A STRATEGY GUIDE FOR GAME CREATION

V.3

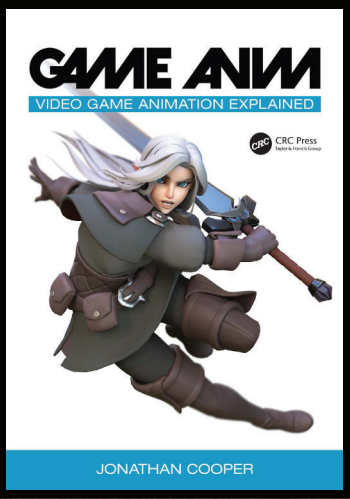
A **CRCPRESS** FREEBOOK

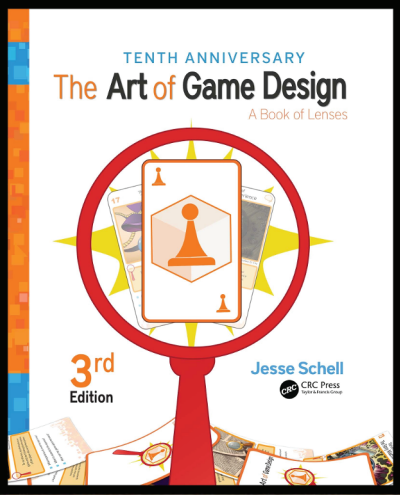
CRC Press
Taylor & Francis Group

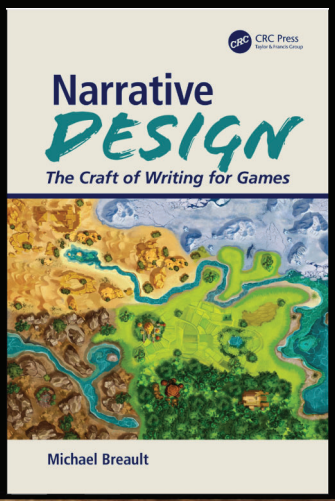
TABLE OF CONTENTS

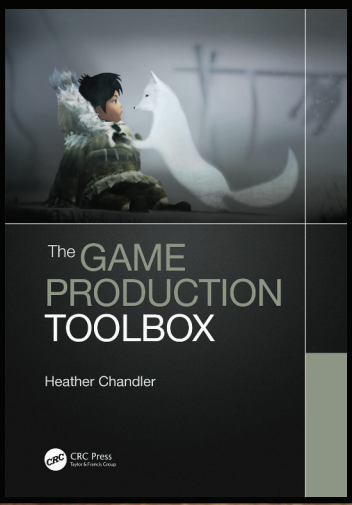
- Introduction
- 1 • The Role of the Game Designer
- 2 • The Designer Creates and Experience
- 3 • Developer and Publisher Overview
- 4 • The Five Fundamentals of Game Animation
- 5 • Story in Games

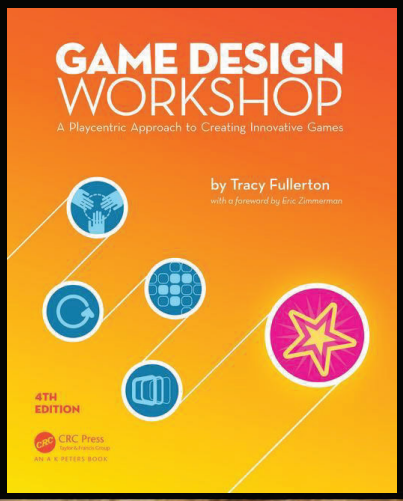
READ THE LATEST ON GAMING AND ANIMATION WITH THESE KEY TITLES











VISIT WWW.ROUTLEDGE.COM

TO BROWSE FULL RANGE OF GAMING TITLES

SAVE 20% AND **FREE STANDARD SHIPPING** WITH DISCOUNT CODE **UBM18**

Introduction

Game Design Workshop *puts you to work prototyping, playtesting, and revising your own games with time-tested methods and tools. These skills will provide the foundation for your career in any facet of the game industry including design, producing, programming, and visual design.*

Presents over 100 sets of questions, or different lenses, for viewing a game's design. Written by one of the world's top game designers, ***The Art of Game Design*** *describes the deepest and most fundamental principles of game design, demonstrating how tactics used in board, card, and athletic games also work in video games. It provides practical instruction on creating world-class games that will be played again and again.*

The Game Production Toolbox *focuses on the nuts and bolts of producing interactive content and how you can organize and support the creative, technical, and business efforts that are all part of interactive game development. This book isn't going to tell you how to design a game or what technologies to use. Instead it provides techniques for and insights into managing, from concept to release, all the pieces that must come together in order to get a game into the hands of a player.*

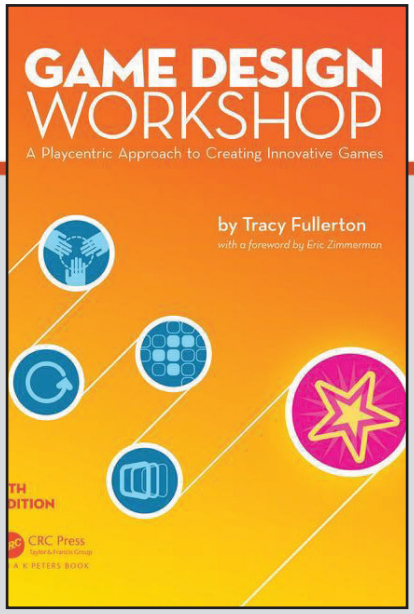
What makes the difference between great video game animation and the purely functional, and how does this relatively new medium of non-linear animation creation differ from the more traditional fields of film and television? ***Game Anim*** *de-mystifies the animation side of game development, explaining every step of the process while providing valuable insights and work philosophies for creating the best game animation for beginners and professionals alike.*

Narrative designers and game designers are critical to the development of digital and analog games. ***Narrative Design*** *provides a detailed look at the work writers and designers perform every day on game development projects. It includes practical advice on how to break into the game industry as a writer or game designer. Readers can use the templates and detailed instructions provided here to create lively portfolios that will help open the door to jobs in the game industry.*

CHAPTER

1

THE ROLE OF THE GAME DESIGNER



This chapter is excerpted from

Game Design Workshop

A Playcentric Approach to Creating Innovative Games, Fourth Edition

by Elizabeth A. Rozanski, John E. Rush.

