Editor: David Pallai
Cover Design: Tyler Creative

CHARLES RIVER MEDIA
25 Thomson Place
Boston, Massachusetts 02210
617-757-7900
617-757-7969 (FAX)
crm.info@thomson.com
www.charlesriver.com

This book is printed on acid-free paper.

Steve Rabin. *Introduction to Game Development.*
ISBN-13: 978-1-58450-377-4
ISBN-10: 1-58450-377-7

Library of Congress Cataloging-in-Publication Data
Introduction to game development / edited by Steve Rabin.
p. cm.
Includes bibliographical references and index.
ISBN-13: 978-1-58450-377-4
ISBN-10: 1-58450-377-7
1. Computer games—Design. 2. Computer games—Programming. 3. Video games—Design. I. Rabin, Steve.
QA76.76.C672158 2005
794.8'1536—dc22 2005007512

Printed in the United States of America
07 7 6 5 4 3

CHARLES RIVER MEDIA titles are available for site license or bulk purchase by institutions, user groups, corporations, etc. For additional information, please contact the Special Sales Department at 800-347-7707.

Requests for replacement of a defective CD-ROM must be accompanied by the original disc, your mailing address, telephone number, date of purchase and purchase price. Please state the nature of the problem, and send the information to CHARLES RIVER MEDIA, 25 Thomson Place, Boston, Massachusetts 02210. CRM's sole obligation to the purchaser is to replace the disc, based on defective materials or faulty workmanship, but not on the operation or functionality of the product.

Introduction to Game Development

Edited by Steve Rabin

CHARLES RIVER MEDIA

Boston, Massachusetts

2.2 Game Design

In This Chapter

- Overview
- The Language of Games
- Approaching Game Design
- Game Artifacts
- Play Mechanics
- Interface—Player and System Communication
- Systems
- Constraints
- Iterating
- Creativity
- Communication
- Psychology
- Summary
- Exercises
- References

Overview

Although the practice of designing games has existed for centuries, recognition as a field to be studied has been late in coming. The fashioning of games seems to have traditionally been the work of communities and societies, evolving as cultural products over decades and centuries. These cultural products of antiquity may be conveniently referred to as "folk" games [Costikyan04], and would include most of the classic games we grew up with, such as chess, checkers, Parcheesi, snakes and ladders, and the like.

It is almost certain that the ability for people to sustain themselves purely by means of the design and construction of games is a historically recent development. There is little in the way of practical history or treatment of game design method until we come to the latter half of the twentieth century.

In this chapter, you will study an introduction to computer game design. More precisely, you will be introduced to a simplified view of some practices that will prepare you for future study in the field. Upon completing this chapter, you should be able to communicate effectively with a wide audience of designers, developers, and enthusiasts and be able to understand a technical approach to solving game design problems.

First, you will be introduced to a superficial caution on the subject of forms and models, which will give way to brief reviews of a term or two. Next, we will describe games in general, as artifacts of human design constructed for the purpose of arousing emotions via structured play. A feature-based approach will be suggested, along with three domains into which features (or at least their aspects) can be roughly categorized. In other words, we will get a handle on a technical view of games. Later, our attention turns to the constraints that define typical problems facing game designers; the workaday world of game design, execution, and testing. After surveying a few models of creativity and decision-making and suggesting methods for managing and communicating vision, we will wind things up with a review of some work in psychology relevant to the work of game designers.

There is no one "right" way to design games, but there are many successful approaches. However, do not assume that this brief chapter will be able to teach you any of them. Instead, it will provide context and direction for your future studies. If you are left excited and a bit more conversant, we could ask for no more.

The Language of Games

The professional game development industry is still quite young, and its standards in theory, practice, and terminology are still being formed. As odd as it may first seem, debates even remain over the definition of *game*. However, for the most part, differences between one organization's culture and another's are related to use of terminology and workflow. In this chapter, care will be taken to define terms where they may be new to the reader, as well as when differences in use exist.

The beginning:

Play: An activity engaged for the purpose of eliciting emotions.
Game: An object of rule-bound play.

The reason why we and other animals play is not fully understood, although there are plenty of theories and potential answers. In fact, one of the only safe assumptions may be that animals, like people, are motivated to play because it's found to be pleasurable [Grier92]. Indeed, studies have begun suggesting similarities in brain activity that may ultimately lead to understanding the impulses in animal and human play [Siviy98]. They, just like us, may play because it's "fun."

Games, one would assume, seem easier to define. On the contrary, there are probably more diverse opinions on what constitutes a game than on the nature of play. In this chapter, we prefer to treat both terms lightly.

The previous definitions are easy to modify, fitting the culture of any organization. For example, if it is believed that all proper games must include conflict, amend the definition here to accommodate. Likewise, add "win" or "challenge" or whatever your needs demand. By being reductive to start, we stay focused on more rudimentary matters.

After offering two simple definitions for game and play, we come to the third term we will oversimplify into something easy to work with:

Aesthetics: The emotional responses evoked during play [LeBlanc04].

Classical aesthetics is a philosophical discipline that deals with understanding beauty and the ways in which objects appeal to and affect our senses. It is a deep subject worthy of serious study and well outside the scope of this chapter.

Our use of aesthetics, inspired by designer/programmer Marc LeBlanc, simply correlates the experiences of playing with the emotional responses generated as a result. As we did with play and game before it, we leave our term general and easy to work with.

The scope of aesthetics can vary at all levels. At the component level, we can describe *tension*, *surprise*, *fear*, *wonder*, and others that work together to form larger aesthetics such as *adventure*, *challenge*, and *fantasy*. Eventually, we reach the scope of the whole, the experience of a game in its entirety.

Aesthetics needn't be restricted to those feelings we would at first think desirable. Just as the experience of a tragic drama can be as moving and satisfying as a light comedy, we cannot so simply claim they are "fun." It is easy to use common sense to understand the appeal of comedy: laughing feels good. Tragedy, on the other hand, is less explainable this way; sadness and loss hardly seem appropriate to fun. However, when the interests of the audience are safe—that is, that the experiences of the entertainment will not have lasting or *real* effects—then people freely experience all sorts of feeling, many of which are no more desirable than a stubbed toe. However, even fear and anger can be part of satisfying experiences.

That play occurs within a context of safety is one recurrent theme, in the study of games. Players know that the experiences they will have are free from potential harm [Crawford03]. This environment is frequently referred to as a *frame* and indicates to players that the actions taken within it are not part of "real life" [Salen04]. In this setting, players allow themselves to experience emotions that might otherwise not be considered pleasurable, and it resembles the contexts of other arts. However, humans aren't the only species to establish a safe context for play; many others (mammals and nonmammals alike) are known to signal each other their intent to engage and remain in play [Beckoff01].

Feelings within the context of the game's frame are considered safe—they can be experienced without real repercussions. However, when a player experiences a feeling such as frustration, he is being disengaged from the safety of the frame; it affects the player in the real world.

So, while a frequent assumption made while talking about games is that they are "fun," nailing this down has proven very tricky. Our goal, as designers, is to make games that people will like to play, which we do by creating opportunity for experiences. These can stimulate a variety of emotions (including those that arise during problem solving) and do not have to be restricted to those ordinarily recognized as typical of games.

Approaching Game Design

Computer game design and creation is an art form that, like most others, requires good technical discipline to be applied successfully and consistently. Most of the early claims that neither the "art" nor "science" of design could be taught or instructed have been slowly set aside in favor of more reasonable attitudes. (That it may not be taught to you *here* is the fault of the author.) Contrary to these early views, game designers do not require special inborn sensitivities that can only be nurtured and refined through years of toil and experience. There is nothing "magic" about the practice of game design, just as there is nothing mystical about playing and composing fine music. Desire and dedication, practice, and the willingness to work methodically are all that are truly required.

Technical skills are essential, and it is easy to draw analogies to other arts such as music, wherein the "art" is an application of these skills to emotionally affect ("move") the audience. If works of art are seen in such terms, then it's easy to argue that the designs of games are artistic expressions in the form of dynamic models.

Models, Forms, and Paradigms

Science is neat but tricky stuff. A common sense view typically assumes that the goal of science is to pursue knowledge or truth through the careful and objective use of experimental observation [Chalmers94]. This "naïve inductivist" approach assumes that observations are conducted independently from theoretical assumption. Theories then arise from considering the experimental results and integrating new findings into the context of previous knowledge. In other words, knowledge is seen as something based on objective evidence, in turn based on other prior knowledge that we might refer to as "facts." Presumably, the foundation of these facts runs deep, for each is supported by yet more and more layers of fact. While popular, this view doesn't fill well to the history of scientific development. It would be better (although by no means wholly "true" either) to view given periods of scientific thought and approach as describing and testing formal models within paradigms.

A *model* is a representation of something else [Chartrand77]. As you may imagine, models are fundamental to our daily lives. Each day, each of us looks to countless models for guidance and understanding in our relationships to and with the world around us.

Suppose that a new acquaintance invites you to dinner. Driving to his address, you use a road map that models the geographic area and its roads. You consult the

back of an envelope where you have written a model of your route—the directions. Along the way, the velocity of your car is modeled on a speedometer, while the fuel gauge models the level of gasoline in the tank. To determine if you should stop to fill up, you invent a mental model of the car's fuel economy; you factor the reported level of fuel and distance of the trip that remain. At dinner, your host offers a choice of several frozen dinner packages, each showing a picture that models the food inside. You can probably see where this is going.

A *mental model* (first suggested in 1943 by Kenneth Craik) is a psychological representation, a "small-scale" conceptual model, of something [Johnson99]. People construct these models after experience, training, and instruction and use them to represent the structure of situations, reason, and explain the world they live in [Norman88]. Will Wright has described games as running on a "coprocessor system" where player's mental models are half of the experience. [Wright03]

Abstract models are another type of conceptual model useful for investigating specific questions about a subject. They are idealized, where the process of abstraction has been taken to the extreme, and, because of this may even include some assumptions known to be false. These inaccuracies are allowed, provided the important questions can be investigated more clearly as a result.

For example, linguist Noam Chomsky created an abstract model, the ideal speaker-hearer, to investigate the way language communicates meaning. To avoid the distraction of issues concerning a given person's ability to use his language (competence), Chomsky proposed an imaginary situation wherein both participants in a conversation could communicate between one another perfectly (i.e., every utterance from one person would be perfectly understood by the other).

For example, we can use an abstract model to help simplify our thinking about basic topics. Figure 2.2.1 shows a simple game. The game equipment is a board (1×2) and a *piece* (or *token*) that will be used for indicating.

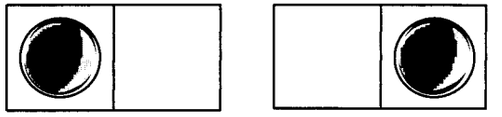

FIGURE 2.2.1 *Two views of an abstract model.*

We begin with a single rule:

- *The piece is moved to the open square.*

Remember that we are working in the abstract and can allow assumptions that would be unacceptable under ordinary circumstances: Figure 2.2.1 won't be any "good" or "fun"; it will barely be a game at all, but will be "game enough" for our needs (observing the results of modification).

Form refers to the shape and structure of anything [MICRA98], and *formal models* are models that attempt to represent this. The basis of modern science is founded on formalizing regularities, both in the structure of things and their transformations [Fauconnier02]. Talk about "scientific" understanding—whether of stars and quarks, neurons and networks, cultural signs and economics, or whatever—refers to knowledge of forms.

Paradigms are like overarching, high-level models; they present a way of thinking about a system [Chalmers94]. They might consist of fundamental laws, theoretical assumptions, standard application of fundamental laws, instrumentation and instrumental techniques, a general metaphysics, and other shared outlooks on one or more fields. They set the way in which questions are asked and how the results of observation are to be interpreted. In other words, paradigms are a way of understanding something, a pattern of beliefs and experiences through which new experiences and beliefs are filtered.

Understand that formal constructs (models, paradigms, theories, etc.) are not reality. Forms may underlie our knowledge of the universe, but just as the photo of the food is not the food [Fauconnier02], they only represent the meaning of that universe.

There are many reasons to be well acquainted with models, but for designers, some are of particular importance. First, our game designs are formal models of the game. Second, we need to understand ways people will create their mental models of the system; much of what we think of as the game experience is better understood as the player experience. Therefore, the game designer is a modeler working within these two spheres: games and minds.

To make the most from our models, we must be ever mindful of scopes and limits. Good models contain details that are appropriate to the purpose at hand. Models with lots of useless data are usually more trouble than they are worth. Ideally, everything significant to its purpose is included within the model boundaries, and all else is abstracted or cut away. For example, a set of blueprints to a house would include details such as architecture (walls, doors, windows, etc.), plumbing, and electrical diagrams. The process of trimming away factors is *abstraction*. Abstracted components might include the exterior paint, location of your wallet, or the nearest shoe store. Even though, at some time, any of these other factors may be important to the resident, regarding the construction of the home they are simply not needed.

Video games not only exploit the latest developments in digital technologies (in turn based on advancements in physical and logical sciences), but are designed and constructed in accordance with paradigms and models of human nature—we need to understand people and how they play. Yet the models of player perception we often work with are not always well accepted by serious scientists working in the fields of psychology and neuroscience. However, game designers are free to accept compelling and applicable models as long as they work.

Designer Jay Minn uses a conceptual model of player reward and reinforcement. His "potato chip loop" operates on an analogy between a player's desire for food and

desire for entertainment. Each reward is imagined as a potato chip that, in all of its salty and fatty glory, is given to the player periodically to encourage him to continue progressing through the game. The chip might be a reward to the player for completing a stage or collecting an object or having performed some interesting feat. At times, the chip may be bigger or smaller, but the most important factor is the temporal spacing between chips—the timing between one reward and the next. Too few, and the player gets too hungry and looks for food elsewhere. Too many, and the player gets full and wants to take a break. Minn's simple idea doesn't attempt to establish formal account of behavior conditioning, but an easily understood, communicated, and remembered device.

Game Artifacts

Games can be described and defined within many paradigms, some more straightforward than others. We can make elaborate, sometimes awkward conceptual models such as games are "stories" or "conversations" or "hallucinations." However, with each attempt to construct new analogies, we risk distancing ourselves from understanding games in simple terms. The typical consumer has little trouble understanding what a game is. Professional developers also know what they are making, more or less; they like to ponder the subject but prefer working to worrying about it. "At the end of the day, all I care about are results."

From a common-sense perspective, it is useful to discuss games simply as *artifacts* [LeBlanc04], objects made by people. For most of us, talk about artifacts is natural; we live in a world surrounded by the products of human invention. The shoes and clothing you are wearing are artifacts. This book you hold as well as cars, diapers, lawnmowers, pennies, golf clubs, and pencils are all examples of artifacts. In the case of each, we understand them as wholes of form and function where identity is tied to physical makeup of the object and its *affordances*—the properties that determine how people believe the artifact is to be used [Norman88].

The idea of treating games as artifacts isn't radical or new. Most developers have been doing this implicitly all along. However, another tradition in cognitive studies has described artifacts used to aid thinking as *cognitive artifacts* [Hutchins99]. Lists, maps, and string tied around a finger are examples, as well as mental artifacts like mnemonics, maxims, and rules of thumb. We might even presume to describe games as cognitive artifacts, but for the time being it's safest to distinguish them as game artifacts; games stimulate the emotions of the players rather than improve their cognition.

Game Frames and Rules

The *frame* of a game is the understood context of play—the time and space that distinguishes play activities from nonplay activities. The moves chess players make are within the frame of the game but their entrance into the room is not.

Rules are the predominant aspect of a game construct. They are often what people refer to by use of the word *game*. Rules provide the formal description of a game's

structure within its frame. The basic purpose of a rule is to contribute organization and context to play. They can be categorized into *procedures* and *delimiters*—two sides of the same coin.

Procedures: The processes and techniques players use to reach goals. As a set of instructions, they describe the methods players may use to perform actions, including:

- Who is eligible to perform the action?
- When is the action to be performed?
- How is the action performed?

Delimiters: Restrictions placed on possible actions. They work, as Chris Crawford describes, to prevent the player from subverting a game's challenges [Crawford03]. Certainly, the experience of challenge is widely seen as critical to building enjoyment.

Without predefined rules, play is an unregulated and nebulous activity. Personal creativity and social skills are used to express and negotiate acceptable behavior. The fantasy play of children (e.g., *House*, *War*, *Cops and Robbers*, etc.) is one common example. Rules for these games exist on a moment-to-moment basis, inconclusive and ephemeral; rules are "made up" while the game is being played. Ad hoc rules such as these force players to make an effort to simply understand the boundaries of the game. By clarifying the manner of play, formal rules allow players to concentrate on exploring different strategies in the system rather than on understanding and interpreting the system itself.

