Chapter 2.1: Games and Society

Overview

Throughout all of world history and culture, games have been a part of human life. But in spite of their place in human experience, professional designers of games were practically unheard of 30 years ago. Between then and now, games that run on microchip computers, commonly known as *video games*, have changed everything. As the business matured into a multibillion dollar industry, the need for skilled game designers increased. In turn, this created a demand for training and education and now we are in a modern renaissance of sorts where game design is in the forefront. This chapter will introduce you to a brief glimpse of topics and issues common to video game design.

While there are many, equally valid, ways to view a game, this chapter will take a functional view, focused on shortening the conceptual distance between what the player experiences and what the game actually does.

With a broad field and limited space, some assumptions have to be made about an "average" reader. Designers, for example, *usually* work as part of a team; *frequently* make games for people other than themselves and for profit; are *sometimes* responsible for documentation; and take care of other things not uncommon in the game development workplace. It will also be assumed that you are not just interested in design but are actively engaged. So do not be alarmed if you read "your game" or "your career" every now and then, because we have faith in you.

Who Is a Game Designer?

To create something you have to be something.
—Goethe

A game designer is someone who designs games. Could it get any easier than that? But, while taking up the mantle is uncomplicated, getting others to see you're really wearing it will require more effort; you will need to continue designing games and probably for some time to come. The designing-a-game part... that can get a little tricky. Now that we come to it, game design can be downright difficult. A game designer needs help at every turn: a friendly ear listening to another way that players might use a summoning spell; a fresh set of eyes and an honest opinion about the feeling of your new levels; perhaps even a mentor to suggest a new way to approach a pacing problem.

Though it does happen from time to time, it is rare that game designers really go it alone. There are certainly many more people who *could* design, program, and decorate games on their own than people who actually do it. Creating video games is really a collaborative art—even for those who are doing it "on their own." It requires listening to people who are sure to tell you things you would rather not hear. Cooperation can be beneficial, and it can help to allow other people to put their handiwork into your game. In fact, just about every member of a game development project will, at one time or another, *be* a game designer too. They are applying countless judgments and perspective shifts while working on the game. Like an artist who repaints textures to improve the visibility of the track or a programmer who laid out the controls as it was described in the design document but made a few "tweaks" after trying it out—each one of these people is a closet designer, making the game design better without anyone noticing it. If there is one thing of value to take from this chapter, you may as well get it up front: *Everything in a game that adds value to the player experience has affected the design of the game*.

So, good news: You are probably a game designer already. Your experience may be limited,

and you may not “know” a thing about what you are doing, but that’s alright because you become a better designer by continuing to practice and learn. Designing a game isn’t half the battle, it *is* the battle, and it begins whenever you do.

When first infected by an interest in some creative craft, who doesn’t want to see where the magic happens? The visible parts and tangible parts are real things that get transformed into a final product that people will enjoy. You can look at prototypes or brainstorming sessions or diagrams or even a giant game design document (GDD) and witness the art in progress for yourself, right? Well...the truth is that, while all of that stuff is vital and every game designer needs to learn how to do the work, equally important is a context for learning—a way to fit everything together in your mind so that you can take in more and more information and put it where it belongs.

Take your time and keep an open mind. But be prepared to think for yourself and to disagree if need be. Game designers who are certain they’ve already worked everything out are going to need those closet game designers to pick up the slack.

Special Definitions

We have too many high-sounding words, and too few actions that correspond with them.
—Abigail Adams

While continually improving and maturing, the video game field is still young. Terms and definitions are not standardized and disagreements on language are common. In part, this is because definitions frequently depend on the particular kind of game being discussed. This language gap reflects a gap in understanding—terms for video games differ because thinking on video games differs. In reality, studios and gamer communities get by just fine.

A well-known philosopher, Ludwig Wittgenstein, asked readers to try describing a definition for the word “game.” Each time he proposed a necessary condition (a game must have competition; a game must be amusing, etc.), he would turn to a popular game that violated that condition. You might be surprised to know that he really wasn’t looking for the answer—he was trying to illustrate that fixed, exact definitions are usually not workable because people do not naturally use them. We communicate with loose and flexible categories instead of tight, precise definitions. But modern folks get uncomfortable when things aren’t well understood. Formal systems, taxonomies, standards... people *really* like to agree and are willing to fight through a lot of disagreement just to get there.

“What is a game?” is a difficult question under any light, one with no shortage of differing opinions and heated argument. All game players, developers, and academics have their preferred beliefs and it is unlikely that a single view will be coming any time soon. Terms here are not a key matter and are mainly presented to support a conceptual schematic—a diagram showing how the many parts of a game relate to the whole. So you are encouraged to construct your own definitions and get comfortable translating them into the languages of others; a necessary skill for communicating design issues with people who don’t think like you do.

The definitions offered in this section are intended to be functional. You should be able to apply them to all games that could ever be made. The rationale behind each definition will be explained so that you can disagree in part or in whole according to your own opinion. These particular meanings may not be “right,” but by the standards of the field today, they are certainly not “wrong.”

Artifacts

An *artifact* is a thing made with an intended function [HMCo00]: a thing someone used or

made to be used. Ancient swords and golden idols, stuff you might first imagine being an "artifact," certainly count, but so do everyday things. Spoons, shoes, pens, screwdrivers, computers, sandwiches, and everything else that has ever been made or used are described as artifacts. Clearly, games are artifacts [LeBlanc04] but we will need to look at the ways people categorize and comprehend artifacts in order to get any more detail.

People distinguish artifacts by what they do (purpose), what they are made of from (form), and how they do it (function) [Norman88]. Normally, all of these things work together, and the kind of sorting that is required has as much or as little detail as needed. For example, a "tool" is something that helps us perform work; a "hammer" is a kind of tool, usually with a handle and a head, used to bang on things; a "mallet" is a hammer with a soft head to prevent damaging the objects it hits; a "copper mallet" is a type used to move iron or steel machine parts without scuffing the surfaces. Once the appropriate level of detail is found, we're finished. If "any old hammer" won't do, maybe you will have to specify, "The blue-handled claw hammer in the bottom drawer."

The materials can be important: for example, how a cup may be made of many substances but a glass cannot. More frequently, we identify things by the ways they are assembled and operate (bulky CRT monitors versus light and slim LCDs).

A *system* is a collection of objects that function together to do something. It's a set of components structured in such a way that their properties, actions, and relationships to each other form a whole that produces a set of behaviors [Meadows08]. The behavior, the doing part, is the key to systems being more than a group of stuff. A system has an operational purpose, a reason for existing even if that reason is meaningless. By the way, add "systems" to the list of things, like artifacts and models, which fill our everyday world.

If there is one thing agreed (and that may be about right) across game development, game studies, and in the public, it is that games are systems. Video games are even systems running on systems! It's the other facets of the artifact that are causing the fuss.

So to answer our questions, we just need to organize them into their category of function. But that kind of organization will lead us, sadly, straight into the rocks with no way out other than through!

Play and Fun

Games facilitate play where either rules or goals or both impose some degree of structure to the interactions [Salen04]; they are "play artifacts." So, then, what is play all about?

With increasing interest in games, recent years have seen the discussions on play intensify. Books, articles, and conferences, large and small, have offered potential answers, from the philosophical to the scientific. A good place to start is to ask scientists how play appears—what does it look like when it happens? There is a great deal of agreement on this topic, so sifting through many descriptions highlights some familiar themes.

- No apparent purpose, play is just to play
- Voluntary
- Different from serious behaviors
- Fun, play is pleasurable
- Begun in relatively safe situations
- Improvisational

These features of play, skimmed from leading voices in the science, have been observed across the spectrum of animals—not just dogs and cats. While it isn't universal, it seems close—primates to birds, lizards, fish, and so on [Burghardt05, Brown09]. With all of this data and evidence, is it known why play exists?

The short answer is that things really aren't clear, at least not to the satisfaction of science. There are some good indications that it is beneficial [Brown09], but evidence on how and in what circumstances is still weak. Play might be an important means of learning, offering an evolutionary advantage, or it might be a recurring side-effect of other biological/behavioral patterns—reappearing because of sensory-emotional structure more than selective pressure [Burghardt05]. Play, in humans, might even be a critical factor that leads to culture, creativity, social classes, literacy, war, market economies, etc. Studies have shown similarities in brain activity that may ultimately lead to understanding the impulses in animal and human play [Siviy98]. At this point in the research, just about the only thing that seems a safe assumption is that playing feels good to the player. Players—both people and animals—are enjoying themselves when they play. We're all having fun.

Fun: What provides amusement—entertains or occupies [Oxford09].

Everyone holds their own opinions about fun because they have particular experiences of it. Cultural attitudes help describe and set expectations, while individual preferences actually govern our reactions to our own emotions. Fun isn't so much a single set of favored emotions as it is a label we put on preferred feelings that fit our personal attitudes.

You've probably experienced something like this: a friend insists you'll have fun trying something. You agree only to discover that, while they had fun, you didn't; and it wasn't because it wasn't fun *enough*. You just didn't like it. No sir.

Why do some people like watching *Schindler's List* or *The Exorcist*? Why do some want to play *Shadow of the Colossus* or *Braid* or *Dead Space*? Why would anyone want to experience sadness, guilt, disgust, loss, fear, or any emotion of that sort? Long-held views have attempted answers, suggesting that negative feelings experienced in entertainment were either not real or that the audience was willing to endure in order to get to the end where "good" feelings were, to feel better when it's over. But recent studies suggest that people can actually *like* bad feelings in the right context; feeling in control of the situation is one such situation. Playing a game, we can feel true, measurable fear—in the same form and intensity as if under real threat—yet enjoy it [Andrade07]. But taste and personality matter; for example, some people won't ever like watching a horror movie.

The interesting mental trick that allows entertainment to work at all, is the way our brains judge feelings based upon our perspective—experiencing things that are not part of the real world. When a situation is fictitious or "not serious," the brain regards emotional responses differently than if the situation were real or "serious." All of this fits with current leading theories on emotion, which we will mention later.

Pause a moment to review where we are. We began by asking "What is a game?" which reminded us that people identify artifacts by function. This led us to ask "What is play?" and then "Why is play?" We looked at some voices from science and found that people play because it's fun and that fun is a description of feelings when they fit the personality and the context.

A good game designer doesn't need existential answers ("Why does play exist?"), but rather practical and functional ones ("Why do people play games?"). We are standing at the answer: People play to have fun. You also know that fun isn't just one emotion. All that is left is a good-sounding term.

In the cognitive sciences, *cognitive artifact* describes artifacts used to aid thinking [Hutchins99]. Examples include lists, maps, string tied around a finger, calculators, and Wikipedia. If you use it to reason, calculate, or remember, it's a cognitive artifact. There are certainly emotions involved during cognition, but games have another purpose.

Since words are free, we create a provisional definition, *emotional artifact:* an object created

and used to experience feelings.
Game: An emotional artifact used through a series of structured interactions.

A Model of Games

A *model* (as a noun) means a representation or an explanation of something [Chartrand77]. *Model* (as a verb) is the act of representing or explaining. Models are everywhere: a blueprint modeling the floor of a building, a number of hit points (HP) modeling the health of a player, and sheet music modeling the arrangement of sounds over time. In our modern lives, models are like artifacts: pervasive, important, and so common that even bringing them up to talk about can seem pretty silly. Photographs, meters, indicators, price tags, audio recordings... the list is endless.

To *abstract* is to remove details from something, and the process of *abstraction* is the universal method for solving problems—simplify matters by ignoring unimportant and distracting details. *Abstract models*, then, are representations with only the important and relevant bits left in them. Look up "abstract model" on Wikipedia, and you will be redirected to "scientific modeling" and find the following:

Modeling is an essential and inseparable part of all scientific activity... [Wikipedia09].

All models are abstractions in some way or other; after all, they represent something else, but that doesn't mean that particular thing is replaced by the model. Models are not reality. A picture of a car won't get you anywhere. The photo of the food is not the food itself [Fauconnier02]. This is the essence behind the saying, "The map is not the territory." Details are expected to be missing and imperfections expected to misrepresent reality, but as long as the model serves the intended purpose, we're happy. A map to your friend's new house needs to show the streets to get there, but doesn't need every side street, alley, and tree along the way. You won't expect to arrive and find, outside on the lawn, letters the size of a school bus spelling "My House." It's just a model. Today, game designers have a lot of choices when it comes to conceptual models of games. You will probably look to several over the course of your career. Programmer and designer Mark LeBlanc's MDA (Figure 2.1.1, left) is a model of the player experience, showing how subjective experiences are generated through play. In the middle is Jesse Schell's "Elemental Tetrad" (from the excellent *The Art of Game Design*), which describes the elements that make up the game artifact. Furthest right is a formal model, by Jesper Juul, which proposes necessary and sufficient conditions for something to be a game [Juul03].

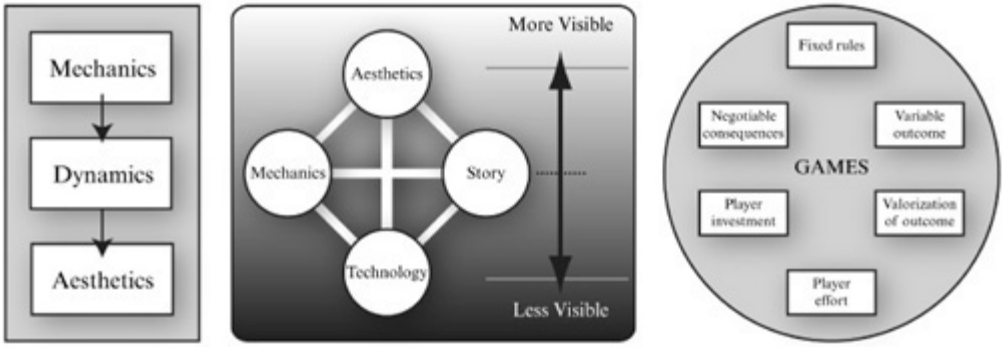

Figure 2.1.1: Three models, MDA, Elemental Tetrad, "6 game features" [LeBlanc04, Schell08, Juul03].

The model of the player-game used in this chapter will organize some major features into two domains: Player and Game. Three pairs of closely related concepts will be placed into the model. Keep in mind that boundaries are fluid and blurry. No attempt is being made to declare

that this is *the* way games are designed or organized; it is an outline for *a* way designers think and organize their design work for the games they create.
Everyone knows that game designers have to consider games and players, so Figure 2.1.2 doesn't hold many surprises. On the left, you see Player and on the right Game. Neither of these domains is meant to represent its subject entirely, just abstractly. In this case, Player can be thought of as *things directly concerning the player* rather than being just about some real person. Eventually, every thought and feeling ends up on this side of the model.

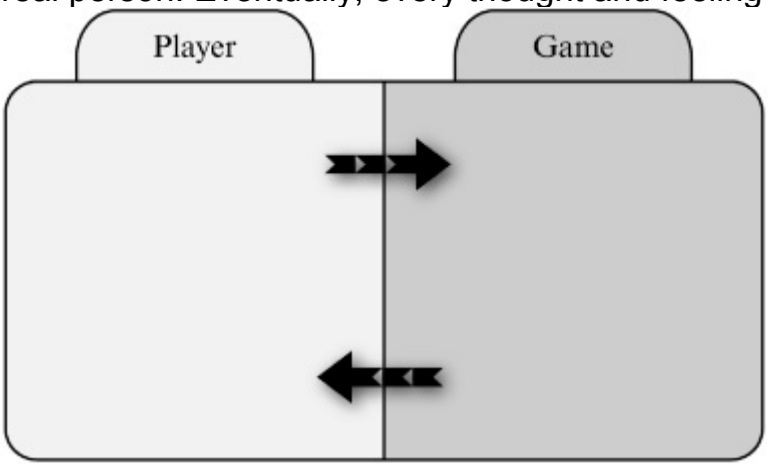

Figure 2.1.2: Player, meet game.
On the right half, into Game, go *things that make up the game*. Just what those things are will vary depending on the particulars of each game (video game, board game, jolly round of pat-a-cake, etc.) at a given time. In video game development, most of the things that fit into this domain will be built out of software. Video games usually run on general-purpose gaming machines that are outside the control of a designer; using the hardware is relevant but designing hardware is not typically considered part of the game design. Exceptions will always exist, of course, and can result in successes like the guitars of *Guitar Hero*, but it's still relatively unusual unless you're Shigeru Miyamoto.
The arrows in the center of Figure 2.1.2 are another abstraction, just hinting at some type of interaction happening during play—for example. Player affects Game, Game affects Player—but not much more yet. There are no clues to the order, rate, sequence, duration, or even type of inputs and outputs.

The Game Half

This is the domain of the "real" stuff. As we mentioned, hardware (if it matters) and software (which always matters) definitely *are* the game. The digital bits that describe what can and will happen in the course of playing are set here (Figure 2.1.3).

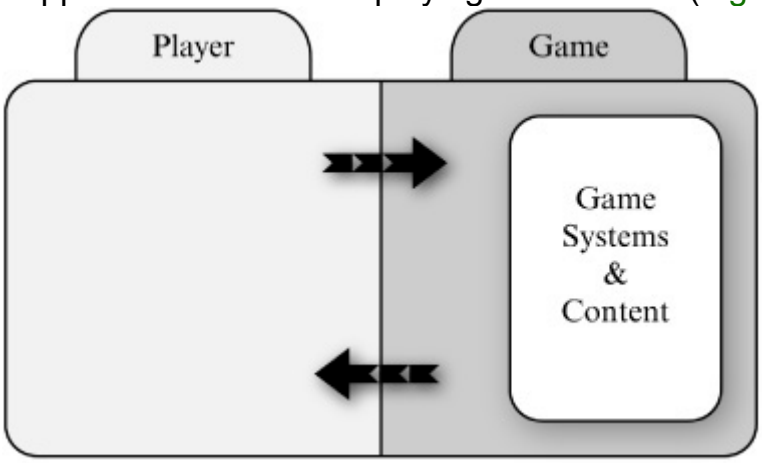

Figure 2.1.3: The rules.
While every game is a system, video games, in particular, are systems built of systems running on systems and so on. But *game system* or *gameplay system* refers to just those that directly affect the things that the player will do (see “play mechanics” in the next section). Often, this is what people think of when imagining the work of game design: creating the rules. Game systems determine the procedures and operations that will utilize game resources and, with a little help from the Player, produce outcomes. Most formal elements that define a game are products of game system structures.
Game *content* is all of the stuff forming and populating the universe that game systems govern. It is the space and substance of your game, from game board to galaxy, and the resources that fuel the game systems during the course of an actual game. Content makes up the what, where, and when of everything operating in the game; all of the materials experienced during the game, whether concrete objects like battle tanks and rooks, or more abstract concepts like “Mission 12,” or states like a running account balance of 2500 credits.

The Player Half

Try as you might, you won’t have direct control over the player. The limits of a designer’s direct control are inside the limits of the Game domain. All of the really important stuff—motivations and feelings—happen across the border, in the mind of the player. People are the most important part of your game.
Earlier, you read that there are all kinds of things about players we would leave out of the model until they mattered. In Figure 2.1.4, you can now see something that really matters, the *experience*—the relevant perceptions, feelings, thoughts, intents, and actions. It’s so important that we show it twice! (One is the overall experience, the other more specific.)

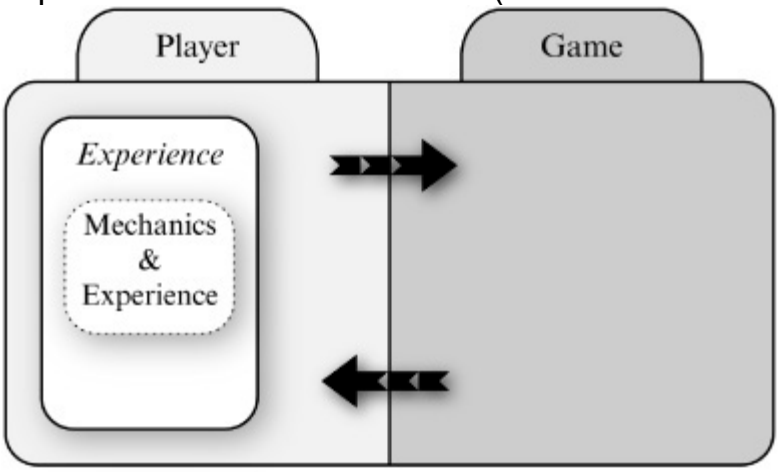

Figure 2.1.4: The player’s experience.
Experiences are enormous sets of mental stuff—packages of psychological and physical states all crammed into a box of percepts and memories and labeled with a context. How we talk and think about them can shed a little light on how they fit into that box. “Driving home was nice today, for a change.” “I got beat last time, but this time I have my defenses set up... Here they come!” “I’m trying to finish as fast as I can, but I’m still going to be late!” The range of possible experiences is boundless. Miniscule experiences might be a single percept like feeling a touch; epic experiences could be a bike ride from Chile to New York.
Our specific inner *experience* is our shorthand for *aesthetics* and *emotions*. Aesthetics are reflections on and considerations of the emotional experiences evoked during play [LeBlanc04, Huizinga55]. The emotions are those feelings. The experience is what players want from your game, why any of this exists in the first place, and why you are reading this

book.
A little later on you will find that emotion covers much more than you might imagine. Solving problems, defeating an enemy, and spending time with friends online all have emotional components that largely determine whether or not you enjoy that experience. The cold facts rarely mean much to us—emotions are the only things that matter.
Mechanics (*game mechanics*) are systems of interactions between the player and the game. More particularly (and a little atypically), this chapter will consider game mechanics to be the player's experience of those interactions. While game mechanics are more than what the player may recognize, they are only those things that impact the play experience. In other words: *what happens during play that affects the player*.
While there can be any number of particular reasons for a person to play a particular game, all rely on it to evoke feelings that the person will value. If it can't, there will be nothing to encourage play. Games are emotional artifacts. When they create the kinds of experiences you intend and give players the kinds of feelings you want, you are an effective artist.

The Third Half

It has been said that video games run on a "coprocessor system" [Wright03]; one made of silicon and powered by electricity, the other inside the player's mind. Before you can run any part of that system, however, you will need to provide a bridge between the two physical entities. Game don't just need an interface, they need two!
In Figure 2.1.5, the Player and Game are connected to each other via a system of *interfaces*—hardware and software devices that connect information and commands between device and user. As a general concept, this is the most straightforward piece in our schematic. Interfaces are the most visible aspect of a Game, and the Player has little difficulty identifying the systems of input and output.

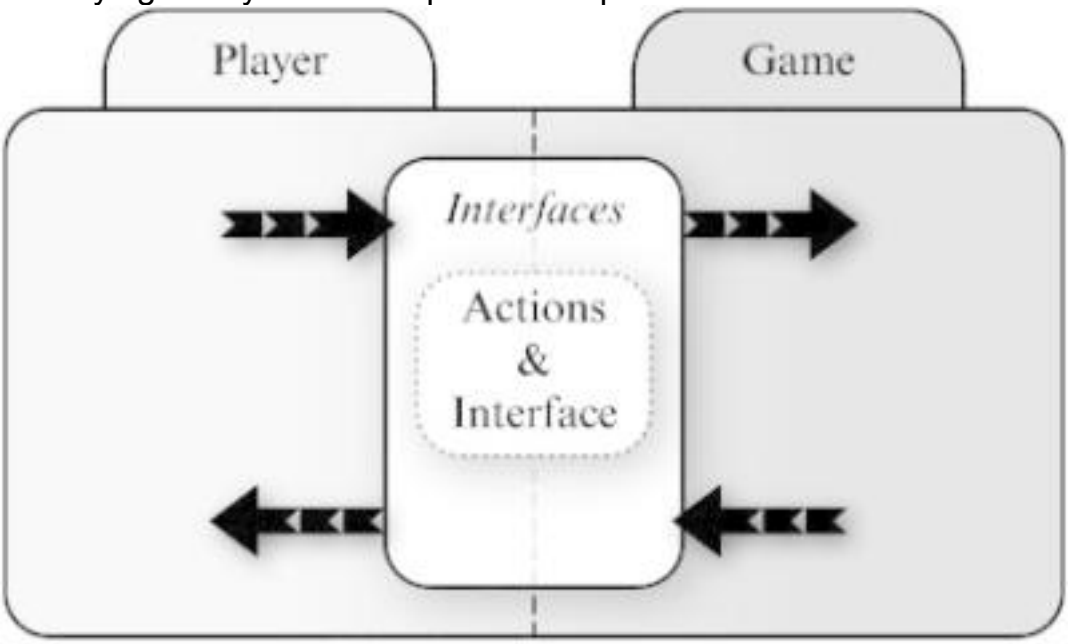

Figure 2.1.5: Interfaces connect Player to Game.
In our model, the *interface* element will contain all aspects of presentation and feedback, regardless of their mode—video, audio, haptic, etc. The reason for this approach is that it makes more sense to organize design thinking around the functional effects (alerting the player to a danger, providing good feedback on a failed command, and so on) instead of by the modalities used to express them.
Players perform *actions* in the game, using the interface to signal their intentions to the game.

There are two dimensions of action: the actions using the controls (pressing a button, moving a stick) and the actions that happen in the metaphorical space of the game (shooting a basketball, unlocking a gate)—inside the coprocessor system of brain and machine.
The arrows attempt to show that information is traveling in a circuit through the whole system. Once things are underway, there is no real point of origin—the player has something to tell the game and the game has something to show the player. Every bit of this information crosses through the interface.

All Together

Now this is not the end. It is not even the beginning of the end. But it is, perhaps, the end of the beginning.
—Winston Churchill

Here we are at the finished schematic (see Figure 2.1.6). As a hybrid, it's not too complex while viewing structure, experience, and the cycle of interaction in a fair way. It does oversimplify and understate, but the leading problem is that all relevant matters of a game will not always fit into neat categorical bins. This is because *the relevant matters of a game do not always fit into neat categorical bins*!

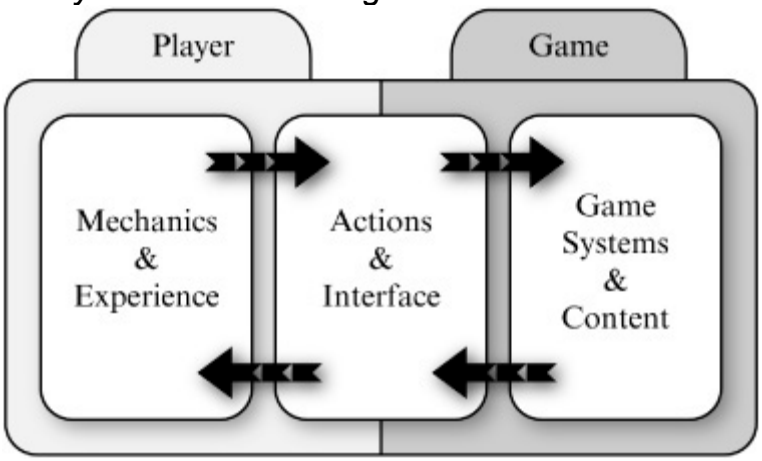

Figure 2.1.6: A model of the Player-Game relationship.