Learn more

Chapter 1
The Role of the Game Designer

The game designer envisions how a game will work during play. She creates the objectives, rules, and procedures; thinks up the dramatic premise and gives it life; and is responsible for planning everything necessary to create a compelling player experience. In the same way that an architect drafts a blueprint for a building or a screenwriter produces the script for a movie, the game designer plans the structural elements of a system that, when set in motion by the players, creates the interactive experience.

As the impact of digital games has increased, there has been an explosion of interest in game design as a career. Now, instead of looking to Hollywood and dreaming of writing the next blockbuster, many creative people are turning to games as a new form of expression.

But what does it take to be a game designer? What kinds of talents and skills do you need? What will be expected of you during the process? And what is the best method of designing for a game? In this chapter, I'll talk about the answers to these questions and outline a method of iterative design that designers can use to judge the success of gameplay against their goals for the player experience throughout the design and development process. This iterative method, which I call the "playcentric" approach, relies on inviting feedback from players early on and is the key to designing games that delight and engage the audience because the game mechanics are developed from the ground up with the player experience at the center of the process.

An Advocate for the Player

The role of the game designer is, first and foremost, to be an advocate for the player. The game designer must look at the world of games through the player's eyes. This sounds simple, but you'd be surprised how often this concept is ignored. It's far too easy to get caught up in a game's graphics, story line, or new features and forget that what makes a game great is solid gameplay. That's what excites players. Even if they tell you that they love the special effects, art direction, or plot, they won't play for long unless the gameplay hooks them.

As a game designer, a large part of your role is to keep your concentration focused on the player experience and not allow yourself to be distracted by the other concerns of production. Let the art director worry about the imagery, the producer stress over the budget, and the technical director focus on the engine. Your main job is to make sure that when the game is delivered, it provides superior gameplay.

When you first sit down to design a game, everything is fresh and, most likely, you have a vision for

what it is that you want to create. At this point in the process, your view of the game and that of the eventual new player are similar. However, as the process unfolds and the game develops, it becomes increasingly difficult to see your creation objectively. After months of testing and tweaking every conceivable aspect, your once-clear view can become muddled. At times like this, it's easy to get too close to your own work and lose perspective.

Playtesters

It is in situations like these when it becomes critical to have playtesters. Playtesters are people who play your game and provide feedback on the experience so that you can move forward with a fresh perspective. By watching other people play the game, you can learn a great deal.

Observe their experience and try to see the game through their eyes. Pay attention to what objects they are focused on, where they touch the screen or move the cursor when they get stuck or frustrated or bored, and write down everything they tell you. They are your guides, and it's your mission to have them lead you inside the game and illuminate any issues lurking below the surface of the design. If you train yourself to do this, you will regain your objectivity and be able to see both the beauty and the flaws in what you've created.

Many game designers don't involve playtesters in their process, or, if they do, it's at the end of production when it's really too late to change the essential elements of the design. Perhaps they are on a tight schedule and feel they don't have time for feedback. Or perhaps they are afraid that feedback will force them to change things they love about their design. Maybe they think that getting a playtest group together will cost too much money. Or they might be under the impression that testing is something only done by large companies or marketing people.

What these designers don't realize is that by divorcing their process from this essential feedback opportunity, they probably cost themselves considerable time, money, and creative heartache. This is because games are not a form of one-way

1.1 Playtest group

communication. Being a superior game designer isn't about controlling every aspect of the game design or dictating exactly how the game should function. It's about building a potential experience, setting all the pieces in place so that everything's ready to unfold when the players begin to participate.

In some ways, designing a game is like being the host of a party. As the host, it's your job to get everything ready—food, drinks, decorations, music to set the mood—and then you open the doors to your guests and see what happens. The results are not always predictable or what you envisioned. A game, like a party, is an interactive experience that is only fully realized after your guests arrive. What type of party will your game be like? Will your players sit like wallflowers in your living room? Will they stumble around trying to find the coatroom closet? Or will they laugh and talk and meet new people, hoping the night will never end?

Inviting players "over to play" and listening to what they say as they experience your game is the best way to understand how your game is working. Gauging reactions, interpreting silent moments, studying feedback, and matching those with specific game elements are the keys to becoming a professional designer. When you learn to listen to your players, you can help your game to grow.

In Chapter 9 on page 277, when I discuss the playtesting process in detail, you'll learn methods and procedures that will help you hold professional-quality

1.2 More playtest groups

playtests and make the most of these tests by asking good questions and listening openly to criticism. For now, though, it's just important to know that playtesting is the heart of the design process explored in this book and that the feedback you receive during these sessions can help you transform your game into a truly enjoyable experience for your players.

Like any living system, games transform throughout their development cycle. No rule is set in stone. No technique is absolute. No particular scheme is the right one. If you understand how fluid the structures are, you can help mold them into the desired shape through repeated testing and careful observation. As a game designer, it's up to you to evolve your game into more than you originally envisioned. That's the art of game design. It's not locking things in place; it's giving birth and parenting. No one, no matter how smart, can conceive and produce a sophisticated game from a blank sheet of paper and perfect it without going through this process. And learning how to work creatively within this process is what this book is all about.

Exercise 1.1: Become a Tester

Take on the role of a tester. Go play a game and observe yourself as you play. Write down what you're doing and feeling. Try to create one page of detailed notes on your behaviors and actions. Then repeat this experience while watching a friend play the same game. Compare the two sets of notes and analyze what you've learned from the process.

Throughout this book, I've included exercises that challenge you to practice the skills that are essential to game design. I've tried to break them down so that you can master them one by one, but by the end of the book, you will have learned a tremendous amount about games, players, and the design process. And you will have designed, prototyped, and playtested at least one original idea of your own. I recommend creating a folder, either digital or analog, of your completed exercises so that you can refer to them as you work your way through the book.

Passions and Skills

What does it take to become a game designer? There is no one simple answer, no one path to success. There are some basic traits and skills I can suggest, however. First, a great game designer is someone who loves to create playful situations. A passion for games and play is the one thread all great designers have in common. If you don't love what you're doing, you'll never be able to put in the long hours necessary to craft truly innovative games.