Explicit rules are a basic formal structure of any game artifact. These are sometimes called the "laws" of the game—binding, nonnegotiable, and unambiguous [Huizinga55]. In nonelectronic games, the explicit rules are written into rule books or formed by the playing equipment and moderated by either the players or a separate referee. In electronic games, they exist within the hardware and software architecture.

For example, the rule we described for Figure 2.2.1 is an explicit rule—a procedure; *the piece is moved to the open square.* While this is easy to understand, it is also necessary to recognize that both the piece and board are explicit rules as well. Rather than provide them with a handsome illustration, we could have described them in two more written rules.

Ideally, explicit rules are clear, every player sharing the same interpretation of their meaning. Vagueness is often harmful to the system, leading to confusion, exploitation, or a breakdown in the play. If ambiguity is revealed, the players must agree, among themselves, to clarifications.

It's safe to say that, as it exists, Figure 2.2.1 has some problems. For one, the rules fail to define the number of players allowed in a game. In addition, we don't indicate when the piece is to be moved. Interestingly, if we specify that Figure 2.2.1 is a one-player game, our job is finished; there's only one player and he may move the piece whenever he so chooses. However, if we allow more than one player, the players may

run into some confusion; "When do *I* go?" So, we might add "Each turn..." to the wording of our first written rule to clear up any confusion.

Implicit rules [Salen04] are unlike explicit rules in two important ways: they are unwritten and are not necessarily binding. To players, implicit rules may appear as critical to the game's enjoyment as are the explicit rules. For example, implicit rules often include an agreement between the players not to exploit certain weaknesses in the game system [Sniderman99].

Robert Fulghum tells of his experience of childhood hide-and-seek games with another player that "hid too well" [Fulghum89]. After looking for a very long time, the other children would give up and continue playing without him. Eventually, he would appear upset about being abandoned and arguments over the rules would erupt. Disagreements about the rules would then turn into disagreements about who made up the rules and so on.

Implicit rules are broad assumptions made in the "spirit" of the game, not amendments to explicit rules. For example, if Fulghum's playmates agreed to a new rule prohibiting hiding in a certain spot (e.g., "no hiding under leaves"), that rule would not be implicit but explicit, known and accepted by the players. However, it would still fall short of the implicit rule under which most of the children were operating: *you shouldn't hide so well that it is impossible to be found.*

We have to leave certain matters up to the implicit rules. In fact, most matters in nondigital games are governed by implicit rules. Few games instructions specify, "You shall not pelt your opponent with rocks and garbage," or, "Pouring a milkshake over the board is not allowed while the game is in play." These countless rules do not need to be stated explicitly because players are expected to have a *lusory attitude*—the willingness to submit to the conventions of the game [Suits90]. At the most basic level, the lusory attitude demands that the explicit rules will be followed. However, at its fullest, the lusory attitude is the willingness to abide by the cultural conventions common to those with which you are playing [Salen04].

Some rules only come into effect at given times or in particular circumstances. These often serve to create variation, govern game progress, and ensure the system remains within preordained limits [Fullerton04].

Rules in electronic games are formed by the platform and software architectures. The first advantage is that no ambiguity in explicit rules is possible; the computer referees the game, and players are forced to abide if they are to continue playing. Second, the rule systems can be modeled in much greater detail because players are not required to process all of the rules themselves. The richness and responsiveness of computer simulation can operate at a level of sophistication impractical by other means.

Features

In software development, a *software methodology* (*method*) is a set of procedures and practices developers use to create programs. There are many of them, each representing a given philosophy on the best way to get software built. Most of these methods

are *feature-based*: they describe a paradigm of software architecture as organizations of small, functional units. In this paradigm, a software application (e.g., word processor, game, virus, etc.) is expected to perform one or more tasks. The application performs a sequence of operations to complete each task. This set of operations, that performs a given task, collectively forms a *feature*.

As with all systems, features vary in size and complexity according to their job. When several features are being discussed as a whole, they are a *feature set*.

How do we determine what is a single feature and what is a set? How fine an approach should we take when describing these units? For example, we think of a map in a game as a feature. However, looking closer, we see that there are smaller features in there. Unit locations are displayed. The color of each unit reflects its team. The terrain is colored. How do we know when to stop? The level of detail that goes into feature descriptions will depend on the development method being used and the culture of the organization.

In this chapter, we discuss the aspects of game features framed in terms of three major domains shown in Figure 2.2.2: *mechanics*, *interface*, and *system*. It is an *approximate model*: one that puts us in the ballpark of form. This starts us with an easy way to conceptualize a structure that we can build up.

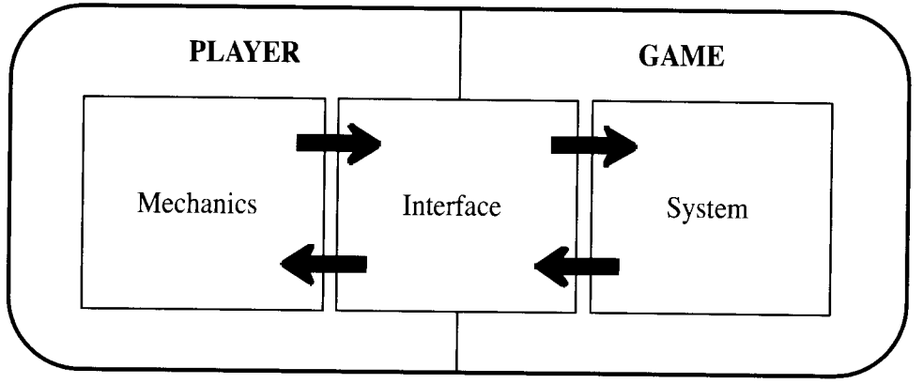

FIGURE 2.2.2 *A model of the player-game relationship.*

The model isn't complicated:

Mechanics: Things the player does.
Interface: The way the player and game communicate.
System: How the game structure behaves.

In our model, the relationship between player and game is shown. Players experience the game through the interface, which processes the inputs and outputs of the system. Back and forth, back and forth, information is moved along the structure.

Players act in the game by manipulating the values of *control variables*, such as changing the angle of a joystick or the pressing of a key. In turn, these inputs affect the various *state variables*, which describe, by their values, the current state of the game [Isaacs65]. If the game were halted at any particular moment, only the values of the state variables would be needed to resume play; this is the data written in a game save file or a character record.

However, there is a problem: we can't build players! All we can do is design the structures of the game. Considering that a player's imagination can be half the experience, we must design systems and interfaces to produce play mechanics.

Play Mechanics

Gameplay was once the term du jour; to refer to an experience in a game, it was handy. Gameplay usually indicates either the feeling of playing a game or the activities engaged in a game [Costikyan02]. However, a lack of precision in its definition has led gameplay to fall on hard times; now it seems out of style. Being hip is probably not worth the effort, but clarity is useful.

Here, when discussing specific player activities within a game, we will use *play mechanics*, *game mechanics*, and *mechanics* synonymously. *Aesthetics* will be used for the feelings experienced during play. If you, however, prefer to use gameplay, please do.

Play mechanics are the activities in which a player participates during the game; what the player does. This can include actions and strategies as well as the player's construction of a mental model of the game.

It might be easy to consider play mechanics as a version of Donald Norman's Seven Stages of Action applied to game playing. Donald Norman describes a model of action, as shown in Figure 2.2.3, in the world at large; it works the same way when applied to games [Norman88]. He structures the action in three aspects: *goal*, *execution*, and *evaluation*.

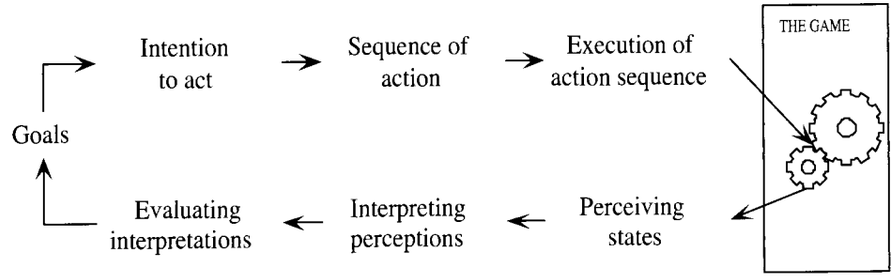

FIGURE 2.2.3 *The Seven Stages of Action [Norman88].*

A *goal* is first formed that models a desired state. This goal does not describe how something is to be done, but what is desired as a result of an action. During the *execution* phase, the goal must be turned into an *intention* to act, what Norman calls the

specific statements of what is to be done. Intentions are put into an *action sequence*, the order in which the internal commands will be performed. At last, this mental processing is turned into a physical manipulation of the controller: the *execution.*

The evaluation stage begins as we *perceive* the state of the game through the interface. Perceptions of the game states are *interpreted*—understood according to our mental model of the system—and *evaluated* by comparing the current states with our intentions and goals.

As our model of a game (in Figure 2.2.1), this is an approximate model. Most actions do not occur in discrete stages. Not all action needs to progress through every stage in the sequence. Most doings in the real world are made up of many steps, each an action in itself.

We can understand a game mechanic, at the finest level of detail, as an action leading to a result in the game system—one complete turn of the seven stages of action that produce a perceived change in the game state. These *primary elements* [Cousins04] can be things such as "move forward" or "jump" or "select item," depending on their context within the game.

In a way, to design games is to devise model systems that create opportunities for player action. We attempt to satisfy the basic desire of players to do something, whether engaging, intriguing, or otherwise significant.

Mechanics must be able to fit into the players' mental model of the game—they must have the opportunity to understand it through their relationship to the system. Often, players are not *aware* of mechanics in the sense that they understand how they are being affected by the results of their actions.

Game designers do well when they remember that mental models in their mind differ from those inside the players' minds. The designer consults a mental model when the game artifact is being built, but the reality of the *system image*—the portion of system that can be perceived from the outside, including documentation and other available information—does not always correlate perfectly to that intention. Users form their mental models based on insights received while interacting with the system image (see Figure 2.2.4). Player and designer are not in direct dialog with each other; the player is not told by the designer how the game is supposed to operate, play, or feel [Norman88].

Designers often spend time visualizing the actions within the spaces of the game. They imagine that players will do certain activities and enjoy them, while fearing and avoiding others. To help understand the player's viewpoint and strategy, designers imagine themselves playing the game. "What would I do?" These thought experiments are crucial to designing well; a cheap and quick way to test ideas. However, you must remember that the player is not you and will not understand the game as you do. Player strategies will *always* be based on their unique conceptual model of the game. Be cautious of relying on too many assumptions about these understandings. We can only expect players to know what they have been told or shown by the game; sometimes a solution that seems painfully obvious to the designer is painfully frustrating to a player.

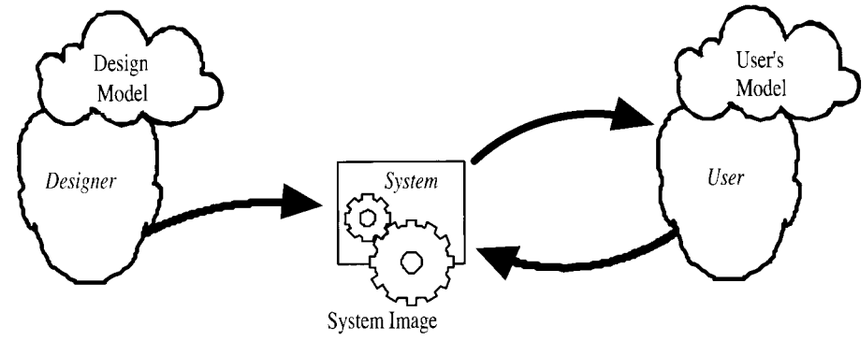

FIGURE 2.2.4 *Conceptual models, ours and theirs [Norman86].*

For every game, there is one or more *core mechanics*—the fundamental action (or set of actions) that characterizes the typical activities engaged by the player. Core mechanics are patterns of behavior where the player cycles through the same kind of same actions repeatedly. In a first-person shooter, core mechanics might be "run, shoot, and explore"; in a strategy game, "harvest, build, expand, and conquer." Finding the proper set of activities to make up the core mechanic will depend on the type of game and audience intended. Balancing those will rely, as the other mechanics, on prototyping, testing, and tuning the system to satisfy the needs of that audience.

Premise—Analogies and Metaphors

The *premise* (or *fiction*) of a game refers to the metaphors of action and setting [Fullerton04]. This premise directs the players' experiences and gives them a context within which their experiences fit. Through the course of play, users *map* the states of the game and their actions to the premise, actively participating in the metaphor.

Consider aspects typically related to the *story* of a game: setting (time and place), characters, motivations, relationships, and so forth. It is easy to see how these all work together to help form the premise. In fact, this method of establishing premise has existed for centuries in narrative forms.

The Legend of Zelda background story forms the backdrop against which play will occur. From the moment players engage the game, they control Link in his quest to save Princess Zelda and the land of Hyrule. Likewise, the terse introduction to *Robotron: 2084*, describing how superior robots decided to eliminate humans, also establishes the backdrop action, even if only to elaborate on the imperative given to the player to save the last humans.

Game premises can be complex, as in the case of character and story-based games. Alternatively, they may be little more than an abstract metaphor that allows people to manipulate agents in some understandable way, like arranging shapes in *Tetris* or stacks of colored balls in *Magical Drop*.

Premise extends beyond the story's canonical narrative. It encompasses all game activities. Link's sword and shield, his abilities to store items in an inventory, and his ability to interact with the inhabitants of Hyrule are all part of the game's fiction. In fact, all of the actions available to Link during his adventure are members of its premise.

Discussing actions in a game, players are rarely using literal terms to depict the performance. Recounting a game session, they will use the language of the premise, primarily describing actions, perceptions, and strategies experienced within its context. The portrayal is largely symbolic, representing activities such as fighting, buying, building, and so forth. All of this conduct occurs within the boundaries of the game's premise. The vocabulary used by players in describing play reflects the metaphors evoked by the game [Crawford03].

Understand the distinction between the model (the game) and the activity being modeled. Playing a typical basketball video game, for example, is no more playing the physical sport than is reading a comic book the same as having superpowers. The goal of the video game is to make the players *feel* as though they were controlling athletes on court. Those without interest in sports might still enjoy sports games, just as people do not play *The Sims* because of an increased interest in taking their own garbage out. Model and reality are different.

Premise is more than just about simply establishing a setting or a tone; it can significantly alter the players' mental model and the way in which they develop strategies. Consider a classic example, a selection task designed by psychologist Peter Wason. Four cards are set before you, each having a letter on one side and a number on the other. You are told that a card with a "D" on one side must have a "3" on the other.

In Figure 2.2.5, which card(s) do you need to turn over to see if this is true?

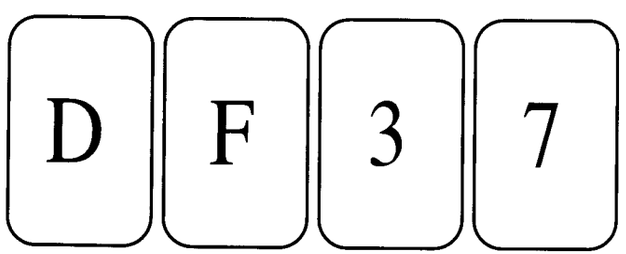

FIGURE 2.2.5 *First task.*

Make a quick note of your answer. Now, imagine that you are working in a restaurant and must make sure that no one under legal age (18) is drinking. You can check what a person is drinking and how old he is.

In Figure 2.2.6, which person(s) do you need to check to make sure no law is being broken?

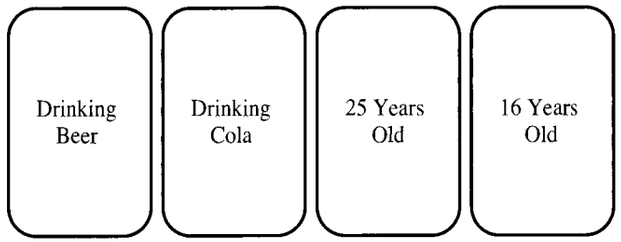

FIGURE 2.2.6 *Second task.*

These tasks have been featured in several studies, and over 90 percent of people, when faced with the first task, answer incorrectly [Pinker97]. Yet most people are quick to give the correct answer for the second task. (Leda Cosmides and James Tooby proposed that people possess an innate capacity for detecting "cheats.") The illustration indicates that people may fail to choose a successful strategy (checking "D" and "7") while operating within one premise, while another allows them to readily discover the solution (checking "Drinking Beer" and "16 Years Old").

One last point worth making on the subject of premise is regarding its content. Writers are often taught to distinguish *plausibility* from *possibility*. When something is considered possible, it is capable of happening in the real world; planes may fly, but muscular men in cape and tights alone cannot. However, in creative writing, you are free to suggest any kind of truth as long as it is plausible within the rules of the universe you've described in your fiction; there must be an explanation that *makes sense* in the particular context of the environment. As it relates to a player's map of the premise, we can view plausibility as the domain (area) of the player's map; when an object or situation is implausible, it is beyond the border of what the player can (or is willing to) map acceptably.