Just remember that it's still a model, not a substitute for the real thing. As an abstraction, it should offer you a reasonable way to organize thinking while learning how to be a game designer the right way—making games!
The rest of this chapter will be split into two major sections. The next eight sections, through Content, are focused on the pieces of a game, and these things are treated like that: in pieces. The sections that end the chapter will deal with design work itself, creating concepts, working in iterations, documentation, and so on.

Game

Before you picked up this book, you knew what a game was. From *Backgammon* to *BioShock*, despite all of the variety in kind and style, we know a game when we play it. But, ever since Plato, people have valued formal qualities as a way to fully explain and understand things.
What follows in this section is a brief description of various formal elements typical of games. No assumptions are being made that *any* game has *all* of these, but most games have some.

Objectives

Objectives are designed requirements that players must satisfy to accomplish a particular outcome [Fullerton08]. Encoded into the structure of the system itself, objectives are formal properties of the game, gating player progress. As a means of establishing conflict and

challenge, objectives motivate player engagement with an offer of finite and solvable problems that players then work toward. At their most rudimentary level, objectives give people something to do.

When a game offers one set of mutually exclusive objectives to everyone, it begins with a natural balance in place. When games offer differing objectives to players, greater effort is needed to balance and minimize the player concern for unearned advantages. (It's not fair!) One approach, popular with online games, is a team versus team arrangement where objectives trade back and forth between opposing teams, from one round to the next. A given player's success is then tied to the choices and actions of fellow players—not the by-product of unit design.

An objective is not quite the same thing as a goal. Objectives are those things players get asked to do; goals are those things that players want to do. We'll go over the difference later in the Player section.

Outcomes

Games have a set of possible and uncertain outcomes that will result from the players' interactions. These conclude play—they end the game—and need to be measurable. Why do they need to be measurable? The full answer involves the way our brains anticipate futures and evaluate the successes and failures in our decision making, giving us good or bad feelings in the process. For now, we can say players need to evaluate their performance for the game to be "meaningful." This is also why outcomes are usually unequal, and some results are better than others. Adding a little risk of failure and potential for reward makes a game even better at holding our attention.

It is not enough that a game's outcomes can be measured and that those measurements be as meaningful as possible. Outcomes must also be agreeable, clearly presenting the end of the game and the measures of performance. Players need a standardized way to agree on the result. When this is absent, things get uncomfortable as the players must negotiate their opinionated versions of "who beat who."

Winning and losing are the two classic outcomes. They are discrete and definitive results of competition, but even among classic examples of the type (backgammon, chess, etc.), games commonly can end without decision—a third outcome, the tie (draw).

Many video games lack distinct outcomes, like win or lose altogether. Nevertheless, players evaluate and compare performance, weigh accomplishment and failure, and have a great time in the process. Sometimes people will describe this as temporary states, such as *win state* or *loss state*. So, if you hear these terms, you may need to clarify whether or not the state in discussion is final or temporary. Was an end condition reached? Or was the commentator just reaching?

Uncertainty

In a situation with many possible futures and no way to foretell which one will happen, you have *uncertainty*; it's any time that things are not certain! In life this is unavoidable and regrettable, but, in games, uncertain outcomes are customarily expected and desired. (Who wants to keep playing when the outcome is predetermined?)

Uncertainty is necessary to the experience of playing games. Final outcomes that feel totally predictable tend to be boring or frustrating or both. When the decisions that remain have obvious answers and conclusions, there isn't much room to build in expectations. Without that space to grow, feelings quickly atrophy, and the whole experience stops being worthwhile. Players need to feel as though the things that they want to do can be accomplished, but have some chance of failure, either through mistake or bad luck. Designers control uncertainty by

requiring physical performance and mental strategy, as well as limiting the information. Random systems can also help; however, you will be cautioned about them later.

Rules and Structure

Rules form the structure of each game, establishing an uncertain relationship between the player and her objective. Without rules, play becomes unregulated and nebulous. It is ironic that less certainty in rules can diminish the expectation of uncertainty in play. 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 do exist on a moment-to-moment basis, inconclusive and ephemeral, "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 uncertain systems rather than spending most of their effort on continually reinventing and maintaining 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 nondigital 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.

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.

Some rules only come into effect at given times or in particular circumstances. These often serve to create variation, govern game progress, and ensure that the system remains within preordained limits [Fullerton08].

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, and it has a perfect understanding of the rules (at least as far as that version is concerned). Second, the rule systems can be much more detailed 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.

Operations are rules describing the methods and procedures players use to play the game. These are concerned with defining what actions players may perform at a given moment. Operations determine the timing and order of actions and the precise instructions for how to do them.

Systemic rules define the possible conditions of the game and its events. They are concerned with the various states (configurations of position, value, etc.), limits (especially resources), and events that result from player action or chance (scoring, penalties, etc.).

Frames

Games and playing create, in our minds, temporary spaces that are separate from the real world. Dutch historian John Huizinga coined the phrase "magic circle" to refer to these places. Later, Gregory Bateson offered a similar description of the *frame* of a game as the understood context of play—"this is just a game"—the time and space setting apart playful and inconsequential activities from the serious and consequential [Bateson72]. Moves made during a game of chess are within the frame of the game, but the players' entrance and the conversation afterward is not.

These contexts aren't a recent invention or exclusively human. Animals, when playing with others, signal their intent to others. These signals let them engage and remain in play. Feelings within the context of the game's frame are supposed to be safe and experienced without real-world effects. That said, not all feelings are as compatible with these frames as others. A betrayal in *Diplomacy*, which is part of the game, can be difficult for some players to enjoy and can be tough to forgive, lingering into the real world. Other feelings, like frustration or humiliation, can disengage players from the game entirely.

Player

I work tirelessly not to laugh at human actions, not to cry at them or to hate them, but to understand them.
—Baruch Spinoza

There is nothing for a game designer to value more highly than the player.

Most of today's veteran game designers grew up playing games. They tend to come from very similar stock, having tuned their awareness and sensitivities playing hardcore games. Passion and excitement have been forged in experience to create the sensibilities that guide them through each day's work. Since most of the industry is still serving the "core" gamer and designers understand that kind of player because they are that kind of player, personal judgment and preference are excellent guides when you are the audience.

But times are changing and the audience is changing with them. The population of game players is growing and becoming more diverse. In the United States, 65 percent of households play video games and the average age of the most frequent purchasers is 40 years old. More than a quarter of adults over 50 play video games, and it is the largest area of growth recently, according to "Family Entertainment" [ESA09].

But how can designers create a successful game for people potentially different from them? The answer is by becoming deeply interested in people and psychology, and most importantly, learning how to listen.

Emotions and Feelings

Any emotion, if it is sincere, is involuntary.
—Mark Twain

Set this book down for a moment and make a fist with each hand. Now hold them together with the heel of each palm touching. That is a rough model, in shape and size, of a human brain. The one in your head has around a hundred billion neurons in various modes of resting and firing while you read this, while you think, and while you feel.

In 1884 psychologist William James posed a question in an article: Why do we run from a bear? Perhaps, he suggested, we see the bear, begin running, and become afraid because we are running. Well what would you say to that? Most of his colleagues knew what to say: poppycock! Common sense told us all that we needed to know. We see the bear, become afraid, and run because we are scared. (Duh!) Well, over a hundred years later, the ideas that James-Lange posed would get a lot more respect since neuroscience technology has advanced enough to allow scientists to watch the brain at work.

While today's most commonly held scientific view (Figure 2.1.7) is more nuanced than James-Lange, it looks far different than the common-sense model most of us regular folk still hold today [Ledoux02, Pinel07]. Seeing the bear creates visceral emotions and bodily reactions before we have even made sense of what we have seen. The likely reason for this supercharged emotional system is efficiency—it is much faster to react than to have to wait for consciousness to make up its mind.

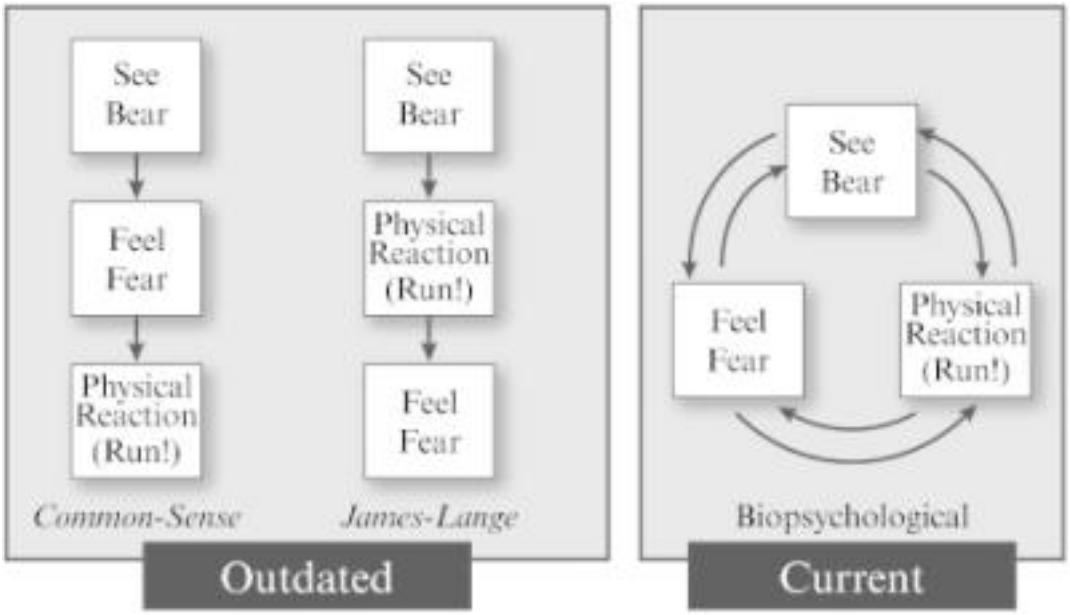

Figure 2.1.7: A modern theory of emotion [Pinel07].

The process looks a little like this:

- 1 Something worth feeling comes up—present or remembered.
- 2 Signals run between emotion systems and conscious systems (cortex).
- 3a The emotion system generates responses.
- 3b The cortex is using explicit memories to recognize and understand.
- 4 The cortex uses information from 3a and 3b to make choices.

Emotion systems are quick and often accurate (the subject of the book *Blink*), but the cortex isn't just along for the ride. We can go with the emotional flow, or the cortex can try to control the rest of the system. For example, when it recognizes that the bear is stuffed and has a tag on its bottom. "That's a teddy bear, silly!"

People spend a good part of their lives unaware of the emotions that they are having. And when they do feel something, they make sense of it by referring to the context of the situation they are in. What's really interesting is how we can confuse these things a bit. In experiments, men have been "made" to find women more attractive than others by speeding up the sound of a heartbeat. And everyone knows (or should) that a scary movie is a great place for a first date. What everyone might not know is that this works because the feelings of fear and romantic attraction are similar and easy to confuse.

Even this view here is getting a little dusty from all the progress neuroscience has been making. But game designers don't need to be scientists—we just need a model that makes sense and makes good predictions, such as understanding that our minds evaluate our emotions differently depending on the situation. Some people enjoy the rush and tensions of being scared in *Left 4 Dead*, and some people find those same sensations unpleasant and, consequently, decide that the game is, "not for them."

Players have preferences, feelings they like to have during entertainment, and feelings they do not like. Not everyone enjoys hunting and gathering. A designer needs to understand the audience that will be playing the game. Emotions are not just for entertainment. Emotions are also covering our goals and everything we care about.

Thinking Is Feeling

Let's not forget that little emotions are the great captains of our lives and we obey them without realizing it.

—Vincent Van Gogh

From antiquity comes a common belief about our minds: one part rational, moral, and recently evolved; one part irrational, reckless, and rooted in the primitive parts of the brain.
But a great many of our decisions happen much “deeper” in the brain. Much of our thinking and calculating, that stuff that most people feel like they have a firm cognitive grip on, is actually leaping up from emotional systems that have channeled the answer to us before we were hardly aware of the question. And even when our rational cortex is in full effect, calculating and figuring, we need the brain’s emotional systems to let us know when the “right” answer has been reached. It feels good when we think we have the right answer. Why do we play games again? (We play because it feels good!)

Working Memory

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.
You will find this figure, 7 ± 2, throughout design fields of all types. Keep the demand on your player’s lower range of memory retention if you want them to remember. Higher if you want them to forget.
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.

Attention

You are in a busy restaurant waiting, with your friends, for a table. Despite the noise and movement of people chatting over lunch, hurrying in and out, you listen to your friend as she tells you about her day. The host steps up and begins “Is there a...,” and before you know it, all you can hear is his voice, waiting to see if you are being called. The sound of your friend’s voice has fallen into the crowd, and you wonder what she just said.
That is *selective attention* (*attention*), the process of focusing; tuning in on things you care about and out on things you don’t. This focus is engaged by either your own thoughts (choosing to listen to your friend) or by an external experience (the voice of the host drawing your ears away) [Pinel07]. Your brain, in order to do this tuning, is able to amplify and dampen the mental representations, depending on whether the thing is attended to or not. In other words, it’s how we turn our focus on things that seem to matter, allowing us to effectively prioritize goals.
Some of the most important studies of attention were conducted in the 1950s and involved people listening to two simultaneous messages. These studies produced several findings:

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

Example: Getting Attention in Hidden Object Games
In the casual game markets, the hidden object genre is popular. Players search for items embedded in and around the details of colorful background scenes, often racing against a countdown timer. Once some minimum number of objects has been found, they are able to

progress; failure usually results in retrying the level over again. It is typical for people to get stuck from time to time, usually because they aren't sure about what one of the requested items should look like (What kind of "bow" do they mean? Bow and arrow? Violin bow?), or they believe that they have already thoroughly looked over some part of the scene and have mentally checked it off a list (I know it's not up there!). It helps to give them a hand when it's been a while since their last find, such as a brief move, or a pulse, or a sparkle. Keep the effect subtle, and it will just tickle their peripheral vision and draw an eye toward it. When they find it (There it is!), they feel a sense of accomplishment, you feel the awesome joy of making something fun, and everybody wins. The next time it happens and they notice (That sparkles when I'm stuck.), it gets added to the suite of mechanics.

Psychological Quirks

There are a number of thinking and feeling oddities that influence decision making and emotional evaluations. Sometimes there are specific circumstances that must be at work, but other biases are in full effect regardless of the situation.

Framing Challenges

Put one way, a problem is easy. Put another way, our brain can have trouble understanding the context of the question, failing to find a good strategy for reasoning.

> **Consider the following puzzle:** You are dealt four cards, as shown in Figure 2.1.8. Each card has a letter on one side and a number on the other. Please pick *only those cards that must be turned over* to verify the following statement. A card with a D on one side must have a 3 on the other.

Figure 2.1.8: D on one side must have a 3 on the other.

Make a quick note of your answer. Now, imagine that you are working at a bar and grill and must make sure that no one under legal age (21) is drinking. Each card in Figure 2.1.9 represents a patron. One side shows the age of the person, the other what they are drinking. Please pick *only those cards that must be turned over* to see if any of these people are breaking the law.

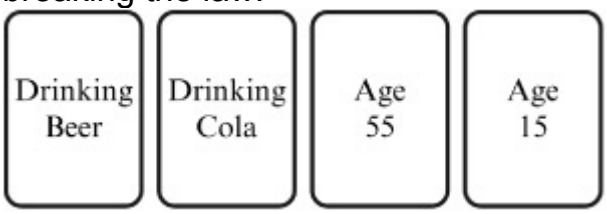

Figure 2.1.9: Which people need to be checked?

The answer to the first question is D and 3. The answer to the second is beer and 15. About 25 percent of people choose correctly in the first case and about 65 percent in the second, even though they are the same task [Pinker97]. (In psychological research, this experiment has caused a lot of debate!) One thing is clear: the way a puzzle is presented matters!

Conditioning

Conditioning is a type of learning through association or reinforcement. 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.
The classic example of Pavlov's dogs illustrates the concept. Before conditioning, a sound heard by the dog produces no response. However, meat in the dog's mouth causes the animal to salivate; the meat is the *unconditioned stimulus*, and the salivating is the *unconditioned response* as the dog's reactions were in their natural state. During conditioning, the sound is played while meat is put into the dog's mouth, causing salivation. After this has gone on for a while, the dog is "conditioned," and needs only to hear the tone to begin salivating (the *conditioned response* or CR) [Zimbardo92].
Operant conditioning describes learning where a behavior is encouraged or discouraged by its consequences. *Positive reinforcement* rewards a behavior (the *operant*) with a positive outcome, making that behavior more likely. In *negative reinforcement*, the behavior is encouraged by the threat of a bad outcome should the subject choose to stop the actions. Note that both positive and negative reinforcement schemes encourage a particular behavior; one is nice about it (have a cookie!), the other... not so nice (do it or else!).
Punishment is the third kind of operant conditioning. *Punishers* reduce the likelihood that the subject will perform the act, and *punishment* is, well, the application of the punisher [Zimbardo92]. Be *very* careful when thinking about using punishment, as it tends not to be a ton of fun. Try reinforcing alternative behaviors rather than simply punishing those you want to discourage.

Aiming at Audiences

The more one pleases everybody, the less one pleases profoundly.
—Stendhal

Your audience is a set of people—from none to everyone—with enough interest in your game to give it some attention. Increasing that number of people is a complex and uncertain problem, but there is a lot you can do to help improve your chances. For the moment, we have time to concentrate on one: targeting an audience.
The basics of this approach are simple: don't try to please everybody. Not because you have any good reason to keep people from liking your game, but because it is nearly impossible for everyone to like your game. By trying to achieve that effect, you only increase the risk of pleasing no one.
1 Identify groups to aim for—your target audiences.
2 Model their preferences.
3 Create a list of aesthetic goals informed by the model.
4 Use the model for guidance in design.
This style of approach has roots in the practices of targeted marketing [Russell02]. In the 1980s, advertisers began to move away from mass market advertising, where markets were represented by size or households. People were being faced with a proliferation of choices and advertisers found a need to address their new and more individual demands. To reach these individual demands, whole markets are *segmented* into smaller groups of people with shared qualities (*demographic variables*). While segments can be based on any kind of relevant feature, age and gender became standard. *Market segments* are still generalizations, not real individuals but a sort of averaged concept, the *demographic profile*.
A *target audience* is a segment you choose to serve with your game. In professional game development, it is a bit uncommon for the developer to have complete freedom in this decision. Most professional studios are working for or with a client (publisher, investor) that will want a defined segment with a purchasing history. If, however, you are the master of your own destiny, then you can choose the target audience that fits the goals you have.
After (or as an exercise in) choosing an audience, commit these to a model. The best way to

do this is by creating *personas*, which are fictional characters representing your targets. Personas have become a popular tool in many customer-centric businesses because they let you address fictional characters as though they were real people. A persona is a little like a role-playing character (sans cape).
Resist the urge to create elaborate personas with exhaustive lists, personal histories, and irrelevant details about traits and quirks. Short simple and memorable, like Figure 2.1.10, are preferred to complex and confusing. “Was frightened by an anteater as a child,” will not help you decide if your players would like to customize their playing piece. On the other hand, you should spend a little time making sure that the personas you create will fit the audience you are targeting. How? By talking to people that fit the demographic profile!

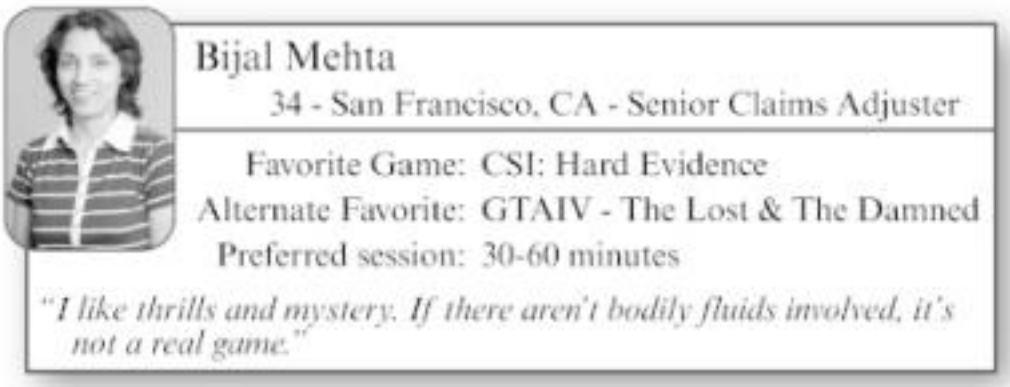

Figure 2.1.10: A persona without too much detail.

Experience

There is no excellent beauty that hath not some strangeness in the proportion.
—Francis Bacon
Experience: The emotions and aesthetic feelings evoked during play.
Experience is huge, but we will concentrate almost exclusively on one part: the emotions. Emotions are our reactions to the game—the feelings we have while playing. These sensations arise from game mechanics, graphic art, sounds, settings and narratives, and everything all at once.
Designers make games that affect and move people. Just what affects and what movement is really the art of creating a design for an audience. Maybe you want to try something new, not limited to the aesthetics typical of other games. There are plenty of opportunities.
Nicole Lazzaro, president of the research and design company XEODesign, was the first to apply a special technique for monitoring expressions (FACS) while observing players and identifying the emotional states they were experiencing. Through that work, came the Four Keys, descriptions of emotional experience that people enjoy in games: Hard Fun, Easy Fun, Experience, and Social Fun [Lazzaro04]. As they are convenient categories of experiences, this will form the basis of the organization in this section.

Mastery

“Winning isn’t everything.”
Mastery has been the hallmark aesthetic throughout the history of games. It is the part of gaming driven by contests of talent and effort. For many, gamer and nongamer alike, this is what games are all about: using skills and strategies to beat the game. For game designers, this has been the single most important and time-consuming aesthetic goal. Many have spent

their careers concerned with nothing more than this.
This is not so much “the desire to win” as it is the reason behind that desire. It is a way for players to prove how good they are. The emphasis of mastery is on satisfaction for a job well done and well tried. Loss is acceptable because it’s a possibility with any worthy challenge. Mastery exemplifies achievement and success in the face of creditable risk. Potential defeat is what makes the challenge meaningful.
Mastery is personal. While a particular challenge may involve defeating opponents, the motivation is pride in oneself for your talent and effort. Players motivated by mastery want to know, inside, that they are good, not just share the news. Bragging is just a bonus. “I beat Ninja Gaiden on **hard**!”
The feeling of this triumph is fiero [Ekman03]. It is the vigorous joy at overcoming adversity. It is cheering “Yes!” and maybe even performing an embarrassing gesture while doing it. This is the fruit of sustained effort—the purity of victory.
To offer mastery, you must also offer control. If players have no control and fail, they are frustrated. If they have no control and succeed, they didn’t do anything special. You must offer players choices. Periodically present players with a few options. Give them opportunities to use strategy so they can delight in their own shrewdness.

- Design mechanics to test player skills.
- Test multiple skills at the same time.
- Minimize the role of luck in mechanics for mastery.
- Lead players to tempting rewards.
- Allow players to choose to accept risky opportunities.
- Separate meaningful objectives from players with significant obstacles.
- Use multiple objectives to allow players to use strategy.
- Create danger and risk that they can choose for themselves.
- Enhance the perception of difficulty where possible.

Flow

The flip-side to mastery is frustration, and this is a huge issue for game designers in their daily work. How can your game let people overcome meaningful obstacles without those obstacles being too challenging?
The most popular model of this issue is called *Flow*. Created by positive psychologist Mihalyi Csikszentmihalyi, Flow is a fully immersed mental state. The player is absorbed with task at hand, unaware and unbothered by things outside of the immediate experience.
Csikszentmihalyi calls Flow “optimal experience,” and it is common to any activity where people are “in the zone.”
Flow starts with a challenging activity, with an uncertain outcome, that needs skills [Csikszentmihalyi90]. In other words, start with an opportunity for mastery.
Clear goals; reachable goals compatible with the player’s skills.
Becoming one with the activity.
Clear and immediate feedback; aware of goals.
Complete concentration on the task at hand.
Effortless control; no concern for losing control.
Loss of self-consciousness.
Time becomes distorted.
As Figure 2.1.11 shows, attaining Flow is a balance between the challenge of the task and the skills needed to succeed at it. The white area rising diagonally is the *Flow channel* showing that the challenge of the task must increase along with the player’s skill (and vice

versa). When someone is here, they are said to be *in Flow.*

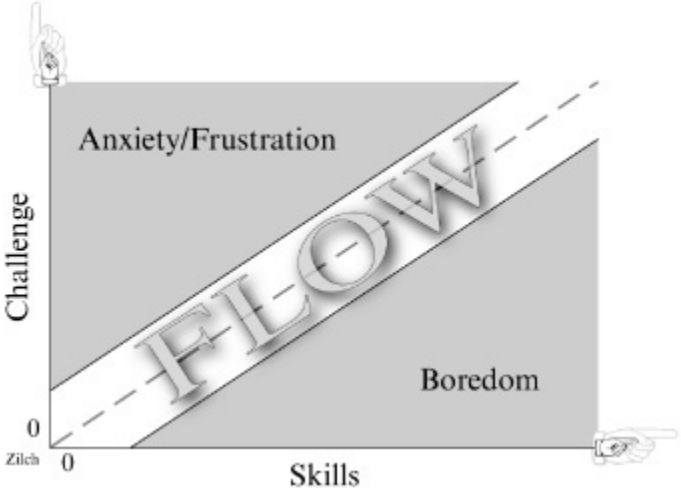

Figure 2.1.11: The flow channel—not too hard, not too easy.

Immersive

Curiosity is lying in wait for every secret.
—Ralph Waldo Emerson

Where mastery challenges, immersion entices. Players are drawn into the game with a suite of feelings from wonder to curiosity. Inspire players to look around the next bend or over the next hill, and keep their minds filled with possibilities.

You have an advantage. When a player chooses to engage a game, he has made a statement: “I am willing to see what is inside.” Now the game needs to deliver. Capture his attention with patterns of concealing and revealing. Show a little bit and hide a little bit.

