "king" if it reaches the other side of the board. These three properties completely define the state of each checker within a game.

What would be an example of a game object with more complex properties? How about a character in a role-playing game? Figure 5.1 shows the main properties of a character from Diablo. As you can see, this list defines an object of much greater complexity than our first example. Also, the properties of this object change over the course of the game, and not in a simple binary way, like the checker. Because of its greater complexity, the object in this case will probably have less predictable relationships with other objects in the system than would a simple object like a checker.

5.1 Diablo: character properties

Exercise 5.1: Objects and Properties

Choose a board game you have at home in which you are able to clearly identify the objects and their properties. Strategy board games often have objects with properties that are easy to identify. Make a list of all of the objects and their properties in the game you have chosen.

Behaviors

The next defining characteristics of objects in a system are their behaviors. Behaviors are the potential actions that an object might perform in a given state. The behaviors of the bishop in chess include moving along any of the diagonals radiating from its current position until it is blocked by or captures another piece. The behaviors of the role-playing character described previously might include walking, running, fighting, talking, using items, etc.

As with the sheer number of properties, the more potential behaviors an object has, the less predictable its actions within the system. For example, let's take the checkers example again. A "normal" checker has two potential behaviors: It can move diagonally one space or jump diagonally to capture another piece. Its behavior is restricted by the following rules: It can only move toward the opponent; if it can jump an opponent's piece it must do so; and if possible, it can make multiple jumps in a turn. A "king" has the same behaviors, but it does not have the rule regarding moving toward the opponent; instead it can move forward or backward on the board. This comprises all the potential behaviors of the checker objects in the game (obviously, a very limited set of behaviors, which result in a fairly predictable game pattern).

Now, let's look at the Diablo character again. What are the behaviors of this character? It can move: running or walking. It can attack using weapons in its inventory or skills like magic spells. It can pick up objects, converse with other characters, learn new skills, buy or trade objects, open doors or boxes, etc. Because of the range of behaviors available, the progression of this object through the game is much less predictable than the poor checker.

Does this make the game inherently more fun, however? I discuss this in the sidebar "Deconstructing Set" on page 134 when I analyze the system of Set, a simple, yet compelling, card game; the fact is, a greater complexity of gameplay does not always equate to a more enjoyable experience for the player. For now it is simply important to note that the addition of more potential behaviors tends to add choice and lessen the predictability of the outcome in a game.

Exercise 5.2: Behaviors

Take the list of objects and properties you created in Exercise 5.1 and add a description of the behaviors for each object. Consider all behaviors in different game states.

Relationships

As I mentioned earlier, systems also have relationships among their objects. This is a key concept in design. If there are no relationships between the objects in question, then you have a collection, not a system. For example, a stack of blank index cards is a collection. If you write numbers on the cards or mark them in several suits, then you have created relationships among the cards. Removing the "3" card from a sequence of 12 will change the dynamics of a system that uses those cards.

Relationships can be expressed in a number of ways. A game played on a board might express relationships between objects based on location. Alternatively relationships between objects might be defined hierarchically, as in the numerical sequence of cards described previously. How relationships between objects in a system are defined plays a large part in how the system develops when it is put in motion.

The hierarchy of cards is an example of a fixed relationship: The numerical values fix a logical relationship between each of the cards in the set. An example of a relationship that changes during gameplay is the movement of the checkers on our

checkerboard: Pieces move toward the other side of the board, jumping and capturing the opponent's pieces along the way. As they do so, their relationship to the board and to the other pieces on it continually changes.

Another example of a relationship is the progression of spaces on a board game like Monopoly. This is a fixed, linear relationship that constrains gameplay within a range of possibilities. On the other side of the spectrum, objects might have only loose relationships with other objects, interacting with them based on proximity or other variables. An example of this would be The Sims, where the relationships of the characters to other objects are based on their current needs and the ability of the objects in the environment to fulfill those needs. These relationships change as the characters' needs change; for example, the refrigerator is more interesting to a character who is hungry than a character who has just eaten a huge meal.

Change in relationships can also be introduced based on choices made by the players. The checkers game exhibits such change: Players choose where to move their pieces on the board. There are other ways to introduce change into game relationships. Many games use an element of chance to change game relationships. A good example of this is seen in most combat algorithms. Here is an explanation of how the combat algorithm works in WarCraft II.[1]

Each unit in the game has four properties that determine how effective it is in combat.

- Hit Points: These indicate how much damage the unit can take before dying.
- Armor: This number reflects not only armor worn by the unit, but also its innate resistance to damage.
- Basic Damage: This is how much normal damage the unit can inflict every time it attacks. Basic damage is lowered by the target's armor rating.
- Piercing Damage: This reflects how effective the unit is at bypassing armor. (Magical attacks, like dragon's breath and lightning, ignore armor.)

When one unit attacks another, the formula used to determine damage is: (Basic Damage − Target's Armor) + Piercing Damage = Maximum Damage Inflicted. The attacker does a random amount of damage from 50–100% of this total each attack. To see how this algorithm tends to introduce chance into the relationship between objects, or units as we have been calling them, let's look at an example from the strategy guide on Battle.net:

> *An ogre and a footman are engaged in combat. The ogre has a Basic Damage rating of 8 and a Piercing Damage rating of 4. The footman has an Armor value of 2. Every time the ogre attacks the footman, it has the potential to inflict up to (8 − 2) + 4 = 10 points of damage, or it could inflict as little as 50% damage, or 5 points. On average, the ogre will kill the 60 Hit Point footman in about 8 swings.*
>
> *The poor footman, on the other hand, with a Basic Damage of 6 and a Piercing Damage of 3, will only inflict 3 to 5 points of damage each time he attacks the ogre, which has an Armor value of 4 (that's (6−4) + 3 = 5). Even if the footman is extremely lucky and does the maximum amount of damage with every attack, it will take 18 swings to kill that 90 Hit Point ogre. By that time, the ogre will have pounded him into mincemeat and moved on.*

This example actually shows two ways of determining relationships: chance and rule sets. As can be seen by the calculation, there is a basic rule set

5.2 WarCraft II: going up against an ogre

determining the range within which damage can fall. Within that range, however, chance determines the final outcome. Some games tend more toward chance in their calculations, while others tend more toward rule-based calculations. Which method is best for your game depends on the experience you want to achieve.

Exercise 5.3: Relationships

Take the list of objects, properties, and behaviors you created in Exercises 5.1 and 5.2 and describe the relationships between each object. How are these relationships defined? By position? By power? By value?

System Dynamics

As I have noted, the elements of systems do not work in isolation from each other. If you can take components away from a system without affecting its functioning and relationships, then you have a collection, not a system. A system, by definition, requires that all elements be present for it to accomplish its goal. Also, a system's components must typically be arranged in a specific way for it to carry out its purpose, that is, to provide the intended challenge to the players. If that arrangement is changed, the results of the interaction will change. Depending on the nature of the relationships in the system, the change in results might be unnoticeable, or it might be catastrophic, but there will be a change to some degree.

Let's say that my example on page 132 of the ogre and the footman from WarCraft II was changed; instead of using the properties of basic damage, piercing damage, and target's armor to determine the range of potential damage from an attack, let's assume the damage dealt by each unit was just a random number between 1 and 20. How would this change the outcome of each individual battle? How would it change the overall outcome of the game? What about the value of resources and upgrades?

If you said that the element of chance in the game would increase in both individual combat encounters and overall outcome, you are right. Also, the value of resources and the upgrades available via those resources would disappear because upgrades to units and armor would mean nothing in terms of determining outcome. The only strategy open to players in this game would be to build as many units as possible because sheer numbers would still overwhelm in battle. So by changing the relationship between units in combat, we have changed the overall nature of the WarCraft II system.

On the other hand, if we changed the original damage calculation by removing the random element at the end and assuming that all units would deal maximum damage, how would that change the outcome of each battle? In this case, a player would be able to predict exactly how many attacks would be necessary for any one unit to destroy another. This sense of predictability would affect not only strategy but also player engagement. While in the first example of a completely random damage system, player decisions lose much of their strategic value, the example of a completely nonrandom damage system would diminish the unpredictability of individual encounters and the overall game as well.

Another important feature to understand about the interaction of systems is that systems are greater than the sum of their parts. By this I mean that studying all of the individual qualities of each system element in isolation does not equal studying these elements in relationship to each other. This is important for game designers to realize because games can only be understood during play when their dynamics become evident. Katie Salen and Eric Zimmerman call game design a "second-order" problem because of this,[2] meaning that when we design a game, we cannot directly determine the player experience—we cannot determine exactly how the rules will play out. We have to craft a "possibility space" as best we can, and playtest it as rigorously as possible, but in the end, we just do not know how each and every play of the game will go.

Deconstructing Set

Set is a card game designed by Marsha Falco in 1988. At the time, Falco was studying as a population geneticist in Cambridge, England, trying to understand if German shepherds that get epilepsy actually inherit it. To help her understand the variables, she wrote information about each dog on file cards; she drew a symbol to represent a block of data, indicating different gene combinations. One day she found her kids playing with her research cards—they had made a game out of them. The game was so fun that they went on to publish it as a family business. The game became an instant classic, winning a number of awards, including the Mensa award.

The Rules of Set

The system of Set is quite elegant. The game is played with a special card deck made of 81 unique cards. The cards are the basic objects in the game, and each has a set of symbols with four properties: shape, number, pattern, and color; there are three options for each property. The diagram below shows how the number of properties and the options for each adds complexity to the deck, which is measured by the number of unique cards.

The procedures of Set are also quite simple. The deck is shuffled; 12 cards are then dealt out, as shown in the next diagram. The players all look at the cards, searching for "sets." A set consists of three cards in which each property is either all the same or all different. For example, in this layout, A1, A2, and A3 are a set because (1) shape = all the same, (2) number = all different, (3) pattern = all different, and (4) color = all different. A1, A4, and C1 are also a set because (1) shape = all different, (2) number = all the same, (3) pattern = all different, and (4) color = all different.

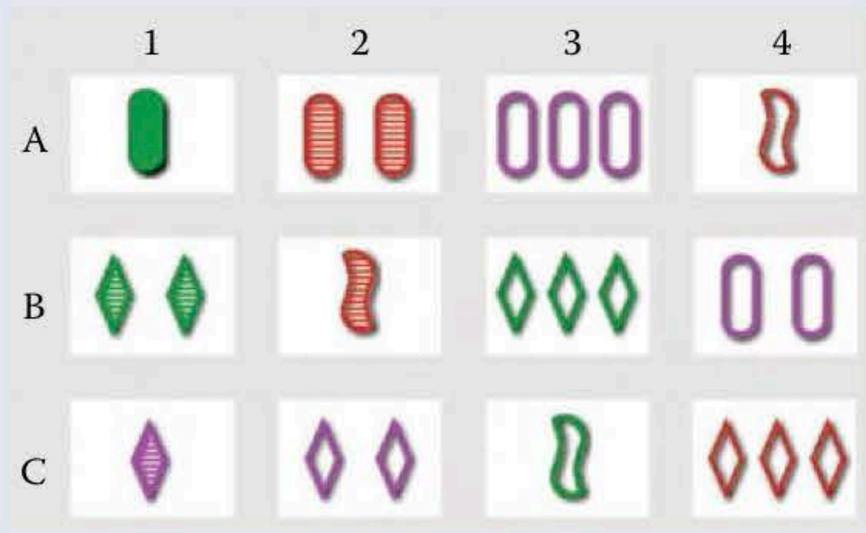
	1	2	3	Unique cards
Shape	Oval	Diamond	Squiggle	3
Number	1	2	3	9
Pattern	Solid	Clear	Striped	27
Color	Green	Red	Purple	81

Elements of Set and how they contribute to complexity

When a player sees a set, she calls out, "Set!" and points out which cards she believes make a set. If she is correct, she takes the cards; three more cards are dealt and play begins again. If she is incorrect, she must give back one of her sets to the discard pile. When there are no more cards, the player with the most sets wins.

Analyzing Set

As we have said, the design of Set is quite elegant. If you look closely at the cards in the figure at right, you will see that for any two cards you choose, you can describe the third card you would need to create a set. For example, look at B2 and C4. What card would need to create a set including these two cards? First, these cards have different shapes, so you would need another card with a different shape: an oval.



Set playing cards

Second, these cards have different numbers, one and three, so you need a card with two ovals. Third, these cards have different patterns, so you need a card with a different pattern: solid. Fourth, these cards have the same color, red, so you need another red. To make a set with B2 and C4, you need a card with two solid red ovals—there is only one such card in the deck, and it is not shown, so we cannot make a set with these two cards.

	1	2	3	Unique cards
Shape	Oval	Diamond	Squiggle	3
Number	1	2	3	9
Pattern	Solid	Clear	Striped	27
Color	Green	Red	Purple	81
Background	White	Black	Grey	243
Border	Silver	Gold	Onyx	729
Animation	Still	Blinking	Rotating	2187

Elements of Set with added properties

Now, how did Marsha Falco decide on this system configuration for the game of Set? Why not more properties? Or fewer? Why not more options for each property? As discussed in the analysis of Mastermind and Clue on page 138, the complexity of a system is greatly affected by the underlying mathematical structures. In Set, a deck of 81 cards provides a challenging, yet playable, number of possibilities. When learning to play Set, players often will remove the property of color to make the experience simpler. Without the property of color, the deck consists of only 27 cards, and it is much easier to find a set. After new players get the hang of it, they add back the remaining cards and the additional complexity that comes with a deck of 81 cards.