The limits of plausibility are subjective, understanding what audiences will and will not allow being essential to pushing those boundaries. For example, games that attempt to blend science fiction and fantasy universes together have to account for the segment of their audience that cannot accept the two fictional conventions in the same space. Similarly, it is common for fantasy massively multiplayer online games (such as *Dark Age of Camelot*) to have to provide special *role-playing* (RP) servers for their customers who find the implausibility of a wizard named "MonsterTruck," since it shatters their enjoyment of the game.

Choices and Outcomes

Choice—the act of choosing—lies close to the root of how we understand our game experiences. Sid Meier, an industry legend responsible for *Civilization* among other titles, is credited with observing that a game is largely a series of interesting choices. This quickly became one of the earliest and most popular maxims to be adopted by

other designers. It gave direct and succinct articulation to an experience everyone understood firsthand: satisfying play.

A choice may be known as a question asked of the player. What unit would you like to build? Which ally will you betray? Which door will you choose? And so on.

Through the course of a game, player choices create imperatives for actions. These actions by the player result in *outcomes*, a result or consequence. If choices are felt at the root of our experiences, then outcomes are the fruits we taste—sweet or bitter—as results from the system are interpreted.

One way to describe the landscape of potential choice is as a *possibility space*. This space represents the set of possible actions in a given environment. We might refer to a game as having a wide possibility space, indicating that there are many possible choices for the player to make. A narrow or small possibility space indicates there are very few options available. Many games, especially those depending on a predetermined narrative sequence, will have possibility spaces that change during the course of play; many things to do and ways to do them at one point, building to momentous, key decisions at others.

The *consequence*, or *weight*, of a choice might be called the "significance of its outcome." This is one of the most important factors in designing choices. We can understand the scale of consequence by relating it to the alteration of the game; the greater the consequence of a choice, the more significant its outcome will be to the course of the game. A well-designed choice will often feature both desirable and undesirable effects [Fullerton04].

Choices that your players must deal with should involve their desire to achieve their goals. Insignificant or irrelevant decisions are almost always detrimental to the experience. The more decisions the players must make outside of this scope, the weaker their connection to the experience will be. While particular distributions will differ for each particular style of game, play should be a progression of actions periodically marked by significant decisions.

Excitement, anticipation, and tension are created when the player is faced with weighty choices. On the other hand, if every choice is too melodramatic (i.e., life and death, save the universe, etc.), then the experience risks violating plausibility.

In keeping players engaged, you try to enable an experience that oscillates within an ever-rising balance of challenge and ability. You will often want the weight of choice to behave similarly—more significant on average as the game progresses.

One way to make decisions more significant is to keep the available choices *orthogonal*—distinctive by nature of quality and property [Smith03]. With a set of orthogonal choices, the desirable and undesirable effects of each choice are different from each other. This difference is not just in scale, but also in kind.

To help assess how effective a choice is, the book *Game Design Workshop* offers eight qualities to characterize [Fullerton04]. These are a starting place; add your own qualities as you need—we add "orthogonal" here.

Hollow: Lacking any real consequence.
Obvious: A decision that leaves no choice to be made.

Uninformed: An arbitrary decision.
Informed: Describes a deciding player with sufficient information.
Dramatic: Strongly connects to the player's feelings.
Weighted: Positive and negative outcomes with each choice.
Immediate: Effects of the decision are immediate.
Long-term: Effects of the decision manifest over an extended period.
Orthogonal: Distinct among other choices.

A hollow choice lacks measurable outcome, while an obvious choice leaves nothing to be decided—a player would be either stupid, self-destructive, or both to choose the nonobvious option. Uninformed choices are those made without enough information to support meaningful judgment; the player selects a strategy at random. Informed choices are those the players can make, confident they know all they need to about the decision. Dramatic choices produce emotional responses within the premise, such as when players must abandon a companion to save themselves. Weighted choices have significant desirable and undesirable outcomes. The outcomes of immediate choices are produced close to the decision event, while long-term choices have lasting results. Lastly, the orthogonal choice is one that is distinctive relative to other potential choices.

Goals and Objectives

Few words are so intimate or have such a history with games as the word *goal*. There are many meanings one might consider, each depending on the context in which it is used. These can be generalized into two of the most common:

- An objective
- An intentional outcome

Objectives are designed requirements that players must satisfy to accomplish a particular outcome [Fullerton04]. Encoded into the structure of the system itself, objectives are formal properties of the game, gating player progress.

Goals (or *incentives*) are subjective notions that direct action toward outcomes [LeDoux02]. Put another way, a goal is "what the player *wants* to do." All games involve goals, even those rare few in which we are unable to define clear high-level objectives (i.e., *software toys*). Goals are personally created and maintained rather than formally structured by the system.

Goals are subject to player discretion and not just those intended by the game's designers. Every opportunity for choice or action is an opportunity for goal setting. Goals provide motivation and direction to the player. These goals most often include reaching game objectives; if players aren't motivated to reach the game's objectives, they usually stop playing.

Typically, objectives are set at many levels—from major requirements to minor steps unessential or unrelated to the main objectives. In keeping with the competitive nature of most games, winning is the most common objective. So common that it is frequently cited as a formal requirement for games. Even in those games that lack

victory conditions, most players will expect objectives by which their progress can be measured. Although there are a growing number of examples that demonstrate players are more than happy to dispense with high-level objectives when system interactions are compelling enough without them (e.g., *The Sims*).

Objectives are key elements to establishing a fixed challenge. When goals are self-directed, players are free to overcome adversity by changing or modifying their goals. For most games relying on challenge to provide dramatic aesthetics, the result is a breakdown in motivation.

Objectives may be the same for each player, or they may differ. Usually, when a game provides objectives that differ for the players, they depend on conditional roles (such as in team sports where a role might be distinguished as *offense* and *defense*).

Just as with other aspects of the mental model, users will not always have the same set of goals the designer has intended, as in Figure 2.2.7. Choices are invented by the player in light of the system's *affordances*—the apparent ways something can be used [Norman88]. Forcing player goals to align to specific intentions requires the use of objectives. Players may still have their own notions, but must make adjustments when they cannot achieve them. Balance your need for players to behave within the system with their desire for *expression*—the latitude for them to set and achieve their own goals.

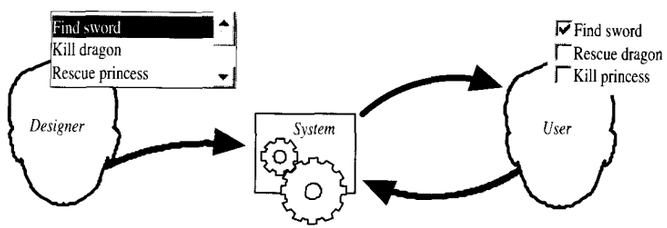

FIGURE 2.2.7 *Objectives and goals aren't always the same.*

Measurable outcomes require objectives. Without them, players become solely responsible for generating their own goals. This is the hallmark of the *software toy*.

Resources and Economies

Whether in games or in the world at large, *resources* are things that are used in support of some activity (such as manufacturing) and are drawn from an available supply—they are the factors of production and the bases of development. Resources needn't only be physical things: for example, entrepreneurship and education are viewed as important resources in capitalist systems. In games, resources are the things used by players and other agents to reach goals [Fullerton04].

Resources may exist within the premise of the game or without. Within the game's premise, we might expect resources such as materials, people, magic power, or health.

Outside of the premise, we might consider functional components as resources, such as save games or lives; these can be provided in limited supply to build tension, challenge, and provide opportunity for further strategy but without necessarily being understandable within the game's premise.

To be meaningful, a supply of resources must not only be useful, but also limited in some way. To limit a resource, we can restrict the total supply to a finite amount, or restrict the rate. We might instead provide special conditions for their use or employment, or create penalties for their consumption.

The relative value of a given resource can be determined by looking at the relationship between its utility and its scarcity. Expect problems to arise if a resource is either useless or readily and infinitely available. To consider these issues in the context of systems, it is helpful to view resources in *economies*—closed systems of supply, distribution, and consumption.

Some typical questions regarding resource economies include:

- What resources exist in the game?
- How and when will a player use the resources?
- How and when are the resources supplied?
- What are their limits?

Player Strategy

People are unlikely to apply formal logic in the course of making everyday decisions; they may not be necessarily "rational" in the sense of computing likely outcomes based on statistical assessments. Instead, we tend to let experience and common sense guide most of our actions and reactions [Norman88]. Why not? What has served evolution probably remains sufficient to serve us in our daily lives. (We will have more to say on economics and game theory later.)

From our experiences with simple systems, we acquire a common-sense world view of linear causation (see Figure 2.2.8); every event has a cause preceded by an infinite history of causes before it. Likewise, the standard approach to strategizing tends to be linear, appropriate to this general outlook. As we will see later, most systems of any complexity, including games, do not often behave in a linear fashion.

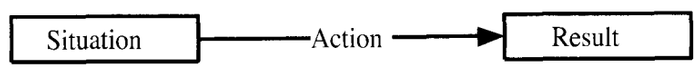

FIGURE 2.2.8 *A linear outlook/strategy.*

Strategic challenges do not need obvious or simplistic solutions, but you must be aware of what assumptions you expect the player to have about the way in which the game works.

Game Theory

Game theory (sometimes known as *multiperson decision theory*) is a branch of economics that studies decision making. It asks questions about optimal strategy in a variety of situations, especially those where outcomes are dependent on the choices of others. Originally developed by mathematicians working on economic problems, it was soon applied in sociology, biology, and many other fields.

Many designers, at first, expect game theory to help them create systems and find themselves quickly disappointed; game theory is concerned with numerical analyses of strategy. In other words, game theory studies playing games, not making them.

In a nutshell, game theory uses models, in the form of games, to study how decisions produce outcomes. The *utility* (a measure of desire) of an outcome is represented by numeric *payoffs*. Higher payoffs correspond to higher utility. The bias of players to utility is known as their *preference*.

Game theory games are played by *rational players*, abstract models with one goal: to maximize their potential utility. Rational players solve problems using pure logic, with complete awareness of the game's state. At any point during a game, they understand every option available given the information they have been allowed. We use these conceptual models to avoid worrying about individual intelligence or ability and to focus on understanding how the optimal strategies can be determined.

Three categories of games describe of how decisions are afforded [Baillie-de Byl04]:

Games of skill: One-player games where outcomes are determined solely by their choices.
Games of chance: One-player games with the outcome determined, in whole or in part, by probability (or as a two-player game with *nature* as one of the players).
Games of strategy: Competitions between two or more players.

In a game of skill, every choice is a *decision under certainty*: the player knows the outcome of any decision beforehand. It is up to the player to select the best available strategy.

Choices made during games of chance are known by two types. In cases where the probabilities are known—such as 1 in 37 for ("European") Roulette—the strategy made by the player is a *risky decision*. When probabilities are unknown, the player makes *decisions under uncertainty*.

Games of strategy model competitions between two or more players, with outcomes determined by their collective decisions. Of these games, there are three types:

Zero-sum: Players' outcomes are directly opposed.
Pure coordination: The best outcome is the same for all players.
Mixed-motive: Payoffs between players do not correspond to each other.

Zero-sum games are two-player games of strategy where the payoffs, when added together, equal zero. Player preferences are directly opposed. This is the classic win-lose arrangement of many popular two-player games: chess, checkers, go, and so forth.

One elementary tool in game theory is the *decision tree* (or *game tree*). Each node on the tree represents a choice made by a player; other nodes along the same row represent alternate choices available in the same turn; games that resolve simultaneously are still discussed in turns. Games start as a single node: the initial condition of the game. From there, lines branch out to nodes representing possible moves that the first player could select. In each successive row, lines branch to more nodes representing moves available to the next player in the sequence.

The biggest limitation with decision trees, as tools, is their complexity. With a very small range of potential moves, they can be useful diagrams. However, with any real game, they quickly become so complicated that they are rendered impractical. In Figure 2.2.9, for example, with only three choices possible at any point, the Rock-Paper-Scissors game has nine potential outcomes. Consider chess, with its 20 possible openings and 25×10^{115} distinct 40-move games (while the number of electrons in the universe is 10^{79}) [Parr02], it is all but practically impossible to map such a tree.

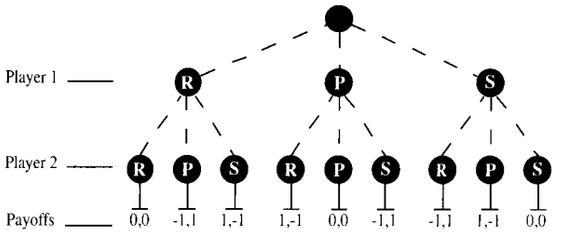

FIGURE 2.2.9 *A game tree for Rock-Paper-Scissors.*

(Outcomes for zero-sum games are usually shown with just one figure, since the other player's score is assumed known. We show both first and second players' scores to introduce the notation for multiple players—each is given in order.)

Cake-cutting is a zero-sum game that we can use to wet our feet. Imagine two children are sharing a piece of cake. Their father has decided, to avoid an argument, that one child will cut the piece and the other will select their pieces.

We can use the *payoff matrix* in Figure 2.2.10 to examine the strategies available to each player and the outcomes that will result. On the left is a friendly version that lists, explicitly, the outcomes possible during our game. On the right, each pair of outcomes is represented with a payoff value signifying utility. Read payoffs left to right, the first value corresponding with the first player and so on.

In *coordination games*, the best outcome for the game is the same for all players. When a condition of perfect information exists, the choices are trivial. When the information is imperfect, with players unable to communicate with one another, a greater challenge arises.

In some of these games, however, there exists a quasi solution—it isn't a proper formal solution, but a very strong tendency toward a particular intuitive choice. As

	Select larger piece	Select smaller piece		
Cut evenly	One crumb more	One crumb less	-1,1	1,-1
Cut unevenly	Smaller piece	Larger piece	-5,5	5,-5

FIGURE 2.2.10 *Cake-cutting payoff matrix.*

Roger Schelling found, some strategies emerge more strongly than others [Baillie-de Byl04]. For example, Schelling asked a series of paired people to select heads or tails (as in coin tossing). They were told that, should both pick the same side, they would win; otherwise, they would lose—86 percent of those questioned chose heads [Schelling60].

In another experiment, Schelling gave people a scenario where they would pick the place and date to meet someone they hadn't met. Then they were told that, without being able to communicate to the stranger, they would have to pick the time to arrive at the meeting place, within a minute of the other person. Almost every participant chose 12 P.M., noon.

These points of prominent strategy are "Schelling points" and the strategies are known as "salient." Because these strategies seem to be based on a broad spectrum of cultural knowledge, a zeitgeist, it is almost impossible to calculate.

Many designers reconsider game theory once they begin to understand how it can be useful to their needs. At a higher level, it can be useful for analyzing mechanics—game theory can help you design and tune choices and outcomes. At a low level, it can be useful within the behaviors of game objects (AI strategies, etc.). Lastly, game theory provides a language that we can use for describing and discussing our games and the kinds of choices they make available.

Interface—Player and System Communication

The interface is the set of components by which players interact with the game system. Both input and output features work together, establishing a link between player execution and system reports. Any information passed from user to system, or vice versa, is channeled through interface components. Players *experience* the game by interpreting their perceptions of these transactions. From a player's perspective, the interface is the form of the game, while the system content is its function [Crawford03].

The interface contains both hardware and software elements. For example, in Figure 2.2.11, the interface consists of display, sound, keyboard, and mouse. The player

receives information from the game system by seeing and hearing it, and executes actions by manipulating the keyboard and mouse.

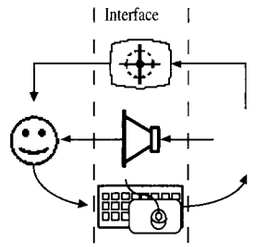

FIGURE 2.2.11 *A typical PC interface.*

The parts are easy to categorize, as most of us are already familiar with the types:

Output: Communicating information to the player
Input: Communicating information to the system

With the technologies currently available to most consumers, players can get information from video games in three ways: seeing (visual), hearing (audio), and feeling (haptic). Of these, only the visual and audio systems can be taken for granted; it's safe to assume that a general audience will be able to see and hear the game. Many platforms have peripherals with *haptic* features available, but these we cannot always rely on to be engaged.

States in video games can be staggeringly complex, but poorly understood systems can risk confusing players. When every consequence is unexpected, every decision misunderstood, players feel cheated. You must constantly consider how the players will understand the system and events and get them the information they need through the interface.

Output—Video

Vision is the dominant sense to inform us of the world beyond our bodies. As well as having large and mobile eyes, we have a large amount of the cerebral cortex associated with visual processing, much greater than any other sense [Goodale99]. Understandably, the most salient features in a video game are usually represented visually.