Curiosity is an inspiration that seeks answers; we want to know because we anticipate something interesting in the result. Present mysteries and ambiguities, and offer answers in small amounts until you are ready to surprise them. If answers are to be found, offer answers in exchange for a little effort. Give people questions to ask and they will say: “What is this?” “Is it safe?” “Is it for me?” “What will happen if I try this?” “What’s happening?”

Wonder is experienced when faced with things that are rare and incomprehensible. We are overwhelmed and fascinated by things that we can’t understand—either what it is or how it has happened [Ekman03]. When the understood is revealed to be something much greater or more complex than ever imagined, the player can be placed into a state of wonder. You will need to maintain a balance between improbable and impossible. The player must believe the wonderful could exist; she’s just amazed that it does.

When wonder is mixed with fear, we have awe. Not only is it something we can’t understand, but we also are struck with the feeling that its danger cannot be measured. Is it a threat? Undoubtedly. Is it threatening us? Uncertain.

Consider the following when trying to increase immersion:

Curiosity—want to figure things out.

Create mysteries but leave some answers ambiguous.

Excite players with possibilities and opportunities; show them new worlds.

Show people the curtain, not the man behind it.

Attention—keep people interested by presenting things that would interest them.

Leave them guessing with ambiguity.

Rhythm—allow people to fall into patterns of behavior.

Incomplete—Gestalt psychologists would say that the mind seeks closure.

Wonder—reveal things at a scale they hadn’t expected—much larger or smaller.

Awe—the dangerous wonder.
Fantasy—give players a space to imagine what could be.
Improbably—not impossible to find the unreal in the real.
Use interesting stories and intriguing characters.
Use audio to fill the world with unseen substance.
Always leave them wanting more.

Internal Experience

We are feeling machines. Our bodies and our minds work together to produce these feelings, and we experience the sensations. Players can be motivated by a desire to change their mood or just to experience new feelings. The feelings that a person desires, of course, depends on the situation, and your game will offer some set appropriate to the experience you want to create.

Excitement is a reaction to the new and the challenging. Our interest is captured by the sudden appearance of something fresh and unexpected. With a rush of adrenaline, the player is ready for action. It is an arousal, like fear and anger, but lacking the context of the dangerous or unjustified. Excitement mixes well with others, intensifying both pleasant and unpleasant emotions.

Where excitement can involve other emotions, relief always does. It is a feeling that can seem to course through the whole body in a large sigh or a gentle lift as a burden disappears. We experience relief when some larger emotional experience is removed. The relieved emotion can be immediate like the fear in a narrow escape of a car crash, or extended like worry over a job application. We can be relieved even after something positive, such as a hard-fought win. Relief can be subtler, following moods, like the escape from everyday concerns and stresses at the end of a workday.

5 A combination of focus and new emotions help offer relief.
6 Offer pleasant emotions with lower effort.
7 Change perspective.
8 Provide support and recognition.
9 Treat the players kindly and caringly.
10 Care for the players.
11 Remind them of their past.
12 Get their mind away from the real world, or at least their part in it.
13 Amuse them with interactions.

Players can enjoy the simpler structure that games can offer. Confident with their understanding of the game, they can enjoy the process of working through to the answer. Here the designer wants to create challenges that most players will overcome, if even by a little trial and error. In fact, the trial and error approach is quite pleasant, provided the player doesn’t have to spend too much time without making progress.

Social Experience

Until they went digital, a primary reason people played games was as a structured activity to do together. As video games shifted from arcades and entered our homes, they became stigmatized as “anti-social.” While that was always unfair, the Internet has now removed all doubt: games can be downright social.

Players do not need to be *collocated*—right next to each other—for social experiences. It is only necessary that the appropriate connection to others is made. Even a cleverly constructed high score system, such as *Geometry Wars 2*, can give players the feeling of being with friends as they try to best their score. *City of Heroes* is a social experience, with similarities

and differences to a room of friends playing *Wii Sports.*
Schadenfreude is a German word for the joy in the misfortune of a rival [Ekman03]. This is "You lose!" and it is an emotion present (whether our culture frowns on it or not) in every competitive game ever created. The more polite might suppress the grin or the laugh or the secret cheer, but it is still there. The less polite... well, schadenfreude is the fountain of gloat, and the longer the rivalry has existed, the more will flow.
In Yiddish, the word *naches* is the pleasure that is felt at the accomplishment of a child or mentee (person mentored) [Lazzaro04]. People will watch friends and others play, sharing in their successes.

- Create opportunities for competition.
- Create opportunities for cooperation.
- Allow players to display their skills.
- Allow players to display their individuality—"peacocking."
- Allow players to display their humor.
- Provide tools for communication and sharing.
- Create a connection between the player and meaningful others.
- Support the ability for players to metagame.
- Naches—pride in the child.
- Give people the opportunity to watch.

Tempo and Rhythm

Time is an entire dimension that you have to alter the game experience. It is an easy tool for building drama ("will they be alright?") or excitement ("I can't wait to get there!") or fear ("I'll never make it in time!") You can turn it into a resource that players have to manage like any other. In fact, time usually is *the* resource most players are contending with.
Whether intentional or not, your game will have a rhythm. The player will experience this through cycles of risk and reward or exploration and discovery. Think about what you would like that pace to be and whether you would like it to change—accelerate or decelerate.

Session Length

Frequently, we are concerned with the length of the game—overall, how long a game will take to complete. However, one of the most practical concerns about time is simply how long will it take for the player to have a meaningful experience that he can leave, feeling satisfied? We now use the term *session length* to indicate the anticipated time required for a single sitting.
A brief comment about the word "anticipated." We know that casual game players can actually play for hours at a time. But what is important is that when they are deciding to play, a choice is being made around perception of the future game session. They aren't sitting down with the expectation that two hours will pass unnoticed. Instead they innocently think, "Oh, I should take a break for five or ten minutes..."
Short session games take around 5–15 minutes to play, and most of the classic casual games fit into this range. *Mid-session games* are 15 minutes to an hour. *Long session* titles are those that start around 30 minutes and can run until self-neglect becomes a serious concern.
Casual gamers could have a complete experience in less than 10 minutes—from launch to play to exit. This became more obvious when consoles began offering downloadable games and developers began making hardcore games with short session lengths.
Short session games must be quick to enter and leave, with a minimum number of options to set or confirmations to acknowledge. For this reason, casual games usually have a profile-based save system so that players may quit at any time without losing progress.

- What is the shortest length of time it will take a player to sit down, play, accomplish something worthwhile, and leave the game safely?
- What is the shortest length of time to make a major step forward in the progression, such as moving to the next chapter?

The average of these two figures should give you something of a rough estimate of the player's operating assumptions when evaluating the length of time she will be playing.

Play Mechanics

Things won are done—joy's soul lies in the doing.
—William Shakespeare

When someone playing *Lego Star Wars* cracks apart a table, jogs in a circle to gather the coins that erupted from the rubble, and receives the value of those coins into her total account, they experience a mechanic. If she continues breaking objects and collecting coins, she might be able to unlock Greedo back at the cantina; that is a mechanic.

The previously described sequences—hitting the table to break it, collecting coins from rubble, buying Greedo—are all simple game mechanics; they represent a single transaction. If we bring our perspective a little further back, looking at all of the sequences together, we can see a compound game mechanic that describes a relationship between the player and the game's coin economy.

A play mechanic is formed when the player applies game actions to game elements. They are interactions that produce a meaningful result—it "matters" within the context of the game. Mechanics don't have to be critical enterprises, but they do need to serve the overall purpose of play in some way. In other words and in an abstract sense, play mechanics create feelings.

As a note, keep in mind one running thesis for all design:

If the feature does not improve the game, it should not exist in the game.

Player Arrangements

In narrative arts (literature, movies, etc.), various types of conflict, related to dramatic struggle, are categorized—man vs. man, man vs. nature, etc. In games, these narrative conflicts can still exist. At a far more specific level, games offer all sorts of other kinds of conflict such as different styles of fighting within a single game. So the word "conflict" can get overloaded quickly.

We will use *player arrangement* for those configurations of player conflict (Figure 2.1.12). In the book *Game Design Workshop*, Tracy Fullerton uses the term *interaction patterns* and adapts a scheme that offers a nice illustration of the various forms these game conflicts can take [Fullerton08].

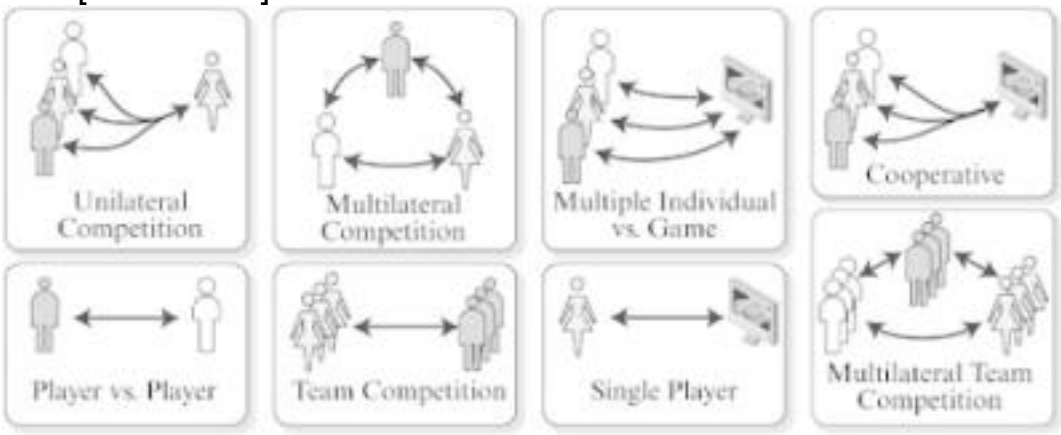

Figure 2.1.12: Player arrangements [Fullerton08].

The following are some typical player arrangements:

- Single player—player contends with the game system.
- Player vs. player—two players contend with each other.
- Multilateral competition—three or more players contend with each other.
- Team competition—two groups compete with each other.
- Multilateral team competition—three or more groups compete.
- Unilateral competition—two or more players compete with one player.
- Multiple individual vs. game—multiple players compete against the system.
- Cooperative—two or more cooperate against the game system.

Core Mechanics

In every game there is one or more *core mechanics*—distinctive and fundamental sequences of actions and results that players repeat throughout the game to advance. Not all repeatable mechanics are considered to be core, only those tied to reaching the overall goal or maintaining desired states, like survival.

In an action game, core mechanics might be based on movement and combat; in a farming game, tilling and sowing; a social networking game might have trading and stealing for core mechanics. You will want to explore core mechanics that immediately support your project's thematic, aesthetic, and audience goals. Discover the proper set of activities through team creative collaboration, prototyping, and user testing. Prove your ideas first and then find efficient ways to build them.

Local Emergent Gameplay

Local emergent gameplay (second-order mechanics) results from the combined use of other mechanics and systems. Most often, these are player inventions, arising from the creative use of properties and relationships among game elements. When second-order mechanics are simple exploitations of game systems, they can often be viewed negatively as exploits, even when they don't violate the game premise.

As a classic example of simple second-order mechanics: id Software's original *Quake* had two explosive weapons, a rocket launcher and a grenade launcher. In addition to the damage system injuring anyone near, the game's physics engine would add an outward force, moving characters away from the source of the blast. Players jumping over an explosion at the right moment would be propelled in the air, an order of magnitude higher than a standard jump. (When performed purposely with your own weapon, this has become known as *rocket jumping*.)

While emergent gameplay offers many interesting and exciting possibilities, not every game is suited to it. Local emergence usually requires that the game systems are operating somewhat beyond the immediate needs of the play mechanics. Often, systems are performing some amount of general-purpose simulation; they are running rules of the world that aren't necessarily relevant to the designed actions. For example, FPS games usually have some level of physics simulation that includes objects with masses and velocities. If there is also a combat system that calculates damage to a character when colliding with an object (where force = mass × acceleration), then nearly anything that can move in the world is a potential weapon. But a Texas Hold 'Em game would probably not have or benefit from such a system.

Local emergence is sometimes called *systemic gameplay*, which is a fair description of the way designers might go about creating it: use several small systems with the potential to interrelate in new ways [Dunniway08]. To support local emergence, game designers need to focus on designing coherent rules for the world and give players tools that operate on those principles (fire that injures, a field that burns, and a match).

Fighting

Combat is no stranger to video games. It used to be the primary mechanic in most games, but this has changed gradually over time, especially with the rise of online casual games. Even further, sites that cater to more traditional gamer audiences and sensibilities, like Kongregate, feature more and more games with new mechanics.

Fighting can be roughly classified by type: melee, ranged, and mounted. *Melee* is hand-to-hand combat with fists or hand-wielded weapons like swords and spears. *Ranged* combat involves weapons that shoot things over some distance and is often categorized as short, medium, and long. The distances involved with each range are unique for each game, but the general reason for the classification is one of balance. Mounted is being on top of another unit, like a horse or such, which often gives a combat advantage over unmounted units. Combat units frequently balance different attributes like speed and range to offer players a variety of ways to do the same thing: kill, kill, and kill.

Inventories and Collections

Inventories are a frequent consideration, coming up whenever a game allows players to hold something for later use. Inventories can be abstract interfaces (“things you have”) without much more explanation. They can also be deeply integrated into the game universe. At one end of the spectrum, you have a single power-up in *Mario Kart*; at the other end, you have an elaborate network of hangars distributed through the galaxy in *EVE Online*.

Just remember that, *any* time you are considering players owning something, you will need to also design a good and useful way for those things to be managed. By their nature, inventories—storing things where they cannot be seen at all times—will add complexity to your game. You should be asking questions to determine whether or not that complexity will add any value to the game.

How large will the inventory need to be? Can it be smaller?

How will objects be stored? Removed?

How will they be retrieved? Used?

Will they need to be organized? Sorted?

The difference between a collection and an inventory is really one of intent. When all of the items can be gathered but not used, you are usually dealing with a collection. Collections are much simpler to design because the interface considerations are small. The collection may need to be viewed, but usually the elements in the collection don’t need to be moved around or managed.

Collection has become one of the most popular mechanics across all games and demographics. Collection mechanics can even provide motivation for players to continue playing after the main game progression has been completed (as in *Lego Star Wars*).

Rewards (and Punishment)

I became a game designer to create joy.

—Anonymous

The mechanics you design should be rewarding in their own right; it should be fun to play. But outcomes need to produce an emotional component to be of significance to the players; they should fall somewhere between coveted and unwanted. This relative value is the meaning behind reward and (to a much lesser extent) punishment, and both share two approaches to one primary purpose. The purpose, of course, is to attach the players to your game; get them interested and keep them there. This is approached by informing the players and by encouraging them.

First, we know that successes are most meaningful when the player feels in control. But

control comes from feeling informed, from understanding what is good and valuable and what is not. Reward and punishment forms a language that game designers use to communicate and teach the players the relative values of rights and wrongs in the game's universe.
The more important the choice, the better the reward should be. If there are multiple ways or degrees of success, consider variable rewards that show it. In one mode of the racing game *Midnight Club: Los Angeles*, players deliver cars, with the payment contingent on the condition of the car. The message is clear: "It really is better if you don't run into everything."
Punishments can be an instructive tool, but be very careful in using them. Chances are very good that players will already understand that they have failed without needing to be reminded. However, careful use of punishment can intensify the player's excitement and pressure later in the game, which will result in more satisfying moments of relief. (If a few early deaths teach players the painful consequence of running from cover into the open, they will learn to sidle up to protection whenever they can.) When your game focuses on the experiences of mastery, punishment is a useful tool for increasing risk.
The second approach to using rewards is more general than the first; here, we simply want to offer players enjoyment. We want people to like themselves and our games. Players are our friends, and we should keep their feelings in mind. Encourage them when they are beginning, praising them to keep confidence and spirits up, especially when they might be struggling to learn. The player's offer to you is his time, so your game's offer should be to reward that time in a delicious assortment of rewards.
As players progress, keep them going with reward loops (gameplay loops) of tasks and rewards; they do something, they get something. These are rewards that are forecast, and the player has a good idea what to expect ahead. Each loop forms a cycle, which, according to industry research, is best kept under 15 minutes [Dunniway08]. Through these sequences come the game's rhythm and tempo.
Offer valuable rewards for valuable achievement.
Keep rewards coming in cycles averaging less than 15 minutes.
Use rewards to establish and emphasize the game's rhythm.
Allow players to chain rewards together, increasing the value.
Create rewards for each experience you want to support.
Use collections of small rewards that lead to huge rewards.
Jay Minn calls the reward loops the "potato chip loop." The game serves potato chips to players periodically, encouraging progress. The chip might be a reward for completing a stage, collecting an object, or some interesting feat. The chip may be bigger or smaller, but the most important thing is the rhythm between one chip and the next. Too few, and the player gets hungry, looking for something else to eat. Too many, and the player gets full and wants to take a break. The potato chip loop doesn't try to establish formal accounts; it's just an easily understood, communicated, and remembered device.

Puzzles

Puzzles are problems. There is, in game industries and studies, debate about whether or not puzzles are games (one gets the feeling there is not a lot of serious work to do). Certainly, puzzles are an essential part of many games, enough that a whole genre of games exists that uses little else but puzzles. One delightful game, *Professor Layton and the Curious Village* has even wrapped an adventure around solving a wide, huge collection of puzzles that aren't (gasp!) even themed in Layton's world! The people of the village just like puzzles.
Puzzles have a solution, a right answer [Kim09]. It is this aspect, more than any other, which makes puzzles a challenge to solve and a challenge to design. The difficulty of a particular

style of puzzle is dependent upon the skills and knowledge of the puzzle solver. A clear understanding of your target audience can help, but that “most people get it” is not much comfort to the poor sucker left out.

Think of puzzle design like game design. You want to offer your players both challenge and control. Help them to understand the goal, that it is solvable, and how they might go about finding that solution. They should be guessing at the answer rather than the point!

A puzzle that displays progress encourages players to keep working at it. Whenever you see a player sitting still, staring into her screen, you are looking at a mounting risk of frustration. It’s better to have distinct failure states and let the player try again immediately. Perhaps your puzzle can, through this trial and error, reveal something about its solution.

The early years of adventure games should serve as an object lesson: any puzzle that is mandatory to the game’s main progression needs to be treated with the utmost care. Expect that players will get stuck, and plan to offer hints or allow them to bypass challenging puzzles. You may also want to have a method of simply offering the answer. As Jesse Schell notes, the “Aha!” doesn’t come from finding the answer, it comes from seeing it. Finding the answer just adds fiero to fire. Victory!

Information

Players set goals and intentions (make decisions) based on subjective understandings. They make choices according to what they believe about the game at any given point. In turn, those beliefs are based on the information they’ve received. Experienced designers know that information is a resource with many uses and applications. To put it to work, they need a few answers.

14 What kind of information is this?
15 What is the value of this information to players?
16 What are the benefits of sharing it with them?
17 What are the benefits of keeping it from them?

Open information is shared freely, while *hidden information* is kept from one or more players. Most games have a mixture of the two, with specific designs determining what is open and what is hidden. So which and when should you use either? Usually, games have some combination of both, but like most of the wishy-washy answers you’re getting around here, it depends on what your game needs.

Open information reduces uncertainty but does not necessarily eliminate it. For games with broad possibility spaces (chess, *Go*, etc.), the effects can be minimal as most of their uncertainty results from the spontaneous choice of an unpredictable opponent with many options. Games with open information typically include chance and player choice.

For example, in the board game *Carcassonne* (Figure 2.1.13), players take turns drawing tiles at random and then aligning them with other tiles already in play. After a tile is placed, the player may place one of her seven followers on it to earn points. There are fourteen types of tile, each with a particular layout, and the count of each varies—some as many as nine, others as few as one. Each time a tile is drawn, it is shown to the group who are supposed to help find places where it can go. Because the board changes shape during the game, and because scoring involves careful placement of followers, the open information contributes to a friendly experience without ruining the uncertainty of the outcome. There are enough possibilities that hiding information isn’t necessary.

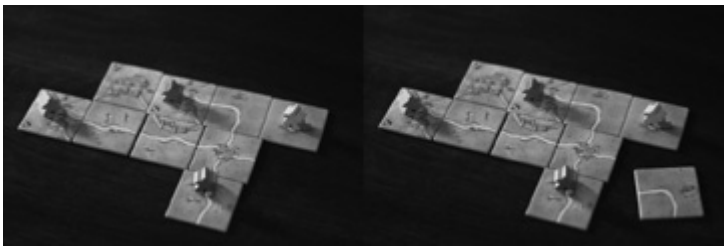

Figure 2.1.13: Carcassonne (courtesy Hans im Glück).

Hidden information builds uncertainty, letting players keep secrets while they strategize, and it provides opportunities for bluffing and misdirection—a key feature in many of the most popular card games. It creates opportunity for inference, where players look at what information is known and imagine likely outcomes, building anticipation and mystery. Without hidden information, poker would be turned into a five-card version of war (see Figure 2.1.14). But, again, we see a blending of hidden and open information in Texas Hold 'Em, which is why it is currently the most popular poker game in casinos.

Figure 2.1.14: Hidden information is a critical element to particular card games like poker and Texas Hold' Em.

Video games with local multiplayers (playing on the same screen) are limited in the amount of hidden information they can feature. Sports games like the *Madden* series compensate by allowing players to input play calls without explicitly showing the selected choice on the UI.

Enhancing Uncertainty

When you are working on your game, if you discover that things are too predictable, resist the *immediate* temptation of quick fixes like "adding randomness." Before adding chaos, look to player performance and strategy first. Hiding a little information (for example a fog of war) is often all that you will need to get the excitement back. A deterministic system can still have plenty of mystery and uncertainty. Just remember that it is seldom fun for the player if randomness plays a key role in a player's success. Random fates tend to frustrate players.

If you do decide to introduce randomness, look for parts of your game systems where the unsystematic behavior won't be too obvious or out of place. For example, you might hide this a bit by using a random outcome in a supporting role; somewhere in your system that will subtly and quietly affect the overall result.

Choices and Outcomes

Choice—"the act of choosing" [HMCo00]—lies close to the root of how we understand our game experiences. A quote by designer Sid Meier: "A game is a series of interesting choices"—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, which lead to outcomes. Choice really is at the heart, and we can describe the landscape of potential

choice as a *possibility space*. This space represents all possible actions as an area; *wide* indicates many possible choices; *small* indicates very few.
The *consequence*, or *weight*, of a choice notes the significance of the outcome. With greater effect comes greater weight; the more a choice will change the game, the heavier it is. This weight is one of the most important factors in designing and balancing choices. Choices that your players must deal with should involve their desire to achieve their goals. A well-designed choice will often feature both desirable and undesirable effects [Fullerton08]. Insignificant or irrelevant decisions are usually annoying.
In keeping players engaged, you try to enable an experience that oscillates within an ever-rising balance of challenge and ability or Flow. 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.

Keep It Simpler

Play mechanics are mirrors of the game systems (funhouse mirrors anyway), and because they are systems themselves, can quickly grow complicated. Keep a watchful eye over them. Careless complexity causes us to spend more time trying to get mechanics under control than we ever spent inventing them in the first place. When in doubt, test, limit, reduce, and scope!
Each option that you offer players comes at a cost. When they make a choice, they are getting something they want, but it often comes at the cost of other things that they *might* have wanted. Studies show that people tend to weigh their satisfaction with a past decision against the imagined results of the options they passed on.
Provide frequent but restricted choices. Allow players to pick and choose, here and there. And reward them constantly, and maybe even reassure them that they made the right decision to play your game.

Actions

I have always thought the actions of men the best interpreters of their thoughts.
—John Locke
When players step out of their pirate freighter, on to the docking platform of a titanic interstellar trading station, they have taken an action within a game world. When they move the right thumbstick on their controller to turn the camera, looking around the platform, they have also taken an action.
Action has two meanings, and they both involve “what the player does.” One regards the events in the *real world*, and the other events in the *game world.* We can understand actions as primary elements of the mechanics [Cousins04]. These can be things such as “move forward” or “jump” or “select item,” depending on their context within the game.
Normally, designers are focused on the second case: actions happening in the game world. But you will be encouraged not to forget about the real world. When you are playtesting, don’t just look at what your users are doing in the game, or what they are reporting they are feeling. Take your eyes down and watch their bodies. Look at their fingers as they use the controller or the mouse. Look for actions that you think might be fatiguing or tedious. Sometimes, they are difficult to see if you are only looking into the screen.
Think carefully about the way that your game will connect with the player physically. How will they do what they need to? Apply these concerns to the interface design issues that follow.

Goals

Goals (or *incentives*) are subjective notions directing our actions toward outcomes [LeDoux02]. A goal is "what the player *wants* to do." All games involve goals, even those few lacking clear high-level objectives (for example, software toys). Goals are the personal property of players, not designers.

During a game, the player's goals are usually aligned with game objectives. But the disconnect between those goals and objectives reminds us that incentives must be communicated clearly.

Kobe Bryant Doesn't Dribble

Imagine your friend is visiting, bringing a new game. It's a basketball game, even though he knows it's not your thing. "You're a game designer... you're going to love this!" he promises and tells you that it's *just* like being in the NBA. He arrives, sets up the game, and hands you a controller. "You're the Lakers. Get ready to *be* Kobe!" The game starts, and the tip is to Bryant. You have the ball, so you move... instantly the whistle blows. Traveling? "Yeah of course... don't forget to *dribble*!" Ten minutes later, and you are wondering if real basketball players ever injure their thumbs like you have. You might have to be on injured reserve until it heals.

Few of us know what it is really like to be a professional basketball player on the court at game time. But we can make a good guess that passing and shooting and picking up the open teammate are things athletes think about; they're a little beyond thinking about bouncing the ball while walking. Dribbling isn't part of what we imagine the NBA experience to be, and therefore it doesn't appear as a feature in games that represent pro ball.

Now imagine you are designing a basketball game of your own. But this time the setting is a kindergarten with children playing using soft rubber balls and big low hoops. Your partner shows you a diagram of the controls (see Figure 2.1.15). Do you shake your head and say, "Kobe Bryant doesn't dribble?" Or do you think about how much fun it will be for players to feel like a little kid loping around the playground, with a giant slapping dribble?

Figure 2.1.15: Playtime or playground?

Real World to Game World

When players hold a controller or click a button, they are trying to tell the game that there is something they want to do. "I want to go here!" "I want to eat that!" For them to make successful game actions, they will need to execute successful actions in the real world.