Imagine adding one more property—a background color, for example. As shown in the figure at top, this would create a deck of 243 cards. If we add a background border, our deck has 729 cards. Let's say we are making a digital version of Set. Now we can add animation! Should we? Well, that would mean there are 2187 cards in our digital game of Set. For a player trying to apply the rules of the game, there are now seven properties to consider and about 30 times less probability that the card you need to make a set will be dealt to the current display. You can see that adding this level of complexity has probably *not* improved your player experience. In fact, it is likely that this version of Set is unplayable.

	1	2	3	4	Unique cards
Shape	Oval	Diamond	Squiggle	Square	4
Number	1	2	3	4	16
Pattern	Solid	Clear	Striped	Hatched	64
Color	Green	Red	Purple	Yellow	256

Elements of Set with added option to original properties

The next figure shows another possibility—adding another option to each of the original properties. This does not change things quite as much; at least the deck is only 256 cards, only 3 times more complex than the original game system. But the game is already quite difficult. If you want to see what this change does to the player experience, build your own Set deck with the new option and playtest it.

Conclusion

This analysis deals with a game that is, compared to many digital games, quite simple. However, as you can see, changing just a few of the system elements can exponentially change the complexity of that simple system and the player experience. It is critical to understand the mathematical structures of your own game design and to test differing levels of complexity by adding to or deleting from your properties. One way to do this is the way we have done here with Set: build a matrix and calculate the level of complexity mathematically. And always keep in mind: A more complex mathematical solution might not offer the most satisfying gameplay result. The goal is always to build a system that is complex enough to delight and surprise your players, but not to confound or frustrate them.

Exactly how the dynamics of any given game system are affected by the properties, attributes, and relationships of its objects is difficult to generalize. A good way to understand how these elements can affect each other is to look at some example systems—ranging from very simple to fairly complex—that exhibit various types of dynamic behavior.

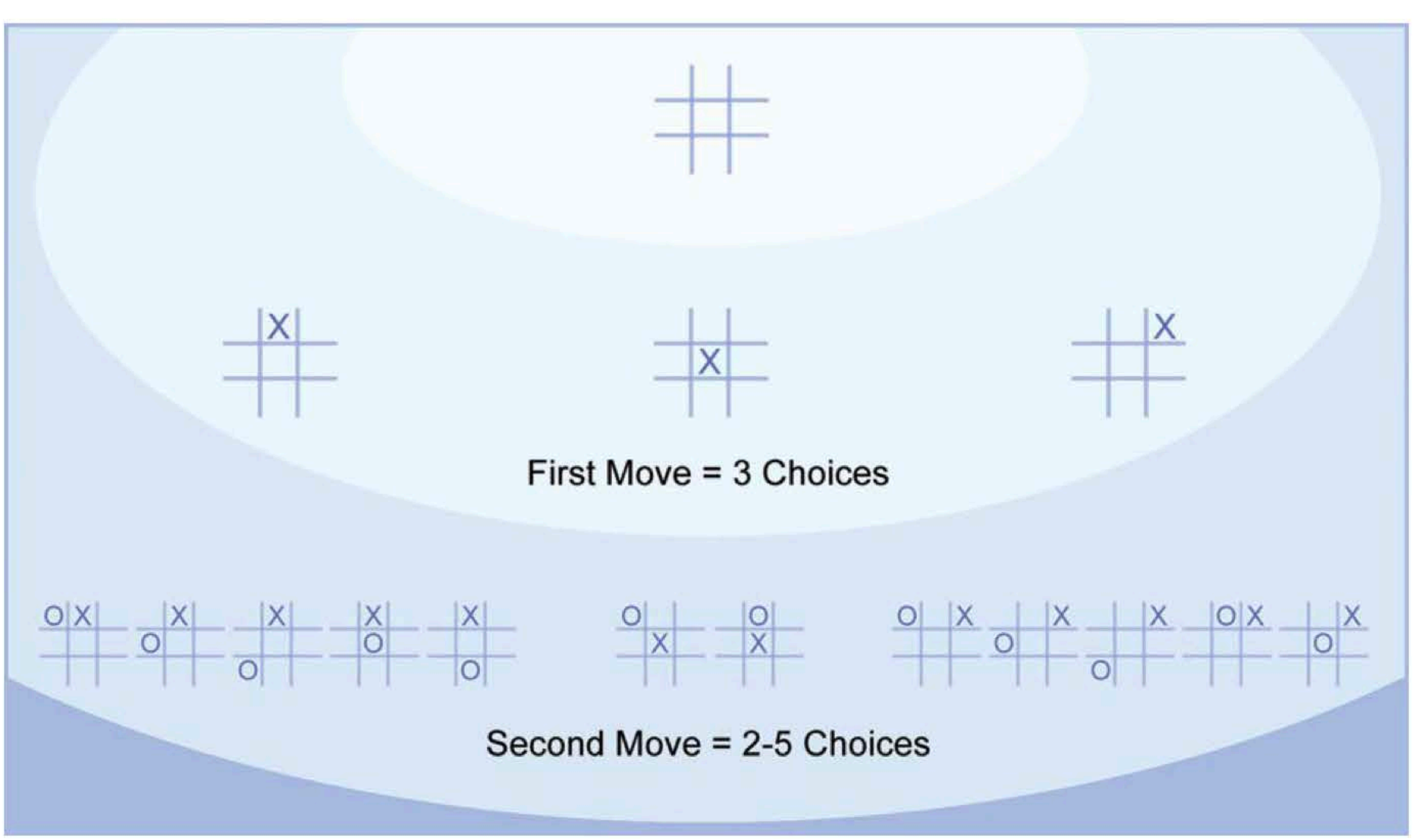
Tic-Tac-Toe

The objects in tic-tac-toe are the spaces on the board. There are nine of them, and they are defined by their properties, behaviors, and relationships. For example, their properties, are "null," "x," or "o." Their relationships are defined by location. There is one center space, four corners, and four sides. When the game begins, the relationship between the spaces is such that there are only three meaningful choices for the first player: center, corner, or side.

The second player has between two and five meaningful choices, depending on where the first player put an "x." You can see by the diagram of potential moves that playing the first "x" in the center reduces the amount of significant next moves to two. Playing the first "x" in a corner or on a side creates up to five next moves.

If I continued this diagram of tic-tac-toe out to its conclusion, you would see that the tree of possibilities is not very large. In fact, when you learn the best possible moves to make, you can always win or tie at this game because of its ultimate simplicity. What is it about tic-tac-toe that makes its system so easy to learn?

First, you can see that the game objects themselves are simple: They have only three properties and one behavior. Also, their relationships to each other are fixed: The locations of the spaces on the board do not change. Because of both the number of objects (i.e., the size of the board) and their relationships, the system has only a few possible outcomes, and these are all completely predictable. And as a result of its limited possibilities for play, tic-tac-toe tends to lose the interest of players after they learn the optimal moves for any given situation.



5.3 Partial tic-tac-toe game tree

Chess

An example of a system that has more than one type of object, and more complex behaviors and relationships between objects, is chess. First, let's look at the objects in chess: There are six types of units plus 64 unique spaces on the board.

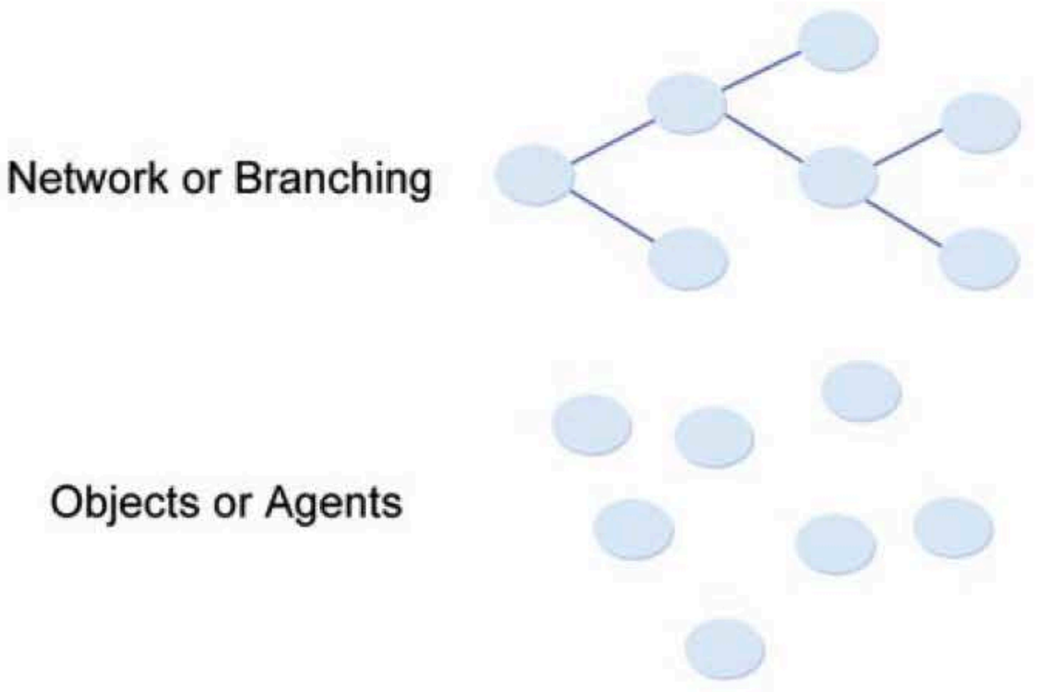
Each unit has several properties: color, rank, and location, as well as a set of behaviors. For example, the white queen has a beginning location of D1 (the space at the intersection of the fourth rank of the first file). The movement behavior of the queen is that it may move in any straight line horizontally, vertically, or diagonally, as long as it is not blocked by another piece. Alone, these properties and behaviors do not make the objects in chess more complex than the spaces on the tic-tac-toe board. However, the varied behaviors of the objects and relationships between them do make the emergent gameplay more complex. Because each unit has specific behaviors in terms of movement and capture, and because those abilities create changes to their locations on the board, the relationships between each unit are effectively changed as a result of every move.

While it is theoretically possible to create a tree similar to the one I drew for the opening moves of tic-tac-toe, it quickly becomes clear that the complexity of potential outcomes beyond the first few moves makes this a useless and physically impossible process. This is not the way players tend to approach the game, and it is not even the way that computer chess applications have been programmed to decide the best move. Instead, both players and successful programs tend to use pattern recognition to solve problems, calling up solutions from memories of previously played games (or a database in the case of the computer) rather than plotting out an optimal solution for each move. This is because the elements of the game system, when set in motion, create such a large range of possible situations that the tree becomes too complex to be useful.

Why does chess have such a vast number of possible outcomes as opposed to tic-tac-toe? The answer lies in the combination of the simple, but varied, behaviors of the game objects and their changing relationships to each other on the board. Because of the extremely varied possibility set, chess remains challenging and interesting to players long after they have mastered its basic rule set.

One of the most important aspects of a game is the sense of possibility that is presented to the players at any given time. As I discussed in the previous chapter when talking about challenge, the goal of the designer is to present a situation that is equal to the abilities of the players and yet grows in challenge over time with their abilities. It is clear from the two examples above that how a system is constructed dramatically changes the dynamics of that system over time and the range of possibilities that face the player at any given point.

The range and type of possibilities within the system is not a situation where more is better in all situations. Many successful games have a somewhat constrained set of possibilities, and yet they still offer interesting gameplay. For example, a linear board game like Trivial Pursuit has a very small possibility space in terms of outcome, but the overall challenge of the game is not affected by this. Some digital games games, like side-scrollers, have a similarly small range of possibilities: Either you successfully navigate the challenge or you are stuck. But this range of action works for these types of games. Story-based adventure games often have branching structures with a limited number of outcomes. For



5.4 Various game structures

players of these games, navigating that defined set of possibilities is part of the challenge.