A *graphical user interface* (GUI) was first known as a paradigm of computer control, a friendly metaphor: files and folders that could be managed with a mouse. In games, the term "GUI" still tends to be applied to those graphic features that either present control metaphors (buttons, etc.) or labels. However, for purely practical reasons, it can be helpful to understand the graphical interface as encompassing all graphical elements, including those within the game environment itself.

Every scene in a game is viewed in two dimensions, through the physical screen (LCD, CRT, etc.). Even 3D games are still limited to rendering scenes onto the plane of the video screen. We identify, collectively, the set of features directly involved in rendering the game environment as the *graphics engine*.

The engine is one of the most significant parts of a game's user interface, determining how scenes will appear. The *camera* represents the vantage from which the current scene is drawn, the viewpoint. We described cameras by their perspective or a combination of perspective and behavior.

Typical perspectives are:

First-person: Camera views from a character's (or object's) point of view in 3D; controls are usually direct, moving camera and avatar together.
Over-the-shoulder (OTS): Camera views from above and behind the subject in 3D.
Overhead (or **top-down**): Camera views from directly above the scene.
Side: Camera views from directly to one side of the scene.
Isometric (or **three-quarter**): 3D view from a fixed perspective, usually high above, giving a view of roughly three-quarters of any given object in the scene.

Powerful 3D engines are now commonplace, both on the desktop and in gaming consoles. The level of detail and variation that can be displayed on the average gaming PC is beginning to approach photo-realism—in real time. New technologies make presentations more compelling and certain perspectives easier to manage than before. These advances enable us to model activities in ever-greater detail and effectiveness, but do not necessarily change *what* we can model. Indeed, many current games are *translations* of earlier mechanics to new interfaces.

The *Grand Theft Auto* (*GTA*) series presents a great opportunity for examining play mechanics translated from one mode of presentation to another. The core mechanics of the original, which was set in 2D, have been retained in each installment, even though the latest titles are in 3D. Players move a character freely about an environment populated by numerous agents (characters and vehicle objects). Agents each follow simple behavioral rules describing how to act in the environment and with other agents. Each title introduces more behaviors and more choices for the player, but the core model—stealing cars and performing mission objectives in an "open" world—is present from the beginning to the end of the series.

First-person perspectives are often assumed to connect the players closely to their character. In fact, it can actually create distance between them because the players revert to assuming their own identity: they play as themselves rather than as the character. This can be overcome with persistent reinforcement of the premise, such as in *Half Life 2*.

Other issues common to FPS games are the nuances of world and object interactions. Without a clear frame of reference, scale can be difficult to judge, and presenting believable world interaction is tricky. Navigating through game spaces often first requires that players acquaint themselves with the dimensions of the game, testing widths, heights, and depths to discover how their virtual body fares. Designers and

artists in these circumstances have to make sure they follow some standard system of scale to help clearly indicate to players when something can be navigated or not.

Third-person perspectives present avatars more plainly. As a result, it is easier for players to identify strongly with them. A typical problem of third-person perspectives is the lack of connection to the character's focus or aim; the camera is usually over the shoulder. This makes action games more difficult to tune, especially those featuring ranged combat. In addition, the avatars can obscure a fair portion of screen real estate when the camera is close.

Output—Audio

Because our vision is so dominant, it is easy to underestimate the importance of sound to compelling experiences. Proper use of sound effects can reinforce the premise, evoke emotions and moods, and inform the player when significant events of the game change.

Our sense of hearing is remarkably sensitive, capable of detecting a wide dynamic range of sounds, both in pitch—*frequency*—and in loudness—*intensity*. Despite this range, our hearing remains finely tuned—we can detect a difference in frequency as fine as 0.2 percent and a difference in intensity as small as one decibel (dB) [Richards99]. Our ability to locate sounds in the world is astonishing: we can detect differences between each ear as small as a few microseconds.

Game audio is typically broken into in three categories: *music*, *sound effects*, and *dialog*. Production pipelines for audio work vary between organizations, but, from a designer's perspective, considering the subject within these three should work well enough.

Sound effects are sounds triggered by game events. They range from the simple feedback of a mouse click to a chiming alert signal to bloodcurdling screams of the murderous War Beast! To a greater extent than the other types of sounds, the effects serve to inform the player of game states. Often they are used to provide information when the player's eyes may be to busy to look away from their current focus.

Table 2.2.1 is a list that designers might supply to the audio team. Categories provide context for the event so the audio designer understands how the sound is to be used. The game designer will often describe the kind of sound he imagines working, to give the audio team a starting place, a reference.

Table 2.2.1 Sound Effect List

Category	Condition/Event	Type
System	Menu select	Soft tap
System	Overflow warning	Short siren call
Action	Flush	Swooshing water
Action	Flush	Swooshing water—loop
Environment	Large pipe	Tight echoes
Power-up	Toilet duck	Brief quack-quack

Dialog is any voice recording that the player will need to understand during play. Occasionally, dialog will be used for warnings and announcements ("Orange alert!"), but the most common use is to give game characters a voice. Although technically dialog can be seen as a sound effect, its unique challenges often require separate studio and production schedules. Game designers must work closely with audio designers to ensure that the intent and meaning behind the words is clearly understood so that they may properly direct the *voice talent* (actors) during recording.

Music is a powerful tool for establishing moods and reinforcing themes, arguably more potent than narrative at tying players to the attitude and tension of the environment. It can motivate, frighten, and transport the imagination.

The unique needs of games have given rise to new technologies and compositional techniques known collectively as *interactive music*. Producing good interactive music requires forethought and planning on the part of the designers, who then work with an interactive music producer. Fortunately, your game's aesthetic objectives will often help determine the basic music categories. From there it is a process of refining the detail and mapping the meanings to events. Table 2.2.2 shows an example of how certain places or events might trigger particular music.

Table 2.2.2 An Example List of Interactive Music Needs

Category	Condition/Event	Aesthetic/Feeling/Meaning
Self	Combat victory	Triumphant
Party	Companion death	Sorrow
Location	Toy shop	Hospitality, friendliness
Environment	Woodlands	Safety, idleness
Environment	The Editor's Lair	Danger, impending doom

Because the aesthetic goals of your game are not likely to change, it is best to begin planning for music early. This will allow time to refine goals, and nail the themes exactly as you want.

Input—Controls

The *controls* are the physical input devices players use to communicate to the system.

Most platforms will have a standard controller layout that designers will map to their game actions, usually some combination of keyboard, mouse, joystick, or game pad. Other, nonstandard controllers are available, but the expense in production and distribution can make these products unappealing to publishers.

Control inputs are the manipulations, by the user, of the game's controls—the way the controls are used. These are patterns of presses, clicks, and movements performed to initiate actions. These inputs are detailed in a *key map* or *control table*—see Figure 2.2.12, a diagram showing control input, action, and context for the event.

Action	Control	Context
Left	←	all
Right	→	all
Forward	↑	all
Backward	↓	all
Sprint	X	all
Pass	○	Offense
Lob	▲	Offense
Shoot	□	Offense
Steal	○	Defense
Block	▲	Defense
Hit	□	Defense

FIGURE 2.2.12 *A simple control diagram.*

For example, in a fighting game we might assign a button (X) to trigger a punch. Each time the player presses X, his fighter hits an opponent for one unit of damage. If the player taps X in a special pattern, rather than a single punch, the fighter performs a three-punch combination that deals five units of damage. The same button on the controller was used, but the control inputs determined which attack was initiated.

Control inputs are not strategies; they are controls. The ability to perform a combo in a fighting game is not a strategic decision—it is not a choice, as such. Deciding *when* to perform moves is the strategy.

The ability for players to express themselves, to do what they want, must be balanced against the intuitiveness of the control design. Generally, the more diverse the actions allowed in the play mechanics, the more demands will be made of your controls.

When a given set of controls needs to be accessible at the same time, try to keep their layout and activations as orthogonal as can be practical: players appreciate clear, understandable mappings between their controls and actions. In other words, controls should be immediately distinct from each other. You should only mix mappings when contexts can be apparent to the player.

Controls are one of the most difficult aspects to innovate in games. For every known style of game, there already exist schemes of control that are accepted as norms. However, if your controls simply mirror the function of all other games in the genre, it is possible that your design may not be creating any new experiences [Dietrich02]. Designers should not be discouraged from inventing control systems that break with common conventions when the new scheme is better suited.

We might look for inspiration in a recent boxing game. Despite having been one of the earliest and most well-known subjects in sports video games, controls in boxing games have been similar among all of the various titles in the genre. Punches are triggered by pressing buttons corresponding with the type of punch (jab, hook, uppercut, etc.), and whether it was thrown with the left or right arm. However, Electronic Arts'

Fight Night 2004 introduced a new scheme—shown in Figure 2.2.13—that uses the right analog joystick to give players better command of their boxers' fists. The result is an interesting device, and a fighting experience unmatched in other boxing games because of the *natural mapping* between control input and action.

FIGURE 2.2.13 *Fight Night 2004 analog control punching.*

When creating control designs, look to your hands frequently. Evaluate your ideas for control mappings by keeping your attention tuned to your hands rather than the game's display. *How* your fingers are moving is just as or more important than to *where* they are moving. In this way, controls reveal a lot about the nature of the model you are making; they can indicate where player attention is being spent.

Current games offer a wide range of actions for the player, and moderating this complexity is a challenge for designers. In a single game, they may need to interact with characters, open containers, push crates, search under beds, or unlock a door. Rarely, however, are these actions performed at the same time, so orthogonal mappings don't necessarily make the most sense.

Contextual controls allow games to map any number of actions to common (i.e., shared) controls. This is done by relating the player's current situation (position in the environment, relationship to nearby agents, etc.) to the actions available. Football games, for example, commonly map multiple functions to the same buttons. The game uses the current context of play—the possession of the ball—to determine which mapping is currently active. In games where the characters are in complex 3D environments, the difficulty of determining context becomes more complex. Even so, the "action" button has fast become a staple of many games.

HUDs and Front-Ends

In application software, the *front-end* is a façade, the visible portion of the product with which the user interacts, while the *back-end* is the portion that processes data. In games we typically use *front-end* to refer to GUI elements not set directly within the environment—in other words, not rendered as part of the scene. Front-ends may include menus, buttons, HUD, help text, and the like.

Some typical elements of a front-end:

- Title
- Main Menu

- New Game
- Load/Save Game
- Options
- Credits
- Quit Game
- Pause Menu
- Map
- Log/Journal
- Game Stats
- Exit to Main Menu

Where possible, strive for thematic consistency, using the style of decoration to support and enhance the game premise. The layout and design of your interface should support the experiences you are trying to create.

The *HUD* (Head-Up Display) overlays the camera display during play, most often to show important information that can't be visualized directly in the game. The HUD elements border the game view.

Some typical information displayable in HUD:

Progress/Scores: Kills, points, levels, etc.
Resource levels: Health, ammunition, money, etc.
Map: Geography, agent locations, etc.
Alerts: Warnings, help, information, etc.
Chat: Dialog from the system, players, team, etc.

Designing an effective HUD requires an artist's skill with visual communication. As Figure 2.2.14 will no doubt attest, a game designer does not a graphic designer make. The duty of a game designer is to understand what information needs to be presented, not necessarily *how* it is best presented.

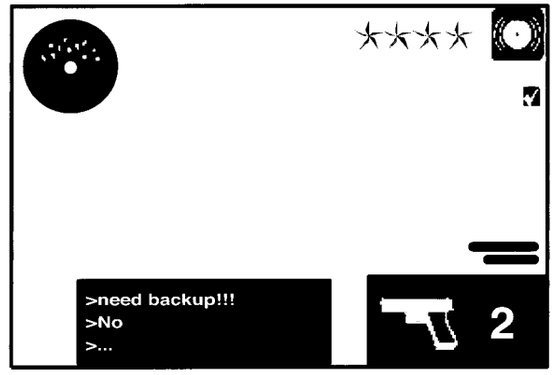

FIGURE 2.2.14 *A head-up display.*

Human-Computer Interaction and Cognitive Ergonomics

A relationship between two things is called a *mapping* [Norman88]. Consider how we navigate by using a (literal) map: looking at a two-dimensional bird's eye view of the world, we relate positions on the map to positions in the three-dimensional space around us. Similarly, we may map a light switch to a lamp, relating one position to "on" and the other to "off."

Human-Computer Interaction (HCI) is a study concerned with creating and enhancing communication between users and computers: it examines the ways in which people design, build, and use interfaces [Hollan99]. HCI is concerned with using current models of perception and cognition to create better interfaces, move beyond the current metaphors of icon and desktop, and develop better support for cooperative work. *Cognitive ergonomics* is a discipline that analyzes the cognitive representations and processes involved with performing tasks to better understand how we interact with systems to achieve goals [Cara99]. Many of the goals and methods of HCI and cognitive ergonomics overlap, as do many of the practitioners.

In the *Design of Everyday Things* (formerly *The Psychology of Everyday Things*), Donald Norman specifies five principles of design [Norman88]:

Visibility: Making the relevant parts visible
Mappings: Understandable relationships between controls and actions
Affordances: The perceived use of an object
Constraints: Prevent the players from acting in ways they shouldn't
Feedback: Reporting to the user what has been done and accomplished

As a set, these principles offer us clues to designing better interfaces, and better games. We can design challenges that are less focused on feats of dexterity or perplexing logic puzzles and more focused on managing the flow of information to the player.

Significant controls should not be obscure, and their functions in hardware should map, as naturally as possible, the functions in-game. The function of our game objects should be intelligible—their use and behavior knowable. Player actions need to be constrained, guided by the game and the interface so that there is no confusion *how* players are able to play the game.

Even as players signal their intent through buttons, keys, mice, and sticks, they need feedback from the interface—reassurance that they have been heard. In many styles, feedback can be managed solely within the mechanics—a button is pressed and a character jumps, for example. If the state of the game will change noticeably and immediately, additional cues are not needed. However, if player actions do not produce outcomes until a measure of time has passed (such as in strategy games), it's good to provide feedback with sound and visual effects.

Players are often ignorant of much of the internal structure of the games they play. In nonelectronic games, players must arbitrate the rules themselves; they develop a direct familiarity with the systems at work. However, when the system is maintained by a computer, the players are no longer required to be even familiar with the formal organization. They know what they can glean through their experience through their relationship to the play mechanics and interface. This puts the onus on us to make our systems clear. By applying Norman's principles to your design, you can increase the player's comprehension of the system without having to provide lengthy tutorials, training sessions, or frustrating trial-and error-sessions to do so.

Systems

Game systems have one purpose: to enable play mechanics. At the end of the day, a player will be unimpressed by beauty and sophistication that isn't experienced. If creating better games were simply a matter of adding more technology or more complexity, design would be a much simpler practice.

A *system* is a set of components structured in such a way that their properties and relationships to each other form a whole [Salen04]. It is a set of pieces that work together. In our daily experience, it is impossible to find anything that is not both a system and a part of a system itself.

Consider a cellular phone, one small piece of a cellular network. In a nutshell, it is a tiny, low-power radio transmitter/receiver, operating among base stations and switching offices. This system connects one caller to another. A closer view of that phone reveals that it is, itself, a system: a circuit board, microphone, speaker, display, and antenna. Further down, each of those parts are systems of yet smaller pieces, and so on and so on.

Conceptually, we seem to prefer dealing with wholes rather than parts, viewing each system as a coherent body. In other words, we think and talk about each system, as well as each of its pieces, as a *thing*. We know a sock is a system of threads, but do we think of this when trying to find the lost mate?

Systems depend on the particular arrangement of their objects, their *architecture*, for their function. Relationships between their pieces determine how they work to produce results. If the architecture of the system changes, so does the operation; it needs every one of its elements for it to work properly. This doesn't mean that a change must appear significant for it to have an effect: sometimes, the smallest adjustment leads to metamorphosis. Generally speaking, however, if objects of a system can be added, removed, or otherwise altered without affecting the relationships of the other objects or the behavior of the whole, then you are working with a *collection* rather than a system [Fullerton04].

Familiarity with modeling an object-oriented *modeling language*—a collection of standardized symbols and techniques for designing software architecture—(e.g., UML) can be helpful when designing systems. Conceptualizing and describing game objects becomes easier and more productive with training.

Objects and Relations

The pieces of a system are *objects*. These can represent just about anything: items, agents, places, and so forth. Just as in the real world, objects can be described by their *attributes*—the properties that determine what they are—and *behaviors*—the things that they do. Additionally, *relationships* are used to describe the ways in which their behavior and influence will change as contexts within the system change.

Once you understand the *requirements*—the things your system must accomplish—list the objects that will make up the system, including everything that seems important. Don't worry about the details when beginning: you will have plenty of time to review and revise the particulars.