To someone on the outside, making games might seem like a trivial task—something that's akin to playing around. But it's not. As any experienced designer can tell you, testing your own game for the ten thousandth time can become work, not play. As the designer, you have to remain dedicated to that ongoing process. You can't just go through the motions. You have to keep that passion alive in yourself, and in the rest of the team, to make sure that the great gameplay you envisioned in those early days of design is still there in the exhausting, pressure-filled final days before you lock production. To do that, you'll need to develop some other important skills in addition to your love of games and your understanding of the playcentric process.

Communication

The most important skill that you, as a game designer, can develop is the ability to communicate clearly and effectively with all the other people who will be working on your game. You'll have to "sell" your game many times over before it ever hits the store shelves: to your teammates, management, investors, and perhaps even your friends and family. To accomplish this, you'll need good language skills, a crystal-clear vision, and a well-conceived presentation. This is the only way to rally everyone involved to your

1.3 Communicating with team members

cause and secure the support that you'll need to move forward.

But good communication doesn't just mean writing and speaking—it also means becoming a good listener and a great compromiser. Listening to your playtesters and to the other people on your team affords fresh ideas and new directions. Listening also involves your teammates in the creative process, giving them a sense of authorship in the final design that will reinvest them in their own responsibilities on the project. If you don't agree with an idea, you haven't lost anything, and the idea you don't use might spark one that you do.

What happens when you hear something that you don't want to hear? Perhaps one of the hardest things to do in life is compromise. In fact, many game designers think that compromise is a bad word. But compromise is sometimes necessary, and if done well, it can be an important source of creative collaboration.

For example, your vision of the game might include a technical feature that is simply impossible given the available time and resources. What if your programmers come up with an alternative implementation for the feature, but it doesn't capture the essence of the original design? How can you adapt your idea to the practical necessities in such a way as to keep the gameplay intact? You'll have to compromise. As the designer, it's your job to find a way to do it elegantly and successfully so that the game doesn't suffer.

Teamwork

Game production can be one of the most intense collaborative processes you'll ever experience. The interesting and challenging thing about game development teams is the sheer breadth of types of people who work on them. From the hardcore computer scientists, who might be designing the AI or graphic displays, to the talented illustrators and animators who bring the characters to life, to the money-minded executives and business managers who deliver the game to its players, the range of personalities is incredible.

1.4 Team meeting

As the designer, you will interact with almost all of them, and you will find that they all speak different professional languages and have different points of view. Overly technical terms may not translate well to artists or the producer, while the subtle shadings of a character sketch might not be instantly obvious to a programmer. These are generalizations, of course, and many team members may come from multidisciplinary backgrounds, but you can't always count on that. So a big part of your job, and one of the reasons for your documents and specifications, is to serve as a sort of universal translator, making sure that all of these different groups are, in fact, working on the same game.

Throughout this book, I often refer to the game designer as a single team member, but in many cases, the task of game design is a team effort. Whether there is a team of designers on a single game or a collaborative environment where the visual designers, programmers, or producer all have input to the design, the game designer rarely works alone. In Chapter 12 on page 391, I will discuss team structures and how the game designer fits into the complicated puzzle that is a development team.

Process

Being a game designer often requires working under great pressure. You'll have to make critical changes to your game without causing new issues in the process. All too often, a game becomes unbalanced as attempts are made to correct an issue because

the designer gets too close to the work and, in the hopes of solving one problem, introduces a host of new problems. But, unable to see this mistake, the designer keeps making changes, while the problems grow worse, until the game becomes such a mess that it loses whatever magic it once had.

Games are fragile systems, and each element is inextricably linked to the others, so a change in one variable can send disruptive ripples throughout. This is particularly catastrophic in the final phases of development, where you run out of time, mistakes are left unfixed, and portions of the game are amputated in hopes of saving what's left. It's gruesome, but it might help you understand why some games with so much potential seem D.O.A.

The one thing that can rescue a game from this terrible fate is instilling in your team the need for good processes from the beginning. Production is a messy business; it is where ideas can get convoluted and objectives can disappear in the chaos of daily crises. But good process, using the playcentric approach of playtesting, and controlled, iterative changes, which I'll discuss throughout this book, can help you stay focused on your goals, prioritize what's truly important, and avoid the pitfalls of an unstructured approach.

Exercise 1.2: D.O.A.

Take one game that you've played that was D.O.A. By D.O.A., I mean "dead on arrival" (i.e., a game that's no fun to play). Write down what you don't like about it. What did the designers miss? How could the game be improved?

Inspiration

A game designer often looks at the world differently from most people. This is in part because of the profession and in part because the art of game design requires someone who is able to see and analyze the underlying relationships and rules of complex systems and to find inspiration for play in common interactions.

When a game designer looks at the world, he often sees things in terms of challenges, structures, and play. Games are everywhere, from how we manage our money to how we form relationships. Everyone has goals in life and must overcome obstacles to achieve those goals. And, of course, there are rules. If you want to win in the financial markets, you have to understand the rules of trading stocks and bonds, profit forecasts, IPOs, and so forth. When you play the markets, the act of investing becomes very similar

1.5 Systems all around us

to a game. The same holds true for winning someone's heart. In courtship, there are social rules that you must follow, and it's in understanding these rules and how you fit into society that helps you to succeed.

If you want to be a game designer, try looking at the world in terms of its underlying systems. Try to analyze how things in your life function. What are the underlying rules? How do the mechanics operate? Are there opportunities for challenge or playfulness? Write down your observations and analyze the relationships. You'll find there is potential for play all around you that can serve as the inspiration for a game. You can use these observations and inspirations as foundations for building new types of gameplay.

Why not look at other games for inspiration? Well, of course, you can and you should. I'll talk about that in just a minute. But if you want to come up with truly original ideas, then don't fall back on existing games for all your ideas. Instead, look at the world around you. Some of the things that have inspired other game designers, and could inspire you, are obvious: personal relationships, buying and selling, competition in the workplace, and on and on. Take ant colonies, for example: They're organized around a sophisticated set of rules, and there's competition both within the colonies and between competing insect groups. Well-known game designer Will Wright made a game about ant colonies in 1991, SimAnt. "I was always fascinated by social insects," he says. "Ants are one of the few real examples of intelligence we have that we can study and deconstruct. We're still struggling with the way the human brain works. But if you look at ant colonies, they sometimes exhibit a remarkable degree of intelligence."[1] The game itself was something of a disappointment commercially, but the innate curiosity about how the world works that led Wright to ant colonies has also led him to look at ecological systems such as the Gaia hypothesis as inspiration for SimEarth or psychological theories such as Maslow's Hierarchy of Needs as inspiration for artificial intelligence in The Sims. Having a strong sense of curiosity and a passion for learning about the world is clearly an important part of Wright's inspiration as a game designer.