In his 1988 book *The Psychology of Everyday Things*, Donald Norman introduced a wide audience to the subject of *usability*—a field that attempts to improve object and interface designs, making them easier to use and more effective [Norman88]. In the book, he shares a seven-stage model of user action, a tool for designers of all kinds to serve as a checklist for preventing and troubleshooting situations where players who are running into trouble perform actions in the real world to perform actions in the game world.

Figure 2.1.16 shows the psychology of performing a task.

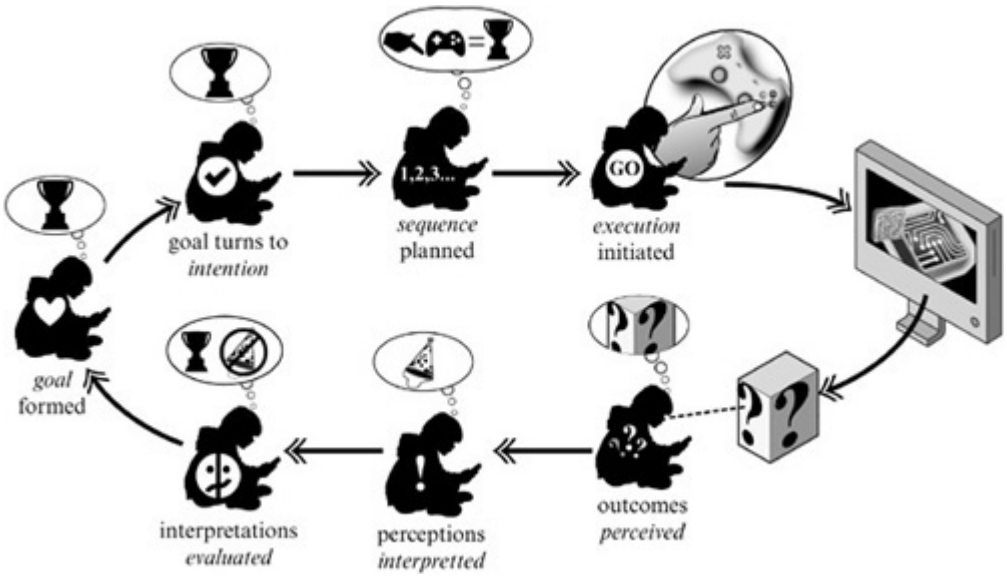

Figure 2.1.16: The seven stages of action [Norman88].

- **The player imagines a *goal*—an attractive possibility—** "I would like that."
- **An intention to reach the goal is formed—** "I am going to do this..."
- **A sequence of actions are planned—** "I need to do this, then this, and this..."
- **The player executes that sequence—** "Move forward, pick up object..."
- **The player perceives the outcome—** "What's this I see?"
- **Perceptions are interpreted—** "Hey it's a thing!"
- **Outcome is evaluated—** "No, that's not what I want."

The player forms a *goal* in his mind. Goals don't have to be grand noticeable things, just something wanted. During the *execution* phase, the goal is transformed into an *intention* to act; the player has decided to go for the goal. These intentions are put into an *action sequence* where the necessary steps are planned out, in the order that the actions will be performed. The *execution* is the actual act of putting the body into motion, manipulating the controller. ("I want to turn left." "OK, here it goes.")

At this point, the action has been "done." The player is now waiting for the system to respond to the action. Keep in mind, this whole sequence can happen in a moment. The player may not have had a conscious thought this whole time.

Presuming that the system did do something, the player would *perceive* the result, noticing its existence. Then the perception is *interpreted* or recognized. Finally, the player can *evaluate* the result against the initial goal.

There is a remarkable amount of time spent fantasizing as a method of understanding the player's viewpoint, choices, and strategies. Designers imagine playing the game and visualizing the behaviors of game systems. These imaginary agents move through game spaces, accepting challenges or pursuing mysteries, engaging activities and reacting to them. These thought experiments are a cheap and quick way to test ideas. "What would I do?"

The limit to these exercises, of course, is that the player is not you. She will not understand the game as you do because she has never seen *inside* the game systems. Player strategies must rely on their unique but imperfect understanding of the game. While working, don't be afraid to return to a very basic question: "Why would the player know this?"

Interface

The interface is a network of components allowing the player to interact with the game—a communication system spanning the physical gaps between both ends of our model. Every

unit of information that the game will need to convey comes through this system and every action the player intends will need to be directed through this system. While a game cannot be made great through its interface alone, as the visible and audible expression of a game, a bad interface can be more than enough to discourage players before even laying a hand on it. Let's step through the interface network shown in Figure 2.1.17. We can begin almost anywhere, but let's start with the physical interface, the display. A graphical element displays some information. The player sees and interprets the meaning. (His meter is nearly full.) The player creates a goal and an intention, sequences his actions, and executes by pressing a button on the controls. This becomes a message sent to the game, which is processed according to the operational rules of the system. Game states are changing and the results update on the virtual interface. A new signal is produced and sent to the physical interface, which updates the graphical element. This is the continual exchange between the player and the game through the interface.

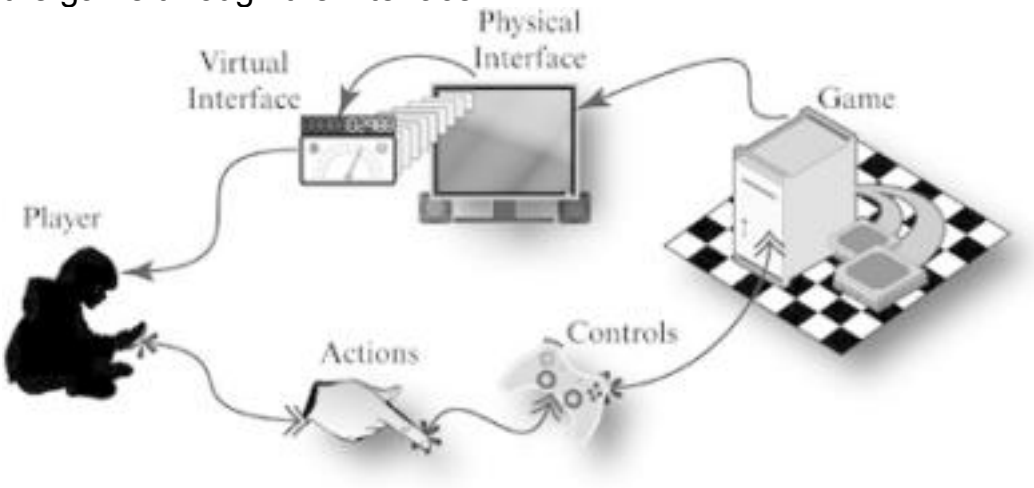

Figure 2.1.17: Connecting the player to the game in a loop.

Controls

Simplicity, simplicity, simplicity!
—Henry D. Thoreau

Controls are the systems of input that convert physical signals from the player (touch, sound, etc.) and convert that information to digital signals that the game system can properly interpret. These are things like keyboards and mice, console controllers, touch screens, and everything right down to a pair of *Samba de Amigo* maracas (Ole!).

The task for a game designer is to take these physical objects and map game actions to it (Figure 2.1.18). More specifically, you will need to consider the entire loop of interaction and map the player's intentions to execute game actions through real-world actions using the controls.

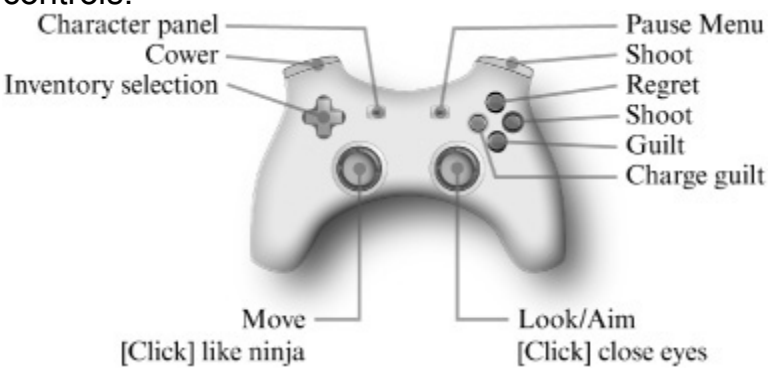

Figure 2.1.18: Two thumbs, four placements.

The ability for players to express themselves, to do what they want, must be balanced against the intuitiveness of the control design. The more things there are to do in your game, the more work you will need to devote to designing controls. Don't simply look at the buttons and sticks you have available and fill them up. Try to carefully build clear, understandable mappings between actions and controls. You should only change mappings when some significant event has taken place in the game, such as entering or exiting a car in *Grand Theft Auto IV.*

Most game types have standardized control schemes that are accepted as the norm. If you are designing a game that will appear similar to something well known and established, start there. Don't fear addressing the individual needs of your game with a unique control method, but be considerate of those players who will expect things to work in a standard way.

It's rare, but you could have an innovation in controls that will set your game apart. For example, boxing games had a standard control for throwing a punch set in 1984 with the *Punch-Out!* arcade game: press a button. For 20 years, every boxing game followed suit. Then, *Fight Night 2004* introduced something truly new (see Figure 2.1.19) to give players a better feel for throwing punches. Rather than just mapping a punch to a button press, they mapped it to the right stick and in that change created a visceral experience: boxing with fists thrown in jabs, hooks, and uppercuts.

Figure 2.1.19: Analog punching à la Fight Night 2004.

Feedback

By and by is easily said.
—William Shakespeare, Hamlet

While ultimately of the same stuff, helping the player to understand, let's discuss feedback from the perspective of controls and game system separately.

Control feedback is the information the player gets about controls and the things that she is trying to do with them. When an action is committed, feedback informs the player of the result or (just as importantly) the *non-result*.

A simple example of feedback's proper role in an interface is the mouse and pointer system used in nearly every graphical operating system (Mac, Windows, etc.). As the user moves the mouse, the pointer moves. Up, right, diagonally, fast, slow, or whatever, the cursor is mapped *directly* and *immediately* to the mouse. Every movement provides feedback, and while clicking is a little tougher to get, it doesn't take new users very long to understand how moving the mouse is supposed to work. It is such a powerful system of feedback, and one of the ways we test to see if our computer is responsive or has crashed is to shake the mouse and see if something happens.

Your game is the ultimate boss and gatekeeper. When a player performs an action, there are two aspects to the request. The player executes a command and implicitly asks for an acknowledgment, such as: "I want to do this. Is this OK?" The player wants to know if she has been heard and if she is allowed. The interface should respond in one of two ways: "Your command has been executed," or "You may not execute that command."

When player input is received, the system will either accept it or reject it. If this is in a menu or other GUI device, a response should be made immediately. Most of the time these responses are so subtle we fail to recognize them, like the brief blink of a button with a warm click. But when this feedback is missing, we notice the uncertainty, disagreement, and low standards. Poor feedback can make games feel “cheap.”

Action games need to be responsive. Spend some time reading reviews, and you will notice that nobody enjoys issuing a command that isn’t executed with urgency. Any delay in response greater than a tenth of a second risks confusing the player and making him wonder what is wrong [Schell08].

Game feedback is the information given to the player about the state of the game systems and their place in it. This is where your needs and designs for hidden and open information matter. If you’ve already worked through the issues concerning open and hidden information, you’re ready to use that knowledge.

Ask the following questions about your game state information:

- What should the player never know?
- What should the player always know?
- What should the player sometimes know?
- When should he know it?
- How should he know it?
- Will he need to be alerted or reminded?

For example, a strategy game might keep the positions of the enemy a mystery until (...sometimes...) contact between friendly and hostile units has been made (When...). The enemy positions show up in the main view and on the map (How...). The player receives a message when there is fighting between units (...alerted...).

Constraints and Context Sensitivity

Constraints are limits on actions. While it may sound counter-intuitive, by limiting the things that a player can do, at any one time, you are helping make the player’s goals clearer. Too many available choices lead to confusion.

Context-sensitive controls allow players to do specific actions in specific circumstances. Most of today’s 3D action-adventure games place players into rich interactive worlds where context-sensitive controls are required to manage the number of different interactions. This is done by relating the player’s current position and state (in flying mode, on offense, etc.) to the appropriate actions.

Viewpoint

Game worlds are viewed on flat displays, like movies, and the camera is a metaphor for our view. The virtual position and orientation of that camera is the *viewpoint* or *presentation*.

Orthogonal Views

Orthogonal presentations place the camera perpendicular to the playing field, either in a top-down or side-view arrangement (see Figure 2.1.20). These were standard views for arcade games for almost two decades. With the massive popularity of 3D engines, these views had often fallen into secondary, supporting roles, like maps. But, in recent years, with the proliferation of short-session games online, orthogonal views have been making a comeback. Bellwether titles like *Desktop Tower Defense* (Flash) and *Geometry Wars* (XBLA) have inspired a new wave of games that have returned to classic orthogonal presentations.

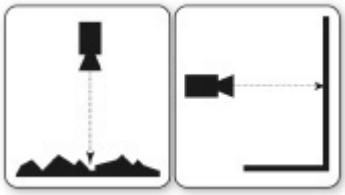

Figure 2.1.20: The virtual camera in overhead and side orthogonal views.

First and Third Person

First-person presentation hardly needs an introduction these days. An unequalled tool for many experiences, it still has some limitations. Seeing and hearing are not the only senses we have. One that is easy to forget is *proprioception*—feeling our body position—it is how we know what *we* are doing at any moment in the real world. It is the combination of this sense, with our others, that puts our real-world selves into something closer to third person than you might think. We are used to sensing what we "look" like through proprioception.

With invisible bodies, so many actions and interactions with the environment are unsatisfying—melee, using furniture, and so on. Even navigation is not trouble free. Players are easily stuck against things in the game world outside of view (a constant consideration for level designers).

The second problem is that, because the main character can't be seen in first person, players can't build real empathy for them; players forget about the character quickly and place themselves in the game instead. So if your game demands that players really feel like someone else, it will be a hard sell.

But, if your goal is to stick the player to the game world, first-person presentation is almost unparalleled. The position is not only at the level of the avatar's eyes, but the center of the screen works well as a natural aiming point.

Third-person perspectives place the camera outside of the body, presenting the avatar to the player more clearly. It becomes easier for players to identify, empathize, and understand the character because they can see it; it isn't just a mask they have on. But the challenge with third-person presentations always involves getting the camera and movement systems to work well.

Graphical Interface

Graphical user interfaces (GUIs) convey information to the player in any number of various abstractions. Get in the habit of listing information that the player might need to know. You can start with the information you were considering hiding or revealing. Go ahead and err on the side of listing too much. Then review your lists and prioritize the items in order of importance. Do the same thing for menu operations. Create lists or mind maps, as shown in Figure 2.1.21.

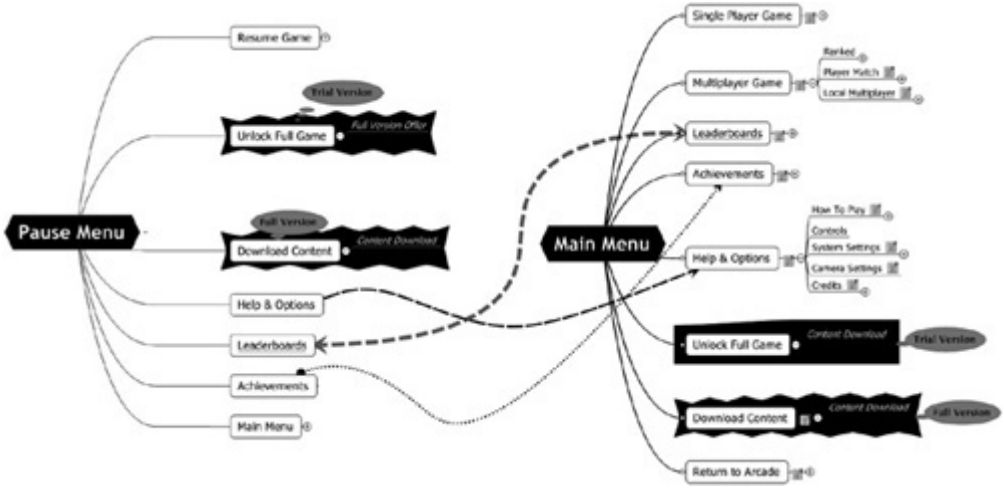

Figure 2.1.21: A mind map for a menu system.

Once you have determined what the GUI needs to do, use sketches to organize those ideas into an arrangement. Think about how the player would interact with all these elements. Take these notes and work with an interface artist/designer and try to turn simple implementations into prototypes. This stage can be difficult if you've created elaborate ideas about the way the GUI *must* work. There are a limited number of ways for people to interact with most GUI configurations and UI designers have often worked through many of them.

As your prototypes and work begin to get more refined, the GUI is taking shape. Make sure to fit the controls into the game's theme and setting (as shown in Figure 2.1.22). Anything that a player will have to see or use should always support the overall experience, even if it's just a button.

Figure 2.1.22: Embody theme and function.

Audio Effects

From environment to interface, games present visible *and* audible experiences. Recorded effects and voices bring games to life. But it is easy to overlook the importance of strong audio cues in your interface. Audio is that underappreciated tool that gets players through menus and options with a minimum of confusion or need for a manual.

Beeps, pops, clicks, and other abstractions communicate to users when their controls have made an appropriate choice such as a selection, or an inappropriate one, like trying to use a deactivated button.

Your interface should offer a small vocabulary of audio effects, standardized sounds for things like selections, advancing (start), return (back), and errors in input as well as warnings and confirmations.

As a general rule of thumb, if an element of the interface (button, slider, window) produces a visible reaction to player input (pressing, closing), there should be an accompanying sound effect to reinforce the action.

Designers will often be responsible for creating a sound event list that maps individual sounds

to various game or interface events. Figure 2.1.23 shows a hypothetical event map, where different scoring events trigger progressively “bigger” sound rewards, reinforcing the player’s successes.

Scoring Feature	SFX						
GAMEPLAY Cities		1 Tile	2 Tile	3 Tile	4 Tile	5 Tile	9 Tile
	Golf Cheer		X	X			
	Crowd Cheer				X	X	X
	Fireworks (#)			X (1)	X (2)	X (3)	X (7)
Roads		1 Tile	2 Tile	3 Tile	4 Tile	5 Tile	9 Tile
	Dust		X	X	X	X	X
	Bricks & Stones			X	X	X	X
	Horse & Cart				X	X	X
	Horse Gallop				X	X	X
Cloister		1 Tile	2 Tile	3 Tile	4 Tile	5 Tile	9 Tile
	Divine Light						X
	Doves						X
	Harps						X

Figure 2.1.23: Sound event list.

It should be noted that there are diminishing returns to adding new sounds once a critical mass has been reached. Sounds are just like any other language, and there comes a point where having more words won’t help your communication.

Game Systems

Joy is not in things, it is in us.
—Richard Wagner

Systems are organizations of related elements that work together to produce a result.

All systems can be described in three aspects:

- **Elements (objects)—** multiple parts form a system.
- **Interconnections (relationships)—** the elements influence each other.
- **Function (purpose)—** what the system does.

Systems are composed of elements. There are no real limits to what an element can be—large, small, simple, complex, physical, mental, etc. System elements are often other systems too.

Figure 2.1.24 shows a ring, a line, and a weight. In this arrangement, these are objects and the objects are collected together, but they do not represent a system. Being close together doesn’t count; the objects need to be connected. Elements must have relationships to each other so that they can create behaviors. Until they operate together to do something as a whole, we are not looking at a system.

Figure 2.1.24: Three unrelated elements.

In Figure 2.1.25, we create the first relationship by attaching the ring to the line. This creates an interconnection by physically attaching the two elements; pull one part and the other will follow. With just that step, a simple system that could lift or pull is created. Attach a weight to the free end of the line, and the weight can be lifted by pulling on the ring. (Or the weight, with the line and ring, can become part of another system for crushing or smashing.) But this

system isn't satisfying yet; the weight can't be lifted very high, and it takes 100 units of force to lift 100 units of weight.

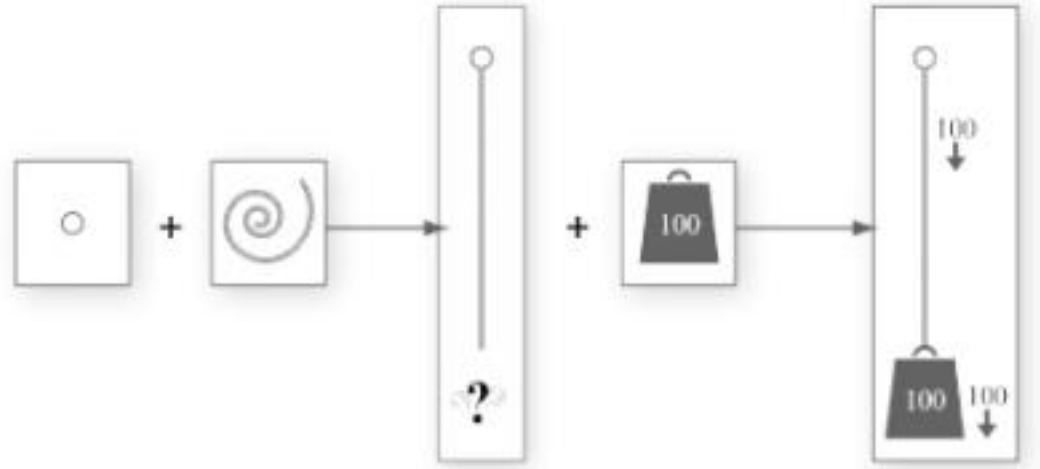

Figure 2.1.25: Interconnected elements arranged in a simple system.
The height of the lift can be improved if we add another system as an element in our system. A pulley changes the direction of applied force and now the weight can be lifted above the operator (Figure 2.1.26). To reduce the force needed to operate, we add an anchor and another pulley and the behavior of the system changes again.

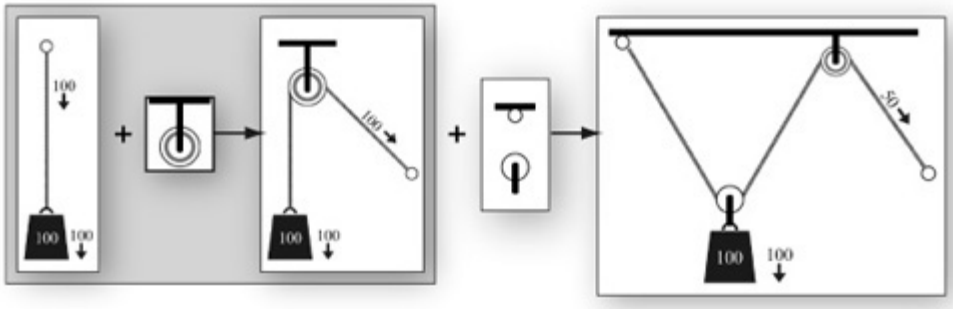

Figure 2.1.26: Adding elements doesn't always have the same effect.
If we keep adding objects, behavior continues to change. But there are two reasons to be careful when adding complexity to a system. First, continuing to add elements seldom keeps improving the ability of the system to achieve the same goal. Second, the more things that are in a system, the harder it is to predict the actual behavior.
Note that *function* and *purpose* both refer to what the system actually does rather than what it is "supposed" to do; the designer's intent is something else entirely.
System designs usually begin at a high level, starting with the intended play mechanic. Each step along the way, things are broken down into more specific details, refining your questions and answers. When at all possible, *build* your systems as prototypes and test them frequently with anybody you can find.

Dynamics of Systems

Game dynamics result from continued interactions between the players and the game system. Because the behavior of a system is dependent on its unique structure, it can be hard to determine just how it will operate. As systems get larger, involving more elements, this challenge can become immense.
One tool is known as *systems thinking* and was begun in 1956 at MIT by Dr. Jay Forrester to study how systems change over time. Systems thinking is a model for understanding general

behaviors we can find in systems. For the game designer, it can explain why some parts of our game that are *supposed* to do one thing are, instead, doing something else.

System dynamics describe everything in two aspects (Figure 2.1.27). *Stocks* (*levels*) are a stored amount of something—gold, enemies, etc. *Flows* (*rates*) are changes in the level of that amount; *inflows* are increases, and *outflows* are decreases. Game designers will often call these "faucets" and "drains." (The clouds on either end of a model indicate that where the stuff comes from is unimportant to the model.) Arrows show the direction of the flow, into or away from the stock. The handles over inflow and outflow indicate where the rates of change are controlled.

Figure 2.1.27: Stocks and flows.

Figure 2.1.28 shows the same basic model as it might be applied in game design. The character has a health that is lowered by damage and raised by healing.

Figure 2.1.28: A basic combat system.

But not everything needs to have both inflows and outflows. The two models in Figure 2.1.29 show systems that only fill or drain. Sometimes you have a drain without any faucets; it just depends on what you want the system to do.

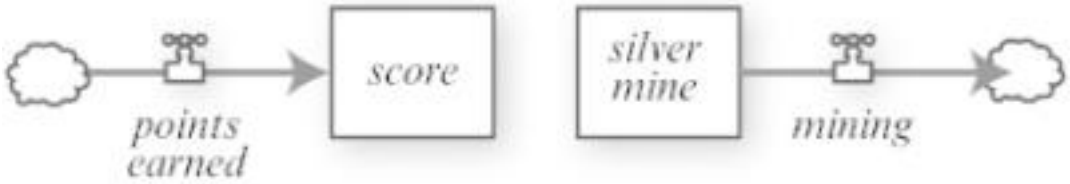

Figure 2.1.29: Earning points and mining a limited resource.

Game systems often have multiple inflows and outflows to a given stock (Figure 2.1.30). When you don't need to show connections to other systems, generalize the increases and decreases with single flows and show what influences those rates.

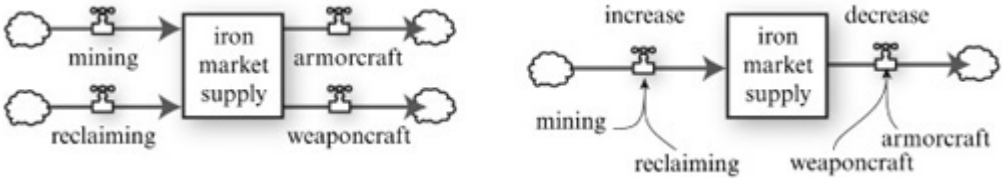

Figure 2.1.30: Multiple inflows and outflows.

Stock and flow diagrams are fine for sketching structure, but we also need a way to view behavior over time. For this, basic graphs are fine; they're easy to read and make. Let's go back to the silver mine we first saw in Figure 2.1.29 and see what happens if our player assigns a worker to mine silver at a rate of 400 units per minute (Figure 2.1.31). The mine starts with 8,000 units, and the worker keeps at it for 20 minutes when the last unit is taken.