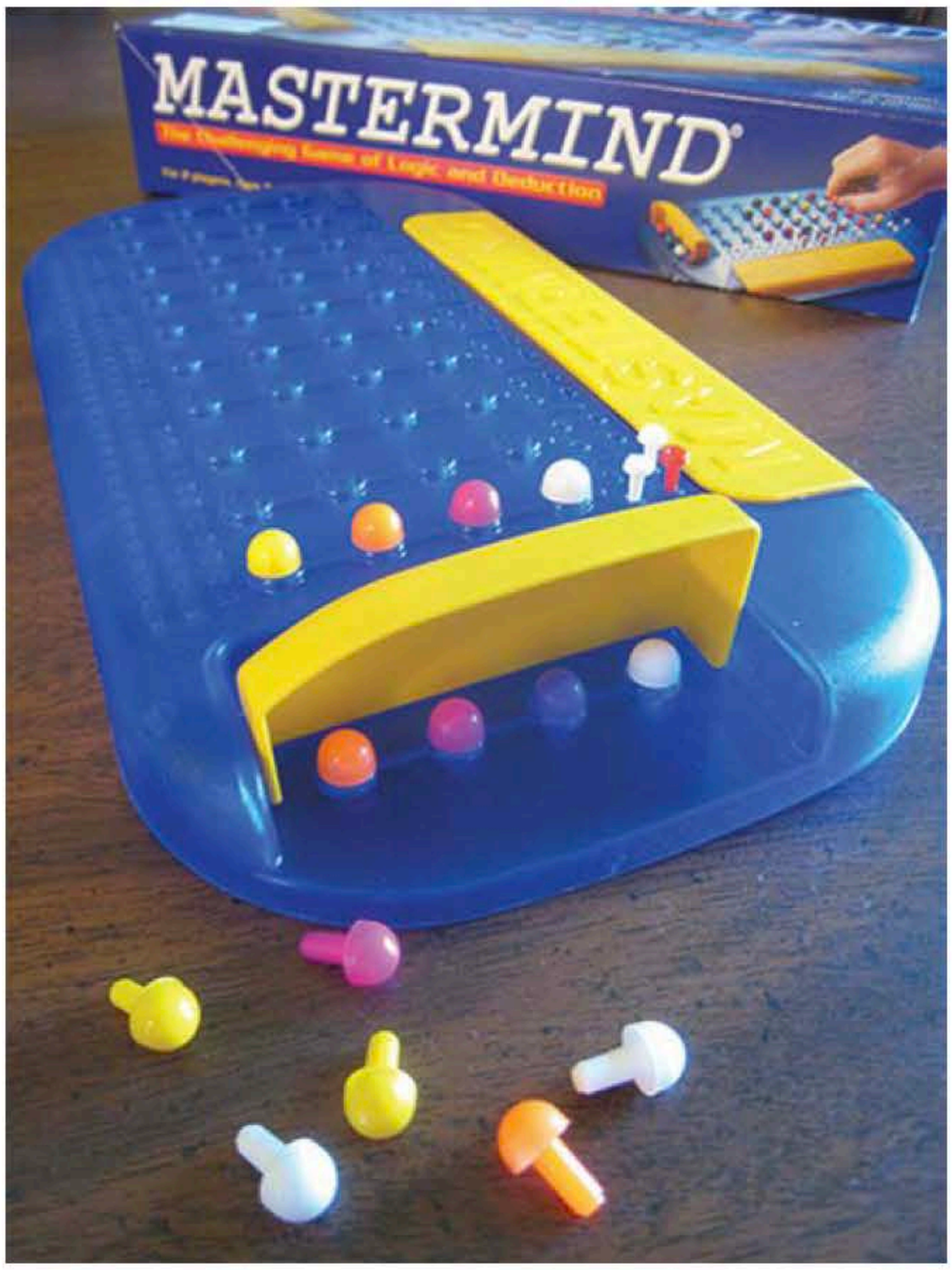
On the other hand, some games have attempted to create larger spaces of possibility for their systems. The way in which they do this is to introduce more objects into the system with defined relationships to each other. Open-world games like the Grand Theft Auto series, real-time strategy games, and massively multiplayer worlds all use such an approach to creating increased possibilities. The desired effects of a larger possibility space are a greater scope of choice for the player, opportunities for creative solutions to game problems, and enhanced replayability, all of which are an advantage to capturing a certain type of game player.

The following example shows how two games with very similar objectives and related system designs can provide extremely different ranges of possibilities, and they therefore create completely different player experiences.

Mastermind versus Clue

The game of Mastermind is quite simple. In case you are not familiar with the game, Mastermind is a two-player puzzle game in which one player is the puzzle-maker and the other is the puzzle-breaker. The puzzle is made up of four colored pegs (from a possibility of six colors), and the object of the puzzle-breaker is to solve the puzzle in as few guesses as possible. The procedures are also simple: On each turn, the puzzle-breaker makes a guess, and the puzzle-maker gives him feedback by showing how many pegs are: (1) the correct color and (2) how many are the correct color *and* correct placement in the sequence. The puzzle-breaker uses a process of elimination and logically attempts to narrow the possibilities and thereby solve the puzzle in as few guesses as possible.

Now let's look at the system structure. The objects in the game are the pegs, the properties are the colors, and the relationships are set by the puzzle-maker when they create the puzzle sequence. In the common version of Mastermind, the puzzle uses four pegs and six colors of pegs; repeated colors are allowed, so there are 6^4, or 1296, unique codes that can be made.

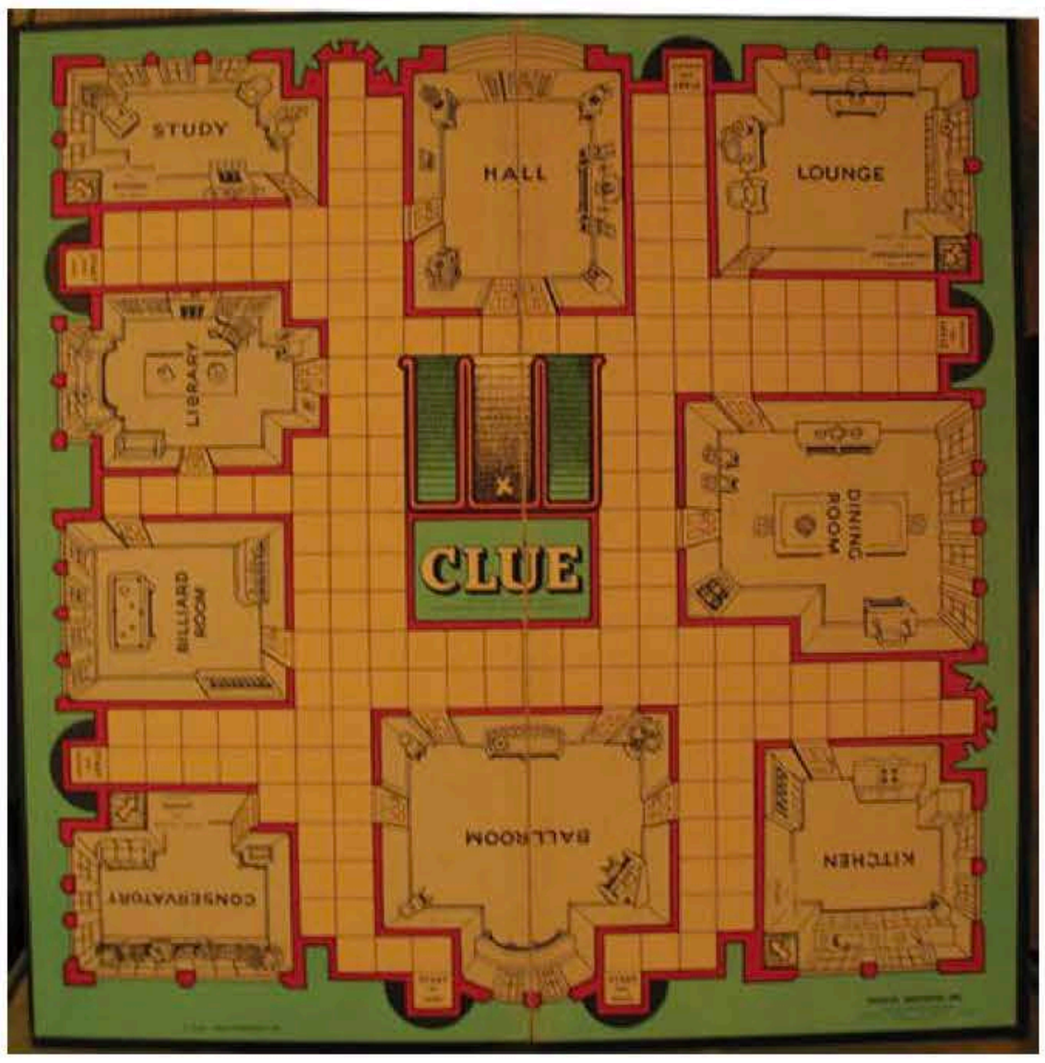


5.5 Mastermind

If you added one more peg to the length of the code, you would have 6^5, or 7776, possibilities. If you added another color, you would have 7^4, or 2401. These choices are part of the game's system structure and define the possibility set for the players.

Even if you aren't a mathematician, you can see that adding another peg to the code would add such an exponentially greater number of possibilities that it would make the game potentially unplayable—or at least a lot harder. Adding another color would not be as great a change, but it would still double the number of potential puzzles and make the game significantly harder. The designers of Mastermind undoubtedly playtested these various combinations of the system before settling on four pegs and six colors.

Now let's look at Clue, or Cluedo as it is called in Europe. Clue has a similar puzzle-breaking objective, but that objective has a slightly different mathematical structure, and it is couched in a different



5.6 Clue board, circa 1947

set of procedural elements, resulting in an entirely different player experience.

Clue is also a game of logic and deduction in which the objective is to solve a puzzle. But Clue is a game is for three to six players, and there are no special roles—all players are trying to deduce the answer to the puzzle. The game maps the premise of solving a murder onto the puzzle system and adds an element of chance into the procedures via the board and movement system.

In terms of its mathematical structure, the puzzles in Clue have a much smaller possibility set. There are six possible suspects, six possible weapons, and nine possible rooms, or $6 \times 6 \times 9 = 324$ possible combinations. The puzzle is mathematically simpler to solve, but players are hindered in their ability to guess the answer by the fact that they must roll the die and move around the board to gain information and make accusations. This addition of chance evens the playing field for those who might not have the best deduction skills (i.e., children) and makes Clue more accessible to a wider range of players. Also, the addition of the mystery premise and colorful characters provides a more compelling experience for some types of players, ergo a "family game."

If we compare Mastermind and Clue we see that both have similar objectives (i.e., solve a puzzle). Both puzzles are combinatorial (i.e., they use combinations of existing sets to create a "random" puzzle for each play). Mastermind has many more possible puzzle combinations, so the puzzle is harder to deduce. Clue has more ways of finding information: a social structure of asking for information, reading other players' faces, etc. Mastermind uses logic and deduction. Clue also uses logic and deduction, but it adds more chance to the system (dice and movement), as well as story (characters and setting).

This comparison is not meant to suggest that there is any one correct way to design a game system, just as there is no one type of game player. But you might find, upon analyzing some of your favorite games, that they share successful system designs in terms of the properties, behaviors, and relationships of their objects. Studying how the dynamics of these systems work can help focus your own thoughts and explorations and help you meet your own player experience goals.

Exercise 5.4: System Dynamics

Now let's take the game you have been working with in Exercises 5.1, 5.2, and 5.3 and see how we can change the system dynamics by experimenting with the properties, behaviors, or relationships of its core objects.

1. For example, if you chose a game like Monopoly, change the prices, placement, and rent of every property on the board or change the rules for movement. How you change these things is up to you, but make significant changes.
2. Now play the game. What happens? Did your changes affect the balance of the game? Is the game still playable?
3. If the system is still playable, make another change. For example, take out all the "positive" Chance cards in Monopoly and leave in only "negative or neutral" cards. Play the game again. What happens?
4. Continue doing this exercise until the game is no longer playable.

What was the crucial change you made? Why do you think that change finally broke the game?

One important type of system structure often found in games is an economy. I am going to look at this structure in more detail because it involves dynamics surrounding the resources in a game, one of the fundamental formal elements of games.

Economies

What is an economy? As I touched on briefly in my discussion of the utility and scarcity of resources in Chapter 3, some games also allow for the exchange of resources—either with the system (i.e., the bank in Monopoly) or among players. When a game allows exchanges of this kind, the system of trade forms a simple economy. In more complex systems, the rules of real-world economies might apply, but more often, games have severely controlled economies that only vaguely resemble real-world markets. Even so, there are some basic concepts of economic theory that we can use as a barometer for the feasibility of our game economies.

First of all, to have an economy, a game must have items of exchange, such as resources or other barterable items; agents of exchange, such as players or the system bank; and methods of exchange, such as markets or other trading opportunities. Also, an economy may or may not have currency, which helps to facilitate trade. As in the real world, the methods by which prices are set in an economy depend on what type of market controls are in place. Prices of market items in games can be free, fixed, or subject to a mixture of controls, depending on the design of the system. Also, the opportunities players have for trade can range from complete freedom to controls on prices, timing, partners, amount, etc. Here are some basic questions a designer should ask before building a game economy:

- Does the size of the economy grow over the course of the game? For example, are resources produced, and if so, is the growth controlled by the system?
- If there is a currency, how is the supply of that currency controlled?
- How are prices set in the economy? Are they controlled by market forces or set by the game system?
- Are there any restrictions on opportunities for trade among participants, for example, by turn, time, cost, or other constraints?

To get a sense for how games handle these economic variables, let's look at some examples ranging from classic board games to massively multiplayer online worlds.

Simple Bartering

Pit is a simple card game in which players barter for various commodities to "corner the market." There are eight suits of commodities, nine cards in each suit. The commodities are worth a varying number of points from 50–100. For example, oranges are worth 50 points, oats are 60 points, corn is 75, wheat is 100, etc. You start with the same number of suits in the deck as there are players—from three to eight.

The cards are shuffled and dealt out evenly to all the players. During each round, players trade by calling out the number of cards they want to trade but not the name of the commodity on the cards offered. Trade continues until one person holds all nine cards of a single commodity—a "corner" on that market.

5.7 Pit

There are several features to note in this simple barter system. First, the amount of product (i.e., cards) in the system is stable at all times—cards are not created or consumed during play. Additionally, the value of each card never changes relative to the other cards in the deck. The value is fixed by the printed point value set before the game begins. Also, the opportunity to trade is only restricted by number—all trades must be for an equal number of cards. Other than this, trade is open to all players at all times.