What inspires you? Examine things that you are passionate about as systems; break them down in terms of objects, behaviors, relationships, and so forth. Try to understand exactly how each element of the system interacts. This can be the foundation for an interesting game. By practicing the art of extracting and defining the games in all aspects of your life, you will not only hone your skills as a designer, but you'll open up new vistas in what you imagine a game can be.

Exercise 1.3: Your Life as a Game

List five areas of your life that could be games. Then briefly describe a possible underlying game structure for each.

Becoming a Better Player

One way to become an advocate for players is by being a better player yourself. By "better," I don't just mean more skilled or someone who wins all the time—although by studying game systems in depth, you will undoubtedly become a more skilled player. What I mean is using yourself and your experiences with games to develop an unerring sense for good gameplay. The first step to practicing any art form is to develop a deep understanding of what makes that art form work. For example, if you've ever studied a musical instrument, you've probably learned to hear the relationship between the various musical tones. You've developed an ear for music. If you've studied drawing or painting, it's likely that your instructor has urged you to practice looking carefully at light and texture. You've developed an eye for visual composition. If you are a writer, you've learned to read critically. And if you want to be a game designer, you need to learn to play with the same conscious sensitivity to your own experience and critical analysis of the underlying system that these other arts demand.

The following chapters in this section look at the formal, dramatic, and dynamic aspects of games. Together, the concepts in these chapters form a set of tools that you can use to analyze your gameplay experiences and become a better, or more

articulate, player and creative thinker. By practicing these skills, you will develop a game literacy that will make you a better designer. Literacy is the ability to read and write a language, but the concept can also be applied to media or technology. Being game literate means understanding how game systems work, analyzing how they make meaning, and using your understanding to create your own game systems.

I recommend writing your analysis in a game journal. Like a dream journal or a diary, a game journal can help you think through experiences you've had and to remember details of your gameplay long afterwards. As a game designer, these are valuable insights that you might otherwise forget. It is important when writing in your game journal to try to think deeply about your game experience—don't just review the game and talk about its features. Discuss a meaningful moment of gameplay. Try to remember it in detail—why did it strike you? What did you think, feel, do, and so forth? What are the underlying mechanics that made the moment work? The dramatic aspects? Perhaps your insights will form the basis for a future design, perhaps not. But, like sketching or practicing scales on a musical instrument, the act of writing and thinking about design will help you to develop your own way of thinking about games, which is critical to becoming a game designer.

Exercise 1.4: Game Journal

Start a game journal. Don't just try to describe the features of the game, but dig deeply into the choices you made, what you thought and felt about those choices, and the underlying game mechanics that supports those choices. Go into detail; look for the reasons *why* various mechanics of the game exist. Analyze why one moment of gameplay stands out and not another. Commit to writing in your game journal every day.

Creativity

Creativity is hard to quantify, but you'll definitely need to access your creativity to design great games. Everyone is creative in different ways. Some people come up with lots of ideas without even trying. Others focus on one idea and explore all of its possible facets. Some sit quietly in their rooms thinking to themselves, while others like to bounce ideas around with a group, and they find the interaction to be stimulating. Some seek out stimulation or new experiences to spark their imaginations. Great game designers like Will Wright tend to be people who can tap into their dreams and fantasies and bring those to life as interactive experiences.

Another great game designer, Nintendo's Shigeru Miyamoto, says that he often looks to his childhood and to hobbies that he enjoys for inspiration. "When I was a child, I went hiking and found a lake," he says. "It was quite a surprise for me to stumble upon it. When I traveled around the country without a map, trying to find my way, stumbling on amazing things as I went, I realized how it felt to go on an adventure like this."[2] Many of Miyamoto's games draw from this sense of exploration and wonder that he remembers from childhood.

Think about your own life experiences. Do you have memories that might spark the idea for a game? One reason that childhood can be such a powerful inspiration for game designers is that when we are children, we are particularly engrossed in playing games. If you watch how kids interact on a playground, it's usually through gameplaying. They make games and learn social order and group dynamics from their play. Games permeate all aspects of kids' lives and are a vital part of their developmental process. So if you go back to your childhood and look at things that you enjoyed, you'll find the raw material for games right there.

Exercise 1.5: Your Childhood

List ten games you played as a child, for example, hide and seek, four square, and tag. Briefly describe what was compelling about each of those games.

1.6 You Don't Know Jack

1.7 Beautiful katamari and tamakorogashi

Creativity might also mean putting two things together that don't seem to be related—like Shakespeare and the Brady Bunch. What can you make of such a strange combination? Well, the designers of You Don't Know Jack used silly combinations of high- and low-brow knowledge like this to create a trivia game that challenged players to be equally proficient in both. The result was a hit game with such creative spark that it crossed the usual boundaries of gaming, appealing to players old and young, male and female.

Sometimes creative ideas just come to you, and the trick is to know when to stand by a game idea that seems far-fetched. Keita Takahashi, designer of the quirky and innovative hit game Katamari Damacy, was given an assignment while working at Namco to come up with an idea for a racing game. The young artist and sculptor wanted to do something more original than a racing game, however, and says he just "came up with" the idea for the game mechanic of a sticky ball, or katamari, that players could roll around, picking up objects that range from paper clips and sushi to palm trees and policemen. Takahashi has said inspiration for the game came from sources as wildly different as the paintings of Pablo Picasso, the novels of John Irving, and Playmobil brand toys, but it is also clear that Takahashi has been influenced by Japanese children's games and sports such as tamakorogashi (ballroller) as a designer and is thinking beyond digital games for his future creations. "I would like to create a playground for children," he said. "A normal playground is flat but I want an undulating one, with bumps."[3]

I recently designed a game about Henry David Thoreau's time at Walden Pond. I was inspired by his writings and by the thought that underlying his philosophical experiment was an interesting set of rules that he was "playing by" when he set out to "live deliberately." The game took ten years to make and required a deep commitment to the original idea over those years. When we started making it, the idea of an indie game "about" something like a philosopher's experiment in living was considered somewhat strange and new. Today, personal games, and games about ideas or experiences, are relatively common, especially in the indie space.