Imagine we are working on a game where children run their own businesses, *Child Tycoon*. They can sell lemonade, popsicles, lawn mowing, or whatever we feel is appropriate. We might begin with a list of objects: characters, businesses, goods, services, and locations.

Each of these top-level objects is a new *class*—a basic type—of object. Once the objects are known, you begin to define what they are and what they do. Of course, the moment you begin deciding what objects will be in the system, you have already begun considering their attributes and behaviors. This is one of the reasons why working with objects is so useful: we already know how to think about them!

Each of our businesses will be operated by characters that sell either goods or services to other characters. Each business is operated at a location that is capable of supporting it. These are relationships between the game objects that will solidify as we describe the rules defining the object behaviors.

When creating a list of behaviors, you are asking, "What can this object do?" Each behavior is therefore an action a character might do in the game. Potential character behaviors might include buys goods, sells goods, performs service, hires characters, and befriends characters. For example, characters can buy items and services or they can hire other characters for help. These behaviors will help define the potential relationships between objects in the system.

Determining attributes is providing answers to more questions about your objects. What are the important qualities of an object? What distinguishes one from another? How do similar objects differ? How are they the same? These object properties will be used to determine which behaviors are appropriate and under what conditions.

At the highest level is an attribute that differentiates one kind of object from another, it is the very type of object at hand (e.g., characters, locations, etc.). From there we begin to describe properties in finer detail. For example, character attributes might include name, organization, ethic, charisma, and money.

Objects are distinguished further by the value of their attributes, which describe the state of the object. These determine which of its potential behaviors are relevant or available at any given moment and how behaviors involving these objects will produce results. Table 2.2.3 shows some different values for our childhood entrepreneurs.

Table 2.2.3 Example Values

Name	Chris Clifford	Jesse Lopez	Michel Christy
Organization	5	8	4
Ethic	10	3	4
Charisma	3	7	10
Money	0.00	4323.75	12278.00

Our object behaviors have to be given rules that define when and how the behavior is supposed to work. For example, it is clear that a character cannot buy goods or hire other characters if it has no money (such as poor Chris Clifford). These conditions will be defined along with the rule itself.

Attributes can also be used to determine the result of a behavior. For example, we might have the sale price of goods increased when a character has a high charisma. Alternatively, we might have a similar increase in pay when the character performing a service has a high ethic—the character puts more effort into the work.

The relationships between objects are manifest in their behaviors. In Figure 2.2.15, we see a diagram of two characters during a service exchange. One character hires a service that the other performs. We include, in the illustration, the attributes of each character that are relevant to the outcome of this behavior.

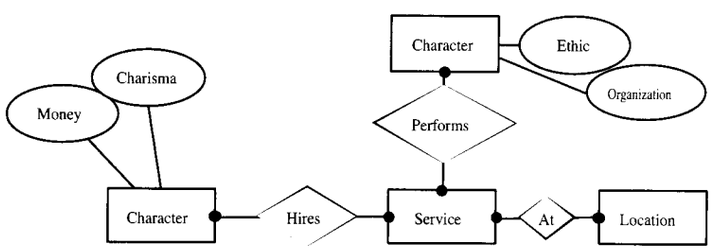

FIGURE 2.2.15 *An entity relationship diagram.*

System designs usually begin at a high level, starting with an intended play mechanic, and then get broken down into more specific details. With each pass, you refine your questions and answers. How does a character trade? What can be traded? When can trade of this object occur? Where? As you go, it is likely the complexity of your system will deepen. It is difficult to predict the behavior of all but the simplest systems. You need to periodically observe the behavior and make adjustments. In other words, you design, prototype, test, and tune. You *iterate*—perform the steps repeatedly—until the level of detail you need has been reached.

Emergence and Systemic Design

Every year, gaming audiences grow savvier. As they become versed in the conventions typical of a style, they demand more novelty as a result. Experienced players can quickly map boundaries of environment and expression, plumb the depth of the simulation, and expose limitations. Our customers grow ever hungrier for larger and larger possibility spaces [Wright03]. In the past, industry has responded by delivering products with more elaborate technology and more content. However, in attempting to add more of everything, the size and expense of the typical project have ballooned and are becoming more difficult to sustain. As a result, most small development studios are having difficulty continuing to operate successfully as independents.

However, many developers are discovering that, perhaps, "more" isn't the only approach. At the 2002 Game Developers' Conference, Harvey Smith delivered a popular lecture contrasting *special case* with *systemic* design. His distinction between *special case*—creating gameplay for a specific, localized occurrence in a game—and *systemic*—creating gameplay out of mixing existing game elements which consist of globally consistent behaviors and characteristics—contrasts two approaches to the demands of players.

Special case refers to game interactions that have been designed by anticipating game states and player actions [Smith02]. The designer visualizes each scene of play as he imagines a user will experience it. Unique events, encounters, and challenges are designed and created for each of these interactions and the emphasis is placed on a "micro" approach; the implementation addresses each special case directly and specifically. Events are mainly prescripted—the sequence and nature of the interaction fully determined beforehand. Player strategies are often concerned with discovering the intentions of the designer.

The *systemic* approach refers to approaches where the structure of the game system provides a set of consistent, universal behaviors. Rather than create many individual and unique problems and solutions throughout the game, the basic rules are generalized and structured into a system. The player experiences the game by working freely within the established rules and premise, as in *Grand Theft Auto* or *SimCity*. Key events and encounters may still be sequenced and scripted on a case-by-case basis, but the *core mechanics* work as part of a rule-based environment. Player strategies are most usually concerned with better understanding the behavior of the world and inventing novel solutions.

Systemic approaches to design often lead to *emergent complexity*—behaviors that cannot be simply predicted from the rules of a system [LeBlanc00]. Player immersion intensifies when the system behaves unexpectedly while remaining self-consistent—events of the game are plausible within its own rules.

In 1843, John Stuart Mill wrote *Systems of Logic* and distinguished "mechanical" from "chemical" modes of producing effects [McLaughlin99]. In the mechanical mode, two or more causes produce results that would be the same had each cause happened on its own rather than together. Imagine pushing a ball in two directions—

once straight-ahead and once to the side. The final position of the ball would be the same regardless of the order in which the pushes were applied. Mill suggested that in the chemical mode (so named because of the way in which chemical reactions seemed to produce results), the resulting product could not be achieved by the effects of each cause independently. For example, methane reacts with oxygen to produce carbon dioxide and water, a process that would not occur with each reactant on its own. Mill called the mechanical products "homopathic" and the effects of the chemical mode "heteropathic." Mill's work would spawn the tradition of British Emergentism. In 1873, George Henry Lewes coined the term "emergence" when he labeled effects that were heteropathic as *emergents* and homopathic as *resultants*.

Emergence is used today in many scientific disciplines to refer to unexpected, novel, and nontrivial phenomena that arise from the behavior of complex systems over time. Indeed, many games exhibit this type of emergence as well. However, some controversy has arisen regarding which are *really* examples of emergent phenomena and which are not. To avoid confusion, you may choose to use *rule combinations* to refer to a game's heteropathic effects, and reserve *emergence* for describing behaviors arising unexpectedly in complex multiagent systems (such as John Conway's *Game of Life*) [Juul02]. When all else fails, Lewes' definition seems broad enough to accommodate everyone.

Uncertainty and Probability

The mathematical study of *probability*, measuring the likelihood a given even will occur, is rooted in the study of games. Italian mathematicians had first solved some specific problems regarding games of chance, but no general theory of probability emerged until the seventeenth century [Apostol69]. A French nobleman consulted Blaise Pascal on a gambling question. In where a pair of dice was thrown 24 times, was it profitable to wager that 12 ("double 6") would be rolled at least once? The discussion that would follow on this and other related matters, between Pascal and mathematician Pierre de Fermat, would develop into the basis of probability theory.

When a system always responds in the same way to a set of conditions and inputs, that system is called *deterministic* [Ledin01]. Assuming the state and control variables remain the same, multiple runs will produce the same result again and again. However, if there is variation from one run to another, while the state and control variables have remained constant, that system is called *stochastic*—in other words, random.

When we discuss *uncertainty*, we are referring to an aesthetic—not knowing what's to come. This lack of foreknowledge, when regarding the outcomes of play, is vital to all games. Without uncertainty, player motivation flattens into a weak signal. When every decision is obvious, every outcome known, people lack the desire to play.

There are four main causes of uncertainty:

- Incomplete information
- Inaccurate information
- Linguistic imprecision
- Disagreement between sources of information

Any of these causes can increase the degree of uncertainty players experience (we support the argument that intentionally introducing linguistic imprecision is not usually fair). The most common method in computer games is to present the player with incomplete information. Some state or aspect of the system is not revealed explicitly to the player. In economics and in game theory, this is referred to as *imperfect information* and is in contrast with *perfect information*—when all players know all there is to know about the state of the game. Both types have existed for a long time: games such as chess and go are of perfect information, nearly all card games are of imperfect information, and so on.

Although uncertainty and probability are related, they are not synonymous. Uncertainty is a psychological experience, probability a mathematical likelihood. While we can make use of probability to generate player uncertainty, we must be careful. "Added randomness" can be easier for a player to detect than one might, at first, suspect, and players tend to resent suffering at the hands of fate rather than by their own actions.

Deterministic systems can still elicit a high degree of uncertainty. One approach is to make outcomes dependent on player performance as well as strategy, similar to sports. For example, first-person shooters are usually deterministic: the variability in player performance and strategy maintains uncertainty. Another method, the most common in video games, is to occlude portions of the system structure or function so that the players operate from a position of imperfect information. They must understand and strategize from knowledge at hand and allow the rest of their decision making to be based on estimates and deductions.

Stochastic systems, on the other hand, can be very difficult to implement and their influences should rarely reach the outcomes of major objectives. When used to introduce variety or in some other way "fuzz" a system to seem more natural (e.g., when propagating environments), stochastic systems can be very useful. Nevertheless, it is usually best to avoid letting pseudorandom fate seal the player's doom.

Dynamics

In game design, we discuss the behavior of systems over time as *dynamics*. Specifically, we are concerned with understanding and generalizing the ways relationships between components of a system can change [Wright03]. However, this has proven difficult to do beyond a case-by-case basis.

The player experiences the game system as a dynamic structure. Figure 2.2.16 shows a model, first used at the 2001 Game Developers' Conference, created by game designer Marc LeBlanc for understanding games [Hunicke04]. *Mechanics, Dynamics, Aesthetics* (MDA) ties the players' emotional responses to the body of a game. The players' view of the game is the result of their interaction with the system over time. (In MDA, *mechanics* refers to the game components, at the level of data representation and algorithms; it encompasses the *system* and *interface* of our model.)

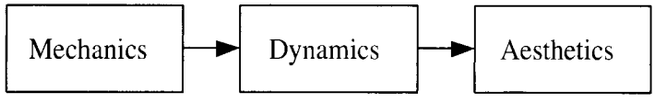

FIGURE 2.2.16 *The MDA model.*

Game dynamics result from continued interactions between the players and the game system. It is hard to determine general behaviors of dynamic systems—each is dependent on their particular structure of objects, properties, and the relationships between the member elements. You can appreciate the complexity of dynamics by reflecting on the problems of creating decision trees.

Feedback and Feedback Loops

Established in the late 1940s, *cybernetics* is a study of communication, control, and regulation in organisms and machines. An early paradigm for modeling systems, it designed and analyzed mechanisms that affect control. Some typical examples of cybernetic systems are governors, thermostats, and autopilots [Wright03].

Figure 2.2.17 shows the elementary cybernetic systems consisting of a *sensor* detecting a condition and reporting to a *comparator*, which evaluates the information, potentially triggering an *activator*, which makes a change in the system that alters the monitored condition [Salen04].

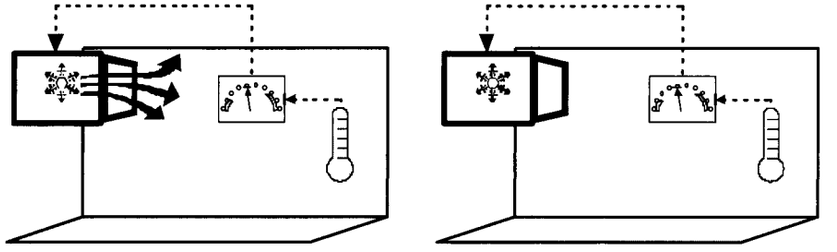

FIGURE 2.2.17 *A climate control system [LeBlanc04].*

When a portion of output from a system is returned back into the same system as an input, the returned portion is known as *feedback*. Literally, this output has been "fed back" into the system from which it came, resulting in further output being altered by the returned input.

The path that is taken by the returned portion is called a *feedback loop*. Feedback is a central concept in cybernetics, control theory, and many other systems disciplines. By connecting several of these simple systems together, more complex systems and behaviors are created.

As Jay Forrester illustrates in Figure 2.2.18, filling a glass with water is more complex than just the flow of water into a container [Forrester96]. The volume of water is being controlled by a feedback loop that spans from the level of water to eye to hand to faucet controlling the water flow.

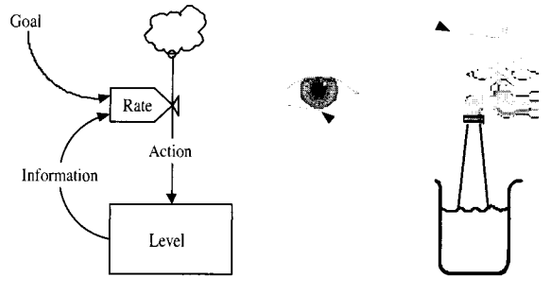

FIGURE 2.2.18 *Simple feedback loops [Forrester97].*

Feedback comes in two basic types:

Negative feedback: Leads to goal seeking
Positive feedback: Leads to runaway behavior—explosion or standstill

Although the name can be a bit misleading, *positive feedback* can be rather difficult to make use of. It either increases an already increasing rate of an accumulation (as in Figure 2.2.19), or it decreases an already decreasing rate. In Figure 2.2.17, a positive feedback on the left would turn the room into an icebox.

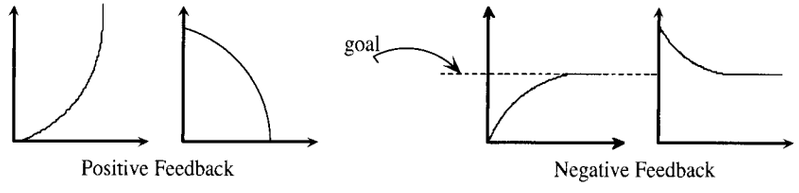

FIGURE 2.2.19 *Curves representing basic feedback types.*

The best-known kind of positive feedback is audio feedback, the Larson effect. A microphone is positioned too near a speaker connected to the microphone's amplifier. The system rapidly overloads itself and a piercing screech threatens all eardrums within range.

Negative feedback can be described as a *goal-seeking* behavior: as it nears a target level, its rate is restricted more and more (Figure 2.2.19). In Figure 2.2.18, a negative feedback loop from water level to eye to hand turns the spigot off as the water reaches the "full" mark. In Figure 2.2.17, a negative feedback loop turns the climate control systems off when the ideal temperature is reached.

LeBlanc has generalized some of the feedback behaviors as they relate to games [LeBlanc99]:

- Negative feedback stabilizes the game.
- Positive feedback destabilizes the game.
- Negative feedback forgives the loser.
- Positive feedback rewards the winner.
- Negative feedback can prolong the game.
- Positive feedback can end the game.
- Negative feedback magnifies late successes.
- Positive feedback magnifies early successes.

System Dynamics—A Modeling Tool

As we've mentioned, it is very difficult to make assumptions about the behavior of a system over time. A variety of tools allows us to simulate system structure and behavior—useful if we will need to be testing and changing this structure frequently. One such tool was started in 1956 at MIT. *System dynamics* is a modeling and simulation discipline created by Dr. Jay Forrester to study how systems change over time.

At first, system dynamics was a tool for policy analysis in organizations. Why would some decisions produce expected results while others catastrophic failure? Since then it has been a popular tool for observing system behavior in urban planning, biology, economics, and a wide variety of other fields. For game designers it provides an easy tool for quickly prototyping systems—object relationships and behaviors can be taken for a trial run.

System dynamics describes everything in two aspects: *stocks* (or *levels*) and *flows* (or *rates*):

Stock: A level; an accumulation of something
Flow: A change in the quantity of an accumulation

Figure 2.2.20 shows a basic system-dynamics model. The clouds on either end symbolize the *boundaries* of the model—factors beyond the current scope. In this model, we are looking at a population of peasants.

We begin with 10 peasants and, each month, we will add to this a number of new peasants equal to the value of "births"; currently 10. Therefore, each month, 10 new peasants are born. Likewise, a number equal to "deaths" will expire each month; 2. When we run the simulation over a period of several months, we get a graphed result like that on the left of Figure 2.2.21.