In this simple barter system, the in-game economy is so restricted by the rules of the game that there is no opportunity for economic growth, no fluctuation of prices based on supply and demand, no chance for market competition, etc. However, the trading system serves its purpose as an excuse for creating a frenetic, social trading atmosphere without any of the complexities of a real-world economy. To recap the features of this system:

- Amount of product = fixed
- Money supply = n/a
- Prices = fixed
- Trading opportunities = not restricted

Complex Bartering

Settlers of Catan is a German board game by designer Klaus Teuber in which players compete as pioneers developing a new land. During the course of the game, players build roads and settlements that produce resources such as brick, wood, wool, rock, and wheat. These resources can be traded with other players and used to build more settlements and upgrade settlements to cities, which in turn produce more resources.

As in Pit, the bartering of resources is a central part of the gameplay, and the trade in this game is also fairly unrestricted, with the following exceptions:

- You may only trade with a player on their turn.
- Players can only trade resources, not settlements or other game objects.
- A trade must involve at least one resource on each side (i.e., a player cannot simply give a resource away). However, trades can be made for unequal amounts of resources.

Other than these constraints, players can wheel and deal as they like for the resources they need. For example, if there is a scarcity of brick in the economy, players can trade two or three of another resource, such as wheat, for one brick.

As you can probably judge already, the economy of Settlers of Catan is much more complex than the simple barter system of Pit. One of the key differences is the fact that the relative values of the resources fluctuate depending on market conditions, an interesting and unpredictable feature that changes the experience of the game from play to play. If there is a glut of wheat in the game, the value of wheat falls immediately. On the other hand, if there is a scarcity of ore, players will trade aggressively for it. This simple example of the laws of supply and demand adds a fascinating aspect to gameplay that we did not see in the Pit example.

Another key difference from the simple bartering in Pit is that in Settlers of Catan, the total amount of product in the economy changes over the course of the game. Each player's turn has a production phase, the results of which are determined by a roll of the dice and the placement of player settlements. Product enters the system during this phase. Product is then traded and "consumed" (used to purchase roads, settlements, etc.) during the second phase of the player's turn.

To control the total amount of product in the system at any time, the system includes a punishment for holding too many resources in your hand. If a player rolls a seven during the production phase of their turn, any player holding more than seven cards has to give half of their hand to the bank. In this way, players are discouraged from hoarding and are encouraged to spend their resources as they earn them.

Another aspect of the economy that is interesting to note is the fact that while the barter system is fairly open, there is a control on price inflation. The

5.8 Barterable resources from Settlers of Catan

bank will always trade 4:1 for any resource; this effectively caps the value of all resources. And, as we also noted, while the trading system itself is quite open, the opportunity to trade is restricted by player turn.

The last difference between these two barter systems is their information structures. In Settlers of Catan, players hide their hands, but the production phase is an open process, so by simply paying attention to which players are getting certain resources on each turn, an attentive player can remember some, if not all, of the game state.

- Amount of product = controlled growth
- Money supply = n/a
- Prices = market value with cap
- Trading opportunities = restricted by turn

Exercise 5.5: Bartering Systems

For this exercise, take the simple barter game of Pit and add a new level of complexity to its trading system. One way to do this might be to create the concept of dynamically changing values for each of the commodities.

5.9 Monopoly money and property

Simple Market

The first two examples we have looked at have both been barter systems (i.e., they did not employ currency). The next type of system we will look at is the simple market system of Monopoly. In Monopoly, players buy, sell, rent, and improve real estate in an attempt to become the richest player in the game. The real estate market in the game is finite—there are 28 properties in the market (including railroads and utilities) at all times. Although properties are not sold until a player lands on their board space, they are still active in the sense that they are available for purchase.

Each player begins the game with $1500 from the bank, which they can use to purchase properties or pay rent and other fees. The growth of the economy is controlled by the rate at which players can circle the board, passing "go" to collect $200. According to the official rules, the bank never goes broke; if it runs

out of money, the player acting as banker can create new notes out of slips of paper.

In terms of trading opportunities, the rules state that buying and trading of properties between players can occur at any time, although "etiquette suggests that such transactions occur only between the turns of other players."[3]

Values of properties in the game are set in two basic ways. First, there is the face value on the title deed; if a player lands on a property, she may purchase it for this amount. If she fails to purchase the property, it goes up to auction and sells to the highest bidder. The auction is not limited by the face value, and the player who passed up purchasing it may bid in the auction. After a property is purchased, it can be traded between players at any price they agree upon. So the second and more important value of properties in the game is a true market value set by the competition of the players.

- Amount of product = fixed
- Money supply = controlled growth
- Prices = market value
- Trading opportunities = not restricted

Complex Market

For examples of complex market economies, we'll look at two historically interesting online games: Ultima Online and EverQuest. These games helped establish the genre of massively multiplayer online role-playing games. The two economies have much in common overall, but different emphases in their designs have created unique situations for each system. The key similarity in these and other online worlds is the fact that they have persistent economies that surpass a single game session by any one player. This immediately puts them in a category of complexity far beyond all the other examples we have looked at so far. The assumption often is that real-world economies apply directly to their systems because of the persistence of economy in these games and because they strive to create a sense of an alternate world.

In both games, players create characters, or avatars, that begin the game with a small number of resources—a little gold in Ultima or platinum in EverQuest, some minimal armor, and a weapon. Now players must enter the "labor market" to gain more resources. In both games, players begin at the lowest level of the labor market—killing small animals or doing other menial work to earn money. They can sell the results of their labor to system agents (in the form of shopkeepers) or to other players, if they can find interested buyers. In addition to the labor market, players can find, make, buy, or sell more complex items than those gained by labor. Items like weapons, armor, and magic items are part of a complex "goods" market.

In both games, goods and labor are traded in two ways: player to player and player to system. In both games, player-to-system trade, controlled by the game designers, exists to keep low-level "employment" steady while encouraging the player-to-player trade in scarce items. For example, shopkeepers will generally buy anything a player wants to sell, even if there is a glut of the object in the market. This keeps newbies steadily "employed." On the other hand, the shopkeepers' offers to buy on higher-level items will not be as competitive as the player-to-player market, encouraging players to seek out higher prices from one another.

In this way, the games create markets that mimic real-world situations in important ways and contradict real-world expectations in others. Supply and demand is a factor for players dealing at higher levels of commerce, with scarce or unique items, but it is not a factor for a newbie just trying to get ahead.

The amount of product in the system is controlled by the game designers, although Ultima at first tried to create a self-regulating flow in which resources were recycled through the system, available to respawn as new creatures and other materials as they were "spent" by the players. This was quickly changed to a system in which the flow of resources into the economy could be directly controlled by the designers. There were several reasons for this, one of which was a tendency for players to hoard game objects, restricting the total amount of products circulating in the game.

In these games and modern games of this genre, a metaeconomy often emerges separate from official gameplay in which characters and game objects

The Jolly Roger **Catskills, Felucca Facet, 85 15' N, 141 11'E**

The Jolly Roger Tavern, a dirty but legendary tavern, is the heart of the town of Red Skull Bay and home to fighting contests and some of the best ale in all Britannia. Landlubbers and sailors alike are invited to have a look at this unique bay, and raise a bottle of ale or two with the residents of Red Skull Bay.

5.10 Player-run establishment in Ultima Online

are sold between players in real-world markets. Characters have been offered on sites such as eBay sometimes sell for hundreds of dollars, depending on the level and inventory offered. While these metaeconomies are not a planned feature of these role-playing games, there are games that have included the concept of a metaeconomy in their designs.

- Amount of product = controlled growth
- Money supply = controlled growth
- Prices = market value with base
- Trading opportunities = not restricted

Metaeconomy

Magic: The Gathering, is another historically interesting game because it was the genesis for the collectible card game genre of play. It is somewhat different from the other example games we have looked at, in that the game itself does not include a trading or exchange component. The main system of Magic is a dueling game in which players use custom-designed decks of cards to battle each other. These cards, purchased by individual players, form the central resource in a metaeconomy surrounding the game itself.

Magic: The Gathering was designed by Richard Garfield and released by Wizards of the Coast, a Seattle-based game company, in 1993. At the time, Garfield was a mathematics professor at Whitman College as well as a part-time game designer. Wizards asked him to design a card game that was quick, fun, and playable in under an hour. But Garfield imagined a game that had the collectible nature of trading cards or marbles, combined with the qualities of Strat-O-Matic Baseball, in which players draft and compete their own teams. The result was a collectible card game that has been likened to "gaming crack." See Garfield's sidebar on the evolution of the game on page 219.

As mentioned, the game Garfield designed is a two-player dueling game with a fantasy theme. Each player has a deck of cards that consists of various spells, monsters, and lands. Land provides mana, which powers spells. These spells summon monsters that you use to attack your opponent. This would seem pretty straightforward except for the fact that a basic deck of game cards does not include all the cards in the system. In fact, it contains a mere fraction of the cards available. Players are encouraged to buy booster sets and to upgrade their deck as new revisions are released. And, important to our discussion of game economies, players can also buy and trade cards from each other, which they do aggressively. The market for Magic cards, and other similar collectible card game cards, is worldwide, greatly facilitated by the Internet.

The publisher of the game has control of the overall shape of this economy, in that it gauges how many cards to release—some cards are very rare, some are just uncommon, and others are not valuable at all because there are simply far too many of them available. But the publisher has no control over where and how these cards are traded after they have been purchased, and, other than rarity, it has no control over the prices set for these game objects.

In addition to the collectible nature of Magic and the metaeconomy formed by the trade of its game objects, there is an in-game aspect to this metaeconomy. Players choose cards from their collection to build decks, adjusting the amount of land and the types of creatures and spells to strike a winning balance.

This process of building and testing decks for effectiveness is similar to the process a game designer would go through when testing the balance of their system, and, of course, the designers of Magic work hard to make sure that any single card or combinations of cards are not so overpowered as to imbalance the game. But the final decisions regarding resource balance are left in the hands of the player and are highly affected by the metaeconomy surrounding the game.

The openness of the Magic system, and its success as a business as well as a game, has spawned a whole genre of trading games. As with online worlds, it is clear that much of the success of these games depends on how their in-game and metagame economies are managed over time.

- Amount of product = controlled growth
- Money supply = n/a
- Prices = market value
- Trading opportunities = not restricted

5.11 Magic: The Gathering cards

Metaeconomies for games have recently become an even greater factor in design with the rise of the somewhat ironically named "free-to-play" model for games. In these games, players can engage with the basic gameplay for free, but are encouraged, through the game structure, to purchase upgrades that can include extra resources, characters, or other items that speed up or improve their gameplay. A good example of this model is Clash of Clans, which is a social strategy game driven by a base resource of green "gems." Players begin with 500 of these gems, 250 of which they will use to complete the entry tutorial. They can earn more gems by meeting in-game goals or by using real-world currency to purchase them. As with the Magic metaeconomy discussed earlier, this means that players willing to invest money in these kinds of micro transactions can significantly increase their playing power. Today's mobile game industry relies heavily on this model of low-barrier-to-entry game combined with a design structure that encourages players to pay as they go.

As you can see, there are a wide variety of economies, ranging from simple bartering to complex and even real-world markets. The task of the designer is to wed the economic system with the game's overall structure and make sure that it supports satisfying gameplay. The economy must tie directly into the player's objective in the game and be balanced against the comparative utility and scarcity of the resources it involves. Every action players perform in relation to the economic system should either advance or hinder their progress in the game. In the case of free-to-play games, this is an important consideration to think about when or whether players should feel the need to make purchases in the game.

In general, economies have the potential to transform rudimentary games into complex systems, and if you are creative, you can use them as a way to get players to interact with one another. There is nothing better for community building than an underlying economy, which makes socializing into a game.

Developing new types of in-game economies is one of the areas in which modern game design has just begun to explore the potential. With the success of games like Clash of Clans and interesting new economic models for mobile games that bridge game economies with real-world economies, we are just beginning to understand the power of these systems. The fusing of economic models and social interaction is one of the most promising areas for future game experimentation, and in the next decade, we will see new types of game economies emerge that will certainly challenge our conception of what a game can be.

Emergent Systems

I have mentioned that game systems can display complex and unpredictable results when set in motion. But this does not mean that their underlying systems must be complex in design. In fact, in many cases, very simple rule sets, when set in motion, can beget unpredictable results. Nature is full of examples of this phenomenon, which is called "emergence." For example, an individual ant is a simple creature that is capable of very little by itself and lives its life according to a simple set of rules. However, when many ants interact together in a colony, each following these simple rules, a spontaneous intelligence emerges. Collectively, the drone ants become capable of sophisticated engineering, defense, food storage, etc. Similarly, some researchers believe that human consciousness might be a product of emergence. In this case, millions of simple "agents" in the mind interact to create rational thought. The topic of emergence has spawned dozens of books that explore links between previously unconnected natural phenomena.

One experiment in emergence, which is interesting for game designers, is called the Game of Life. (No, it is not related to Milton Bradley's board game The Game of Life.) This experiment was conducted in the 1960s by a mathematician at Cambridge University named John Conway. He was fascinated with the idea that rudimentary elements working together according to simple rules could produce complex and unpredictable results. He wanted to create an example of this phenomenon so simple that it could be observed in a two-dimensional space like a checkerboard.