Our past experiences, our other interests, our relationships, and our identity all come into play when trying to reach our creativity. Great game designers find a way to tap into their creative souls and bring forth the best parts in their games. However you do it, whether you work alone or in a team, whether you read books or climb mountains, whether you look to other games for inspiration or to life experiences, the bottom line is that there's no single right way to go about it. Everyone has a different style for coming up with ideas and being creative. What matters is not the spark of an idea but what you do with that idea once it emerges, and this is where the playcentric process becomes critical.

A Playcentric Design Process

Having a good solid process for developing an idea from the initial concept into a playable and satisfying game experience is another key to thinking like a game designer. The playcentric approach I will illustrate in this book focuses on involving the player in your design process from conception through completion. By that I mean continually keeping the player experience in mind and testing the gameplay with target players through every phase of development.

Setting Player Experience Goals

The sooner you can bring the player into the equation, the better, and the first way to do this is to set "player experience goals." Player experience goals are just what they sound like: goals that the game designer sets for the type of experience that players will have during the game. These are not features of the game but rather descriptions of the interesting and unique situations in which you hope players will find themselves. For example, "players will have to cooperate to win, but the game will be structured so they can never trust each other," "players will feel a sense of happiness and playfulness rather than competitiveness," or "players will have the freedom to pursue the goals of the game in any order they choose."

Setting player experience goals up front, as a part of your brainstorming process, can also focus your creative process. Notice that these descriptions do not talk about how these experience goals will be implemented in the game. Features will be brainstormed later to meet these goals, and then they will be playtested to see if the player experience goals are being met. At first, though, I advise thinking at a very high level about what is interesting and engaging about your game to players while they are playing and what experiences they will describe to their friends later to communicate the high points of the game.

Learning how to set interesting and engaging player experience goals means getting inside the heads of the players, not focusing on the features of the game as you intend to design it. When you're just beginning to design games, one of the hardest things to do is to see beyond features to the actual game experience the players are having. What are they thinking as they make choices in your game? How are they feeling? Are the choices you've offered as rich and interesting as they can be?

Prototyping and Playtesting

Another key component to playcentric design is that ideas should be prototyped and playtested early. I encourage designers to construct a playable version of their idea immediately after brainstorming ideas. By this I mean a physical prototype of the core game mechanics. A physical prototype can use paper and pen or index cards or even be acted out. It is meant to be played by the designer and her friends. The goal is to play and perfect this simplistic model before a single programmer, producer, or graphic artist is ever brought onto the project. This way, the game designer receives instant feedback on what players think of the game and can see immediately if they are achieving their player experience goals.

This might sound like common sense, but in the industry today, much of the testing of the core game mechanics comes later in the production cycle, which can lead to disappointing results. Because many games are not thoroughly prototyped or tested early, flaws in the design aren't identified until late in the process—in some cases, too late to fix. People in the industry are realizing that this lack of player feedback means that many games don't reach their full potential, and the process of developing games needs to change if that problem is to be solved. The work of professional user research experts like Nicole Lazzaro of XEODesign and Dennis Wixon of Microsoft (see their sidebars on pages 282 and 303) is becoming more and more important to game designers and publishers in their attempts to improve game experiences, especially with the new, sometimes inexperienced, game players that are being attracted to platforms like smartphones or tablets. You don't need to have access to a professional test lab to use the playcentric approach. In

Designers You Should Know

The following is a list of designers who have had a monumental impact on digital games. The list was hard to finalize because so many great individuals have contributed to the craft in so many important ways. The goal was not to be comprehensive but rather to give a taste of some designers who have created foundational works and who it would be good for you, as an aspiring designer yourself, to be familiar with. I'm pleased that many designers on the list contributed interviews and sidebars to this book.

Shigeru Miyamoto

Miyamoto was hired out of industrial design school by Nintendo in 1977. He was the first staff artist at the company. Early in his career, he was assigned to a submarine game called Radarscope. This game was like most of the games of the day—simple twitch-game play mechanics, no story, and no characters. He wondered why digital games couldn't be more like the epic stories and fairy tales that he knew and loved from childhood. He wanted to make adventure stories, and he wanted to add emotion to games. Instead of focusing on Radarscope, he made up his own beauty-and-the-beast-like story where an ape steals his keeper's girlfriend and runs away. The result was Donkey Kong, and the character that you played was Mario (originally named Jumpman). Mario is perhaps the most enduring character in games and one of the most recognized characters in the world. Each time a new console is introduced by Nintendo—starting with the original NES machine—Miyamoto designs a Mario game as its flagship title. He is famous for the wild creativity and imagination in his games. Aside from all the Mario and Luigi games, Miyamoto's list of credits is long. It includes the games Zelda, Starfox, and Pikmin.

Will Wright

Early in his career, in 1987, Wright created a game called Raid *on* Bungling Bay. It was a helicopter game where you attacked islands. He had so much fun programming the little cities on the islands that he decided that making cities was the premise for a fun game. This was the inspiration for SimCity. When he first developed SimCity, publishers were not interested because they didn't believe anyone would buy it. But Wright persisted, and the game became an instant hit. SimCity was a breakout in terms of design in that it was based on creating rather than destroying. Also, it didn't have set goals. These things added some new facets to games. Wright was always interested in simulated reality and has done more than anyone in bringing simulation to the masses. SimCity spawned a whole series of titles, including SimEarth, SimAnt, SimCopter, and many others. His game The Sims is currently the bestselling game of all time, and Spore, his most ambitious project yet, explores new design territory in terms of user-created content. See "A Conversation with Will Wright by Celia Pearce" on page 183.

Sid Meier

Legend has it that Sid Meier bet his buddy, Bill Stealey, that within two weeks he could program a better flying combat game than the one they were playing. Stealey took him up on the offer, and together they founded the company Micro Prose. It took more than two weeks, but the company released the title Solo Flight in 1984. Considered by many to be the father of PC gaming, Meier went on to create groundbreaking title after groundbreaking title. His Civilization series has had a fundamental influence on the genre of PC strategy games. His game Sid Meier's Pirates! was an innovative mix of genres—action, adventure, and role-playing—that also

blended real-time and turn-based gaming. His gameplay ideas have been adopted in countless PC games. Meier's other titles include Colonization, Sid Meier's Gettysburg!, Alpha Centauri, and Silent Serv.