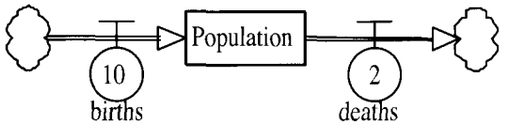

FIGURE 2.2.20 *A simple system-dynamics model of a peasant population.*

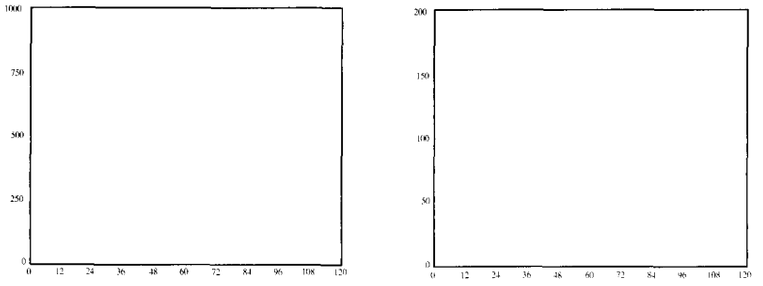

FIGURE 2.2.21 *Two population graphs.*

Every year 10 peasants are added to the town's population and 2 are removed, for a net gain of 8 peasants. By the end of 10 years, there are 970 peasants living in the town. To turn this linear relationship into something more interesting, we change the birth and death numbers into rates that are fractions of the population.

Figure 2.2.22 shows how a diagram of our improved simple system might look. Now, rather than a fixed rate of births and deaths, we have a "birth rate" and a "mortality rate" that are based on the total peasant population—5 and 2.8 percent, respectively. Now, when the simulation is run, we see a population graph (Figure 2.2.21, right) that appears to rise more naturally. Our fictional peasants are prospering nicely.

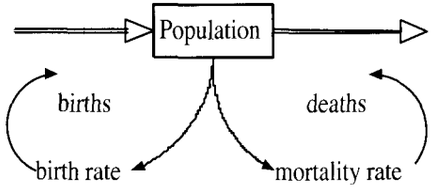

FIGURE 2.2.22 *Population feedback.*

Remember that a positive feedback loop produces change in the same direction, which may be either a rate of increase or of decrease. It is also known as a *reinforcing loop*. Negative feedback, on the other hand, produces change in the opposite direction. Negative feedback is goal oriented, in that it attempts to modify a condition to some target level (indicated in Figure 2.2.19 by the horizontal line).

Other types of behaviors result from a compound of positive and negative feedback types—oscillations and S-shaped curves. When a delay is introduced to a negative feedback loop, oscillations are produced as in Figure 2.2.23. The system responds to a condition, but, because the results of that response aren't immediate, the condition passes the goal in the other direction and the system has to compensate. Typically, the oscillations will diminish as the policy directs the system closer and closer to its goal.

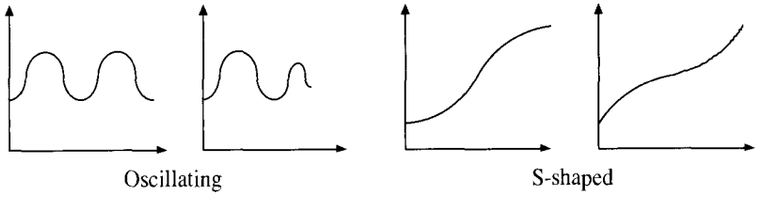

FIGURE 2.2.23 *Compound feedback curves.*

You've probably had, at some time or another, the experience of filling a bathtub where the water is, at first, too cold and then, after adjusting the spigot to compensate, too hot. You may have had to trade too hot and too cold back and forth a few times until the temperature was comfortable enough to step into the tub. The delay in that system would have been the time it took for heat to be exchanged through the volume of water in the tub, at which point your testing (with a finger or a toe perhaps) would indicate when the goal temperature had been reached.

S-shaped behaviors (Figure 2.2.23, right) occur when a system changes from a positive to a negative feedback loop, or vice versa. Objects might use a given behavior until a certain condition is reached, at which point another behavior takes over.

Constraints

At first glance, a topic on *constraints*—the condition of being restricted within fixed bounds—doesn't promise to be very exciting. However, constraints are at the kernel of game design and most veteran designers would probably agree that working within them often leads to the best features [Hecker02]. In fact, it could be said that design is, itself, a process of dreaming within constraints. Most of this chapter has already

been addressing constraints implicit in our work; this topic will clean up a few we have not yet addressed.

When designing, think of constraints as *parameters* rather than as *limitations—measurable factors that define a system* versus *a restriction*. The semantic difference may seem slight, but constraints are easier to appreciate when seen as a help to understanding, rather than as an obstacle to progress.

Consider an analogy to motorcycle riding: When riding a motorcycle, it's good not to run into things. People, however, tend naturally to look at threats. They also tend to steer their body and vehicle in the direction they look. Because, to a motorcycle, any object larger than a poodle is a potential threat, these two behaviors lead to "undesirable" results; in a car, a fender-bender, but on a motorcycle an emergency room. New riders must be taught to look away from obstacles, finding paths that avoid rather than paths that collide into them. Game designers deal with constraints by creating solutions that move around and within, rather than against them.

This approach to constraints is similar across the various roles designers fill, from senior to associate, game to level. Each of these differ more by scale and context than by kind. Whether you are designing systems, interfaces, or placing interactive props in a scene, you are inventing solutions to problems within constraints. What does the player do within these constraints? How do the objects relate within these constraints? Where should I position these crates within these constraints?

Platforms

A computer game *platform* is a general description of hardware and software that collectively determine how the device will function, applications will operate, and how the user will interface with the appliance. It is no surprise that the capabilities of these devices are frontline concerns to designing games.

Typical platform categories include:

Personal computer (PC): A multipurpose system supporting applications ranging from spreadsheets and Web browsers to games (e.g., Windows PC).
Console: A dedicated gaming system designed to integrate with consumer audio/video appliances (e.g., PlayStation 2).
Handheld (console): Dedicated handheld gaming devices with integrated controller, display, and sound (e.g., Game Boy Advance).
Mobile device: Handheld multipurpose devices; cell phones, PDAs, etc.
Arcade: Proprietary coin-operated systems, for public use (e.g., *Asteroids!*).

From a production perspective, the PC is the easiest platform for which to develop games. Most of the tools used in production run on the same machine as the game being developed. The memory and storage capacities of the average desktops outstrip that of even the most powerful console systems currently available; dedicated gaming machines built by hardcore PC gamers push that technology to extremes.

Developing for console and handheld systems, on the other hand, can be technically challenging. First, proprietary development kits are required. These range from

special development environments, such as Sony's TOOL for PlayStation 2 development, to more generic "homebrew" solutions that require a fair amount of patience to accommodate.

Next, console performance and capacity limitations mean that designers must watch resources to a much larger extent than on a similar PC project. Limited system RAM leads to careful asset loading management, and limited memory card storage resulting in careful use of state variables and persistent data. Being judicious with space is not as unwelcome a constraint as it sounds; often, what is left out of the model is unimportant.

Mobile devices suffer similar technical constraints to handhelds, although they are typically easier to develop for. Distribution is often from the Internet so costs can be lower. Typically, the problem has been finding the consumers. However, over the past few years, the mobile market has taken hold, and millions are spending more time playing games on their phones than talking into them.

Arcades have been in steady decline since the end of the 1980s and beginning of the 1990s. There are any number of reasons why this may be so, the most significant being that the technology, once far superior to that available in home consoles, has been met and sometimes outmatched by consumer devices. The main advantage left to arcade machines is the integration of unique, specialized controllers such as the mock cockpits for racing games.

So why doesn't the personal computer dominate the video game market? Several advantages to consoles make them attractive to game developers. First, they are dedicated systems with controllers tuned to game playing; in contrast, keyboard-mouse interfaces are often unwieldy for many types of games. Consoles are known as *lean-back* devices—designed to work either in the hand or with entertainment systems that afford comfortable seating positions and relaxed environments—you play them while lounging in an easy chair or on the couch. In contrast, PCs are known as *sit-forward* devices where one sits upright at a desk—a position more usually associated with work than with play.

During development, testing a console title can be more focused and productive because there is one standard hardware configuration; in testing PC games, QA teams may have to test over 100 different configurations and *still* not cover more than a portion of those configurations possible.

Lastly, although PCs may have an *installed base*—the estimated number of units out in the world—of over 700 million [Shiffler04], only a small portion of those are capable of playing the latest games. The PS2, however, currently has an installed base of over 70 million units, *all* of which are capable of playing the latest games. As a result, the typical *hit* PC title might sell several hundred thousand copies, while hit console titles sell in the millions.

Game Saves

The ability to save progress is expected in nearly all digital games outside of the arcade, but raises many challenges for designers. The subject is somewhat controversial, as

there are many personal philosophies, biased by negative experiences with one system or another.

Save triggers automatically save the game state when the player performs a particular action such as reaching a location or overcoming a challenge. They can make it easier to tune the pace of the game; players cannot "game" the save system by frequent, incremental saving. Objections to this scheme often cite the lack of control this affords the players to moderate their own progress and play session length—sometimes, the real world does interrupt our gaming. Pitfall to avoid: the risk of saving players' progress at inopportune moments, leaving them vulnerable to events immediately following a reload.

Save-anywhere systems (a hallmark of PC games) allow players to maintain safety in the most challenging of situations—each encounter can be replayed with little effort and the outcomes will tend toward minimal losses and maximal gains [Imlach02b]. Balancing difficulty with these systems can be challenging, as even unintentional overuse of the system can make the most difficult game a cakewalk.

Save points allow users to save progress at any time they wish, while mandating set locations to save. Most consoles are known for this type of system in lieu of the more resource-intensive save-anywhere schemes. Save points allow the system to write a smaller set of state variables when storage space is at a premium.

Coded-text or *password* systems may be similar to triggered or save points systems. Rather than use hardware storage, an algorithm codes relevant state variables into a text string (a password) that the player must write down and remember. Because it must be relatively easy to record, the amount of data these systems encode is usually quite limited. These days, even the simplest systems can afford a few bytes of storage, so coded-text saves are becoming rare. They still find occasional use when it is desirable for players to share saves with each other.

Genres

The entertainment arts are classified first by media (film, literature, games, etc.) and then further into *genres*—categories describing generalities of conventions, style, and content. Genres are commonly criticized for loose definitions. Most games fit more than one at the same time, sometimes leading to designation as a *hybrid*—a blend of two or more genres—such as the ubiquitous *action-adventure*.

A few examples:

Action: Emphasize a fast pace, quick reactions, and motor coordination.
Adventure: Players engage in story-driven exploration and puzzle solving.
Arcade: A type of action game where mechanics vary little during play.
Casual: Gradual learning curves, short play sessions, and broad appeal.
Educational: Teaches real-world knowledge or skills to the player.
Fighting: Typically fast and short one-on-one bouts with elaborate controls.
First-person shooter (FPS): Emphasize quick reactions and precision targeting.
Platform: Players performing feats in an environment; jumping, swinging, etc.

Puzzle: Solve logical or geometric puzzles as the core mechanic.
Racing: Driving along a track, trying to achieve shorter (i.e., faster) times.
Rhythm: Emphasis on matching rhythms through the controls.
Role-playing (RPG): Adventure games focusing on character development.
Simulation: Games that emphasize model detail (i.e., perceived realism).
Sports: Emulate contests of physical sport.
Strategy: Core mechanics focus on strategy and managing resources.
Traditional: Digital renditions of nondigital games.

Audiences

During design, we frequently consult mental models of our *target audience*—the group of consumers expected to buy and play our game—to make judgments. We imagine their preferences, abilities, and habits. The accuracy of these intuitions can vary wildly; mappings between mental models and the real world are strongest when the population in question resembles our own. Unfortunately, it is difficult to make good decisions on the sole basis of intuition.

One tool to appreciate audiences objectively is *demographics*, the study of relevant economic and social statistics about a given population. The relevant factors are called *demographic variables* (e.g., age, gender, income, etc.) and are used to segment consumers into groups known as *markets* and smaller groups known as *market segments*. The attributes of a typical market member are known as a *demographic profile*.

For game designers, demographics provide a starting point to defining the audience, but typical demographic techniques (e.g., polling, surveys, etc.) do not always produce the information most interesting to us: understanding the play preferences of the segments. Knowing *what* a market segment likes is not as valuable to designers as understanding *why* it likes something.

For a long time, game genres themselves have been used to classify player preferences. For example, the preference for detailed realism is often attributed to fans of the simulator genre (typified by *Microsoft Flight Simulator*). However, these models are informed by the product, not the players, and are really a kind of folk demographic—a guesstimate by nonspecialists.

Various efforts have been made to typify players' behavior and preference by their motivation to play. Most of these studies rely on personal experience and anecdote to form categories such as *explorers*, *collectors*, *competitors*, *jokers*, *storytellers*, and so forth [Fullerton04].

Similarly, the industry has long divided consumers into two broad groups: *hardcore* (core) and *casual*. While it is generally acknowledged that this distinction is no better than any other arbitrarily invented scheme, it follows that it is no worse.

Some typical attributes of the hardcore gamer include [Ip02]:

- Playing games over many long sessions
- Discussing games frequently and at length

- Being knowledgeable about the industry
- Having a higher threshold for frustration
- Desire to modify or extend games creatively
- Have the latest game systems
- Engage in competition with themselves, the game, and others

Casual gamers, it is implied, are everyone else. As you may imagine, this type of distinction's usefulness can be somewhat minimal when the largest potential audience is left essentially undefined.

Recently, there have been more rigorous attempts to understand player motivations and types. In particular, Nicole Lazzaro (XEODesign) has presented the *Four Keys*, a breakdown of what players like best about games [Lazzaro04]. Successful games contain elements of two or more of these keys simultaneously:

The Player: The Internal Experience Key
Hard Fun: The Challenge and Strategy Key
Easy Fun: The Immersion Key
Other Players: The Social Experience Key

Internal experience: Refers to the enjoyment from visceral activities—resulting sensations such as excitement or relief.
Hard fun: Encompasses challenging aesthetics such as strategy and problem solving.
Easy fun: Comes from intrigue and curiosity—stimulated by exploration and adventure and moved by wonder, awe, and mystery.
Social experience: Involves stimulating our social faculties. Experiences range from competition, teamwork, bonding, and recognition.

"What surprised us most was the dramatic contrast in emotional displays between one vs. several people playing together. Players in groups emote more frequently and with more intensity than those playing on their own. Group play adds new behaviors, rituals, and emotions that make games more exciting."—Nicole Lazzaro, *Why We Play Games* [Lazzaro04].

Iterating

One long-standing approach to software development is the *waterfall method*. In this paradigm, software design and production are broken down into distinct phases. First, the needs of the customer are assessed and documented as *requirements*. These are then analyzed and a plan for the software functionality is designed. After design, the product is developed and tested before finally being released.

There are strengths and weaknesses to any methodology, and waterfall development does put an emphasis on planning, which can be useful. However, it also works best when the needs of the customer (and therefore the form of the product) are reasonably well understood. Because games are dynamic, it is difficult to predict formal requirements with great precision. What is needed is a structured way to adjust the design of the product as it takes shape.

Iterative development refers to the practice of producing things incrementally, by refining and rerefining the product. This approach to building games is marked by use of *prototypes* and frequent *play-testing* in repeated cycles of designing, building, and tuning. Rather than attempting to design the game in complete detail, before production, designers produce a framework—a fundamental organization of features—around which the rest of the game can be built.

Prototyping

Prototypes are early models of the finished product, used to test ideas and techniques. They can be used during all phases of design; you don't have to wait until production is underway [Fullerton04]. All prototypes, whether built in software or from physical material, serve the same purpose: to present a rough, working model that can be evaluated and tuned. Sometimes they may approximate the whole of the game, but more often they are used as disposable test beds—an idea or feature is mocked up and taken for a test drive.

Physical prototypes are nonelectronic models made from paper, cards, chips, tokens, dice, and other common items. Use pens to draw figures, words, and numbers on playing surfaces while concentrating on the utility of the art, not its aesthetics; it needs to be effective while play-testing, not pretty. Make changes quickly as testing and tuning demand and do not let romantic attachments to features get in your way.

Software prototypes are implemented in code. These are the prototypes that, once in production, you are most likely to make use of regularly. They may be separate from the main body of the game's code, or they may be implemented within the main *branch* but in a discrete location where they are easy to remove. Often, prototypes are written in a scripting language (such as Lua or Python) where performance concerns are secondary to ease of development.

Testing

Software testing, also known as *quality assurance* (QA), is the process of verifying the performance and reliability of a product. Expected behaviors are verified by a *tester*, a person trained in the methods of evaluating quality in software products. When a *bug*—a discrepancy between what should and what is happening—is found, a *problem report* or *bug report* (also called a *bug*) is written. The bug describes the current behavior and the steps to be followed to reproduce it.