Building on the work of mathematicians before him, Conway toyed with rules that would make squares on the board turn "on" or "off" based on

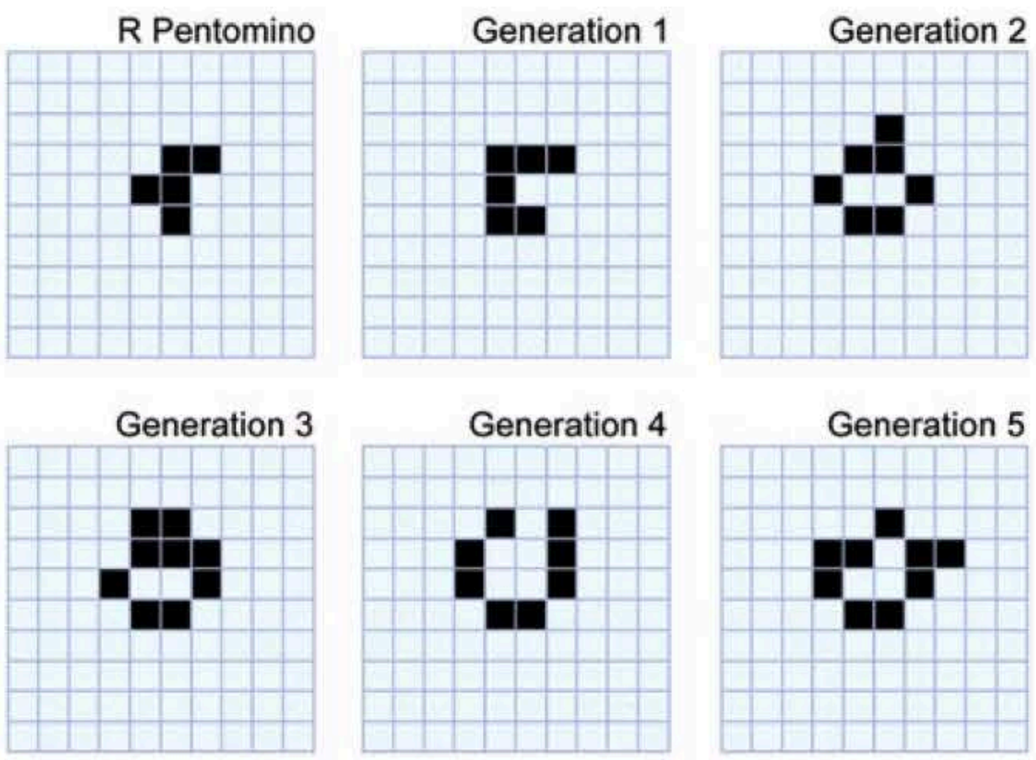
their adjacency to the other squares around them. Acting very much like a game creator, he designed, tested, and revised different sets of rules for these cells for several years with several associates in his department at Cambridge.

Finally, he arrived at this set of rules:

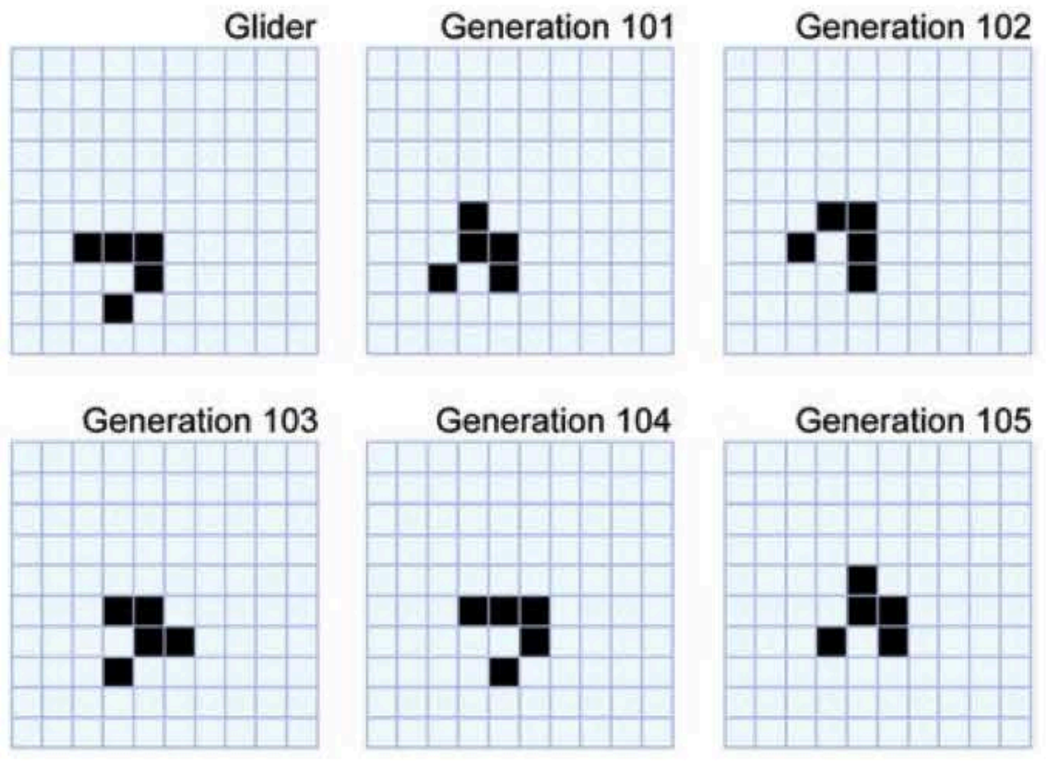
- *Birth*: If an unpopulated cell is surrounded by exactly three populated cells, it becomes populated in the next generation.
- *Death by loneliness*: If a populated cell is surrounded by fewer than two other populated cells, it becomes unpopulated in the next generation.
- *Death by overpopulation*: If a populated cell is surrounded by at least four other populated cells, it becomes unpopulated in the next generation.

Conway and his associates set Go pieces on a checkerboard to mark populated cells, and they administered the rules by hand. They found that different starting conditions evolved in vastly different ways. Some simple starting conditions could blossom into beautiful patterns that would fill the board, and some elaborate starting conditions could fizzle into nothing. An interesting discovery was made with a configuration called R Pentomino. Figure 5.12 shows the starting position for the R Pentomino, followed by several generations.

One of Conway's associates, Richard Guy, kept experimenting with this configuration. He



5.12 R Pentomino over several generations



5.13 Glider "walk" cycle

administered the rules for a hundred generations or so and watched a mishmash of shapes appear. Then suddenly a set of cells emerged from the group and appeared to "walk" on its own across the board. Guy pointed out to the group, "Look, my bit's walking!"[4] Guy worked on the configuration until it walked across the room and out the door. He had discovered what the group would call a "glider." A glider is a configuration that cycles through a set of shapes and moves along the board as it goes. Figure 5.13 shows what several generations of Guy's glider look like.

Conway's system was dubbed the Game of Life because it showed that from simple beginnings, life-form-like patterns could develop. There are a number

5.14 The Sims

of emulators online that you can download and experiment with. You will see that some use different rules and allow you to create your own starting conditions.

Emergent systems are interesting to game designers because games can employ emergent techniques to make more believable and unpredictable scenarios. Games as different as The Sims, Grand Theft Auto 3, Halo, Black & White, Pikmin, Munch's Oddysee, and Metal Gear Solid 2 have all experimented with emergent properties in their designs.

One interesting example, as mentioned in Chapter 4, is the character AI in the Halo series. Nonplayer characters have three simple impulses that drive them: (1) perception of the world around them (aural, visual, and tactile), (2) state of the world (memories of enemy sightings and weapon locations), and (3) emotion (growing scared when under attack, etc.).[5] These three sets of rules interact—each consulting the other—as a decision-making system in a character. The result is semirealistic behavior within the game. The nonplayer characters do not follow a script written by a designer but rather make their own decisions based on the situation they are in. For example, if all of their friends have been killed and they are facing overwhelming firepower from the enemy, they tend to run away; otherwise, they stay and fight.

Different games utilize different methods for creating emergent behavior. The Sims embeds simple rules in both the characters and in items in the environment. Will Wright, creator of The Sims, built the household items in the game to have values. When a character gets near an item—such as a bed, refrigerator, or pinball machine—the rules in the character interact with the rules in the item. So if a character's rules say he is sleepy, then an item that provides comfort, such as a bed, might attract his attention.

All of the previous examples share the same basic concept that simple rules beget complex behavior when interacting within a system. This concept is an exciting and fast-moving aspect of games today, and it is wide open to experimentation and innovation.

Interacting with Systems

Games are designed for player interaction, and the structures of their systems are integrally related to the nature of that interaction. Some of the things that need to be considered when designing for interaction are:

- How much information do players have about the state of the system?
- What aspects of the system do players control?
- How is that control structured?
- What type of feedback does the system give the players?
- How does this affect the gameplay?

Information Structure

For players to make choices about how to proceed in a game, they need information about the state of the game objects and their current relationships to each other. The less information players have, the less informed their choices will be. This affects the sense of control they have over their progress. It can also add to the amount of chance in the system and allow space for misinformation or deception as a part of the gameplay.

To understand the importance of information in a game system, think about what types of information you are given in some of the games that you like to play. Do you know the effect of every move you make? What about the other players? Is there information that you only have access to some of the time?

How information is structured in a game has a large influence on how players come to their decisions. In classic strategy games like chess and Go, the players have complete information about the game state. This is an example of an open information structure. An open structure emphasizes player knowledge and gives full disclosure on the game state. It will generally allow for more calculation-based strategy in the system. If this is the type of play you want to encourage

in your system, you need to make sure that important information is available to your users.

On the other hand, if you want to create play situations built around guessing, bluffing, or deceiving, you might want to consider hiding information from players. In a hidden information structure, players do not receive certain data about their opponent's game state. A good example of this is the 5-card stud variation of poker in which all cards are dealt facing down. In this game, the only information players have about their opponents' hands is how many cards they are dealt and the way in which they bet. This hidden information structure allows for a different type of strategy to develop—one based on social cues and deception rather than calculation. It tends to appeal to an entirely different type of player.

Exercise 5.6: Hidden Information

Many strategy games have open information structures that allow the players access to perfect information about the game state. Examples are chess, checkers, Go, mancala, etc. Take a game with an open information structure and change the system so that there is an element of hidden information. You might need to add new concepts to the game to accomplish this. Test your new design. How does adding hidden information change the nature of the strategy? Why do you think this is so?

Many games use a mixture of open and hidden information so that players are given some data about the state of their opponent's game, but not all. An example of such a mixed information structure might be the 7-card stud variation of poker where several cards are dealt down and several up over the course of the betting cycle, giving players only partial information about their opponents' hands. Another example of a mixed information structure is blackjack, for the same reasons.

The amount of information that players receive about their opponents' states often changes during the course of the game. This might be because they have learned information by interacting with their opponents, or it might be because the concept of a dynamic information structure is built into the game. For example, many real-time strategy games, like the classic WarCraft series, use the concept of "fog of war" to provide dynamically changing information to players about their opponents' status. In this game, players can see the state of their opponents' territory if they move a unit into that territory. When they move the unit out of the territory, the information freezes until another unit ventures back to the territory.

A dynamically changing information structure provides an ever-shifting balance between strategy based on knowledge and strategy based on cunning and deceit. For the most part, this type of advanced information structure has only been possible since the advent of digital games and the computer's ability to orchestrate the complex interactions between players.

Exercise 5.7: Information Structures

What type of information structures are present in Unreal Tournament, Age of Empires, Jak II, Madden 2004, Lemmings, Scrabble, Mastermind, and Clue? Do they have open, hidden, mixed, or dynamic information structures? If you do not know one of the games, pick a game that I have not mentioned and substitute it.

Control

The basic controls of a game system are directly related to its physical design. Board games or card games offer control by direct manipulations of their equipment. Digital games might use a keyboard, mouse, joystick, gesture, multitouch, speech, or other alternative types of control devices. Platform games usually provide a proprietary controller. Arcade games often use game-specific controls. Each of these types of controls is best suited to certain types of input. Because of this, games that require specific input types have been more successful in some game platforms than others. For example, games that require text entry have never been as popular on consoles as they have been on PCs. Today's designers have a wide range of control options to

consider, and each brings with it different potential play situations. For example, gestural controls like the now-defunct Kinect system offered the potential for full-body input. They are useful for prompting big, physical actions in games that are more about the emotion and experience of movement than precise input. Recently, we have been seeing more use of speech controls in mobile and alternative platforms. Home control systems like the Amazon Echo or Google Home can play simple verbal games that offer an interesting opportunity for innovation.

One type of control system is not inherently better than another. What matters is whether the control system is well suited for the game experience, and it is the job of the designer to determine this. I encourage you to think about the games you like to play. What type of controls do you enjoy? Do you prefer to have direct control over the game elements, such as you might have when moving your character through a 3D shooter? Or indirect control, as in a game like SimCity? Do you prefer touch controls, as in Angry Birds, or gestural controls like Kinect Star Wars? These decisions will have a huge impact on what type of game you design and how you go about structuring it.

Direct control of movement is a clear-cut way for players to influence the state of the game. Players can also have direct control over other types of input, like selection of items, directly presented choices, etc. Some games do not offer direct control, however. For example, in a simulation game like Rollercoaster Tycoon, players do not have direct control over the guests at their theme park. Instead, players can change ride variables, trying to make certain rides more attractive by lowering price, increasing throughput, or improving the ride design. This indirect control offers ways for players to influence the state of the game that is one step removed from the desired changes and provides an interesting type of challenge within certain game systems.