Warren Spector

Warren Spector started his career working for board game maker Steve Jackson Games in Austin, Texas. From there, he went on to the paper-based role-playing game company TSR, where he developed board games and wrote RPG supplements and several novels. In 1989, he was ready to add digital games to his portfolio and moved to the developer ORIGIN Systems. There, he worked on the Ultima series with Richard Garriott. Spector had an intense interest in integrating characters and stories into games. He pioneered "free-form" gameplay with a series of innovative titles, including Underworld, System Shock, and Thief. His title Deus Ex took the concepts of flexible play and drama in games to new heights and is considered one of the finest PC games of all time. See his "Designer Perspective" interview on page 27.

Brenda Romero

Brenda Romero began her career at Sir-tech Software as part of the Wizardry role-playing team, where she worked her way up from testing to designer for Wizardry 8. While at Sir-tech, she also worked on the Jagged Alliance and Realms of Arkania series before moving to Atari to work on Dungeons & Dragons. Throughout her career, she has been a passionate advocate for diversity in the industry and was awarded the Ambassador Award from the Game Developers Conference as well as a special British Academy for Film and Television Arts award for her contributions to the industry. On page 88, she discusses her groundbreaking analog game series The Mechanic Is the Message.

Richard Garfield

In 1990, Richard Garfield was an unknown mathematician and part-time game designer. He had been trying unsuccessfully to sell a board game prototype called RoboRally to publishers for seven years. When yet another publisher rejected his concept, he was not surprised. However, this time the publisher, a man named Peter Adkison doing business as Wizards of the Coast, asked for a portable card game that was playable in under an hour. Garfield took the challenge and developed a dueling game system where each card in the system could affect the rules in different ways. It was a breakthrough in game design because the system was infinitely expandable. The game was Magic: The Gathering, and it singlehandedly spawned the industry of collectible card games. Magic has been released in digital format in multiple titles. When Hasbro bought Wizards of the Coast in 1995 for $325 million, Garfield owned a significant portion of the company. See his article "The Design Evolution of Magic: The Gathering" on page 219.

Amy Hennig

Amy Hennig began her career in the game industry working as an artist and animator on games for the NES. While she was working at Electronic Arts as an artist on Michael Jordan: Chaos in the Windy City, the lead designer left the project and Hennig landed the job. Later, she moved to Crystal Dynamics, where she was director, producer, and writer for Legacy of Kain: Soul Reaver. She is well known for her work as a game director and writer on some of the most successful titles in the industry, including the Uncharted series for

Naughty Dog and Sony. She has been awarded two Writers Guild of America Video Game Writing Awards in addition to numerous other awards for her work on the Uncharted games. She describes her writing work on this series as being on the "bleeding edge" of the genre of cinematic video games.

Peter Molyneux

The story goes that it all started with an anthill. As a child, Peter Molyneux toyed with one—tearing it down in parts and watching the ants fight to rebuild, dropping food into the world and watching the ants appropriate it, and so on. He was fascinated by the power he had over the tiny, unpredictable creatures. Molyneux went on to become a programmer and game designer and eventually the pioneer of digital "god games." In his breakout title, Populous, you act as a deity lording it over tiny settlers. The game was revolutionary in that it was a strategy game that took place in real time, as opposed to in turns, and you had indirect control over your units. The units had minds of their own. This game and other Molyneux hits had a profound influence on the real-time strategy (RTS) games that were on the horizon. Other titles he has created include Syndicate, Theme Park, Dungeon Keeper, and Black & White.

Gary Gygax

In the early 1970s, Gary Gygax was an insurance underwriter in Lake Geneva, Wisconsin. He loved all kinds of games, including tabletop war games. In these games, players controlled large armies of miniatures, acting like generals. Gygax and his friends had fun acting out the personas of different pieces on the battlefield such as commanders, heroes, and so forth. He followed his inclination of what was fun and created a system for battling small parties of miniatures in a game he called Chainmail. From there players wanted even more control over and more character information about the individual units. They wanted to play the role of single characters. Gygax, in conjunction with game designer Dave Arneson, developed an elaborate system for role-playing characters that was eventually named Dungeons & Dragons. The D&D game system is the direct ancestor of every paper-based and digital RPG since then. The system is directly evident in all of today's RPGs, including Diablo, Baldur's Gate, and World of Warcraft.

Richard Garriott

Richard Garriott—a.k.a. "Lord British"—programmed his first game right out of high school in 1979. It was an RPG called Akalabeth. He sold it on his own through a local computer store in Austin, Texas. The packaging for this first version was a Ziploc bag. Akalabeth later got picked up by a publisher and sold well. Garriott used what he learned to create Ultima, one of the most famous game series of all time. The Ultima titles evolved over the years—each successive one pushing the envelope in terms of both technology and gameplay—eventually bringing the world of the game online. Ultima Online, released in 1997, was a pioneering title in massively multiplayer online worlds. Garriott continues to push the boundaries of online gaming with work on the science fiction MMO Tabula Rasa.

Dona Baily

Dona Baily was a young programmer in 1981 who, along with Ed Logg, created the classic arcade video game Centipede. At the time, when Baily joined Atari's coin-op division, she was the only woman employed there. When given a notebook of ideas for possible games to program, all of which involved "lasering or frying

things," she chose a short description of a bug winding down the screen because, she said, "it didn't seem bad to shoot a bug." Centipede went on to become one of the most commercially successful games from the arcade era's golden age.

Gerald Lawson

Gerald Lawson was an electronic engineer known for his work in the 1970s, designing the Fairchild Channel F video game system and inventing the video game cartridge. The Fairchild Channel F console, while not a commercially successful product, introduced the idea that game software could be stored on swappable cartridges for the first time. Prior to the Channel F, most game systems had the game software programmed into the architecture of the hardware, so games could never be added to or updated. Lawson's invention was so novel that every cartridge he produced had to be approved by the FCC before distribution as new product. Quickly, his invention became the standard for all future game consoles. Lawson was one of the few African-American engineers working in the industry at that time.