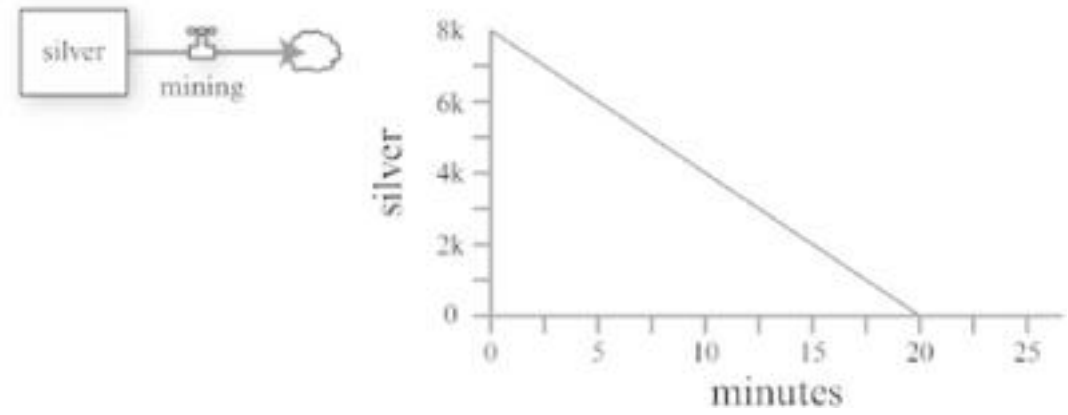

Figure 2.1.31: Mining silver over time.

Feedback Loops

The systems we have been looking at have all been static. While stocks have risen or fallen, it has all happened at a constant rate. Game systems don't get really exciting until they start changing. If every fight was just a slow slog to the death, even the violence wouldn't help. So we need to put a hand or two onto those valves to get things really changing.

Feedback is what happens when changes in a stock alter the rate of the flow in or out of the same stock [Meadows08]. Feedback is part of a tool for controlling system behavior; the name of that tool is called a *feedback loop* (see Figure 2.1.32). The arrow point from the stock to the controller of the flow.

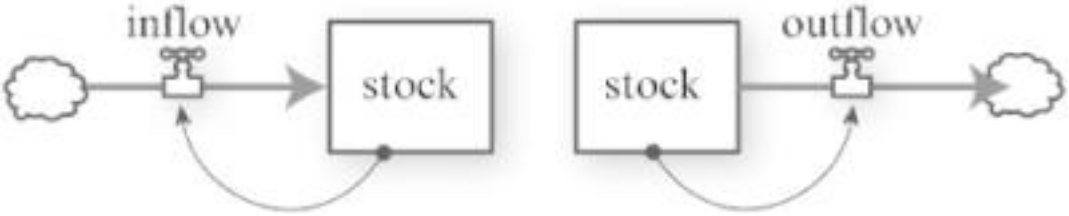

Figure 2.1.32: Feedback loops diagrammed.

There are two basic types (Figure 2.1.33): *balancing loops* (*negative feedback*) try to keep the stock at a certain level; *reinforcing loops* (*positive feedback*) produce more change in the same direction, either an increase or decrease. ("Balancing" and "reinforcing" are easier to understand than "negative" and "positive" feedback because negative feedback is usually desirable and positive feedback will usually break a game. Not a particularly positive effect.)

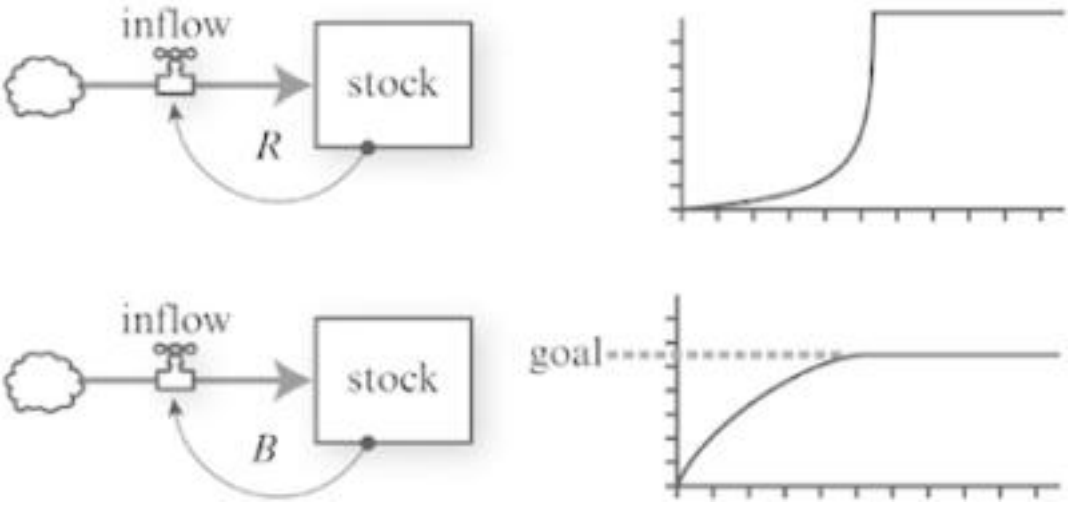

Figure 2.1.33: Comparing reinforcing and balancing feedback.

Figure 2.1.34 shows the balancing loops in a "rubber banding" system. To keep a racing

game exciting, the designers want the racers to stay closer together (a typical trick). So balancing feedback loops—the *B*s—slow a car that gets too far ahead and speed up a car that is lagging behind. Balancing loops try to move a stock toward a goal.

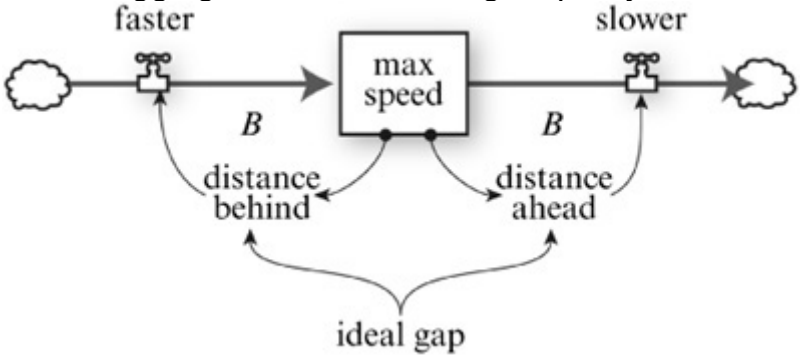

Figure 2.1.34: Balancing feedback in a racing game for a car.

Reinforcing loops, on the other hand, cause runaway behavior and havoc. Look at the imaginary MMORPG character level system in Figure 2.1.35. As the player defeats enemies, she increases her level, which improves her ability to defeat enemies. The *R* shows the reinforcing loop that related her current level with the rate of combat victories. On the right, you see how the situation would get rapidly out of hand.

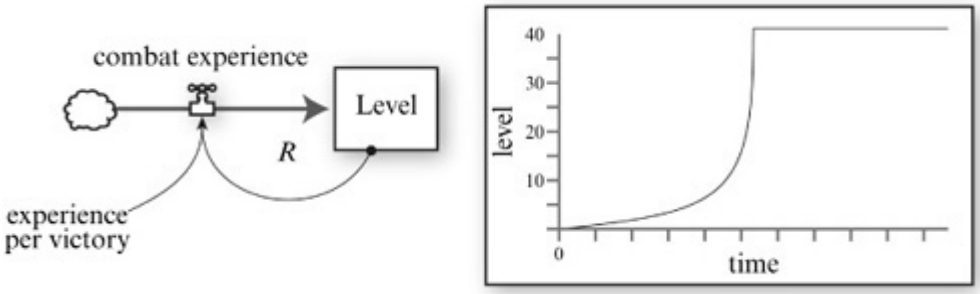

Figure 2.1.35: Leveling system run amok.

Like with many MMORPGs, we want the players of our fictional game to advance quickly at first, and then have to progress through levels slowly as they near the endgame. Our system in Figure 2.1.35 needed to be clamped down, and Figure 2.1.36 shows this clamping. First, the new figure added a progressive experience requirement for each new level; players need more and more XP for each subsequent level. Then, just to be sure, we added a modifier to the experience earned for defeating monsters to prevent players from simply lighting anthills on fire.

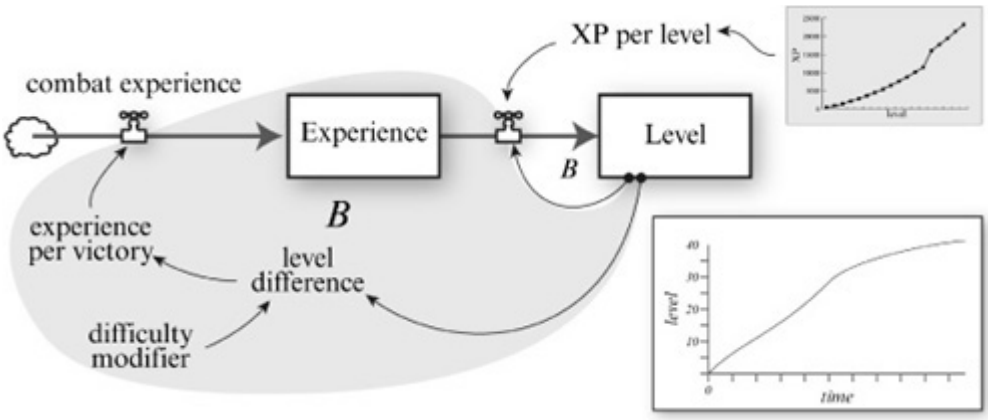

Figure 2.1.36: A normal approach to leveling.

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

Balancing loops stabilize the game.
Reinforcing loops destabilize the game.
Balancing loops forgive the loser.
Reinforcing loops reward the winner.
Balancing loops can prolong the game.
Reinforcing loops can end the game.
Reinforcing loops magnify early successes.
Balancing loops magnify late successes.

Simulation versus Emulation

Games are models. They are interactive representations of something like driving, city planning, war, baseball, etc. Like any model, we make choices about what to include and what to ignore—we abstract. In designing game systems, you will not only choose *what* things to represent but *how* to represent those things. These choices form a large part of the art and work of game system design.

Simulation is modeling with an emphasis on imitating structure first and behavior second. *Emulation* is modeling with an emphasis on approximating the behavior and without real regard to the structure of the thing being modeled. Simulation is a model of systems, where the behavior is a result of the model system; emulation simply models the behavior.

Simulations model systems by also modeling the elements of that system. For example, in complex flight simulators like *X-Plane*, a plane flies because the simulator calculates the effects of air flowing over its wing. If the shape and the size of the wing are not right, the plane would not get off the ground. In emulation, the plane would fly because it would be assigned a behavior: “planes fly.” There would be no deeper system, it would just be so.

Emulation is a tool for abstraction and all games use it, even the most hardcore simulator. A racing game may simulate much of a car’s engine—the mixture of fuel, the efficiency of the exhaust system, but there comes a time when emulation steps in. For example, the internal engine operation (pistons and valves) would be emulated.

Simulation tends to be more expensive because it requires more calculation than emulation. So you will choose when and when not to use it based upon the need. Generally, all else being equal, emulation is usually the correct choice. Even when all else isn’t equal, emulation usually has the upper hand. Simulation is more fragile and harder to tune. When an emulated behavior is broken, you can just change the behavior; simulations require looking at structure and properties and looking at causes of behavior.

Simulation is very effective at making more natural-seeming behaviors and can make complex systems easier to design and manage because rules are defined generally for the world. When agents in the world behave according to these rules, it can often feel natural and immersive. The entire *Sims* franchise has been built based on the satisfying behaviors resulting from simulation.

Nevertheless, most game systems are emulations. Not only is it usually easier to design behavior but also to play with.

Variable Difficulty

Challenge can be difficult to gauge, as you will find once you begin any real playtesting. Even if the designer follows all of the rules and gets all of the feedback and processes in all of the right ways, she still can’t hope to make a game that fits everyone’s tastes and abilities. But it is less tragic to us that our game might not be the right *kind* of game for a player than if someone failed to enjoy it because they were frustrated.

Offering players the ability to adjust the overall difficulty can help lessen the burden and allow

them to enjoy the game more fully.
Some factors that can be adjusted:
Adjust time—speed or slow game times.
Opponent populations—adjust the number/rate of enemies.
Opponent characteristics—slow, weaken, or limit the AI.
Augment resource income—increase the rate and volume.
Power-up distribution—populations, rates, and value.
Dynamic difficulty is a procedural approach where game systems will monitor player performance, attempting to regulate challenges closer to their current abilities. The earlier example of rubber banding in racing games is a classic example of a simple dynamic difficulty system.
If balance and challenge are important for your game and you believe that variable difficulty is called for, make a list of the elements that contribute to the game's challenge. Because you want the player experience to still hold a feeling of tension (otherwise this system is probably more expensive than simply tuning your game to be easier), try looking for elements that are the least visible yet still offer enough leverage to be effective. Prioritize the tunable features and begin creating your difficulty templates.
Contrary to what you might expect, playtesting and tuning times usually increase with variable difficulty systems. You will want to tune each level relative to the others, and you will want testers who represent likely players of those difficulties.

Probability

Probability is a mathematical tool for understanding the unpredictable; it describes the likeliness that a given outcome may occur during a random test. This is one of the main engines behind countless game systems. Any game where cards are shuffled or dice rolled have relied on probability. It is one of the key areas of mathematics that designers need to become familiar with.
Basic probability is easy to grasp, but for now, we will just get your feet wet.
Calculating a simple probability is straightforward:
18 Count all of the outcomes that are possible.
19 Count the outcomes you are interested in.
20 Divide the number you are interested in by the number possible.
Equation 2.1.1 shows the basic formula.

(2.1.1)
$$probability(A) = \frac{outcomes\ being\ looked\ for}{total\ possible\ outcomes}$$

When we write a probability, we use a number from 0 to 1; zero represents "impossible" and 1 is "certain." Just remember that fractions and decimals and percents are all the same; they are all numbers. So ¾ is the same as 3 ÷ 4; which is the same as 0.75; which is the same as 75 percent. After going through some of these examples, we will explain why using 0 to 1 is useful for probability and video games.
Figure 2.1.37 shows two simple probabilities, the first looking for the chance—the *probability*—of rolling a 4 on a six-sided die. With six possible outcomes and one of interest to us, we calculate our chances to be one in six. You knew that already. To get the decimal number, you just divide 1 by 6 (1/6) which gives us 0.1667—just about 17 percent.

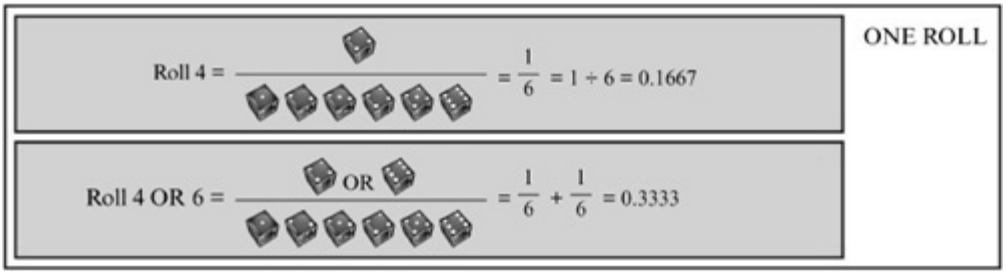

Figure 2.1.37: A basic probabilistic problem.

The second example is almost the same as the first with the exception of the event we are interested in: rolling a 4 or a 6. There are still six possible outcomes, but now we are looking for two numbers (4 or 6) rather than just one. Since these are mutually exclusive events—you can only have either a 4 or a 6, not both—we can just add the two probabilities.

Things get a little trickier when you start rolling more than one die (Figure 2.1.38). So you know that any one number on any one roll has a chance of 1 in 6; in decimal it's 0.1667. But each time you want to add a die roll to that experiment and look for another number, you multiply the two probabilities. Rolling a six has a chance of 1 in 6, which you then multiply by the chance of rolling another six and arrive at 0.0278, or just under 3 percent. (Rolling a 12 or a 2 are the rarest outcomes using two dice.) As long as you keep adding rolls and looking for one outcome each time, you keep multiplying. For example, rolling four times and getting (1, 2, 3, 4) in a row would be 1/6×1/6×1/6×1/6 or 0.000077; in other words, pretty unlikely.

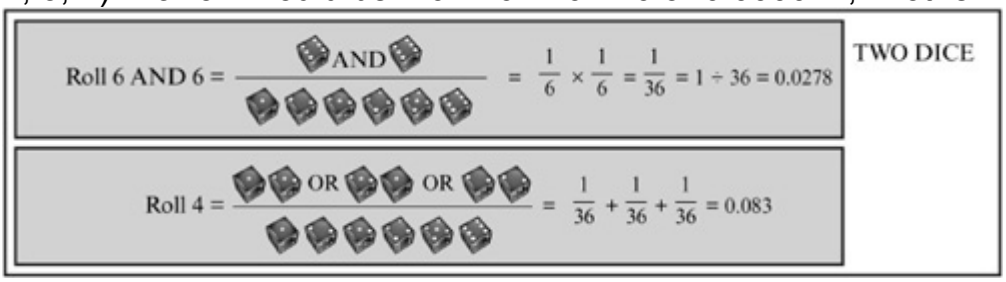

Figure 2.1.38: What are the chances?

The second problem in Figure 2.1.38 considers the result of rolling two dice and looking for a total value, in this case: 4. Just remember that *each roll is an individual test* and the order of events matters. So rolling a 1 and a 3 is *not* the same as getting 3 and a 1. You will need to include probabilities for (1,3) and (3,1) as well as (2,2). Once you have worked out all of the possible combinations (in this case three), you then add the probabilities together just like in the second example in Figure 2.1.37.

Rolling a 4 with two dice has around an 8 percent chance, while rolling 12 is less than three percent; this makes sense when you consider that there are three ways to roll a 4 and only one way to roll a 12. Figure 2.1.39 shows how die rolls stack up. Consider the six ways to roll a 7: (1,6), (2,5), (3,4), (4,3), (5,2), and (6,1). Looking at Equation 2.1.2, you can see that it is no more difficult than any simple probability once you know how many events you are dealing with.

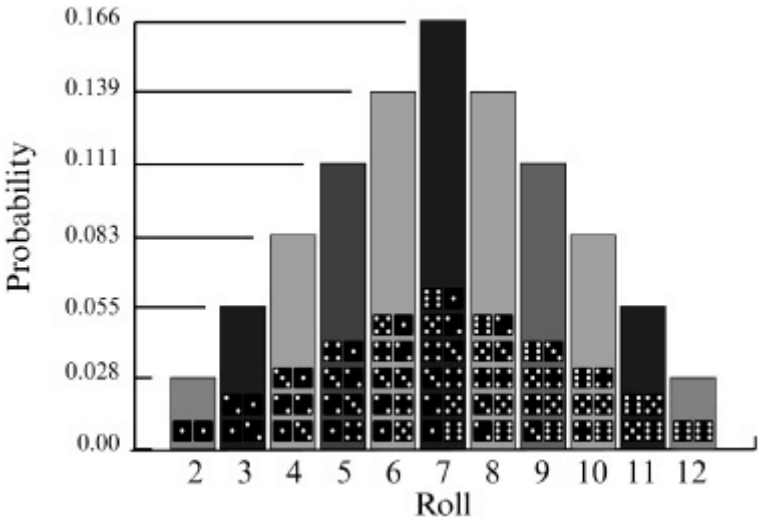

Figure 2.1.39: The probabilities with rolling two dice.

(2.1.2)
$$probability(7) = \frac{6}{36}$$

Our last diagram on probabilities (Figure 2.1.40) illustrates a different problem. Here we are looking for kings or clubs. Of course, there is a king of clubs, and we don't want to count him twice. So you take the first set, kings, and calculate the probability (4/52). Do the same for clubs (13/52). You know, from Figure 2.1.37, that looking for more than one outcome increases your chances—you're open to more results—so you will add the two probabilities. But you need to make sure you're not adding in a given outcome more than once. So you need to remove the king of clubs from your calculation as in Equation 2.1.3.

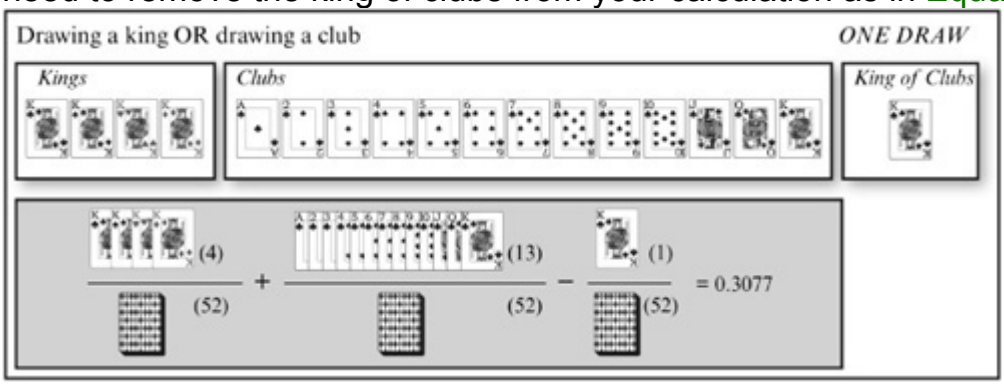

Figure 2.1.40: What are the chances?

(2.1.3)
$$p(king\ or\ club) = \frac{all\ kings}{52\ cards} + \frac{all\ clubs}{52\ cards} + \frac{king\ of\ clubs}{52\ cards} + = 0.3077$$

Why is it useful to express probability as a decimal value between 0 and 1? The reason that decimals are preferred over fractions: they are easy to work with. Even common fractions like ⅔ and ⅝ aren't as simple to evaluate as 0.667 and 0.625; math with decimals is painless, too.

If you feel more comfortable talking in percentages, feel free. In the long run, however, being comfortable expressing probabilities in decimals will pay off, especially if you work on systems with lots of potential outcomes (e.g., RPGs).

Save/Load Systems

Saving is the process of recording the current state of the game, and loading reinitializes the game according to the state in which it was saved. There are many ways save systems can function, and each game has particular needs that should help decide which scheme is most appropriate.

There are three primary ways players utilize a save system:

- Stop play and return later without losing progress.
- Protect current progress from future failure.
- Branch their progression to explore alternative choices.

It comes as a surprise to learn that players have lives in the “real world” with biological and social demands that call people away from their games. A save system is a mechanism that allows the player to retire from the game, knowing that progress will be secure and waiting for her when she returns.

Save systems are also useful for games that feature challenging and risky mechanics. Players can be comforted by knowing that, if they fail, all will not be lost, and they can continue to enjoy their game.

When the player comes to a moment of decision, it can often be difficult to choose which way to go. Players may use a save system to buy themselves some leeway. (I’ll try this out, and if I don’t like it, I’ll go back.) Or they may be intent on seeing everything that the game has to offer, so they would like to return to the decision point just to experience the alternative.

Some different save systems:

- Limited—finite number or special locations.
- Checkpoint—system automatically saves at key environmental points.
- Unlimited—users may save anywhere in the game.
- Autosave—saves happen periodically or after key events.
- Profile—autosave system tied to the player identity/profile.

If you are working within memory constraints or it fits the aesthetics of your game (risk, challenge, etc.), you may consider limiting saves. One approach limits saves by requiring players to use special locations or limiting the overall number that the player can access during a given unit of content (level).

Checkpoint saves offer players a system where they do not have to worry about safeguarding their progress. Usually, checkpoints are placed between challenges or automatically during transitions from one area to another. Plan for situations where checkpoints might catch the player in a weak moment. For example, set minimums for important attributes so that players will not find their game has saved just at the moment before failure.

Save anywhere systems are powerful, and players often prefer to save where and when they want. Unfortunately, there are two downsides: 1) they can be used to circumvent designed challenges, making the game easier; 2) they require more work for the player to maintain.

Autosave systems provide some of the benefits of checkpoints, while retaining some of the flexibility of unlimited. Frequently, autosaving is an option that modifies an unlimited save system. Players have the option of turning the autosave off if they would rather manage their own saves.

Most games in the casual industry use profile-based systems. When the player begins the game, he is asked to name the profile. This is, effectively, the save slot. At any time, the player may simply shut down the game, knowing that his progress has been saved.

Resources and Economies

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 [Fullerton08].

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?

Content

Content is the space of your game and everything that fits inside of it: the combined total of all areas, elements, and states throughout any moment in time.

Developers spend the workday making games and to us, no surprise, content ordinarily means things like levels, models, missions, back-stories, enemies, animations, textures, dialogue, sound effects, music, particles, characters, and so on. We tend to think in terms of the things we need to create during the production of a game—stuff we need to work on. You will be encouraged to hold a broader and more inclusive view of content. For example, the range of content might include the following information:

- Game spaces—chessboards, The Barrens
- Game objects—a rook
- Narratives—back stories
- Characters—Mario, Frodo
- Scripted events
- Models and animations
- Sounds and music

A game's content and its systems are intertwined—two parts of the same whole. While the game is being played, there is no clear distinction between them. Through play, the player's experience results from the synthesis and the quality of execution in both the content and its systems. A superb combat system could be transformed into a broken and frustrating mess if the game's levels are choked with too many enemies. Beautifully detailed woodland landscapes can be a total bore if the process of collecting what you need from them is tedious.

That the two are not synonymous can be seen in water cooler talk over the influence of each in the experience of game progression. Someone might describe something as "system heavy" because the progression focused on the interplay of rules and behaviors (*Civilization*); another is "content heavy" for using environments, narratives, and characters to do the same (*Metal Gear Solid, Gears of War*). Additionally, developers tend to say that players "experience" systems while they "consume" content (the greedy scalawags!).

Theme

Theme is the core idea or message that your game will convey. Theme is not the setting, mechanics, backstory, characters, graphic art, sound, or any other individual element; it is produced by, and the result of, everything in the game when considered as a whole. Understanding what theme is and how it operates is easier if you understand the difference between two ways to convey meaning. The first, *denotation*, is the literal meaning of something expressed. This is where we operate most of the time, at school or work or with our friends. Something is said, we listen, and we interpret the meaning more or less literally. You say, "I went to Washington D.C.," and we accept that as plain fact.

The second way we get ideas across is by using *connotation*, where statements are considered along with links to other meanings. This is where the audience exercises their interpretive skill, picking up on metaphors and emotions encoded in the statement. You say, "I hadn't realized there were so many snakes in the capitol," and we have something connotative going on—Washington D.C. is "full of snakes." Clearly, we think, this is political. Your game may not have a theme, as you see it, but themes will get in there anyway. Connotation is part and parcel of arts, and it's almost entirely unavoidable because it is produced by inference; the audience is the one connecting things together in its mind, even if you're not trying to say anything. Maybe you visited the National Zoological Park during your trip, and you really had no idea that the Herpetological department was so big. See? No politics, just snakes.