When the designer chooses what type of control to offer to players, he is deciding a very important part of the game. This decision forms the top-level experience that players will have with the system. Control often involves a repetitive process or action

5.15 Indirect control: Rollercoaster Tycoon

performed throughout a game. If this basic action is hard to perform, unintuitive, or just not enjoyable, the player might stop playing the game altogether.

In addition to deciding the level of control, the designer also needs to consider restricting control of some elements completely. As I discussed when I talked about designing conflict, games are made challenging by the fact that the players cannot simply take the simplest route to a solution. This is true in terms of designing controls as well. Some games allow a high degree of freedom in terms of player control. For example, a 3D shooter allows for spontaneous, real-time movement throughout the environment. Other games restrain player control tightly, using this structure to provide part of the challenge. An example of this would be a turn-based strategy game, like Go or chess.

How do you decide what controls to allow players and what not to allow? This is a central part of the design process. You can see the impact that different levels of input have on games if you look at a familiar game and imagine how it would work if you took away some of the player input.

For example, let's look at a real-time strategy game like StarCraft II. In this game, the player selects certain units to mine minerals, others to harvest gas, etc. How many units are doing these tasks at any given time is a function of availability, but it is basically under the player's control. What if this opportunity for control

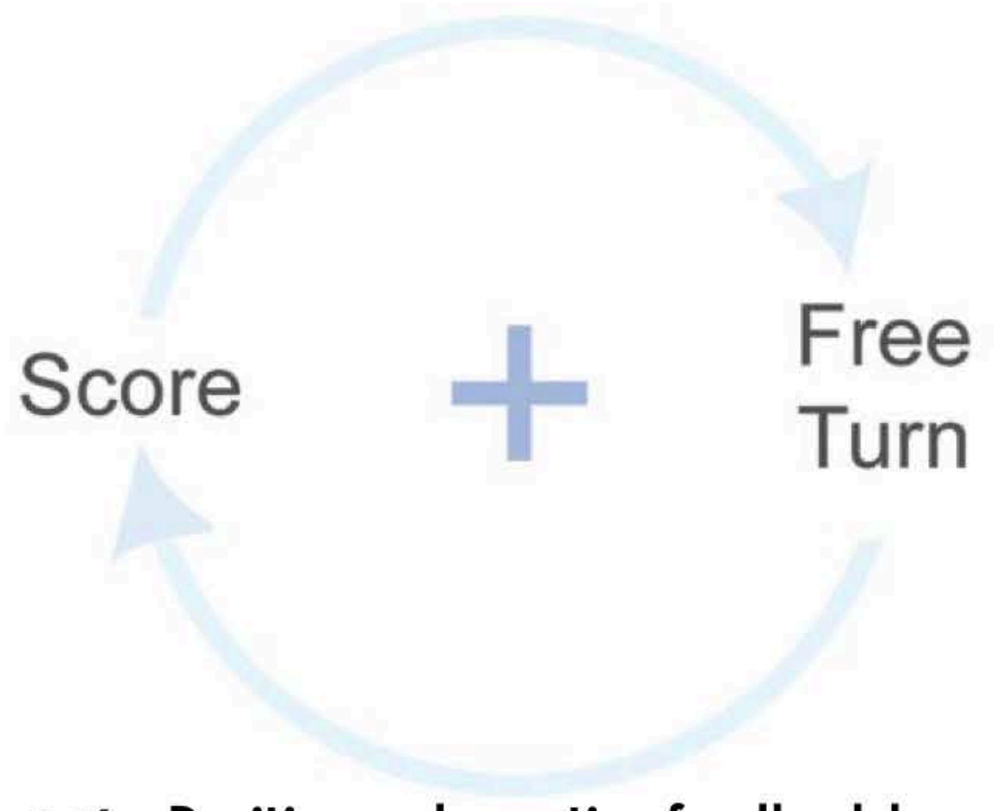
were taken away? Imagine how the system would work if the designer determined that system would always assign 50% of the available units to mine minerals and 50% to harvest gas. How would taking this input away from the player affect the system? Would it create too much balance between various players' resources? Would it take away some tedious parts of gameplay? Or would it take away critical resource management? These are questions a designer faces when thinking about how much and what type of control to give players in the system.

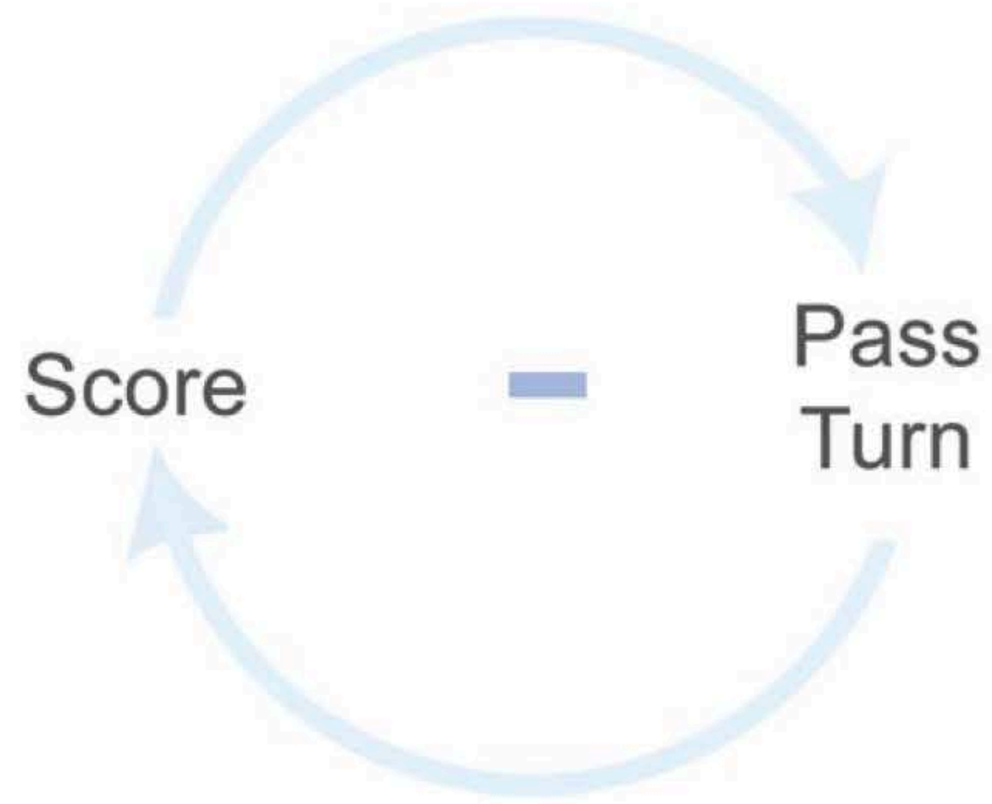
Exercise 5.8: Control

For the same games mentioned in Exercise 5.7, describe the methods of control they use: direct or indirect, real time or turn based. Are there any cases in which these distinctions are mixed?

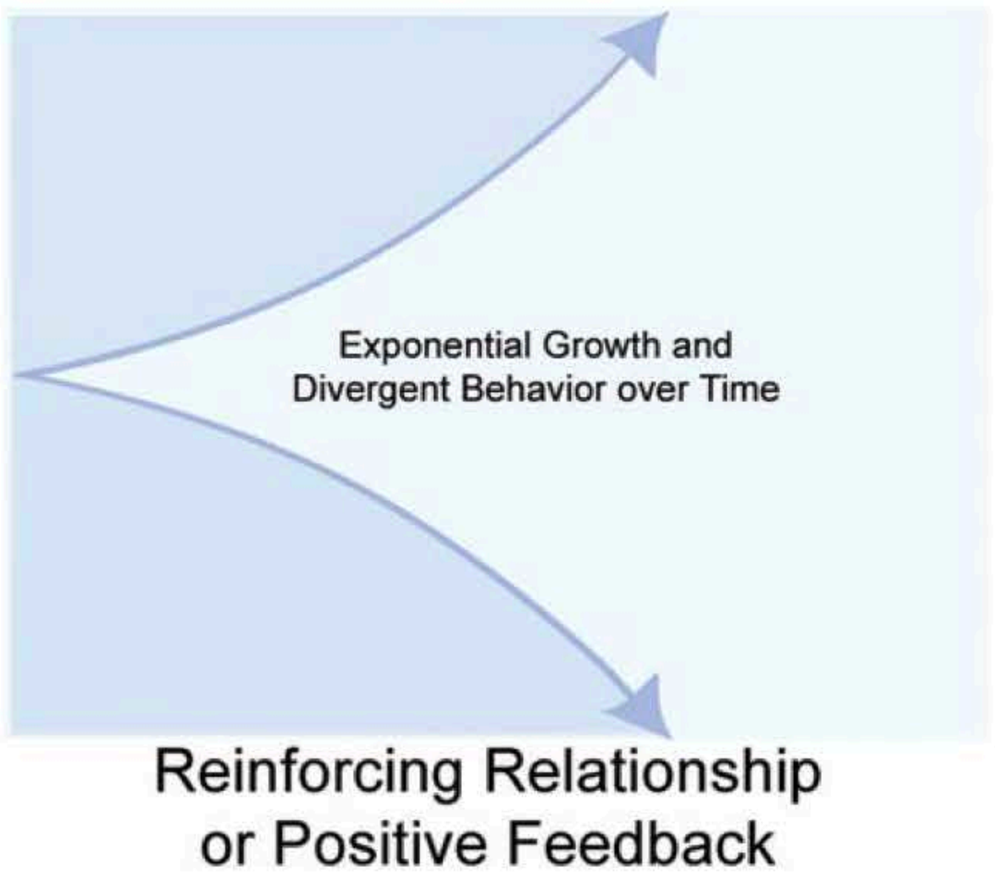
Feedback

Another aspect of interaction with the system is feedback. When we use the word "feedback" in

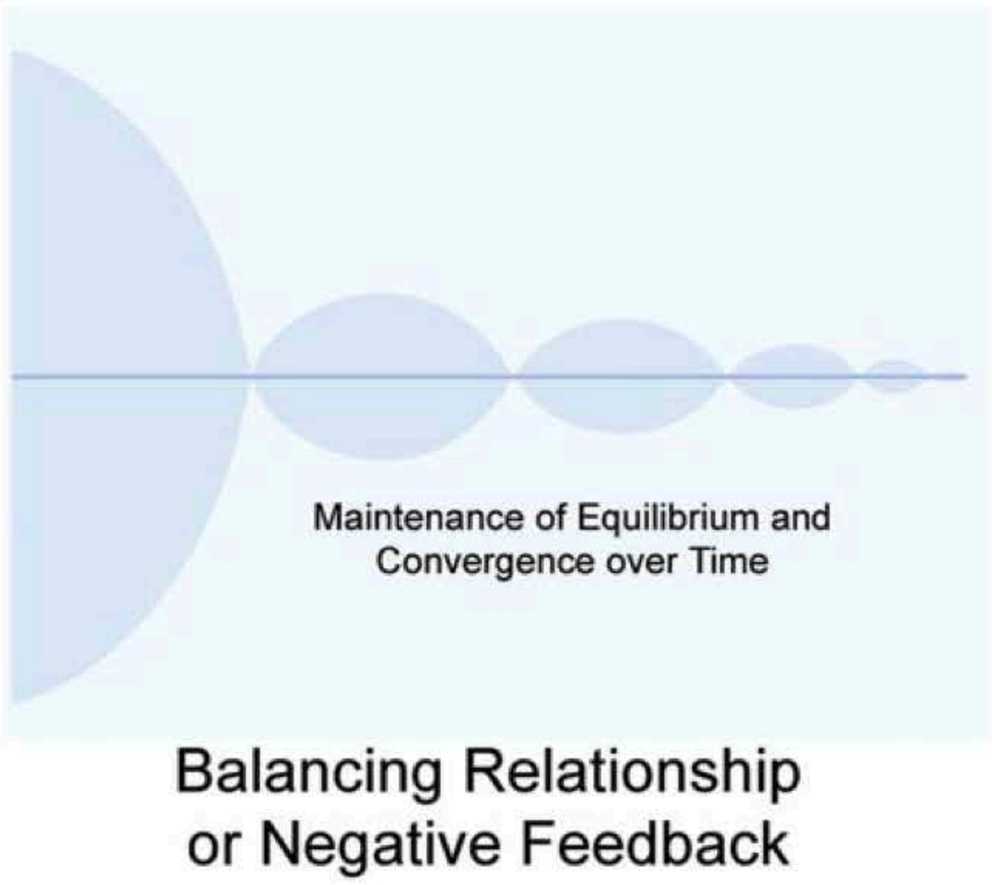




5.16 **Positive and negative feedback loops**



Reinforcing Relationship
or Positive Feedback



Balancing Relationship
or Negative Feedback

5.17 **Reinforcing and balancing relationships over time**

general conversation, we often are just referring to the information we get back during an interaction, not what we do with it. But in system terms, feedback implies a direct relationship between the output of an interaction and a change to another system element. Feedback can be positive or negative, and it can promote divergence or balance in the system.

Figure 5.16 shows example feedback loops for two different types of game scoring systems. In the first example, if a player scores a point, they get a free turn. This reinforces the positive effects of the scored point, creating an advantage for that player. A negative feedback loop, on the other hand, like that on the right, works against the effect of the point. In this example, every time a player scores a point, they must pass the turn to the other player. This has the effect of balancing the system between the two players rather than allowing one player to get a larger advantage over the other.