Chapter 9, I describe a number of methods you can use on your own to produce useful improvements to your game design.

I suggest that you do not begin production without a deep understanding of your player experience goals and your core mechanics. This is critical because when the production process commences, it becomes increasingly difficult to alter the software design. Therefore, the further along the design and prototyping are before the production begins, the greater the likelihood of avoiding costly mistakes. You can ensure that your core design concept is sound before production begins by taking a playcentric approach to the design and development process.

Iteration

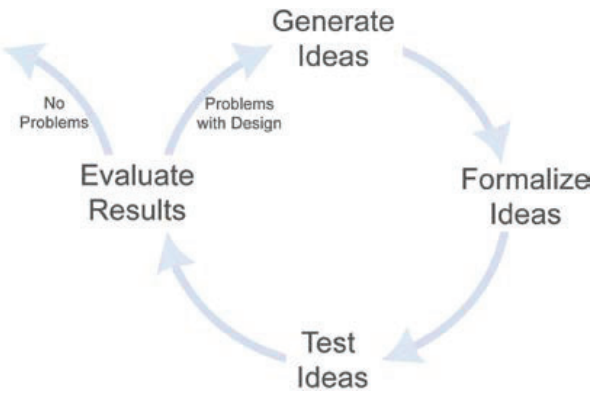
By "iteration" I simply mean that you design, test, and evaluate the results over and over again throughout the development of your game, each time improving upon the gameplay or features, until the player experience meets your criteria. Iteration is deeply important to the playcentric process. Here is a detailed flow of the iterative process that you should go through when designing a game:

- Player experience goals are set.
- An idea or system is conceived.
- An idea or system is formalized (i.e., written down or prototyped).
- An idea or system is tested against player experience goals (i.e., playtested or exhibited for feedback).
- Results are evaluated and prioritized.
- If results are negative and the idea or system appears to be fundamentally flawed, go back to the first step.
- If results point to improvements, modify and test again.
- If results are positive and the idea or system appears to be successful, the iterative process has been completed.

As you will see, this process is applicable during every aspect of game design, from initial conception through final quality assurance testing.

Step 1: Brainstorming

- Set player experience goals.
- Come up with game concepts or mechanics that you think might achieve your player experience goals.
- Narrow the list down to the top three.
- Write up a short, one-page description for each of these ideas, sometimes called a treatment or concept document.



1.8 Iterative process diagram

- Test your written concepts with potential players (you might also want to create rough visual mock-ups of your ideas at this stage to help communicate the ideas).

Step 2: Physical Prototype

- Create a playable prototype using pen and paper or other craft materials.
- Playtest the physical prototype using the process described in Chapters 7 and 9.
- When the physical prototype demonstrates working gameplay that achieves your player experience goals, write a three- to six-page gameplay treatment describing how the game functions.

Step 3: Presentation (Optional)

- A presentation is often made to secure funds to hire the prototyping team. Even if you do not require funding, going through the exercise of creating a full presentation is a good way to think through your game and introduce it to team members and upper management for feedback.
- Your presentation should include demo artwork and a solid gameplay treatment.
- If you do not secure funding, you can either return to step 1 and start over again on a new concept or solicit feedback from your funding sources and work on modifying the game to fit their needs. Because you have not yet invested in extensive artwork or programming, your costs so far should be pretty reasonable, and you should have a great deal of flexibility to make any changes.

Step 4: Software Prototype(s)

- When you have your prototyping team in place, you can begin creating rough digital models of the core gameplay. Often, several software prototypes are made, each focusing on different aspects of the system. Digital prototyping is discussed in Chapter 8 beginning on page 241. (If possible, try to do this entirely with temporary graphics that cost very little to make. This will save time and money and speed up the process.)
- Playtest the software prototype(s) using the method process described in Chapter 9.
- When the software prototype(s) demonstrate working gameplay that achieves your player experience goals, move on to develop plans for the full feature set and levels of the game.

Step 5: Design Documentation

- While you have been prototyping and working on your gameplay, you have probably been compiling notes and ideas for the "real" game. Use the knowledge you've gained during this prototyping stage to develop a full list of goals for the game, which are documented in a way that is useful and accessible for the team.
- Recently, many designers have moved away from creating large static documents for this purpose, moving instead toward online groupware like wikis and smaller, as-needed form documentation because of the flexible, collaborative nature of modern design processes. The design documentation that comes out of your production process should be thought of as a collaboration tool that changes and grows with production.

Step 6: Production

- Work with all team members to make sure your goals are clear and achievable and that the team is on board with the priorities for these goals.

The Iterative Design Process

by Eric Zimmerman, game designer and professor, NYU Game Center

Eric Zimmerman is a game designer and a twenty-year veteran of the game industry. Eric cofounded Gamelab, an award-winning New York City-based studio that helped invent casual games with titles like Diner Dash. Other projects range from the pioneering independent online game SiSSYFiGHT 2000 to tabletop games like the strategy board game Quantum and Local No. 12's card game The Metagame. Eric has also created game installations with architect Nathalie Pozzi that have been exhibited in museums and festivals around the world. He is the coauthor with Katie Salen of Rules of Play and is a founding faculty and arts professor at the NYU Game Center. Also see his article with Nathalie Pozzi on playtesting methods on page 293.

The following excerpt is adapted from a longer essay entitled "Play as Research," which appears in the book Design Research, edited by Brenda Laurel (MIT Press, 2004). It appears here with permission from the author. Iterative design is a design methodology based on a cyclic process of prototyping, testing, analyzing, and refining a work in progress. In iterative design, interaction with the designed system is the basis of the design process, informing and evolving a project as successive versions, or iterations, of a design are implemented. This sidebar outlines the iterative process as it occurred in one game with which I was involved—the online multiplayer game SiSSYFiGHT 2000.

What is the process of iterative design? Test, analyze, refine. And repeat. Because the experience of a player cannot ever be completely predicted, in an iterative process design, decisions are based on the experience of the prototype in progress. The prototype is tested, revisions are made, and the project is tested once more. In this way, the project develops through an ongoing dialogue between the designers, the design, and the testing audience.