Play-testing is akin to software testing, but focuses on detecting problems in the play mechanics. *Play-testers* are individuals who play the game, in either directed play or freely, and report feedback on the experience. Designers will often observe play-testers, taking notes and occasionally asking questions of them [Federoff03].

Some considerations:

- Can the players use the controls? Do they understand them?
- Is the GUI clear? Menus navigable?
- Can the levels be completed? With how much effort?
- Can the needed skills be learned? In how long?
- Are they entertained? In what ways?

A *focus test* is a particular kind of testing session, when a group of play-testers (a *focus group*) representing the target audience is brought to a single location to play-test. These sessions are good for getting lots of feedback at once, but the social dynamics among testers in a group often add noise to the data that compromises its usefulness. On the subject of user feedback, Nathaniel Borenstein recommends listening to the concerns of the users but ignoring what they say in favor of determining what they really mean [Borenstein94].

It is an easy thing, during the course of creating a game, for developers to become blinded to the actual experience of play. Some aspects of the game may have undergone several revisions and, with each change, the risk that the player experience is overlooked is increased. In *Game Design Perspectives*, Sim Dietrich offers the following list of warning signs for faulty game design [Dietrich02]:

- New players can't play the game without assistance.
- New players don't enjoy the game without assistance.
- Excessive saving and loading.
- Unpopular characters.
- The all-offense syndrome.
- Players frequently reconfigure controls.

Tuning

Problems found in the play mechanics during testing must be solved by the designers. *Tuning* refers to the process of developing solutions by adjusting the properties and behavior of systems and interface.

Balance is a property of relationships, usually characterized by equilibrium. In classical aesthetics, it is a harmonious relationship of elements—like the distribution of shape and form in a painting or the movement of dancers on a stage. In games, balance is used in a few ways, all indicating a similar relationship to the preceding definitions.

We might seek to balance:

Player relationships: A lack of bias toward one player versus another.
Mechanics: Keeping the player in the flow channel.
System: Game objects are balanced amongst each other.

Rock-Paper-Scissors (RPS) is the common name of a balancing approach that uses *intransitive relationships*—three or more elements offer weakness and strengths relative to each other as a whole, so that balance is reached [Rollings00]. Figure 2.2.24

shows the relationship in RPS and its counterpart between light infantry (archers), heavy infantry, and cavalry. Table 2.2.4 shows a payout matrix.

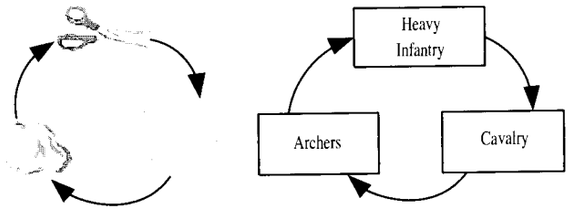

FIGURE 2.2.24 *Two simple intransitive relationships.*

Table 2.2.4 Rock-Paper-Scissors Payoffs

	Rock	Paper	Scissors
Rock	0	−1	1
Paper	1	0	−1
Scissors	−1	1	0

Intransitive relationships are common in multiplayer games because they allow each group of players to feel powerful in a given set of circumstances. They can also provide incentive for players to form groups; one character's abilities are not enough for it to perform a task on its own.

Creativity

Creativity is the ability or power to create. In psychology it is specifically understood as the ability to produce an idea, action, or object that is considered new and valuable within the context of the culture at hand [Csikszentmihalyi99]. To the work of a designer, however, the novelty and acceptance of a creative idea are secondary to its effectiveness. In other words, it's more important that the solution *work* for the game and not as important that it be widely recognized as a creative approach. Think of the creative approach as a mode of decision making.

Most people believe creativity to be a talent or an aptitude. People inherit creativity through the luck of the gene pool, and you either have it or you don't. If you're of the unlucky majority, so the wisdom goes, you may as well pack it in. The truth is that anyone can learn to be creative.

A Classic Approach

In 1926, Graham Wallas proposed a general form for creative thought. It described the creative process in four distinct stages, and later work on the subject of creativity

commonly looks to this early framework for guidance [Wallas26]. Later versions of this model would integrate a fifth stage, elaboration, to appear as follows:

Preparation: The background of research and comprehension of a subject, preparation is an intentional effort to become immersed in a symbolic system or domain. You read, study, and consider every lesson on the subject you encounter. Known and common solutions are reviewed or deconstructed in a process typical of reverse engineering.
Incubation: "Mulling things over," is the thinking and reflection applied to an idea. This work may or may not occur consciously and can continue during unrelated activities [Campbell85]. During this time, ideas are subjected to broad censorship and discrimination, most of which occurs either too subtly or too rapidly to be noticed.
Insight: Whole answers that are resistant to unconscious censorship become revealed to awareness in a moment of sudden illumination. These revelations are the "Eureka!" or "Aha!" experience, hallmarks of the creative process. This is the "one percent inspiration" of Edison's famous quote: "Genius is one percent inspiration and ninety-nine percent perspiration."
Evaluation: Validating the revealed insight. Strive for balance in the evaluation of your ideas, as there are equal tendencies to be overly critical or not critical enough. Solutions should be discarded if a lack of significant novelty is revealed.
Elaboration: Although Wallas did not include this phase in his first description of the creative process, it is a common addendum. Elaboration is the transformation from concept to object; transforming the idea into substance.

These phases are recursive, to be repeated in part or in full as many times as necessary. Failure at any given stage often returns creative thinkers back to the preparation phase where they incorporate what has been learned by the failure into their assessments.

Brainstorming Techniques

By far, the most popular and common approach to stimulating creative thoughts is *brainstorming*. Nearly all brainstorming involves forcing ideas to be driven through the incubation stage to insight by eliminating discrimination during early stages. Instead, thinkers are encouraged to evaluate only after a wide catalog of ideas have been noted.

This technique works both for groups and alone; the practice remaining essentially the same.

Start with a board or piece of paper with a clear view of the goals and constraints that define the problem. Write a single sentence summary of the brainstorm's purpose. Follow this with cycles of free association ("saying whatever comes into your head") and elaboration on these ideas, writing down key words and statements along the way. Once the participants are satisfied with the volume of ideas or overcome with exhaustion, the processes of normal review and selection can begin.

Critics point out that brainstorming is often unproductive because the process is not directed. Participants are encouraged *not* to think critically and time is wasted elaborating on plainly bad ideas.

Six Thinking Hats

Edward de Bono created a mnemonic thinking tool for assuming different perspectives when engaged in the decision process [deBono85]. These modes of thought are described as colored hats that one "puts on" in order to consider other viewpoints that may not be natural to his or her default outlook. When a given hat is on, the thinker role-plays from the perspective represented by the color:

White Hat: Neutral and objective, wearing this hat involves analyzing known facts and detecting gaps to fill in with information. The emphasis is on assessing the decision.
Red Hat: Intuition, gut reaction, and emotion are all qualities of the red hat. Use your feelings and anticipate those of your audience. Allow views to be presented without justification or explanation.
Black Hat: Dark and gloomy, the naysayer's hat is worn when judging and criticizing ideas. Identify all the bad points of a proposed decision cautiously and defensively and actively play the part of devil's advocate.
Yellow Hat: Pollyannaish attitude typifies this hat. Optimistic logic is applied, looking for benefits and profitable outcomes that could result from an idea.
Green Hat: Symbolizing vegetation, growth and creative possibilities are explored. New ideas or modifications to earlier suggestions are offered with an emphasis on novelty.
Blue Hat: The cool mediating influence of organization is symbolized by blue sky. Wearing this hat, you maintain a process and control-oriented perspective, organizing and reviewing the work of the other hats.

Inspiration

Countless new ideas are littering the world at large. To find them you have only to maintain a mind receptive to playfulness and the structure of games. For game designers, play is the thing. Look for opportunities to play at all moments. Look at elements of life around you and reconfigure them into amusements. Play with your friends. If you lack friends, learn how to have them; a number does not matter—one is enough—but the ability to relate to people is coupled with anticipating their feelings.

Other media types are another endless source of inspiration. Don't just consume passively, but take joy in analyzing them: deconstruct their signs and techniques.

A few sources of inspiration:

Board games: Spatial relationships
Card games: Resource management strategies
Paper and dice role-playing games: Dynamic narratives

Books: Fantasy and agency
Film: Continuity techniques
Television: Serializing stories
Music: Rhythm and temporal relationships
Martial arts: Disciplined competitive sport
Children: Endless invention and capacity for play

Communication

One of the most important roles played by game designers is that of communicators; many describe communication as their primary duty. As part of a team, they must liaise for many groups within the organization, helping build and maintain a shared vision for the game. Often this vision is formed in documentation that may include written specification, asset lists, and diagrams.

Documentation

The subject of documentation is controversial. Each developer will have a set of opinions on the proper form and function documents should have. Each organization will have preferred ways to create, manage, and share documentation among its members; a "treatment" here is a "high-concept document" there. Furthermore, each project will have its own specific needs and challenges.

Remember that documentation is a written, descriptive model of the game you are to build. The depth of detail needed will vary in accordance with the depth and complexity of the feature sets you will be creating as well as the culture of the organization. In general, you should not concern yourself with page counts, but with content, clarity, and usefulness to the situation.

A *treatment* (also *high-concept document* (HCD) or *general overview document* (GOD) or *proposal*) is a brief, general description of the game and the fundamental concepts. François Laramée writes: "A treatment is a document containing the smallest amount of data that can allow a reader to make a reasoned decision on whether he wants to be involved with a project," [Laramée02].

Organizational needs will vary, but a treatment may include:

Concept statement: Two- or three-sentence summation of the game
Goals and objectives: List of the overall purpose of the game
Core mechanics and systems: Overview of all *significant* features
Competitive analysis: Situation of other games and the profile of the market
Licensing and IP information: Intellectual property ownership/requirements
Target platform and audience: Who the customers will be, how they will play it
Scope: Overall budget and timeline estimates
Key features: A "back-of-the-box" list of prominent features

Depending on the particular organization, other documents may include a preliminary design document (PDD) or initial design document (IDD), a revised design document (RDD) or general design document (GDD), an expanded design document (EDD) or technical design document (TDD), and a final design document (FDD). Each of these works within a similar paradigm: the documentation begins at a general scope, and then describes the product in finer and finer detail.

One approach that is growing in popularity is the use of intranet *Wikis*—Web site systems that allow users to contribute content (including new pages, links, and formatting) from within standard Web browsers. Often freely available, they have many of the attributes of commercial documentation management systems while being easier to edit from any machine on the local network because no special client installs are required.

Be careful of letting documentation take on a life and a game of its own. In particular, do not mistake the volume or complexity of the documentation as an indication of the thoroughness of the design. Instead, measure how the documentation is being used as a reference and how relevant its material during production. If it is largely unused or inaccurate, the time spent writing might have been better spent prototyping.

Some developers have proven that it is possible to produce good games consistently using lightweight documentation methods. Notably, Mark Cerny (of Cerny Games) is known for his *antidocumentation* stance as manifest in his process dubbed "The Method" [Cerny02]. Cerny's *macro design* is a short document that forms a roadmap for production. For example, a typical five-page macro design may include:

- Character and move sets
- Any exotic mechanics described
- Description of level structure size and count
- Level contents
- Overall structure—level, progression, etc.
- A "macro chart"—showing dependencies and distribution of mechanics

No other general document is created. Instead, the *micro design* that follows is the day-to-day work of the designers. Using lessons being continually learned through production (including prototypes), the level maps, enemy descriptions, mini games, and other elements are designed "on the fly."

It bears mentioning that many *stakeholders*—directly responsible for or otherwise directly interested in a work—may be reluctant to entrust such a degree of responsibility upon someone lacking Mr. Cerny's 20+ years of proven experience.

Flowcharts

Flowcharting is a very typical technique for diagramming steps in a process. Most developers will be familiar with flowcharting to some degree. This wide familiarity is one of their largest benefits. Once the basic conventions are understood, they can be "read" without needing to understand technical details. Figure 2.2.25 shows the four most common steps in a flowchart.

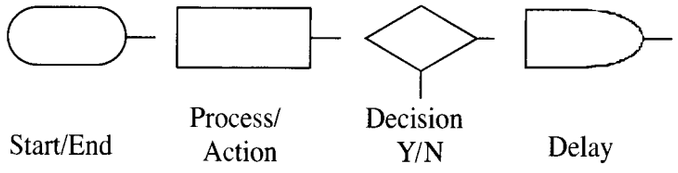

FIGURE 2.2.25 *Common steps in a flowchart.*

Flowcharts are commonly used to describe a series of events ranging from player actions to system behaviors. Because flowcharts are put together step by step, they are helpful for working through sequencing problems. Figure 2.2.26 shows the start of an adventure in a multiplayer game, where party members split up to gather resources and then regroup before heading off to the main encounter.

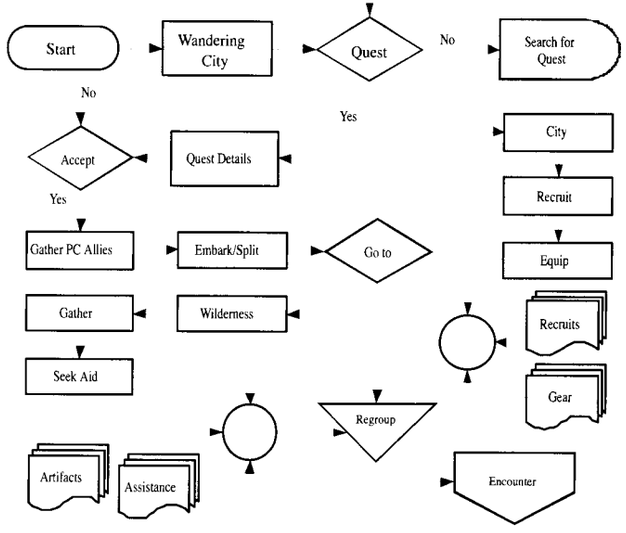

FIGURE 2.2.26 *Flowchart of a game sequence.*

Flowcharts have a downside: as detail and complexity increase, the diagram can become complicated to read. Because of this, flowcharts are most often used to illustrate constrained examples, not detailed accounts of an entire game.

Associative Diagrams

An *associative diagram*, such as the Mind Map in Figure 2.2.27 [Buzan96], is a drawing that helps us manage and organize complex networks of information visually. Elements are linked together illustrating their relationships and context. There is a variety of techniques for creating associative diagrams, all of which can be useful both in creative exercises and in organizing. As documentation tools, they are most useful when those who will be using the diagram are involved in making it.

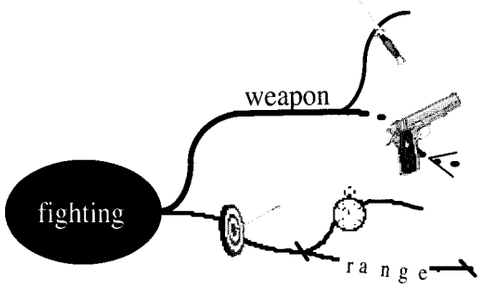

FIGURE 2.2.27 *A Mind Map.*

A central word (or image) begins each diagram, and around this, arms are drawn with key words representing related ideas written along their length. From each of these, more associations are drawn, and the process continues iteratively.

Psychology

Because so much of games and play occurs at the mental level, it is natural that game designers need a basic appreciation for the systems that produce and support the activities inside peoples' minds. For most of us, this is tricky business because we begin with an amazing amount of bad information.

For example, have you heard that people use less than 10 percent of their brain at any particular moment? However, natural selection isn't known for favoring large, complex organs that don't work but 10 percent of the time; especially not when over 50 percent of all genes in our bodies are believed to be in the brain [LeDoux02]. The brain, during periods of relative inactivity, isn't sitting idly by, waiting for something interesting to come along. It is actively processing stimuli, regulating bodily functions (including its own), and producing all kinds of cognitive activity—only a small portion of which is conscious.

So much is happening inside our heads, it's difficult to choose a reasonable starting point to discuss. Fortunately, as designers we need only a basic, approximate model that will be more or less relevant to building opportunities for play.

Working Memory and Attention

Working memory, or short-term memory, is one of our most important cognitive systems. It allows us to keep a limited amount of information, roughly 7 ± 2 items at any

one time [Zimbardo92], for a few seconds, while other portions of the brain perform computations on it. When a new task is begun, the old information is bumped out to make room [LeDoux02], and if we aren't done with the first, too bad.

Attention (also known as *selectivity* or *selective attention*) is the method in which our perceptions of certain stimuli are enhanced relative to other stimuli in the same environment when granted lesser immediate priority [Duncan99]. In other words, it's how we turn our focus on things that seem to matter and how we prioritize some goals over others.