"Positive" and "negative" are somewhat loaded terms, and some systems theories use the terms "reinforcing" and "balancing" instead. Generally, reinforcing relationships are ones in which a change to one element directly causes a change to another element in the same direction. This might force the system toward one or the other extreme. By contrast, in balancing relationships, a change to one element causes a change to another in the opposite direction, forcing the system toward equilibrium.

For example, in the game Jeopardy!, when a player answers a question correctly, she retains control of the board. This presumably gives the leading player an advantage in answering the next question, reinforcing her lead in the game and moving the system toward resolution in their favor. This is a reinforcing relationship, or loop. An example of the same type of relationship, but in the opposite extreme, would be if a Jeopardy! player who answered incorrectly were forced to sit out the next question. This is not a rule in the game, but if it were, it would reinforce the repercussions of answering a question wrong.

Reinforcing loops cause output that either steadily grows or declines. Many games use reinforcing loops to create satisfying risk/reward scenarios for players that drive the game toward an unequal outcome based on player choices. To keep the game from resolving too quickly, however, balancing relationships are also used.

Balancing relationships, on the other hand, try to counteract the effects of change. In a balancing relationship, a change to one element results in a change to another element in the opposite direction. The classic example of a balancing relationship is in football. When one team scores, the ball is turned over to the other team. This gives a boost to the nonscoring team, attempting to balance the effects of the points won. If the advantage were given to the scoring team instead, it would be an example of a reinforcing relationship.

Some balancing relationships are not as easy to distinguish. For example, the board game Settlers of Catan has a procedure that attempts to create balance in the number of resources each player can hold at any one time. In this game, every time a seven is rolled with two six-side dice, any players holding more than seven cards in their hand must give up half of them. This has the effect of keeping prosperous players from becoming too powerful and resolving the game too quickly.

To improve gameplay, a good designer must be able to evaluate how quickly or slowly the game is progressing, understand if there are patterns to growth or contraction in the system caused by reinforcing loops, and know when and how to apply a balancing factor.

Interaction Loops and Arcs

by Daniel Cook

Daniel Cook is Chief Creative Officer at Spry Fox. His games include Alphabear, Triple Town, Realm of the Mad God, Tyrian, and many others. In his spare time, he also writes about game design theory and the business of games at Lostgarden.com

All games (and in fact all interactive systems) are composed of common structural elements called *interaction loops*. These describe how a player interacts with a game and how the game in turn responds to the player. Once you learn to spot them in your games, you'll find they provide a powerful analytical tool for debugging both broken systems and confused players.

Interaction loops help you understand:

- Exactly how players go about learning your game
- The skills players acquire and the order in which they acquire them
- Which exact portions of your game are breaking and causing confusion
- How the smallest interactions in a game are tied into the most complex interactions and why players care

Interaction loops

Any interaction within a game involves the following steps:

1. The player starts with a **mental model** that prompts them to...
2. Make a **decision** to...
3. Apply an **action** in order to...
4. Manipulate the game **rules** and in return...
5. Receive **feedback** that...
6. **Updates** their mental model. Or, in other words, they start to learn how the systems of the game work.

Players then start the interaction loop all over again to gain more practice with a specific **skill**. Or the player decides that what they did wasn't worth it and instead kicks off whole new interaction loop!

Consider Mario from Nintendo's Super Mario Bros learning to jump.

1. You hand a new player the controller. They've never played Mario and have little idea what to do. But they notice that the controller has buttons and they've got a pretty good pre-existing mental model of how you interact with buttons.

2. They don't have much pre-existing information on evaluating **costs and benefits**, so they throw caution to the wind and randomly choose a button. Even a random sampling of an unknown system will likely teach them something.
3. Then they press the button.
4. The mysterious black box within the game, the game **rules,** begins executing hidden code. Time, an abstract resource, advances. There's anticipation at this moment for the player because they don't know yet what might happen. If this system were completely known, it would not be nearly as interesting to play with.
5. Finally, the computer generates visual **feedback** on the screen that Mario is jumping. This is a form of functional feedback because it tells the player how the black box operates.
6. Aha! After a few more loops through the interactions, the player updates their **mental model** that Pressing the A button *causes* Mario to jump. This realization of repeatable **cause and effect** is a moment of **mastery**, where the player acquires a new mental tool for manipulating the world.

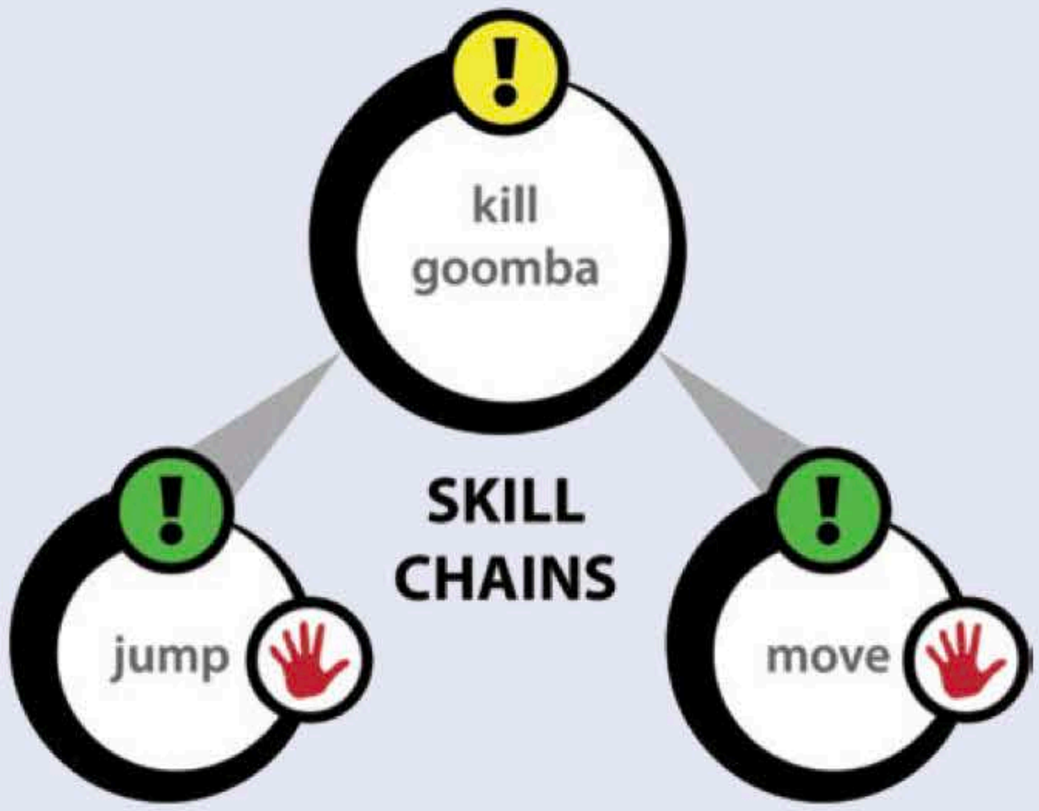
An interaction loop is a **loop** because mastery only arises after repeated passes through each of the steps. Players practice a skill by attempting it again and again. Each time they sample the game, they gain a bit more information about why the system behaves as it does.

Skill chain

Jumping isn't the only interaction loop in a Mario game. Mastering simpler interactions enables players to complete **compound interactions**. For example, to kill a Goomba, the player must first learn how to both jump and move. First they master the simple jump and move. Then they use both those actions together to learn how to kill the Goomba.

Two things are worth keeping in mind when constructing skill chains.

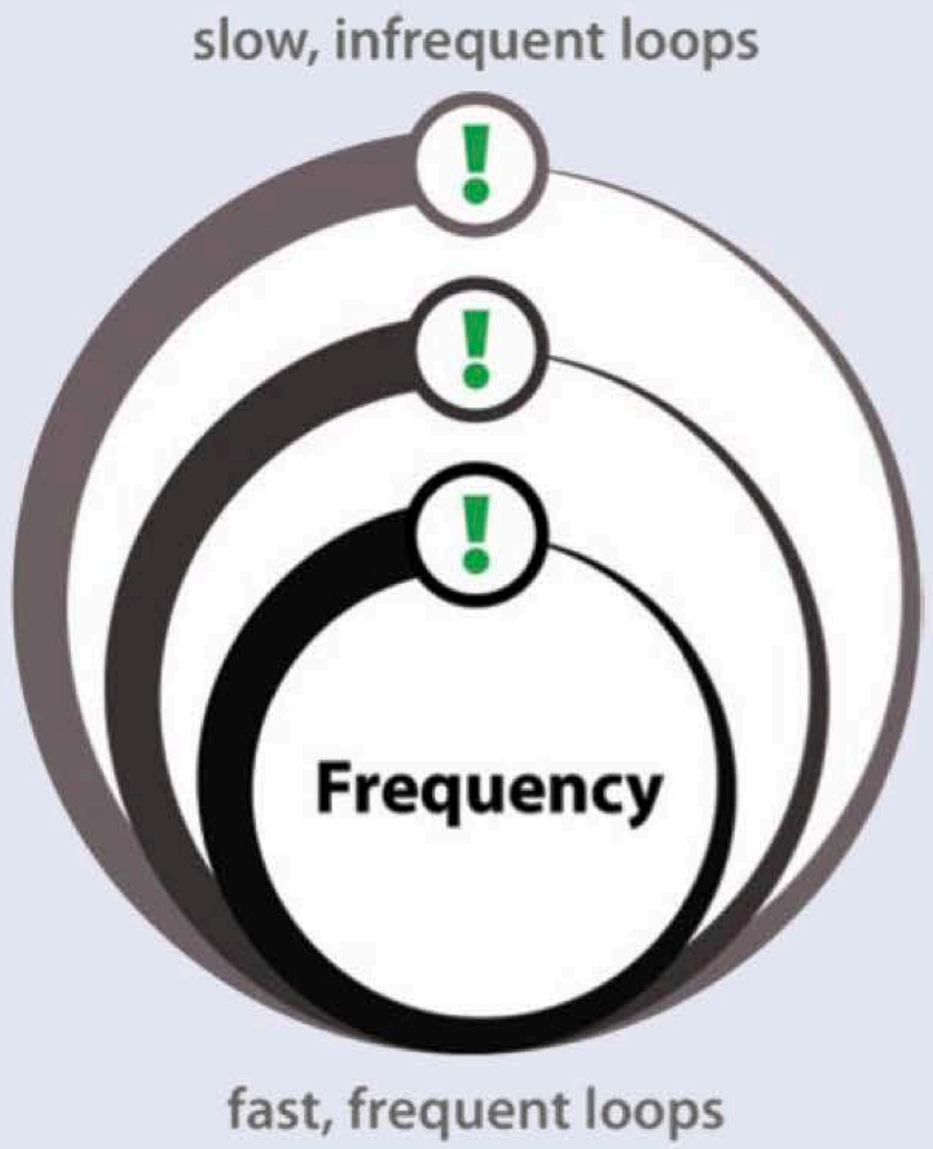
- These loops are arranged in a hierarchical structure where order matters. If a player hasn't learned lower-order skills, they'll be blocked from learning higher-order skills. If a player is failing to learn something about your game, be sure to ask if they've fully mastered precursor skills.
- Each node of the skill tree involves a distinct interaction loop that contains all of the same steps listed above in our breakdown of a generic interaction loop. So if there's a problem with your game, you can zoom in on the exact loop and the exact step that is causing the issue.



A hierarchical skill chain showing how lower-order interaction loops contribute to a higher-order compound skill (that is also an interaction loop!)

Frequency

Interaction loops also occur at different **frequencies**, or time scales, throughout a game. A skill like jumping happens nearly every second, while killing a Goomba might happen only a few times a level. By understanding the interaction loops of a game, you gain a greater appreciation of game pacing.



The fastest interaction loops are real time, often looping in less than 200 ms. The slowest in a progression-based MMORPG might take weeks or months.

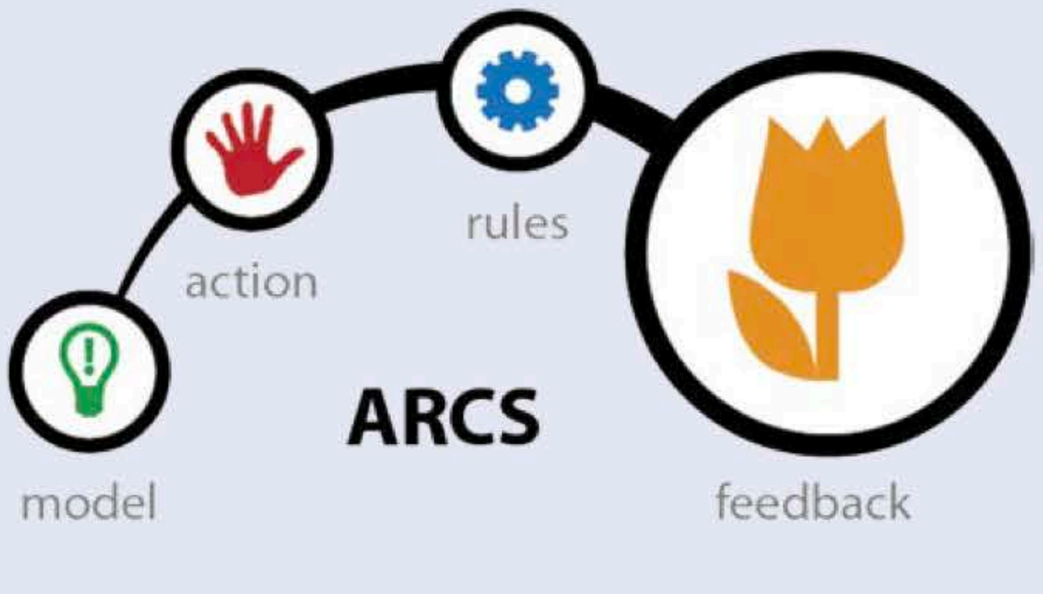
When to use interaction loops

I think in terms of interaction loops when I'm planning out the skills I want a player to learn in a game. Interaction loops and their associated skill chains are a wonderful structure methodically building up a player's "wisdom," a holistic, intuitive understanding of a complex system. As the player works their way through the game, they create a mental model that contains a thousand branches, successes, failures, and nuances. Game structures rich in interaction loops empower your player to approach unexpected new situations with confidence and a flexible set of proven mental tools.