Premise

All games have a *premise* that ties together the environment and action [Fullerton08], relying on a set of rules that the fictional universe will be bound to. The premise can be formed with just a few words. A fine example comes from the middle of the 1980s, when a player could sink a quarter into *Robotron: 2084* and, with five words, be transported into a understandable future place: "Save the last human family."

It is impossible for a game to completely lack a setting, though it may be radically abstract with nothing for the mind to grasp—completely indescribable as a real place, but our minds are not so quick to fail us as is our language. Within the first few moments, a premise is forming in our minds whether the game is providing it for us, or we have to invent it on our own. We need context to guide our experiences and the mapping of our actions. The premise is the source of the game's individual context.

When you are developing a concept, you should be considering the premise from the outset. Summarize, in a short statement, what the game is all about. For example: *"Jenny Briar: Investigator" is a puzzle detective game that challenges you to collect evidence to solve crimes in 1970s New Orleans.*"

The Game Setting

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.

Content and Progression

Game systems provide the framework, and content forms the shape of your game. You could also think of game systems as building codes and standards, defining the rules and guidelines for construction. But it is the layout and arrangement of the content that puts the roof over the player's head. Ultimately, the structure of your content will reflect the qualities of your game, but keep in mind that it will be the content that players are actually getting their eyeballs, ears, and hands on.

The good news is that, when it comes time to plan content structure, you will already have some ideas about how it will look. As a rule of thumb, if you are able to formulate a description of your game's core play mechanics, you should be sketching ideas for the way your content will be structured. Trying to "tie things together" halfway through a project (or later!) is a desperate situation for everyone and has been responsible for countless disasters over the years.

While there are good arguments for bottom-up and top-down approaches to creating mechanics, systems, and content, you are strongly encouraged to structure that content from the top—the whole game experience—and work in more and more detailed plans.

So you can begin with big questions like:

What things do the mechanics and systems require of the content?

How does the game's start relate to its end?

What does a single session (sitting) of play look like?

Will the game need to be divided to support these sessions?

How will units of content be related to each other? To the whole?

Frequently, these questions are easy to answer because the game concept has been developed with assumptions about the content structure. You will begin to consider the overall layout of your game spaces and the player's progression through the course of the game.

By conceptually sketching the content structure, you can begin experimenting with different progression layouts. There are a variety of basic arrangement elements that you will see in various mapping and planning, as shown in Figure 2.1.41.

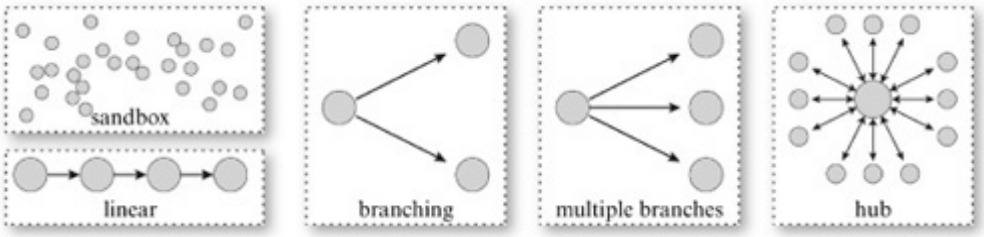

Figure 2.1.41: Basic elements of content arrangement.

Level Design

The most common way for games to divide their content is into *stages, rounds*, and *levels*. They are all names for the same thing: an arbitrary unit of content or progress. The size and form of a level varies greatly from game to game, but they ordinarily include distinctive spatial or temporal elements: levels each form a unique setting. Within a game, levels are known by

similarities and differentiated by unique elements.
For example, *Peggle* levels each have unique background art and placement of pegs, blocks, and obstacles. In *Halo*, a level is an arrangement of space and geometry, game objects, and scripted narrative events. In *BookWorm*, levels are sets of letter tiles where the distribution of common to uncommon letters changes over time to increase the difficulty.
Some typical questions you may ask about levels:
What is the structure of the game's content? How do levels support this?
How will levels be distinct from one another?
What will the player do to transition between levels?
What elements are required in every level?
What kinds of features can be unique to a level?
During production, level designers are touching the ground at every moment. Their eyes and thoughts are set at the player's perspective. They don't have to be a member of the game's target audience, but they do need to understand and appreciate that view in order to make the hundreds of small decisions that, collectively, will make each level successful.

Progression

A field having rested gives a bountiful crop.
—Ovid

Progression is how the player's experience of the game changes over time. It is usually expected that a game's progression will ramp in difficulty to track the player's skills. But progression can also determine how content is distributed throughout the game. For example, it is important to ensure that assets get enough use to justify their expense, as shown in Figure 2.1.42. (This is a business after all!)

Level	Name	Chapter 01	Chapter 02	Chapter 03	Chapter 04	Chapter 05	Chapter 06
LAS	*Left Aloure*	X				X	X
WOB	*Washed Overboard*		X	X			
WIB	*Woken in Bed*	X			X		X
HFP	*Hungry for Pancakes*				X	X	
OOO	*Oscar Oscar Oscar*		X				X
THH	*They Invented Hamburgers*				X		X
NMO	*No Means On*			X		X	X
SPF	*Superfind*		X	X		X	
IMF	*I'm Finished*			X	X		X

Figure 2.1.42: A map of levels to chapters for a casual game.

Figure 2.1.43 shows a few examples of different progressions. Using a *linear* progression, players are directed from one level to the next as they complete objectives. There is no decision to be made, they simply continue on. While this may sound boring (and many critics will argue that it is), players have continued to enjoy the crafted experiences that this kind of progression can support. Because you have control over the order that players will encounter content, you can have a tighter control over their experience. A *branching* progression allows the player some choice in the overall progression, while still keeping them on a reasonably predictable path through the game. Branching progressions typically expand for a few steps and then contract again, funneling the player back to certain key points like bosses or significant story events. *Hub-based* progressions offer even more freedom, but the designer must sacrifice a little of the narrative sequence to player whim.

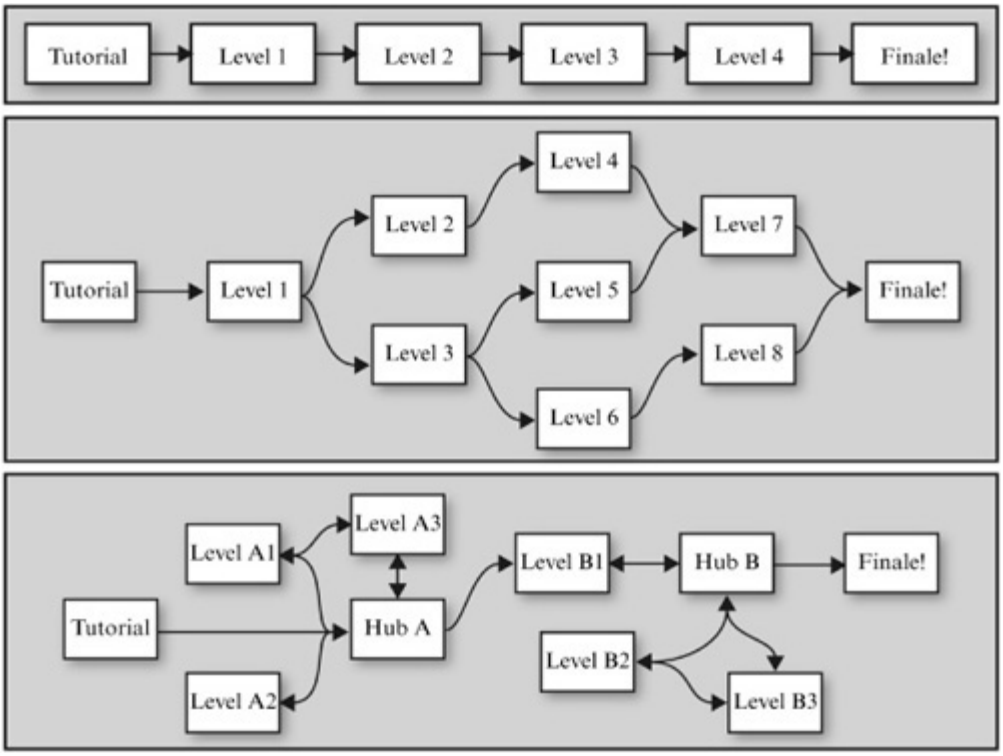

Figure 2.1.43: Examples of linear, branching, and hub-based progressions.
Game designers map other distributions to and for game progressions: player skills, likely powers and inventory, and other attributes that can be used to help judge the appropriate challenge.

Shaking the Line

Many people think that a game's difficulty should, if it were graphed, look something like the ones in Figure 2.1.44. If humans were learning robots, these would be great. Sadly, people aren't able to continue to get better and better on an endless cycle. You are fighting against the diminishing returns of our minds.

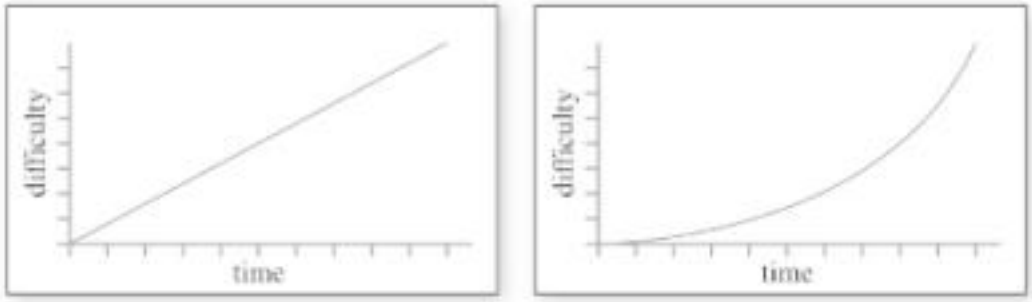

Figure 2.1.44: How should difficulty rise?
The flow diagrams in Figure 2.1.45 illustrate what it feels like when the difficulty of a game continues going up. We run into our limits, and we become frustrated and often angry. A more player-friendly difficulty curve looks like Figure 2.1.46, where over time it rises, but periodically the difficulty decreases after a sharp rise. These decreases should naturally occur after a boss fight or the conclusion of a level, in order to give the player a short break and a period of recuperation.

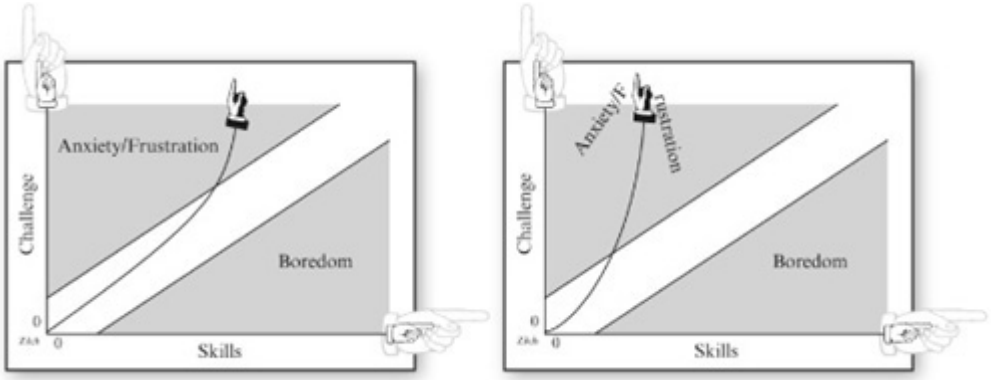

Figure 2.1.45: The player experience of the relentless rise in difficulty.

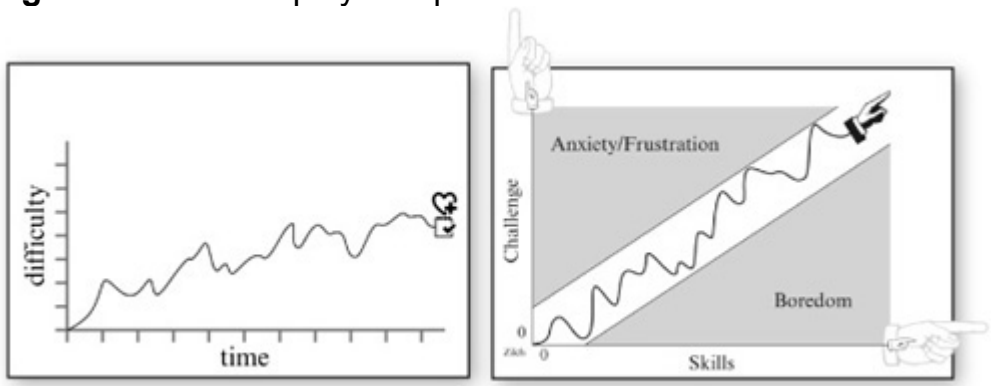

Figure 2.1.46: The ideal difficulty progression.

Environment Sounds

Today's computer visuals are stunning, and they can go a long way to bringing players into your worlds. But it is the audio that can sink them deeper without even realizing what is happening. The ambience of the world and the player's place in that world need to be supported and reinforced through audio. An adventure in the wilderness would not feel complete without the sounds of wind sighing through grass and trees, or the sound of a small stream growing louder as you approach.

Consider one classic sound effect of first- and third-person action games: the time-honored footstep. Without it, players do not walk through a world; they hover on a magic carpet over it. But it is not enough to have a footstep, repeating over and over. There needs to be a set of footstep sounds that vary in pitch and volume. Not only that, but if there is more than one material to walk on (concrete, sand, grass, metal, snow, etc.), then there will need to be sets of those footsteps as well.

During system or content design, it's not unusual to find yourself thinking of sounds that would accompany the things you are working on. You might even find yourself making some pretty strange noises with your mouth. (Be careful of this if you are working in an "open" office, unless you don't mind a little teasing.) As you work and think of these sounds, keep running lists of those that you imagine will help support the player's experience. As these lists grow, periodically take a few minutes to sort and categorize what you have. If you maintain lists of sounds while you work, you will capture tons of great inspiration and make the job of designing the audio requirements easier and more effective.

Unless you have experience working with audio or are willing to put in the time to really learn, resist the temptation to do it yourself. The Internet is littered with game audio that is copied from the same files or junked together without care. If you are working independently on a budget, it can be tough to beat the value of good sounds.

Design Work

You can accomplish anything in life, provided you do not mind who gets the credit.
—H.S. Truman

One of the most exciting things about game design is the endless opportunity for solving new problems and learning new skills. Despite the various roles you may have in your career, most of your work will be approached in styles and techniques familiar to you. These will get bundled together, and people will describe you by your general approach to design.

Designers Solve Problems

When starting out, problems you have never seen before can be a little scary. Maybe it's a procedural mission system, a scripted event for a level, or a new UI for virtual item trades; if you lack a process for solving problems, it is easy to get overwhelmed.

To describe a simple problem-solving method, we will use the Plan Do Check Act diagram (Figure 2.1.47). There are similarities between PDCA and the way design is generally carried out.

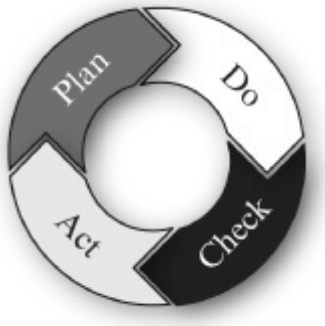

Figure 2.1.47: PDCA, the Deming Cycle.

Plan: Begin by developing an understanding of the current problem—the things that need to be done. Ask questions, write answers. Make lots of lists! You are creating a clear picture of what success looks like.

Make lists of answers to questions like:

- What are the requirements?
- What are the goals?
- What are the options?
- What has been tried before?

Sketch proposed solutions using brainstorming, mind maps, flowcharts, diagrams, spreadsheets, and any other tool you find helpful. Test those ideas with others or against objective criteria, and decide on a reasonable solution that meets the needs of the problem effectively and efficiently.

Remember not to worry; you're making games!

Do: Take that reasonable plan and put it to work. Complete the work to the appropriateness of the problem. Resist the temptation to put extra effort into the solution just yet. You are going to need to see if you're on the right track.

Check: Test and measure the results, comparing them with what you had expected. List and describe the differences you find.

Act: Review the results. Did they measure up to the expectations of the plan? Is the problem solved? If not, what didn't work? Will you need to change the plan? Will you need to change how the plan was carried out? Did you make a mistake while checking?

Repeat: Cycles like this will rule your work. Something needs to be created or

changed, and we need to figure out the best way to do it.

When the problem you are facing looks large, remember not to worry; you are making games! It gets easier over time, and when you are comfortable solving problems, you won't worry about unknown challenges.

Creating Concepts

If we could first know where we are, and whither we are tending, we could better judge what to do, and how to do it.

—Abraham Lincoln

Developing a concept for a new game is like sketching. An idea is taken from some early state and transformed into something more elaborate. Through that process, details are worked out and the concept becomes more "real."

Creativity

Creativity is the ability to produce an idea, action, or object that can be considered new and valuable within the culture at hand [Csikszentmihalyi99]. Just as a doctor's job is not to "use a stethoscope," a game designer's job is not to "be creative." A stethoscope is an important tool for a doctor. Being creative is an important tool for a game designer. Your job is to create, maintain, and share a conceptual model of the game.

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 one 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 as long as they are willing to risk a little.

Going it Alone

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.

Going Together

Going alone is a lot of work and fails to use your biggest asset: your network of coworkers, friends, and family (see Figure 2.1.48). Get other people involved!

Figure 2.1.48: The whiteboard during a creative meeting

By far, the most popular and common approach to group creativity is *brainstorming*. Nearly all brainstorming involves eliminating criticism during early stages. Evaluate ideas only after many of them have been noted.

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. But effective brainstorming is a learned skill. For all practical purposes, there are such things as "bad ideas." The trick with brainstorming is to identify, appreciate, and cut the bad ideas as soon as possible.

Six Thinking Hats

A role-playing system for creative problem solving, this approach uses a metaphor of colored hats to symbolize the different ways people think. Encouraging groups to change the way they work together, each person "wears" a different approach to the problem at hand [deBono85]. It can be a refreshing change of pace to your collaboration, especially if your group is unable to progress because everyone continues to repeat themselves and restate their position.

21 **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.

22 **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.

23 **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.

24 **Yellow Hat:** Pollyannaish attitude typifies this hat. Optimistic logic is applied, looking for benefits and profitable outcomes that could result from an idea.

25 **Green Hat:** Symbolizing vegetation, growth and creative possibilities are explored. New ideas or modifications to earlier suggestions are offered with an emphasis on novelty.

26 **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 ideas fill the world around you. 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 (Figure 2.1.49). 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.

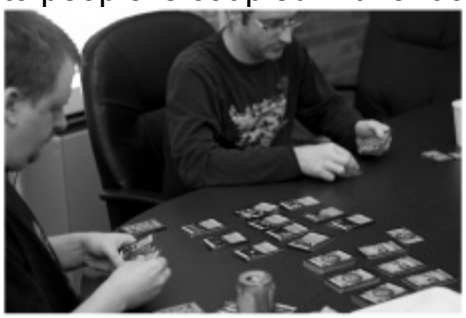

Figure 2.1.49: Paul and Dave playing a great card game, getting inspired.

One of the greatest sources of inspiration and learning for any video game designer is the vibrant world of board games. If “board game” makes you think of *Monopoly* or *Risk*, then you don’t know what you’re missing.

Here is a list of a few board games that you should play:

- Agricola
- Blokus
- Bohnanza
- Carcassonne
- Dominion
- Hive
- For Sale
- Power Grid
- Puerto Rico
- The Settlers of Catan
- Ticket to Ride

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. Render them into system diagrams. Enjoy being a game designer!

Prototyping and Playtesting Cycles

All life is an experiment. The more experiments you make the better.
—Ralph Waldo Emerson

Without a doubt, the best way to know if a design is good is to play it as soon as possible. The best way to know if your game is good for your audience is to have some of them play it as soon as possible. *Prototypes* are the tool for this. All prototypes, whether in software or physical material, serve the same purpose: to quickly build a rough, working model to evaluate an idea, answer a question, or as a test run at solving a larger problem. Prototypes are disposable sketches to be thrown away once your questions have been answered. They can be used during all phases of design and can take any shape you need to prove your ideas.

Physical prototypes are games you make 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.

This is where you should start putting your ideas together. Chances are good that, in the real world of professional game development, you will hardly ever get to *start* with a prototype, but there is always time to take control of your own work and make them for yourself.

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.

Playtesting

Playtesting is finding problems from the user's perspective. *Playtesters* are the people who play your prototypes or game and report feedback on the experience. Designers will often observe playtesters, taking notes and occasionally asking questions of them.

Ask questions continually:

- 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?

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.

Five Tips

Here are five quick suggestions that just about any experienced game designer will share.

- **Carry a notebook.** When you have an idea, write it down. When you want to show something to someone, make sure it's captured in your notebook. Sketches and diagrams and anything that you want to remember can go here. Just the process of writing it down will help you remember.
- **Grow creativity in everyone by sharing yours.** Your job isn't to generate ideas, it is to harvest them, and the most fertile fields are the people around you. Encourage people to share ideas by offering yours, and the earlier the better. Test and validate your thoughts with everyone or, better, share your *thinking*. How you come to a given answer or inspiration can be just as important as the result itself.

- **Listen to people deeply.** Whether talking with a friend, colleague, or customer, listen to everything they are telling you but ignore how they are saying it. Look beyond the words and even their assessments. Look for core causes. Ask questions to draw the answers out. Repeat their positions in your own words, asking "Did I understand what you were saying?"
- **Learn how to do the rest.** A good designer understands game design first and foremost. But a working knowledge of the other parts of development (art, programming, audio, etc.) provides several benefits. First, and most importantly, you create designs that make the most of the available resources. Next, you have more tools to help communicate with others. Last, if you go far enough to be able to create your own digital prototypes (using Flash, for example), you can reduce the time it takes to create effective digital gameplay sketches.

Kill your bad ideas! If your wonderful idea bombs in the prototype *and* you can't find a path to fixing it, kill it. If you can't find anyone in the office that "understands" your great game idea—fix it or kill it.

Summary

This chapter has introduced you to a brief picture of concepts, vocabulary, tools, and other issues related to video game design. Our goal has not been to prepare you as a game designer but as a student of game design. With this snapshot, you should be able to fit the issues you encounter into a basic view of game systems.

After beginning with some reassurance, we established some basic terms and highlighted the importance of communicating with other developers. After moving through these provisional definitions, you were shown one model of games. This schematic reflects the need for game designers to solve problems by isolating issues into parts. Remember that this model is not reality but a representation. In any real game, pieces do not fit into neat categorical bins; things get complicated, and we create models to simplify them. Play mechanics and experience were put into the Player, emphasizing the personal and subjective nature of both. Game systems and content were put on the opposite end of the relationship, indicating the physical reality of where those materials lie—in the artifact. Actions and interface were grouped under Interface to show both how the player and game connected and to emphasize that the actions of the player are every bit as relevant as the controllers that listen to them. You were then brought through many issues and concerns facing game designers every day, organized into the areas of our schematic model. An adequate discussion on any one topic could easily fill whole chapters or books, and here you only got a glimpse. With the strongest and most sincere emphasis, you are encouraged to continue learning about game design. Something happens, along the way, where it becomes difficult *not* to relate just about everything you learn to game design; this is a diverse and expressive art.

The only thing you will ever *need* to do to be a game designer is to be one: start now and don't stop.

Exercises

1 A target audience has been chosen for you: women 35–55. Create a game

. concept to meet the demands of this audience.
   - Start by modeling the preferences of this audience. Begin by creating a list using public sources of information (e.g., library, Internet). Include considerations for playing styles, learning styles, themes, and settings.
   - Create a list of game features, mechanics, and actions with this information.
   - Gather a group (4–8) of people belonging to this demographic and conduct a focus test that you will record using audio or video. Share your feature list as a starting point, asking them for opinions. Ask general questions about their gaming habits, likes and dislikes before narrowing to your list of preferences from 1a. Encourage group discussion. Make notes of opinions that resonate strongly with the group. Keep the conversation moving but do not contribute your own opinions.
   - Create personas (2–3) representing your audience. Using focus test data and your initial list of preferences, look for agreement between the two sources.
   - Brainstorm concepts (2–3) addressing the preferences of your audience. List features and sketch mechanics and interfaces. Use diagrams and short, clear descriptions. The writing should fill one-half to one page, no more.
   - Gather a second focus group to record and present your concepts. For each concept, start by having the group read and review printed material. Then offer a short description of the concept. Begin to solicit opinions by taking a quick show of hands to see like/dislike. Then work through what was liked and disliked, getting more and more detail as the discussion moves on. Provide explanations for questions but do not defend your concepts.
   - Review the test results and suggest possible ways the concepts could be improved to meet the concerns of the participants.

2 Analyze one of the *Lego* series games by TT Games.

. List objectives. Sort these into primary/required and secondary/optional.
Create a list of play mechanics and relate these to the objectives. Make sure to note mechanics requiring the use of specific characters.
Create two or three system diagrams showing, as a best guess, how the game systems are arranged to support play mechanics. Choose systems that have at least two variable stocks. (For more information on systems thinking, see *Thinking in Systems*, by Donella Meadows [Meadows08].)
List the game actions that players can perform in the game.
Map player actions to the interface. Create a control diagram and a schematic map between player actions and the feedback from the interface.

3 Convert the design of a physical board game to a video game.

. Summarize the objectives.
List all of the physical components: pieces, dice, board layout, etc.
Describe the sequence of play. How are turns structured? When are players allowed to act? What actions are allowed and when?
List all of the game information (position, cards, etc.) and determine which can be open and which must remain hidden.
Design an interface that allows players to perform all of the necessary game actions and understand the state of the game at any time.

4 Prototype a game concept.

. 27 Choose an audience that you would like. (It can be you.)
28 Create a list of experience goals for the project.

29 Create goals and core mechanics to satisfy those goals.
30 Design a system that supports those mechanics. (Note: 4c and 4d almost always cycle back and forth.)
31 Design an interface mapping the player actions, which hides or reveals appropriate information.
32 Create a physical or digital prototype (for more information on prototyping, see *Game Tuning Workshop, 2nd Ed.* by Tracy Fullerton [Fullerton08]).
33 Test the prototype with your audience and adjust the design using this information.
34 Repeat 4d to 4g until the concept meets your goals.