Some of the most important studies of selectivity were conducted in the 1950s and involved people listening to two simultaneous messages. These studies formed the basis of much subsequent research.

These studies produced several findings:

Limited capacity: Identifying both messages at once is difficult.
Conditions for selectivity: One message can be identified and the other ignored if the messages differed in physical property (pitch, location, etc.).
Consequences of selection: Listening to one message while ignoring the other resulted in only the crudest recollection of the ignored message.

Any professional dealing with the abilities and capacities of others must respect both of these precious capacities; don't squander or abuse them. As a designer, you must balance the decisions and choices you ask of your players at any given moment so as not to frustrate them. This includes overwhelming them with information or requiring that their attention be spread over too many areas at the same time.

Conditioning

Arguably, Behaviorism's most significant contributions to psychology are its insights into the type of learning known as *conditioning*. The best known of these is *classical conditioning*. In classical conditioning, one stimulus that does not elicit a particular response naturally is paired with another that does until the subject learns to respond to both in the same manner.

Three states are shown in Figure 2.2.28: before conditioning, during conditioning, and after conditioning. Before conditioning, a tone (sound) heard by the dog produces no salivation, but meat (the *unconditioned stimulus* or UCS) in the dog's mouth does (the *unconditioned response* or UCR). During conditioning, the tone (the *conditioned stimulus* or CS) is played while meat is put into the dog's mouth, causing salivation as normal. After conditioning, the dog needs only to hear the tone for salivation (the *conditioned response* or CR) to begin [Zimbardo92].

Operant conditioning (or *instrumental conditioning*) describes learning where an action is encouraged or discouraged by the consequences of the action. The *operant* is a response behavior (e.g., smashing a crate) that produces observable effects in the environment. A *reinforcement contingency* is a consistent relationship between the operant and a change in the environment that results (e.g., get health). The resulting

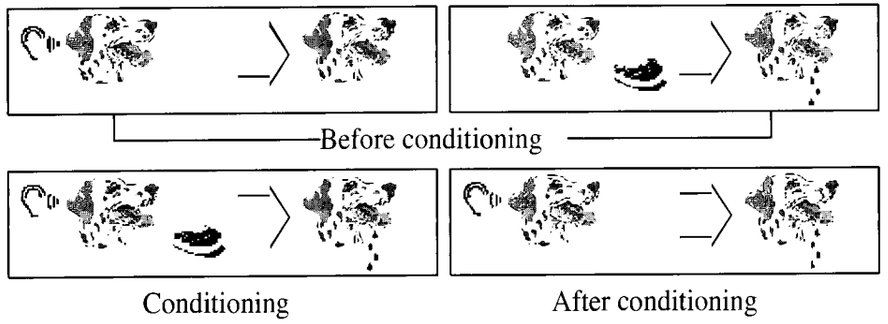

FIGURE 2.2.28 *Classical conditioning.*

changes in the environment are *reinforcers*—they increase the probability the operant behavior will be repeated, in the future and in similar circumstances.

There are two types of reinforcers—*positive reinforcers* and *negative reinforcers*—both of which increase the probability of the behavior. Positive reinforcers (e.g., more health) present a positive stimulus, and negative reinforcers present a negative stimulus (e.g., death). In Figure 2.2.29, the positive reinforcer, staying dry, is contingent upon the use of an umbrella in the rain. Getting wet negatively reinforces the use of an umbrella. In both cases, the user is being trained to use an umbrella.

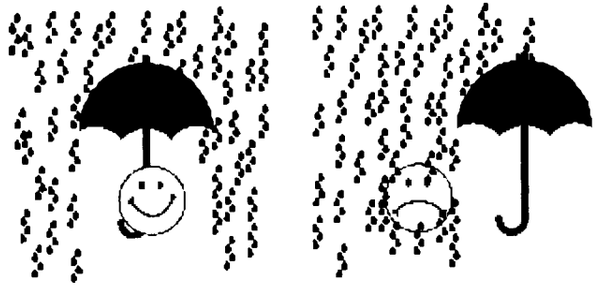

FIGURE 2.2.29 *Positive and negative reinforcers.*

The process of *operant extinction* withholds positive reinforcement so that an existing operant may be eliminated. Operant extinction is most successful when, in addition to withholding reinforcers, another operant is reinforced by producing the player's desired result. For example, if we wish to train the player to stop smashing crates to find health, we would first stop placing health kits in crates. Then we would use another device for the player to interact with that would result in an increase in health (medics, robot nurses, etc.).

Punishers decrease the probability of an operant when made contingent [Zimbardo92]. *Punishment* is the application of the punisher. The classic example of punishment is being burned by touching a hot stove, the resulting pain being the punisher.

It is easy to get confused by negative reinforcement and punishment. They are related, but work differently:

Punisher: Reduces the likelihood of a behavior
Negative reinforcer: Increases the likelihood of a behavior

Be judicious regarding the use of punishment to eliminating undesired behaviors. Psychologist Philip Zimbardo suggests reinforcing alternative behaviors rather than punishing the undesired. Punishment, as an aesthetic, is not one that will appeal to most players. The punished response will be suppressed grudgingly and a negative attitude toward the game can result.

Keep conditioning in mind when considering player *training*—introducing the player to new mechanics. In the past it was common to instruct the players using tutorials or user manuals, but these methods tend to get set aside; even when the players know they *should* go through the process of learning how to play the game properly, they would rather spend their time actually engaged in the game.

More recently, games have been dispensing with tutorials in favor of training the player in-game. Objectives can be structured to present the player with one or two new features, allowing the player to become acquainted with them before moving on. By tying the outcomes to the actual game progression, players are more easily motivated to go through the training scenarios and less likely to feel as though time is being wasted.

Summary

This chapter surveyed a broad spectrum of vocabulary, tools, methods, and practicalities of game design. We began by describing games as artifacts of play. Much as with other forms of entertainment, we interact with these to experience feelings within a safe frame, removed from repercussions in the real world. Viewing games like this, we began to explore their different aspects through a feature-based lens.

We examined play mechanics as the models experienced by the players: premise, choices, outcomes, goals, strategies, and other interactions with the game. These actions are mediated by the interface, which we discussed in terms of input and output systems, governed by principles being explored by human-computer interaction and cognitive ergonomic practitioners.

Our examination of game artifacts proper ended with an introduction to game systems. We touched on systemic design and emergence and introduced a few basic examples of modeling systems with objects. We briefly discussed positive and negative feedback systems as well as probability and uncertainty and their relationship to gaming experiences.

We touched on some common constraints that designers must work with on a day-to-day basis (especially those of platforms, genres, and audiences) and followed this with a quick introduction to iterative development—honing through repeated testing, prototyping, and balancing.

To help direct our creative efforts, we reviewed a few views of creativity itself. This was followed by discussing methods by which communication is propagated in a development organization using documents, flowcharts, and associative diagrams.

Our final topic scratched the surface of some issues in psychology that are immediately relevant to designing games, including working memory, attention, and conditioning.

As the game development industry matures, so too do the approaches and techniques of design. Traditional methods for creating media artifacts, which first inspired a broadening of the game design field, now seem to be reaching the end of their usefulness. Game designers are beginning to explore new forms of interactive expression that do not fit well within the earlier paradigms of art. Our art requires that designers develop not only aesthetic sensibilities, but also a technical capacity for defining and tuning systems. Our talents lie in the ability to model activities, both imagined and real, in dynamic simulations that are understandable through the lens of screen and game pad.

Exercises

1. Select a game with which you are familiar. Create a list of objectives in the game. Next to each, describe how a player might structure goals to reach that objective.
2. Find a volunteer to observe at play. For 20 minutes, create an exhaustive list of mechanics down to primary elements. For 10 minutes, ask the player to describe immediate goals; correlate these self-reports with mechanics you have observed.
3. Describe six qualities of choice. Provide examples of each from a released game. Do the same for all of Norman's principles of design.
4. Plan a level in a horror-themed game. Provide at least one diagram of a successful player experience, beginning to end, and one diagram of a failure.
5. Design a small game system of no fewer than five object classes. Detail each type of object, its attributes, behaviors, and relationships.
6. Plan a save game feature for *Pac-Man*. Choose between save-anywhere and triggered schemes. Explain how it functions and what state variables will need to be stored.
7. Build a physical prototype of a game; it may be an original design or a translation of a video game of your choice [LeBlanc04].
8. Create a treatment for a video game version of traditional marbles.

9. Design a "catch-up" feature for a racing game; explain how feedback would be used.
10. Describe the control scheme for a barbershop game where players cut hair.

References

[Apostol69] Apostol, Tom, *Calculus, Volume II*, 2nd ed., John Wiley & Sons, 1969.

[Baillie-de Byl04] Baillie-de Byl, Penny, *Programming Believable Characters for Computer Games*, Charles River Media, 2004.

[Borenstein94] Borenstein, Nathaniel S., *Programming as if People Mattered*, Princeton University Press, 1994.

[Buzan96] Buzan, Tony, *The Mind Map Book*, Plume 1996.

[Campbell85] Campbell, David, *Take the Road to Creativity and Get off Your Dead End,* Center for Creative Leadership, 1985.

[Cerny02] Cerny, Mark, John, Michael, "Game Development: Myth vs. Method," *Game Developer Magazine*, June 2002: pp. 32–36.

[Chalmers94] Chalmers, A. F., *What is this thing called science?* 2nd ed., University of Queensland Press, 1994.

[Chartrand77] Chartrand, Gary, *Graphs as Mathematical Models*, Prindle, Webber & Schmidt, 1977.

[Cosmides99] Cosmides, Leda and Tooby, John, "Evolutionary Psychology," *The MIT Encyclopedia of the Cognitive Sciences*, The MIT Press, 1999: pp. 295–297.

[Costikyan02] Costikyan, Greg, "Talk Like a Gamer," available online at *http://costik.com/gamespek.html*, 2002.

[Costikyan04] Costikyan, Greg, "Pastimes and Paradigms," available online at *http://costik.com/weblog/2004_06_01_blogchive.html*, June 18, 2004.

[Cousins04] Cousins, Ben, "Elementary game design," *Develop*, October 2004.

[Crawford03] Crawford, Chris, *Chris Crawford on Game Design*, New Riders, 2003.

[Csikszentmihalyi96] Csikszentmihalyi, Mihalyi, *Creativity: Flow and the Psychology of Discovery and Invention*, Harper Collins, 1996.

[Csikszentmihalyi99] Csikszentmihalyi, Mihalyi, "Creativity," *The MIT Encyclopedia of the Cognitive Sciences*, The MIT Press, 1999: pp. 205–206.

[Damasio94] Damasio, Antonio, *Descartes' Error*, Gosset/Putnam, 1994.

[deBono85] de Bono, Edward, *Six Thinking Hats*, Little, Brown, and Co., 1985.

[Duncan99] Duncan, John, and Hollan, James D., "Attention," *The MIT Encyclopedia of the Cognitive Sciences*, The MIT Press, 1999: pp. 39–40.

[Fauconnier02] Fauconnier, Gilles, and Turner, Mark, *The Way We Think*, Basic Books, 2002.

[Forrester97] Forrester, Jay, "System Dynamics and K-12 Teachers," Jay Forrester, 1997.

[Fulghum89] Fulghum, Robert, *All I Really Need to Know I Learned in Kindergarten*, Random House, 1989.

[Fullerton04] Fullerton, Tracy, et al, *Game Design Workshop*, CMP, 2004.

[Goodale99] Goodale, Melvyn, "Visual Processing Streams," *The MIT Encyclopedia of the Cognitive Sciences*, The MIT Press, 1999: pp. 873–874.

[Grier92] Grier, James W., and Burk, Theodore, *Biology of Animal Behavior,* 2nd ed., Mosby Year Book, 1992.

[Hecker02] Hecker, Chris, "Jay Stelly: Technically Speaking," *Game Developer Magazine,* January 2002.

[Hollan99] Hollan, James D., "Human Computer Interaction," *The MIT Encyclopedia of the Cognitive Sciences*, The MIT Press, 1999: pp. 379–380.

[Huizinga55] Huizinga, Johann, *Homo Ludens: A Study of the Play Element in Culture*, Beacon Press, 1955.

[Hunicke04] Hunicke, Robin, LeBlanc, Mark, Zubek, Robert, "MDA: A Formal Approach to Game Design and Game Research," available online at *http://algorithmacy.8kindsoffun.com/MDA.pdf.*

[Ip02] Ip, Barry, and Adams, Ernest, "From Casual to Core: A Statistical Mechanism for Studying Gamer Dedication," Gamasutra.com, June 2002.

[Isaacs65] Isaacs, Rufus, *Differential Games: A Mathematical Theory with Applications to Warfare and Pursuit, Control and Optimization*, Addison Wesley, 1965.

[Johnson99] Cara, Francesco, "Cognitive Ergonomics," *The MIT Encyclopedia of the Cognitive Sciences*, The MIT Press, 1999: pp. 130–131.

[Johnson99] Johnson-Laird, Philip N., "Mental Models," *The MIT Encyclopedia of the Cognitive Sciences*, The MIT Press, 1999: pp. 525–526.

[Juul02] Juul, Jesper, "The Open and the Closed: Games of Emergence and Games of Progression," Computer Games and Digital Cultures, 2002.

[Laramée02] Laramée, François, "Writing Effective Design Treatments," *Game Design Perspectives,* Charles River Media, 2002.

[Lazzaro04] Lazzaro, Nicole, "Why We Play Games: 4 Keys to More Emotion in Player Experiences," XEODesign® Inc., available online at *www.xeodesign.com/whyweplaygames*, 2004.

[LeBlanc00] LeBlanc, Marc, "Formal Design Tools: Emergent Complexity, Emergent Narrative," (Game Developers Conference), available online at *http://algorithmancy.8kindsoffun.com/gdc2000.ppt*, 2000.

[LeBlanc04] LeBlanc, Marc, "Game Design and Tuning Workshop Materials," (Game Developers Conference), available online at *http://algorithmacy.8kindsoffun.com/GDC2004/*, 2004.

[Ledin01] Ledin, Jim, *Simulation Engineering*, CMP, 2001.

[LeDoux02] LeDoux, Joseph, *Synaptic Self: How Our Brains Become Who We Are*, Penguin, 2002.

[McLaughlin99] McLaughlin, B. P., "Emergentism," *The MIT Encyclopedia of the Cognitive Sciences*, The MIT Press, 1999: pp. 267–268.

[MICRA98] MICRA Inc, *Webster's Revised Unabridged Dictionary*, MICRA Inc., 1998.

[Norman86] Norman, D. A., *The Psychology of Everyday Things*, Basic Books, 1988.

[Norman88] Norman, D. A., and Draper, S. Eds., *User Centered System Design: New Perspectives in Human-Computer Interaction*, Erlbaum Associates, 1986.

[Pinker97] Pinker, Steven, *How the Mind Works*, W. W. Norton & Co., 1997.

[Richards99] Richards, Virginia M., and Kidd, Gerald D. Jr., "Audition," *The MIT Encyclopedia of the Cognitive Sciences*, The MIT Press, 1999: pp. 48–49.

[Rollings00] Rollings, Andrew, and Morris, Dave, *Game Architecture and Design*, Coriolis, 2000.

[Salen04] Salen, Katie, and Zimmerman, Eric, *Rules of Play: Game Design Fundamentals*, The MIT Press, 2004.

[Shiffler04] Shiffler, George III, "Global PC Installed Base Reaches 716 Million," Gartner, 2004, available online at *www4.gartner.com/DisplayDocument?doc_cd=125049.*

[Siviy98] Siviy, S. M., "Neurobiological Substrates of Play Behavior: Glimpses into the Structure and Function of Mammalian Playfulness," *Animal Play: Evolutionary, Comparative, and Ecological Perspectives*, Cambridge University Press, 1998: pp. 221–242.

[Smith02] Smith, Harvey, "Systemic Level Design for Emergent Gameplay," Game Developers Conference, 2002, available online at *www.gamasutra.com/features/slides/smith/index.htm.*

[Smith03] Smith, Harvey, "Orthogonal Unit Differentiation," Game Developers Conference, 2003, available online at *www.gdconf.com/archives/2003/Smith_Harvey.ppt.*

[Sniderman99] Sniderman, Stephen, "Unwritten Rules," *The Life of Games*, No. 1 1999, available online at *www.gamepuzzles.com/tlog/tlog2.htm.*

[Suits90] Suits, Bernard, *Grasshopper: Games, Life, and Utopia*, David R. Godine, 1990.

[Wallas26] Wallas, Graham, *The Art of Thought*, Harcourt-Brace, 1926.

[Wright03] Wright, Will, "Models Come Alive," *PC Forum 2003*, EDventure Holdings Inc., 2003.

[Zimbardo92] Zimbardo, Philip, *Psychology and Life, Thirteenth Edition*, HarperCollins, 1992.