Interaction arcs

Interaction arcs are another structural element found in games. Unlike loops, which are used to develop skills, arcs are used to deliver evocative content like a story or movie.

Arcs have a similar structure to loops. The player still starts with a mental model; they apply an action to a system of rules and receive feedback. However, in your typical arc, the mental model is updated with a package of predigested practices or evocative stimuli. The general goal of the content author is to convey key ideas as efficiently and effectively as possible.



Consider the ending of Super Mario Bros. You've just beaten the final boss by dropping Bowser into the lava.

1. **Mental model**: You know you've beaten the big enemy, and in the past, that usually leads to a small cutscene with a toad.
2. **Decision**: You don't have a lot of options, so you decide to run to the right. The same thing you've been doing all game long.

3. **Action**: Run to the right.
4. **Rules**: The game shows you some new visual feedback. There's almost no other simulation going on in this screen.
5. **Feedback**: You see the princess! She thanks you and tells you your quest is over.
6. This is a rich update of your mental model. The symbol of the 'princess' triggers a cascade of all the stored details your brains know about princesses. The word 'quest' does the same. You mix the two together with your mental sense-making engine. The cutscene content delivery arc is complete.

An arc delivering a narrative payload in Super Mario Bros.

Arcs are built to deliver these carefully authored payloads of preprocessed information. You'll typically find many arcs have the following functionality:

- **Simple modular actions** such as turning a page or watching a movie. Or, in the case of Mario, walking right. These rarely enable player expression because skill isn't a goal. You want players to get to the content as easily and quickly as possible.
- **Simple systems** that display content to player. A page of a book is a simple system that conveys content in the form of text to the player.
- **Evocative feedback** that links together existing mental models in some unique, interesting, or useful manner. For arcs, the feedback is 99% of the payload, and the actions and rules are simply a means to an end. Once this payload is fully delivered, the value of repeated exposure to the arc drops substantially. Note that there is very little of the *functional feedback* we saw in interaction loops because the systems in arcs are kept intentionally simple and obvious.

Where a loop might be executed hundreds or thousands of times, an arc is typically executed once or twice. The game moves on and only very rarely will players return to squeeze out more insights. The movie is watched. The book finished. A handful of players return at a distant later date for nostalgia, comfort, or nuanced insight. But most do not. Within a game session, an arc is a loop you exit almost immediately.

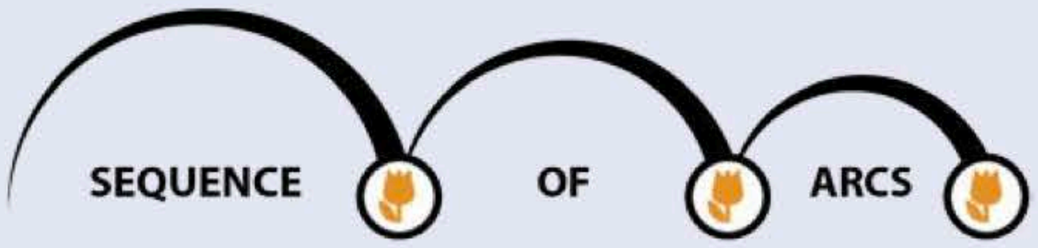
Arcs deliver success stories

Arcs excel at communicating "success stories," a singular ideal **golden path** through a system that someone else previously explored. The best teach a lesson, either informative, positive, or negative. A wise woman might tell you the best way through a level. Or a danger to avoid. This is a brilliant learning shortcut. A story that tells how to build a sturdy bridge or succeed in a difficult relationship might save the novice years of experimentation.

However, the acquired knowledge is often quite different and less robust in the face of change than "wisdom" found in loops. With even a slight shift in context, the learning gained from an arc becomes no longer directly applicable. It is not an accident that we make the distinction between "book learning" and

"life experience." Arcs are neither better nor worse than loops. Both are essential tools that serve different design needs.

Sequences of arcs



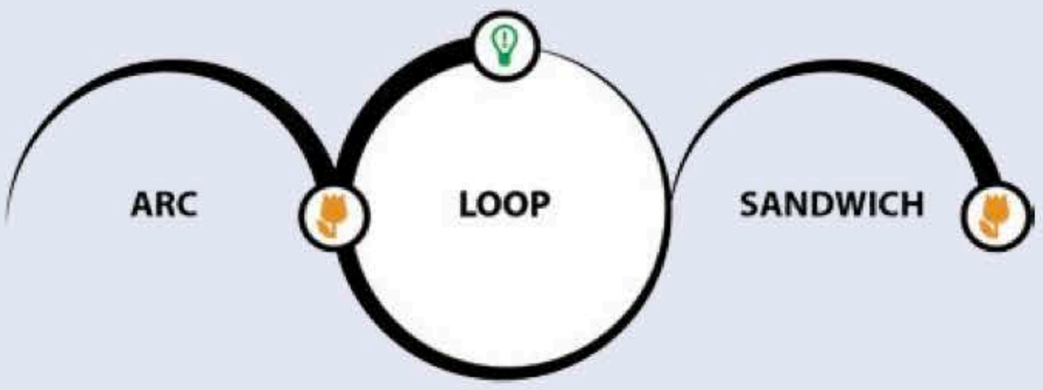
A common game design issue with arcs is that players **burn out** on them rapidly, rarely desiring to engage with an expensively authored experience more than once. It is possible to give arcs a bit more staying power by stringing them together serially in a **sequence of arcs**. This is a proven technique and is at the base of most commercial attempts to give content arcs longer retention. The pages of a book or the scenes in a movie are a sequence of arcs.

Businesses that rely on a constant sequence of arcs to bring in ongoing revenue often find themselves running along the **content treadmill**. If you stop producing content, the business fails. Game studios that sell expansion packs to stay in business are on a content treadmill.

Expanding loops into arcs

As you analyze your game design, any interaction loop can be superficially described as a series of arcs with one arc written down for each pass you make through the loop. This step-wise or turn-based description is an **expanded loop**. This is useful for recording a single person's **personal experience** of incremental insights. However, the expanded loop tells you little about the broad possibility space described by the loops. Where loops neatly describe a statistical spectrum of outcomes, the expanded arcs laboriously describes only a single sample.

Game architecture: Mixing loops and arcs



Since both loops and arcs can be easily nested and connected to one another, in practice, you end up with chemistry-like mixtures of the two. The structure of loops and arcs is a design's **game architecture**. This can get a bit messy to tease apart. The simplest method of analysis is to ask **"What repeats and what does not?"**

Narrative games are the most common example of mixing loops and arcs. A simple combination might involve intermixing a segment where the player is engaged with loops with a segment of arcs. This is your typical cutscene-gameplay-cutscene sandwich.

However, the architectural patterns can get far more detailed. For example:

- **Parallel Arcs**: In a typical game, you can treat the emotional payload of game music as an arc that plays in parallel alongside the looping core gameplay.

- **Levels**: The spatial arc of navigating a level provides context for exploring variations on a central gameplay loop. The Golden Path through a level is an authored experiential arc, but players may loop in corners of the level playing with various challenges.
- **Micro Parallel Arcs**: A game like Half Life combines both levels and parallel arcs to deliver snippets of evocative stimuli as you progress through the level.

These structures also exist in traditional media. For example, if you look at a traditionally arc-based form such as a book, you find an odd outlier in the form of the Bible. At one level of analysis, it can be seen as a story arc that you read through and finish. However, it is embedded in a much larger set of interaction loops we casually refer to as a religion. The game-like loops include everything from worship rituals to the mining of the Bible in order to synthesize weekly sermons. The content arc delivers pieces of a rulebook for executing a larger game consisting primarily of loops.

Even in such a complex case, understanding which interactions are a loop and which are an arc helps tease apart the systemic behaviors. Of the two, loops are rarely discussed in any logical fashion. Consumers will note the arcs in a media project and comment on them at length while being quite blind to the loops driving experiential outcomes. Religion is a lovely example of how loop analysis can provide a practical description of a game's ruleset even when the actual players are only vaguely aware of the structures driving their skill and knowledge acquisition. Try applying the same loop-based deconstruction to your games and the communities that surround them.

Untangling loops and arcs in existing game forms

Over time, common game genres become encrusted with a complex mix of arcs and loops. Mature genres often try to woo jaded players by cramming an ever-increasing amount of flashy (and expensive) content arcs into their game. It can be hard to decipher the game's essential structure.

One exercise I've been performing on various games is identifying loops and arcs in a popular genre and then removing the pieces to see if what is left stands on its own.

First, take your favorite genre (such as platform games) and remove all the content payloads:

- Narrative sequences that don't teach skills.
- Big evocative elements like character art or music. Anything that reminds you of a brand or IP should go.
- Small evocative elements like textures, colors, lighting, or other elements that help the player feel, but don't help them learn.

You should be left with a functional, but structurally complete game. This is something that plays mostly the same, but each of those empty moments can now be filled with new content. Most studios organized around servicing a content treadmill create production efficiencies by defining this exact sort of empty template and then filling in the blanks to create a new deliverable.

But we aren't yet done. Next remove the game architecture, those elements that bind loops to arcs.

- Puzzles, missions, or quests
- Levels
- Menus and buttons that take you from one activity to the next
- Any elements of a computer game that you can "beat" or that render the game boring or meaningless upon repeated play

After boiling the game down, what pieces remain? Can you bind them together into a new architecture? Can you flesh out the bones of this new beast with an entirely different set of content? Most of the time you can. And in the process, you'll start learning how to be an inventor of new games instead of a copier of stagnant historical forms.

Now, the point here is not to get rid of arcs or elements like narrative. Instead, it is to take something you know and understand its structural components. Loops and arcs are ingredients and the goal is to create a new recipe with a different mix rather than unquestioningly recreating the same meal again and again.

Tuning Game Systems

As mentioned earlier, the only way to fully understand a system is to study it as a whole, and that means putting it in motion. Because of this, after a game designer has defined the elements of his system, he needs to playtest and tune that system. The designer does this first by playing the game themselves, possibly with other designers, and then by playing with other players, who are not part of the design process. In Chapters 10 and 11, I will discuss the tuning process in detail and a number of specific issues that your game system can exhibit, but in general, there are several key things that a designer is looking for when balancing a game system.

First, the designer needs to test to make sure that the system is internally complete. This means that the rules address any loopholes that could possibly arise during play. A system that is not internally complete creates situations that either block players from resolving the conflict or allow players to circumvent the intended conflict. This can result in "dead ends" of gameplay and, sometimes, in player conflict over the rules. If players argue over how the rules should deal with a particular situation, it is probably because the system is not internally complete.

When the system is judged to be internally complete, the designer will next test for fairness and balance. A game is fair if it gives all players an equal opportunity to achieve the game goals. If one player has an unfair advantage over another, and that advantage is built into the system, the others will feel cheated and lose interest in the system. In addition, a system can be unbalanced by the availability of dominant strategies or overpowered objects such as those described on pages 323–325. In these cases, the fact that one strategy or object is better than the others effectively reduces the meaningful choices for the player.

When a system is internally complete and fair for all players, the designer must test to make sure the game is fun and challenging to play. This is an elusive goal that means different things to every individual game player. When testing for fun and challenge, it is important to have clear player experience goals in mind and to test the game with its intended audience of players. Generally, this is not the designer or the designer's friends.

For example, when a designer is testing a game for children, she might not be able to accurately judge the difficulty level, making the game too hard for its young audience. Determining the needs and skills of the target players and balancing a system for them requires a clear idea of who the target players are and a process for involving players from that target in the playtesting. Chapter 9, on playtesting, will talk more about how to identify those players and bring them into the design process. Testing for fun and challenge brings up a number of issues regarding player experience and improving the opportunity for meaningful choice that will be addressed in detail in Chapter 11, including the idea of tuning your game for a dramatic conclusion, as discussed in Stone Librande's sidebar on page 368.

Conclusion

In this chapter, I have described the basic elements of a game system and shown how the nature of the objects, properties, behaviors, and relationships can create different dynamics of interaction, change, and growth. I have discussed how player interaction with these elements can be affected by the structures of information, control, and feedback.

One of the real challenges in designing and tuning game systems is to isolate what objects or relationships are causing problems in gameplay and to make changes that fix the issue without creating new problems. When the elements are all working together, what emerges is great gameplay. It is the job of the game designer to create that perfect blend of elements that, when set in motion, produce the varieties of gameplay that bring players back time and again.