Chapter 2.2: Game Writing and Interactive Storytelling

Overview

While stories have been a part of electronic gaming since the beginning, in the majority of cases, they seem to exist as merely an afterthought, secondary to graphics and game mechanics. They are often overlooked by game developers and written off as being all dialogue or simply not interesting. Unfortunately, this sells the story short, for a story is, at its heart, conflict, and conflict, after all, is exactly what gaming is about. Furthermore, as technical elements are reaching their apex, developers are now turning to alternate ways to make games realistic and immersive. The emerging art of interactive storytelling in games is an excellent way to achieve this.

Storytelling has been an integral part of the human experience since the cave-painting era. Stories allow people to escape reality and to be people they could never be, doing things they could never do, in places they could never go. This holds true for stories in video games as well, and in fact, the gaming medium takes storytelling to a new level. For example, even narratives written from a first-person view in a traditional story are still about someone else, whereas in a game, the player is playing the role of the character and events that affect the character, in essence, affect the player's own personal experience. While interactive storytelling within electronic gaming shares many elements with more traditional storytelling, it has, over the past few decades, evolved into its own unique medium.

There are three primary ways a story can be experienced. A person can be told a story, either orally or through text, he can be shown a story as through a movie or cut scene, or he can experience the story by interacting dynamically with the storyline. While the first two ways can be quite immersive, when delivered alone, they fall short in the gaming arena. The average gamer is an impatient soul, with action on the brain, and will almost always prefer to experience the story, rather than hear, read, or see it. This, combined with increasing player sophistication, calls for the growth of the interactive story.

While not possible, or even desirable, in every game situation, the interactive story is an excellent way to fulfill this wish to experience the story, rather than passively absorb it. At its best, the interactive story can create within the player, the psychological state of flow, where reality fades and all consciousness is focused inside the game. The story also provides the player with the motivation to continue playing and a reason to press forward through game obstacles. The player is continually rewarded with uncovering new parts of the story as he plays the game. This maintains his sense of immersion and creates a more fulfilling game experience.

Immersion is the main reason to create amazing game stories and to build impressive game worlds. Giving players a presence in the world and making them a part of the story is essential in creating a sense of immersion [Krawczyk06]. As a writer, it is also important to know your audience and be fully aware of the scope of the project so that the writing is appropriate to the type of game being developed. Generally, when writing for a game, the rule is to be as efficient with resources as possible, while still creating an entertaining experience for the player. Game making is a for-profit business, and as much as writers would love to try new and fantastical ways of telling a story, it is usually not within the scope or budget of the game being developed. However, there are many methods and techniques for creating an immersive game experience through writing that aim to maximize immersion, while still keeping the scope of the game within reasonable parameters.

Know Your Audience

It is essential for the development team, including the game writer, to be on the same page and to have complete understanding of the scope and vision of the game they would like to create. Different types of games, traditionally, demand a different type of story, and even within each game genre, there will be differences that will affect the type of narrative that will best suit the game [Dille08]. For instance, it would be a huge waste of resources to include several very expensive, cinematic cut scenes to explain the background of a character in an action game for the Nintendo DS.

For most action games, such as SEGA Corporation's *Sonic Unleashed* or Nintendo's *Super Mario* games, the purpose behind the game is to survive and to use timing and fast reflexes to overcome obstacles. While an exciting setting and a fun character are usually essential to these types of games, an elaborate story usually is not necessary. The character is driven by one major goal, such as to rescue the princess or to simply survive in the treacherous game. The testing of reflexes required of such games is usually sufficient to keep players entertained and to motivate them to continue playing. Furthermore, unless the storyline is delivered as quickly as the game is running, it will not be well received, and will instead be seen as an intrusion on the action. Therefore, for the majority of action games, creating an elaborate or interactive storyline will probably be a waste of resources.

The case is often the same for shooters, although several recent releases have expanded to include a more complex story. Moreover, some shooters have become crossed with other genres and share many game elements with puzzle games, action games, and RPGs. Valve's *Half-Life 2* and 2K Games' *BioShock*, for example, are still both shooters; however, they are also both also rich with narrative elements usually seen in RPGs and twitch elements traditionally used in action games. Shooters have also moved online with games such as *Left 4 Dead* and *Halo 3*, where there is a basic story and setting; however, for the most part, players create their own stories through their play. Ask any players that have been playing a game online for a day or two and they will have plenty of stories to share about what events transpired in their gameplay world with their online partners and enemies. These stories are personal, and could never be anticipated or designed into the game.

Because of the large diversity among shooters, it is really up to the scope of the project as to how much and how complex the narrative should be. It has been shown that narrative has been well received in recent shooters, and many of the top-selling games over the past few years have been shooters with a storyline. Other, more pure shooters without a plot are also still selling well. There will always be impatient players for whom the story just gets in the way of the mass slayings. For shooters, then, it truly is a case of "know your audience."

This same “know your audience” philosophy can be assigned to other genres as well. Traditionally, racers, fighting games, strategy, puzzle, and rhythm games have been thin on plot. However, this does not mean that there is not room for narrative within these genres. Incorporated correctly, a strong narrative can easily increase immersion into these games. In the near future, it is likely that many genres will expand to include more intricate narratives into their gameplay.

The genres that customarily have demanded strong narrative elements are the role-playing game (RPG) and the adventure game. Players in these genres not only appreciate a good story, but they also have come to expect it. This is the place where a well-thought-out setting, plot, and narrative are integral to the gameplay. With these genres, the players usually have time to think about and analyze their next move. They want to make their own decisions, and they want their decisions to matter. They want to be immersed in rich and fantastical worlds they can explore as they develop and nurture their character. It is within these genres that the interactive story can really enhance a player’s experience.

Not only is genre a major consideration, but it also needs to be known which platform the game is intended to be played on. The audience between console games, PC games, handheld, and mobile games is very different, and the players have very different expectations and tolerance levels. The age group the game targets also needs to be considered, as again, different ages have different expectations of a game. The details of these expectations will not be explored in this chapter; however, it is important to be aware that these differences exist and to make sure when writing for a game that these differences have been taken into consideration.

Budget and Other Limitations

Ask any group of developers, and they would agree that, of course, the writing is important to the game. Unfortunately, a look at the budget tells a completely different story, and generally, only a small portion of time and funds are devoted to the writing. Games with good plots don’t necessarily sell any better than those without, so when budget cuts are necessary, the story is often the first to go [Jeffries08]. It is key, then, for writers to be clear about what limitations are being placed on them, and to find ways to maximize immersion while staying within budget limitations.

Implementing stories in games has been a slow process and has not always proven profitable, and in general, the game industry does not like to take financial risks. Warren Spector of *Deus Ex* fame compares the current situation to the movie, *Citizen Kane* [Kosak 05]. For all of its critical acclaim, *Citizen Kane* failed to turn a profit. In games, Sony’s innovative 2004 game *Ico* is an example of a groundbreaking and critically acclaimed design that failed to meet with commercial success. *Ico* uses a simple, yet compelling story to motivate the player through a series of puzzles. It is done in such a clever and simple way that the player becomes completely immersed in the boy meets girl tale. However, despite its innovation and current cult following, it never turned a profit for its developers.

Commercial failures of this sort often keep game developers reluctant to take a financial risk on a game based on a cutting-edge story design. Often, they prefer to stick to guaranteed moneymakers, like sequels and games based on commercial characters. This means that authors might be forced to only make small strides away from time-honored game-writing techniques. Clever stories are not always welcomed with open arms in gaming studios. Be aware of how far the studio is willing to go in implementing innovative ideas and be sure they are open to it before attempting experimental story designs.

Another cost consideration is that of hiring professional actors. Stilted and poorly spoken dialogue can be the death of immersion for a player. If the acting is not convincing, the game will be unbelievable, thwarting the entire reason for having dialogue in the first place. Even the most superbly written script will fail in the hands of a poor actor. If the budget does not allow for the hiring of professionals, it is preferable to have dialogue delivered in text form, rather than poorly delivered expositions by an amateur. Therefore, know ahead of time if there will be a budget for professional actors and plan accordingly.

The writer might also be bound by technical limitations. For example, the writer might write that a giant bat comes down and swoops the character off to a new area, but the game engine doesn't support flying. One of the biggest problems with game writing is lack of communication about what is and isn't possible within the game [Jeffries08]. Although it might seem as though a game world is filled with unlimited possibilities, most games are actually fairly limited in what can and cannot be accomplished. Therefore, writers need to be fully aware of the technical limitations of the game they are writing for.

Basic Storytelling Techniques

It is important for a game writer to understand classic story structure. It is a tried and true method that is simple, works, and will repeatedly meet audience expectations. There are many ways for the experienced writer to stretch her creative limbs and move from this path; however, for the beginner, it is usually best to stick to a formula that has stood the test of time. These are the roots from which the vast majority of stories are created.

The perception of many developers is that story is primarily dialogue. Not only is that not true, but a lot of dialogue is usually not ideal in a game, and will try most players' patience. Instead, think of the story as a series of conflicts and obstacles that build upon each other. Place these conflicts within a classic story structure, and a story is created.

A story usually begins as a basic concept or idea. This is generally a broad idea that places some character, in some situation, in some setting. For example, a concept might be that a woman wakes up on another planet with amnesia and must find a way to get back home. Meanwhile, she uncovers a devious plot to dominate earth and finds a way to stop it. Or a rookie cop discovers that he has the power to read minds and takes down the mob without letting his supervisors become aware of his power. These are the basic premises of the games and the details can be fleshed out later.

By following basic story structure when writing for the game, the writer is much more likely to provide the player what she expects and the tools to be satisfied. While other story formulas exist, and an experienced storyteller can take creative license and try some new methods, the story structure outlined here is simple and fairly easy to implement. The basic story structure goes like this: Begin with the inciting incident, followed by rising action, then a heart-stopping climax, and finally, a satisfying resolution.

Inciting Incident → Rising Action → Climax → Resolution

Inciting Incident

Every story begins with what is called an "inciting incident," where the major conflict of the story is introduced. Up until this point, the character's life was going along a predictable path, then, bam, some event occurs that throws things out of balance. This is the hero answering the call to adventure. The moment when Jack's plane crashes in 2K Games' *BioShock* is an example of an inciting incident, and it is at this moment his character is thrust into the story, and the wheels begin turning. You also see this in Bungie's *Halo*, when the Covenant attacks your ship. From this point forward, the hero's path has been irrevocably changed, and the

story begins.
It is important at this stage to really grab the player's interest. For many gamers, if they are not drawn in within the first 10 minutes, they are lost and may not choose to ever attempt to play the game again. This means beginning the game with immediate action and conflict. It can be brief and minor, but ideally, there should be an immediate obstacle for the player to overcome right from the outset.

Rising Action

Following the inciting incident, the character begins to experience a series of conflicts that drive the story forward. This is the *rising action*. It is during this time that the majority of the gameplay occurs, with the character encountering numerous obstacles and challenges as he journeys to his end goal.
In a book, much of the conflict is internal and takes place within the character's own mind. With a screenplay, the majority of the conflict is usually interpersonal and is played out with conflict between the characters. With games, however, while the other types of conflict exist, the majority of the conflict is going to be external and come from the environment. It is important, therefore, to make the environment rich and filled with conflict. This can come in the form of puzzles to solve, enemies to fight, people to coerce, resources to gather, and many other ways, but what is key here is to keep the player moving through the story.
It is during this portion that the players, through the game narrative, discover who they are and what they are capable of. They also find out who the enemies are and why they are fighting them. Basically, the rising action and the conflict uncovered provide justification to the players to keep going and a reason to continue pushing the A button.
It is important for the writer to set up goals for the character during this portion and to make these goals meaningful [Krawczyk06]. Goals can be given meaning and significance by making them about human wants and needs and relating them back to the story. For example, sending the character out to find a hat has so much more significance if it is a special hat that has the power to make women fall in love with the hero. Later, that hat creates a scene of conflict between two NPC women in the story fighting for the hero's attentions. Don't send the character out on meaningless, empty quests to fill time. If you want your character to collect pink orbs or silver coins, give him a reason why these are useful and a way for him to use these items in the game. Make the quests and missions creative and interesting and keep them relevant to the story and to the character.

Pacing

Pacing is very important here. You want to build tension and suspense within the player. This tension and suspense can be released in small revelations to the character, and this is actually desirable to maintain motivation; however, there needs to be an overall sense of building suspense and tension. There needs to be a balance, revealing enough to the character to keep her intrigued and to reward her, but not so much that she feels she will know how it ends already. This is done by the character continually resolving small conflicts within the game, but not being allowed to solve the main conflict until the end. Furthermore, a good formula for achieving this is to put the cut scenes and slower action points after high action points, such as a dramatic confrontation, a huge revelation, or a boss fight.
Another way to keep the pace up is to use a concept borrowed from screenplays: *Come late, leave early.* What this means is to start the scene at the point of action and end it at the moment the action is resolved. For example, if the scene is to take place at a sporting event, the audience doesn't need to see the characters park their car, buy their tickets, and find their seats. Instead, the audience should be dropped right where the action starts—in this case, in

the stadium.
The same methods apply at the end of the scene. The audience doesn't need to see anything past when the action stops, and doesn't want to watch the characters exit the stadium if it is not furthering the plot. In this case, the action stops and then the next scene starts at a new location. With games, this would translate to having enemies and obstacles right from the beginning of any transition period. This inserting of characters at action points will keep the pace from slowing and the player's interest from waning. This is not to say you shouldn't ever slow the pace, but just make sure there are some sort of obstacles at the beginning of a new story segment, and that the character can leave the module once the final action piece is finished.

Climax

All of this rising action leads to the next element in the story, which is the climax. Everything so far in the game has led up to this event. The climax is the point in the game story where the major conflict of the story is resolved and all of the player's questions get answered. This is the point where all the clues have led and the hero slays his archenemy, or rescues the princess, or disables the bomb and saves the world. Players expect this portion of the narrative to be intense and worth the investment they have given up to this point. This means the stakes should be high and the drama intense. The players need to feel like they have won. It is up to the writer to make sure that the clues and story elements that lead to this moment make sense and that it connects to the rest of the story.

Resolution

The final portion of the game narrative is the resolution. This is usually a brief portion of the narrative that gives the happy ending. It shows the hero enjoying his success and is the reward for having won. This can take place in a full motion video (FMV) or just a brief cheer from the hero. Primarily, it is just to satisfy the players that have succeeded, and it helps them bring closure to the story. As most players feel finished after the climax, this is not a time to introduce new conflicts.

Plot Types

The plot of a game is revealed in segments. Think of these segments like scenes in a play or film. A series of actions take place within one scene; then the characters move onto the next scene or series of events. Many games do this using a level system. A player completes one level and then moves forward through the subsequent levels. Or a game might employ a series of missions for the character to complete or dungeons for him to conquer. Whatever the method used, these segments can be viewed as *modules* that contain conflict, action, and plot. This is different from modular storytelling, which will be described later in this section. How these modules are arranged dictates the way the player will experience the plot.
There is often contention among designers about which method of arrangement is superior. The thinking is that sandbox or modular storytelling is somehow superior to the linear plot; however, this is not always the case. Both types of games can create a successful entertainment experience and sell well on the market. A real-life example would be the Expedition Everest ride in Disneyworld's Animal Kingdom. A great attraction, it leads its riders on rails through a brief and exciting encounter. People have lined up for hours to get a chance to experience this. Contrast this to the actual Mt. Everest experience. Sure, the real experience is going to be more immersive, but it is also much more expensive, time consuming, dangerous, and requires far more fortitude. Currently, there is a market for both types of experiences. Again, it all depends on the target audience.

Linear Plots

The linear plot is the most simple of the plot lines. Picture a series of modules placed in order, from beginning to end. In this type of narrative, the plot points are revealed in a predetermined sequential manner. Once a player has accomplished all of the goals in one module, he must move to the next in the sequence to progress the story. The player can still interact with the game world; however, his interactions will not change how the story unfolds or which game module comes next. The majority of the content in a linear story is explicit and created by the game writers and game designers. The experience variation from player to player in this type of game is minimal. Most players will have a fairly similar and predictable experience.

Narrative is often delivered between the modules in the form of cut scenes, briefings, or voice-overs. Reverse movement through the modules may or may not be allowed, but once a section is cleared, no new plot elements or challenges should be available to the player. Games of this sort are often compared to an amusement park ride on a rail, with one clear path through the fun. In a linear game, there is only one prescribed ending.

Linear narrative in games often gets a bad rap. However, there are times when this type of story best fits the game. For one thing, they are cheaper to make. It is familiar and comfortable to both the player and writer, and keeps control of the story in the hands of the author [Bateman07]. Also, they usually follow a winning formula that will have a guaranteed audience. There are times when a gamer doesn't necessarily want to think or solve puzzles and just wants to be taken on a ride. Games of this nature are probably not likely to win awards for their narrative, but they do have a place in the gaming world, and it is still important for them to have good writing. A good example of a well-done linear game is *Call of Duty 4*. The game sends the character out on missions that are completed in order and down hallways with doors that cannot be opened. Players are left not caring that they have no choices, as they happily blast away the enemy on an adrenaline-pumping ride.

Branching Plots

Implementing a branching plot is a tempting approach to take in order to have a player's choices make a large impact on the development of the story. Depending on which choice a player makes at critical junctures in the game story, the plot will play out completely differently. Branching plots contain multiple endings and have no clear spine, but rather, several spines. An implementation of this type of plot is seen in the game *Dragon's Lair* that Ernest Adams, in his 2006 Game Developers Conference lecture, dubbed the "decision tree of death." That is because there was only one correct, golden path through the branching that would lead to the hero's success. All other paths led to the hero's demise and the goal of the game was to uncover the sole correct path through the branches. This is not true of all games, however, and the possibility exists for many successful endings. While effective in creating different experiences for different players and conveying player empowerment, branching has many drawbacks.

One of the drawbacks of a branching plot is the waste of resources. Game elements need to be created for each area of the game, and some players will never even get to experience or enjoy the areas that fall outside of their chosen path. Even if they replay the game several times, there will still be areas that will remain unexplored. Also, the branches can grow to be large and unwieldy very quickly. For example, if a player is given just 12 binary choices, the result would be 2^{12} = 4,096 unique paths. Furthermore, despite the choices, the story is still, at its heart, linear and constrained by the author.

Modified Branching Plots

There are other ways of implementing branching that prevent some of these limitations. These are often referred to as *parallel paths*. For example, there is the branching plot that leads the character back to the spine. The player will have choices and will have varying game experiences, but the road eventually leads back to the spine. This keeps the paths from growing exponentially.

Alternately, the author might only offer a couple of junctures, usually toward the end of the game. Here, after experiencing the same rising action sequence as every other player, the players are presented with a choice. For example, at the end, the hero might have the option of keeping all of the treasure she has accumulated and going evil, resulting in a climax that is different than the one for the player who chooses to return the treasure to the proper owners. This gives the game alternate endings, but still keeps the game fairly constrained and allows for the vast majority of game content to be enjoyed by all players [Sheldon04].

Modular Storytelling

The modular storytelling structure is a relatively new concept [Sheldon04]. It is a method where the modules of the story can be experienced in any order, and the plot is still understood. Quentin Tarantino's *Pulp Fiction* and *Kill Bill* are examples of stories in film that are told out of chronological order and yet the arc of the story is still understood by the audience. In a game, the player has complete control over which area of the story to explore next and can move through the story at his whim. This can be a difficult concept to grasp, and very challenging to implement as it removes some authorial control [Sheldon04]. Furthermore, the twists and turns a character might take navigating through the plot can wind up looking like a ball of yarn. However, it can be done. Imagine, for example, a 10-day vacation. Each day of the vacation, something new and interesting happens. It really doesn't matter in which order the days happen. Each day is a story in and of itself, and they add up to one large experience, or story, called a *vacation*. Stories of this nontraditional type can be very immersive and can be a sound way of granting a player a new experience.

Nonlinear Plots

In contrast to the linear plot on a rail, is the *sandbox* game where the game elements are not as formally scripted. Instead, the player is given an environment and a set of tools to manipulate that environment. Content is implicit and is made up of stories, goals, and objectives that players create for themselves, and these can be completed in any order the players choose. The story comes from what the player thinks and feels while playing the game. This is most often seen in simulation games, such as *The Sims* series.

Quasilinear Plots

Some games combine the elements of linear and nonlinear plots. This can be an effective method of making players feel as if they have free will within the game, while keeping the main story along a prescribed path with one fixed ending. Basically, linear gameplay is being integrated into a nonlinear world [Bateman07].

For example, the player has free access to what would appear to be the entire game world. He can wander the map at will. However, there are dungeons or missions that unfold in a chronological order. The player must complete "dungeon one" before entering "dungeon two." To add to the illusion, the player can enter dungeon two first, but won't have the skills necessary to defeat the monsters. In this way, the player feels like where he or she goes is their choice. In another example of this, the world is open to exploration, but missions are given in a particular order to be carried out within the already open world.

The quasilinear plot can be seen in some of *The Legend Zelda* games and in the *Grand Theft Auto* series. No matter what decisions or actions the character chooses in the main game

world, the outcomes and ending are the same. All players will see the same cut scenes and experience the same basic story. Players might start businesses, purchase clothing or weapons, earn extra money, or take up a hobby, however, despite these decisions, all players will eventually wind up in the same locations and will experience the same game ending. This can be a very effective method of giving players the feeling they are in control, while still keeping the explicit narrative controlled and keeping the story constrained. The effect can be so concealed, that oftentimes these types of games are referred to as *sandbox games*. This is technically incorrect as the linear mission structure forces the player to stay on the path in order to reach the prescribed ending.

Another model that has elements of a linear plot, but is not truly linear, is one that allows the player or character to move away from the main plot and move through side quests and subplots. These side quests might advance the character, add interesting, yet nonessential plot elements, or change a player's faction. Ubisoft's *Far Cry 2* works similarly to this. There are a handful of plot points and transitions that the main character must pass through, such as the character awakening in an old slave outpost after becoming unconscious. Between these plot points, the player can choose which missions to take and in which order, who to help, who to befriend, and which weapons and vehicles to use. Or perhaps the player would like to go on a quest to collect all seven pieces of epic armor, and this quest takes him to all corners of the world. The game can be completed without this armor, but some players will take great pleasure in ensuring that their character has the best gear possible. These side quests will give the game a very open and interactive feel, while still providing linear progress [Chandler 07].

Backstory

The backstory, which consists of all relevant parts of the story that occurred prior to the story opening, is often an important part of whom a character is and why he is where he is and why he has to do what he has to do. However, uncovering a backstory can be tricky. It is often instinctive for a writer to want to present the audience with the backstory all at once in the beginning. Doesn't the player need to know these details to understand the story? Usually not. Oftentimes, giving it all to them at the beginning backfires, and the audience winds up feeling dumped upon and wonders why they are given all this information about a person they have yet to care about. It is equivalent to hearing your Aunt Myrtle drone on and on about a childhood friend. The audience is left thinking "so what," and this is not a good way to grab attention.

It also has the effect of ruining the suspense. Part of the mystery is discovering what led the player to be in his current predicament. It is okay for the audience not to know, and, in fact, it is intriguing to them. It causes them to ask questions and then want to know answers, further engaging them. Another problem with presenting the backstory at the beginning is that the player doesn't know what details are important to focus on, or what details are important from the backstory. This means that key information presented here isn't always absorbed and is often quickly forgotten.

Instead, it is preferable to drop a player in the middle of the action, engaging him immediately, and then have the backstory unfold through plot and narrative elements. This is especially true in games, where the impatient player wants to get right to the action. Though many games in the past have used cinematic cut scenes at the beginning of a game, this is not always the best place for them, as they often go ignored. Some players do enjoy this, so it is okay to include them, but just allow the player the option of skipping the cut scene and make

sure that any essential information in the cut scene is given to the player in another form elsewhere. Don't forget that in games, unlike written stories, the story is there to serve the game and that the goal is active engagement and interactivity, not passive absorption.
So, if there are details from the past that the player needs to know to understand the storyline, allow those details to be uncovered through active gameplay. For example, if you need the player to know that he is an orphan, have him meet someone that he knew in the orphanage. If he is in a destroyed city that was brought down 20 years ago in an alien attack, have clues in the environment, such as signs, posters, radio broadcasts, city alerts, and anything that is natural within the environment. *BioShock* does a good job of this in uncovering the details of what happened in the city, Rapture, prior to the start of the game. You can also add NPCs that say things like "Yes, ever since our city was destroyed by aliens 20 years ago, fresh water has been hard to find." This way, the player learns about the backstory actively and while engaged, instead of passively absorbing it.

The Interactive Story

The aim of the interactive story is to grant the player the ability to make choices and develop relationships within the game world [Cooper08]. With the interactive story, no two players will have the exact same experience, and the story is essentially co-written with the player. There are two stories: the explicit game story and the implicit player story. The way the two come together determines how the player experiences the game. Each person that plays the game will come to it with different desires and expectations. Some will run through the game, trying to get to the end as quickly as possible, while others will attempt to experience every available bit of game, exploring every nook, performing every side quest, and talking to every NPC. There is even a group of players that take a perverse pleasure in trying to "break" the story, and others that will run around the world doing odd and nonsensical things just because they can.
The game writer needs to be aware of the different types of play styles and prepare the game world for the numerous ways players will use and abuse it. He also needs to be aware of what actions are available to the character and how these actions can be used in the environment. Will the character have the ability to kick? Can the character kick a box? What happens when a character kicks a box? Can the character kick a bunny? What happens when the character kicks a bunny? Can he kick NPCs, walls, doors, enemies, furniture, or any other game item, and what happens when he does so? Can a character turn on a faucet? What happens if he leaves it on? How about turning on the television or shooting the television? What if the player shoots the television, but the television needs to be used later to deliver game content? The writer needs to be aware of all of these possibilities and plan consequences for these game actions by the character.

Player Agency

A primary element in interactivity and immersion is *player agency*. Player agency is the ability of the player to make decisions within the game world that affect the outcome of events. Player agency can also refer to the way game content is revealed in response to an action performed by the player [Bateman07].
With low-level agency, the player has very minor decision-making abilities. For example, the player can choose where to move her avatar or troops, which methods to use to fight enemies, what equipment to carry, and so on, but, these decisions do not affect the eventual outcome of the game and as long as the player successfully achieves the goals of the level, she will continue on the same game path as every other player that plays the game.

With high-level agency, the decisions a player makes in the game directly affect the path the player will take. This can be seen in Bethesda Game Studios' *Fallout 3*, where the player can choose to either detonate or disarm a weapon of mass destruction. The decision he makes will affect how primary NPCs will interact with him later in the game. Obviously, the more agency that is allowed the player, the more interactive the game will become. Therefore, the ideal in an interactive story is for the player to decide and have control over how he experiences the story.

This also increases his suspension of disbelief, which means that the player will be so immersed in the game world, that details that defy real-world logic are accepted for truth. This is what allows us to believe that Harry Potter is capable of performing magic. Only when someone is truly immersed in the story can he experience this.

The Spine

The *spine* of the game consists of all of the narrative elements that are absolutely necessary for the player to experience in order to complete the game. The game can have many side plots and subquests; however, in the spine are only the plot points that the player must pass through to progress in the game.

With a linear game, the entire narrative is contained in the game spine. On the other end of the spectrum is the sandbox game where there is little or no spine to follow and the player chooses which plot points to experience, if any at all. In the middle are the games that have a clear spine, but the player is allowed to wander away from the path, performing subquests, exploring, and generally interacting with the world. For example, in *Fable 2*, the main character can purchase property, get married, take jobs, and help out NPCs, but none of these actions affect the spine of the game or the overarching plot.

The Golden Path

Along with the spine is the *golden path*. The golden path is the optimum path a player would take through the game in order to experience the game as intended and to experience the maximum rewards. It is the duty of the writer to encourage the player to stay on the golden path and force him to return to the spine, without making it feel unnatural. The player needs to know the reason to move to the next area and perform the next task. It doesn't work to merely tell the player, "now move onto the library," or "now kill the emperor," without giving him a reason to do so. The writer must provide appropriate motivation for the player to execute the next task.

The writer must also have in place a mechanism for dealing with a character that chooses not to execute the next task. What if the player decides he doesn't want to go to the library? The writer must have a plan in place for when things don't go according to plan, because the player's motivation does not always match that of the motivation written for the character. In another case, what if the player somehow fails to pick up on the directions to go to the library next. The player should not be forced to wander around aimlessly. Therefore, hints need to be added to the environment clueing the player into what he should do next, and elements need to be built into the game writing that direct the player back to the golden path.

Keeping the Player on the Path

There are some simple methods for dealing with this issue, but most of them come at the cost of player immersion. For instance, redirection is easily handled if the character has an internal monologue and the character basically tells the player what his or her desires are. This detracts from player immersion, however, and causes a rift between player and character. It points to the obvious discrepancy in player desire vs. character desire. Another simple method is using a map. Here, a line is drawn on a map, or appears along the path in the

game world, that physically directs the player where to go to find the task at hand. Again, this can detract from immersion, but the player still maintains the choice in whether to follow the path or not.

Another less-than-subtle alternative for motivating the player is the use of NPCs. In this case, NPCs might refuse to talk to the player about anything but their next task. Alternately, they might offer reminders, advice, or encouragement that lead the player in the correct direction. The more direct the NPC's dialogue, the more effective it is in pushing the player in the right direction; however, the cost is that it feels less interactive. Preferable are comments where the directions are cleverly embedded in the speech, while the rest of the NPC's speech is creating deeper meaning to the game and its world of characters. The existence of a journal is an additional, frequently employed tactic to keep the player focused. All vital information that is presented to the player is saved in a journal that the player can refer to at any time. That way, if the player somehow misses a key plot element, the journal will document the necessary tidbits and lead the player back to the golden path. This is also useful for players who return to a game after a lengthy break. They can consult the journal and pick up the story right where they left off.

For a game to be truly immersive, more subtle means should be explored. One method employed by theme parks is to place highly visible, intriguing items that draw the players in and interrupt them from their aimless wander. This way, players are guided toward the item, but are still moving toward it of their own free will [Cook09]. Another subtle method is to keep primary objectives on a wider physical path, while side quests are placed, literally, off to the side, with smaller roads leading to them. The size of the road can cue a player consciously or subconsciously where the important things will be found. Whatever methods are employed, they should be as unobtrusive as possible.

Algorithmic Interactive Storytelling

Can a computer algorithm dynamically create unique original stories around what the player does? It's a compelling idea and one that has seen a great deal of exploration from academic AI researchers in a field referred to as *narrative intelligence*. Certainly algorithms can be employed to understand what the player has done and how to bend the story slightly around them, but what is being proposed by many researchers in this field is something much more ambitious.

A few game designers who have been working on this problem, such as Chris Crawford, believe that algorithms can create architecturally valid stories, but that only artists can create stories that are interesting to humans [Crawford99]. To exacerbate the problem, there exists no rigorously objective measure of success for such an algorithm, since a successful story is inherently subjective and can only be accurately judged by humans. However, this might be a straw man argument since the short-term goal at least is not absolutely unique stories, but rather varied and compelling stories based on copious amounts of background material, structure, and pre-engineered characters. Such algorithms can dynamically adapt characters and plan a story, sub-story, or conclusion through recombination and randomness, all based around the actions of the player [Ong03].

The approaches around algorithmic interactive storytelling are extremely varied and diverse. Some approaches concentrate on plot construction while others are character-based. Plot-based systems might use story templates to guide the plot or break the plot into many subplots that can be algorithmically planned. Character-based construction involves changing a character's opinions and beliefs as a result of interaction with the player, thus having the characters drive the story through their modified needs and wants. Since characters and story

are inextricably linked, various approaches must consider both, but the emphasis may be on one or the other.
One of the most successful examples of algorithmic interactive storytelling is the game *Façade*, which can be downloaded and played for free [Mateas09]. In *Façade*, the player is inside an interactive drama where he must engage with a couple, Trip and Grace, who are having marital problems. The player has significant influence over the events that unfold and how the story concludes. This is accomplished through an algorithmic drama manager that guides the characters by adding and removing behaviors and speaking points. The plot's dramatic arc is constructed dynamically by sequencing story "beats" (small segments of story) based on the moment-to-moment interactions with the player [Mateas03, Mateas05]. As a whole, the game *Façade* has had a positive impact on the game industry by proving that an algorithmic interactive story is both feasible and compelling. However, it is still an area of active research, and most publicized algorithmic solutions from researchers remain untested in large commercial games.

Story Mechanisms

In order to move the plot forward, the player must be provided with needed story information. There are several mechanisms to achieve this goal and to ensure the information is conveyed to the player. These include cut scenes, scripted events, artifacts, nonplayer characters, and internal monologues. These mechanisms are usually prompted to occur via a triggered event.

Cut Scenes

Cut scenes have been a part of conveying story elements to the player since the mid eighties with the eruption of the Lucas Arts adventure games. They have ranged from short camera changes, where the player sees something happening elsewhere in the game world, to cinematic masterpieces. They can consist of prerendered full-motion video, created separately from the game engine, or they can directly use the game engine and game graphics. What the various types share though, is that they take control of the action *away* from the player, and should be used with caution.
However, there are situations where cut scenes are called for. For instance, there are many players that enjoy a well-orchestrated cut scene, and oftentimes view them as rewards for completing a level. They also work as a way to slow down the action and create a rest point after a stressful boss battle. This change of pace can be refreshing.
On the other side are the impatient sorts, which will always bypass the cut scene if it is in any way possible. Those types of players view the cut scene as an obstacle to move past as quickly as possible. To satisfy the latter group, it is important to allow the player to fast forward through, or to skip the cut scene. This means that key plot points should not be unveiled in only the cut scene, and that information necessary to complete the game should also be available elsewhere. This could be unveiled through the action, or simply be added to the player's journal if they have one. Just make sure that if you allow them to skip the cut scene, it does not hurt later gameplay.
Allowing players to skip cut scenes means that the resources devoted to them will not be enjoyed by all players, and this could be frustrating to the team that put forth the effort to create the masterpiece. Decide if it would be better to use those resources to enhance gameplay rather than create an expensive and space-hogging cut scene. If the team decides that a cut scene is worth the extra time and money to develop, by all means, include one. Just be aware that for some people, it will detract from, rather than add to, the immersion.

Scripted Events

Scripted events are another way of conveying necessary plot information to the player. During a scripted event, the player generally no longer has entire control over what he sees and does. This can be done quickly—for example, when a character enters a scene and the camera moves to show a battleship full of enemies the player will soon be encountering, or it can be elaborate and show entire conversations or action sequences of NPCs.

In *Call of Duty 2*, an example of a scripted game event occurs when the character is shown carrying a teammate or kicking down a door, which are both actions that the player interface doesn't allow the player to perform. *Half-Life* and its sequel also demonstrate good use of scripted events. As Gordon journeys through the game world, he sees NPCs being dragged into air ducts, scientists fighting off head crabs, and soldiers being dragged away by alien invaders. These events help to immerse the player in the world and bring the world to life.

Fable 2 also shows good use of scripted events. What is interesting about *Fable 2* is that the player still holds control of the camera and can choose what to watch during the event. This also adds to the player feeling more in control. Scripted events don't always take control away from the player. For example, in MMORPGs, in the case of escort missions, the player still controls the character, but the actions of the person they are escorting are scripted and out of the player's control. You can watch what they do, but if you leave them and go too far away, the mission resets.

While a cut scene can be considered a scripted event, and there is some ambiguity of terms at this time, generally, a scripted event is one that is programmed into the scene using the game engine and shows the action that is taking place in the current scene. Usually, a cut scene is prerendered and shows plot pieces that took place in the past or are taking place elsewhere in the world. For the most part, scripted events have been well received in recent games and are a good alternative to conveying story when interactivity is not possible.

A small warning about scripted events and cut scenes: Do not force the player to experience them repeatedly. For example, in a game where the player must restart a level if they fail the objective, do not force them to watch the scripted event each time through. Allow them the choice to skip it, letting them decide if they sufficiently understand the scene. Being forced to watch the same scene each time through can be annoying to the player and should be avoided.

Artifacts

Another way to convey the storyline to the player is through artifacts. Artifacts are items like posters, radio broadcasts, journals, letters, photos, CDs, laptops, and other items that contain information that advance the narrative. Leaving information on artifacts to be discovered by the player leaves the player feeling in control. They can choose when and for how long to view the item, and they decide what significance it has to the plot. The trick here is to keep the player's interaction with the item brief and the amount of information small on each item to prevent the player from being overwhelmed by information.

Nonplayer Characters

Well-written nonplayer characters (NPCs) are essential for creating an immersive experience. A good NPC should possess individual desires, goals, needs, beliefs, and attitudes and should not merely be a fountain of information [Spector07]. Even if the player is never explicitly told what the NPC's specific attitudes and beliefs are, by writing them and a backstory for the NPC, the NPC becomes much more lifelike and believable. If the player empathizes with the characters and understands their motivations, the information that is received by the player becomes much more meaningful. The player feels more motivation to complete the given tasks.

Ideally, each time a player has an encounter with an NPC, the interaction should be unique. In the real world, people don't repeat themselves and respond in exactly the same way each time they are spoken to. Characters that utter the same diatribe each time the player encounters them can quickly put a damper on immersion. Therefore, when a character revisits an area, the NPC should have a new script to follow. Even if the information the NPC is conveying is exactly the same as the last time the player spoke with them, they should say it in a different way. For example, the first time a player asks a question, the NPC responds pleasantly. Each time the player returns, the urgency of the request could increase, or the NPC might begin to exhibit annoyance or accuse the character of having a crush on them. Keeping conversations changing and still appropriate, isn't easy, however, and can involve some high-level artificial intelligence [Krawczyk06].

Some games employ the use of companion characters to join the main character on their journey. These characters are designed to provide pertinent information to the player and keep them on track. *Far Cry 2* features very well-developed companion characters, offering 12 in which each has a different role in the game. The player can choose whom to spend time with, in turn influencing the path of the game. The companions can greatly help in the completion of missions, or might try to lead the hero astray. *Fable 2* features a dog as a companion character. In this case, the companion helps find hidden treasure and alerts the player of oncoming danger. Companion characters can be an innovative and immersive way to transmit needed information to a player.

Internal Monologues

The internal monologue is the self-talk that goes on inside a character's head. If implemented correctly, the internal monologue can be a good method of conveying the story and essential information to the player. This only works if the voice is that of a professional actor and the writing is top-notch. Imagine a character, upon running into a wall, says "Ouch, that hurt" in his head. Later, the player hears the character wonder to himself, "Hmm, what is behind that door?" prompting the player to open the door. This tool for forwarding the story can be seen in *Max Payne* and the *Splinter Cell* series. In *Area 51*, this internal dialogue is presented during load scenes and gives the player insight into the emotional state of the character and creates more empathy toward Ethan Cole's character.

The internal dialogue can also be used to funnel a character back to the spine or golden path of a game. She could say to herself, "Wow, I'm running out of time. I'd better get that bomb disarmed." One thing to be careful of while using this technique is to avoid repetitive dialogue. She should not say, "Wow, I'm running out of time," repeatedly; instead the dialogue should change and become increasingly urgent. It gets annoying to hear the character say the same lines of dialogue repeatedly. Also, lengthy dialogue is to be avoided. The thoughts should be kept brief and to the point. Nonessential thoughts can be included, but they should somehow still relate to the gameplay. A final warning is to not have internal dialogue be triggered during heavy action sequences, unless they directly relate to the action at hand. When a player is heavily engaged, he or she might miss crucial elements of the speech, or find that it distracts them from what they are currently trying to accomplish.

Triggered Events

A *triggered event* is one that is preplanned into the script and is prompted to occur once the player has activated a certain mechanism in the game. For example, if the character opens the kitchen door, it might trigger an onslaught of zombies that come stumbling out of the kitchen. Or if the player kills the boss at the end of a dungeon, it might trigger an NPC to step out of the dark and thank the hero and offer a reward. A trigger could even be as simple as

the character entering a new area. In all examples, the event is prompted by some sort of character action.

Interactive Story Techniques

In an interactive story, it is important that a player feels empowered to make choices. However, creating player choices can be expensive, time-consuming, and generally outside of the scope of the project. However, there are some tricks to fool the players into thinking they have more control than they do and to create the illusion of greater player agency. This is similar to telling a child that she can choose to wear the red dress or the blue dress. The child feels like she made the choice, but by placing parameters on the choices, the parent is the one controlling the decision and making sure the child is attired appropriately [Krawczyk06]. By creating the illusion, the player feels that what he is doing has an effect on the game world; however, these actions do not have any effect on the eventual outcomes in the game.

This technique is used frequently in *Fable 2.* The hero can make many choices along the way. The choices he or she makes will affect things like where they live, who they marry, where they can shop, what they look like, how much money they have, and who their friends are, but not the major plot points. This makes the players feel like it was them, rather than the writers, who created this unique character that they can now control. The open world also creates an arena for a great deal of implicit gameplay, where the player is allowed to create stories and subplots of his own. However, these choices don't change where the story eventually takes the character.

An idea that can be borrowed from modular storytelling is to create a narrative that will still make sense when encountered out of order. Think of each event as a stand-alone component of the larger story. Take each of these events and place them on note cards. Try shuffling the note cards and see if there are events that can occur out of sequence. If this cannot be achieved, work with the events until you can. Once you have some story modules that don't need to be in sequence to further the story, you can allow the player to access them at will. If the note cards fall into groups, it might be beneficial to try to use them to *gate the story*. Gating the story is a method of creating the illusion of agency where the challenges and plot points are grouped, and can be encountered in any order within their grouping [DeMarle07]. This way, while each plot point is not encountered in a linear manner, each group is. One way of keeping a player within a group is not giving the character a type of skill or enough hit points until he has completed all of the challenges within a grouping. He can go to later groupings, but will be unsuccessful in his attempts to defeat the later challenges. A writer can go further and allow different choices to effect what challenges the player will encounter within the grouping; however, inevitably, all characters will wind up experiencing the same number of groupings.

Another trick to making the game world seem more interactive is to make the world more dynamic. Allow the player to tinker with and interact with the game world as much as possible and show the player things that are happening outside of his own story. The story isn't just about the character, but the story exists in a living, breathing world. Let the player do things like flush toilets, turn on faucets, lay down on beds, turn off lights, and mail letters. Any one of these things can be used to trigger a dramatic event, that is, essentially, caused by the player. Imagine for example, the character is allowed to break windows and intentionally breaks a storefront window. Later, when he enters the area, it triggers him overhearing a conversation between the shop owner and the glass repairmen that are now fixing the broken window

about “kids these days.” Alternately, he could flip a switch in an apartment building that shuts down power to the entire building. Later, he might hear a woman on the bus complaining that she is late to work due to the power being turned off in her building. This type of dynamic world is seen in *Grand Theft Auto IV*. It is possible to cause a car crash between two NPCs and then step back and watch the drama unfold. The two parties involved might jump out of their cars and begin fighting and then the people on the street may join in. Emergency vehicles arrive on the scene, and it turns into utter chaos. All of this drama is caused by the main character; however, it really doesn’t directly affect the character or the plot. It just adds to the fun and interactivity of the game.

The Director

Another method for making a story interactive is to employ a director entity. This entity would perform the task of game master from a traditional pencil and paper role-playing game. The director evaluates the current state of the game and decides which set of obstacles to give to the player next [Cooper08]. This was the method employed by *Left 4 Dead* to control pacing and enemy spawns based on the character’s current stress level. This concept could be extrapolated to pertain to storyline. The director would evaluate in real time what obstacle would be most appropriate to present next to the character, based on what the player has previously accomplished. This could also result in multiple endings.

Finally, take full advantage of implicit narrative and the types of stories people create for their character. For example, it is possible in many games, especially the sandbox style, to steal without a direct consequence. It doesn’t affect faction point or alliances; however, it makes the players aware that their character is a thief. “Thief” becomes part of the identity of the character, even though it wasn’t written into the script. Or perhaps there are many abandoned apartments in the city and the player chooses to make one of them a home base. They were never told to do this, but by doing so, they have created ownership within the game. The more open a world is and the more options a player has for creating his or her own story, the less narrative that has to be written into the game.

Characters

Character writing is another responsibility of the game writer. A well-written character can bring a story to life and make the experience more memorable for the player, suspending his disbelief and allowing him to become lost in the game. The goal of a well-written character is to evoke emotion within the player and to cause him to have strong feelings toward the characters. For example, anyone who has played *Portal* will immediately recall GLaDOS, the quirky, psychopathic, robot boss and the frustration she caused the main character. She was so well written that some of her lines of dialogue have reached iconic proportions. These feelings created by characters can provide a strong motivation for the player to move forward through the game. For instance, creating a really vicious and convincing villain will make a player feel enraged toward them, and the villain’s timely defeat will bring a much greater satisfaction to the player than a ho-hum stereotypical villain.

In order for a game world to be convincing and immersive, the characters that exist within that world need to be compelling and three dimensional [Sheldon04]. Because the characters are three dimensional, they need to be built upon a frame to provide support, otherwise they are no more interesting than a pile of goo. A large part of this frame is composed of the character’s backstory. This includes all the major details that brought them to this point in their lives and explains why they are in their current emotional state. The audience need not know all the details of the history, but by providing one for the character, it deepens the character

and makes the other elements of his character make sense [Krawczyk06]. The character needs roots, and these roots should show in what the character says and how the character acts.

The other part of this frame is the character's attributes, such as his motivations, goals, aspirations, attitudes, character flaws, and temperament. Ask yourself, "Why is this person here, and what do they want?" The more layers and depth built into the character, the more alive and human they will seem and the more they will strengthen the player's feeling of immersion. Again, the audience may never overtly see these attributes. However, by the writer knowing these attributes and taking them into consideration when writing a character, the character and their dialogue become that much more believable.

A trick for adding believability and defining and strengthening characters is to deliberately design flaws into the characters. For example, a companion NPC might be scared of dogs, or have a really short attention span, which might get him into a bit of trouble. Adding character flaws can add humor, interest, and realism to the story and further enrich the game world. Character flaws can also be beneficial to add to the hero. A hero that is just a wee bit too arrogant could cause fun things to occur in the storyline as his mouth gets him in trouble. As long as he is a likeable hero, his flaws will not only be tolerated, but will also make him seem more authentic and human.

Additionally, watching NPCs perform and have a life outside of the interaction with the character also makes them more believable. Frequently in games, NPCs seem to just stand in one spot all day, every day, just patiently waiting to talk to the hero. Their only job in the game is to transfer information. This is not only boring, but neglects all sorts of possible opportunities to enhance the game [Sheldon04]. A better alternative would be for the NPC to be busy doing things that are in character whenever he is not actively being spoken to. For example, the NPC could be a gossip, and every time the hero nears him, it triggers an event, and he overhears him relaying some juicy tidbit. Then, instead of clicking on the character to get his attention, the NPC initiates the conversation with the player character. This is much more dynamic, and also allows for later drama when the NPC's gossip could get them both mixed up in some trouble.

Having rich supporting characters can open up new storylines and help with the game story by providing reasons for extra quests and content. This is enhanced by writing characters that the player will feel sympathetic toward and want to help. If the player likes the person sending her on a quest, she will feel much more motivated to help him and will receive more satisfaction upon completing the person's request. The hero and player should have feelings of empathy toward the NPCs they are trying to help and be able to understand their plight. Furthermore, the relationship between the hero and the supporting characters should evolve over time and change upon increasing interaction. If a hero repeatedly visits an NPC, the attitude toward him should change and become increasingly friendly, or wary, depending on the relationship.

When playing a game, the player often projects parts of himself onto the hero. This means that the hero's identity is made up of both the attributes written for him and the attributes of the player. Therefore, it is important to make a hero that a player will be able to identify with. Furthermore, because players bring so much to the character, it is not always vital to create a rich hero. Think about iconic video game characters such as Link, Solid Snake, Samus Aran, and Lara Croft, and you will realize that while compelling, these characters are actually rather shallow and undeveloped in comparison to anything you would see in a novel. The characters are, however, incredibly sympathetic and trustworthy. They possess traits that people want to have for themselves and whatever is lacking, the player fills in with his own personality.

Simplicity can actually work to the advantage of the game writer when writing for the hero, because creating traits in a hero that a player cannot identify with can ruin the experience for the player.
The hero is often the most memorable part of a game, more so than the storyline. In fact, the profit that a well-received character can create for a game studio through commercial franchise can be significant. The most well-known characters are made into all types of commercial products such as action figures, movies, lunchboxes, posters, and even books. There have been occasions where the commercial products produced from a gaming license have produced more profit than the game itself. Because of the power a well-written hero can have, a writer should ensure that the hero is one that will be well liked.

Dialogue

Video games have become notorious for cheesy, corny, and just plain bad dialogue. Hopefully, this will change as studios hire more professional writers and actors. While good dialogue writing is an art, there are some things to be aware of when creating dialogue. For one thing, the writer needs to know if the words will be delivered via audio or text. Spoken dialogue usually requires a different tone than written dialogue. If voice actors are to be used, they need to be aware of the history, background, and goals of the characters they are reading. For the writer, it is not enough that the character and delivery style of the lines is clear in his own mind, he needs to make sure this is clearly conveyed to the actors.
Furthermore, a video game is not a place for lengthy expositions, no matter how well written. Dialogue should be fairly brief and to the point, while still keeping it conversational and in character. A short delivery also makes it more likely that the player will absorb what was said. If the dialogue is too long, much of what was said is often quickly forgotten by the player. By keeping it brief, it maintains the pace of the game and keeps the focus on the necessary facts.
However, players do not usually take kindly to orders without being provided with some sort of motivation to follow them. When giving players directions, it is important to make sure they not only know what it is they have to do, but why it is they have to do it. They need to know, and care, why it is so important to the NPC that they go and find the NPC's lost amulet. It would also be important in this situation to provide a reason for the NPC to want this amulet. Why is it so important to him and how does getting this amulet help him or her achieve their larger goals? By fleshing out the motivations behind an NPC's actions, it creates a more human-like character. It makes the game feel more compelling and authentic.
While the standard method of interacting with an NPC is to walk up to its avatar and click a button that initiates the opening of a dialogue box, there are other methods that can be employed. For instance, instead of the player initiating the conversation by clicking on the NPC, some other, less obvious trigger initiates the conversation, such as getting close to the NPC. Then the NPC is the one that initiates the dialogue. This can be done in interesting ways, like having an NPC flag the character down or follow them around. Or conversations can take place through other means rather than in person, for example, through walkie-talkies, cell phones, e-mail, secret hidden notes, or over loudspeakers. Varying the way a player receives information through dialogue can add interest and keep the game from feeling stale.

Dialogue Trees

The dialogue tree is a commonly used method for experiencing a conversation in a game. Good examples of the use of dialogue trees can be seen in BioWare's *Star Wars: Knights of*

the Old Republic, and *Mass Effect.* A *dialogue tree* is basically a graphical flow chart of conversation where the player chooses what to say to an NPC from a menu. The NPC's response is based on the player's choice, and this initiates a new list of choices for the player to choose from. These can be branching, where previous choices are lost to the player, and each exchange opens up entirely new choices, or they can allow the player to keep options from the previous menus to ensure they don't miss anything. While this type of conversation method can cause a break in the pacing and can sometimes be tedious for a player to work through, it continues to be one of the most widely used methods for interactive conversations in games.

Summary

While traditional writing, such as that for novels or screenplays, shares much with game writing, game writing offers some unique challenges. It is a great deal more complicated than traditional storytelling, with writers having to deal with things like multiple storylines, player choice, and technical limitations. It is imperative that a writer understands these challenges and knows techniques for dealing with them. As more games begin to incorporate stories and genres continue to merge, the challenges will increase. That is why it is of the utmost importance for game writers to be aware of the target audience for the game, as well as the funding and game engine limitations in order to keep their writing within the scope of the project.

Furthermore, there is a growing demand for the interactive story. While traditional storytelling techniques should be understood and incorporated into the game, it is not enough, and further methods for dealing with interactivity issues need to be incorporated into the writing. There are many methods for adding interactivity to the narrative game, and the writer needs to work with the developer to decide which methods will work best for the type of game being developed. Currently, hybrid approaches are the best way to grant the player the illusion of interactivity and player agency, while still keeping the game constrained enough to fit within the limitations of a project. The future, however, might allow for more experimental interactive storytelling techniques that combine artificial intelligence with narrative, creating entirely new gameplay experiences.

Exercises

1. Which type of character needs more depth: the companion character or the hero character? Explain.

2. Give an example of an inciting incident in film. Explain how this pivotal moment changed the direction of the hero or heroine's life.

3. In what game genres are game stories most likely to be well received? In what genres do you think there will be room for increased narrative elements and how do you see them being implemented?
4. Give two reasons why the writing is often the first place developers make budget cuts when funds start to run low.

5 Explain the difference between a branching plot and a modified branching plot.
.

6 Describe player agency and explain why it is important to a story being interactive.
7 What are some ways to make an NPC seem more authentic?

8 Research ways in which artificial intelligence is influencing interactive stories.
.