In the case of games, iterative design means playtesting. Throughout the entire process of design and development, your game is played. You play it. The rest of the development team plays it. Other people in the office play it. People visiting your office play it. You organize groups of testers that match your target audience. You have as many people as possible play the game. In each case, you observe them, ask them questions, then adjust your design and playtest again.

This iterative process of design is radically different from typical retail game development. More often than not, at the start of the design process for a computer or console title, a game designer will think up a finished concept and then write an exhaustive design document that outlines every possible aspect of the game in minute detail. Invariably, the final game never resembles the carefully conceived original. A more iterative design process, on the other hand, will not only streamline development resources, but it will also result in a more robust and successful final product.

Case Study: SiSSYFiGHT 2000

SiSSYFiGHT 2000 is a multiplayer online game in which players create a schoolgirl avatar and then vie with three to six players for dominance of the playground. Each turn, a player selects one of six actions to take, ranging from teasing and tattling to cowering and licking a lolly. The outcome of an action is dependent on other players' decisions, making for highly social gameplay. SiSSYFiGHT 2000 is also a robust online community. You

can play the game at www.sissyfight.com. In the summer of 1999, I was hired by Word.com to help them create their first game. We initially worked to identify the project's play values: the abstract principles that the game design would embody. The list of play values we created included designing for a broad audience of nongamers, a low technology barrier, a game that was easy to learn and play but deep and complex, gameplay that was intrinsically social, and, finally, something that was in line with the smart and ironic Word.com sensibility.

These play values were the parameters for a series of brainstorming sessions interspersed with group play of computer and noncomputer games. Eventually, a game concept emerged: little girls in social conflict on a playground. While every game embodies some kind of conflict, we were drawn toward modeling a conflict that we hadn't seen depicted previously in a game. Technology and production limitations meant that the game would be turn based, although it could involve real-time chat.

When these basic formal and conceptual questions had begun to be mapped out, the shape of the initial prototype became clear. The very first version of SiSSYFiGHT was played with Post-it Notes around a conference table. I designed a handful of basic actions each player could take, and acting as the program, I "processed" the actions each turn and reported the results back to the players, keeping score on a piece of paper.

Designing a first prototype requires strategic thinking about how to most quickly implement a playable version that can begin to address the project's chief uncertainties in a meaningful way. Can you create a paper version of your digital game? Can you design a short version of a game that will last much longer in its final form? Can you test the interaction pattern of a massively multiplayer game with just a handful of players?

In the iterative design process, the most detailed thinking you need at any moment is that which will get you to your next prototype. It is, of course, important to understand the big picture as well: the larger conceptual, technical, and design questions that drive the project as a whole. Just be sure not to let your design get ahead of your iterative research. Keep your eye on the prize, but leave room for play in your design, for the potential to change as you learn from your playtesting, accepting the fact that some of your assumptions will undoubtedly be wrong.

The project team continued to develop the paper prototype, seeking the balance between cooperation and competition that would become the heart of the final gameplay. We refined the base rule set—the actions a player can take each turn and the outcomes that result. These rules were turned into a specification for the first digital prototype: a text-only version on IRC, which we played hot-seat style, taking turns sitting at the same computer. Constructing that early, text-only prototype allowed us to focus on the complexities of the game logic without worrying about implementing interactivity, visual and audio aesthetics, and other aspects of the game.

SiSSYFiGHT 2000 Interface

While we tested gameplay via the text-only iteration, programming for the final version began in Director, and the core game logic we had developed for the IRC prototype was recycled into the Director code with little alteration. Parallel to the game design, the project's visual designers had begun to develop the graphic

language of the game and chart out possible screen layouts. These early drafts of the visuals (revised many times over the course of the entire development) were dropped into the Director version of the game, and the first rough-hewn iteration of SiSSYFiGHT as a multiplayer online game took shape, inspired by Henry Darger's outsider art and retro game graphics.

As soon as the web version was playable, the development team played it. And as our ugly duckling grew more refined, the rest of the Word.com staff was roped into testing as well. As the game grew more stable, we descended on our friends' dot-com companies after the workday had ended, sitting them down cold in front of the game and letting them play. All of this testing and feedback helped us refine the game logic, visual aesthetics, and interface. The biggest challenge turned out to be clearly articulating the relationship between player action and game outcome: Because the results of every turn are interdependent on each player's actions, early versions of the game felt frustratingly arbitrary. Only through many design revisions and dialogue with our testers did we manage to structure the results of each turn to unambiguously communicate what had happened that round and why.

SiSSYFiGHT 2000 Game Interfaces

When the server infrastructure was completed, we launched the game to an invitation-only beta tester community that slowly grew in the weeks leading up to public release. Certain time slots were scheduled as official testing events, but our beta users could come online anytime and play. We made it very easy for the beta testers to contact us and e-mail in bug reports.

Even with this small sample of a few dozen participants, larger play patterns emerged. For example, as with many multiplayer games, it was highly advantageous to play defensively, leading to standstill matches. In response, we tweaked the game logic to discourage this play style: Any player that "cowered" twice in a row was penalized for acting like a chicken. When the game did launch, our loyal beta testers became the core of the game community, easing new players into the game's social space.

In the case of SiSSYFiGHT 2000, the testing and prototyping cycle of iterative design was successful because at each stage we clarified exactly what we wanted to test and how. We used written and online questionnaires. We debriefed after each testing session. And we strategized about how each version of the game would incorporate the visual, audio, game design, and technical elements of the previous versions, while also laying a foundation for the final form of the experience.

To design a game is to construct a set of rules. But the point of game design is not to have players experience rules—it is to have players experience play. Game design is therefore a second-order design problem in which designers craft play, but only indirectly, through the systems of rules that game designers create. Play

arises out of the rules as they are inhabited and enacted by players, creating emergent patterns of behavior, sensation, social exchange, and meaning. This shows the necessity of the iterative design process. The delicate interaction of rule and play is something too subtle and too complex to script out in advance, requiring the improvisational balancing that only testing and prototyping can provide.

In iterative design, there is a blending of designer and user, of creator and player. It is a process of design through the reinvention of play. Through iterative design, designers create systems and play with them. They become participants, but they do so in order to critique their creations, to bend them, break them, and refashion them into something new. And in these procedures of investigation and experimentation, a special form of discovery takes place. The process of iteration, of design through play, is a way of discovering the answers to questions you didn't even know were there. And that makes it a powerful and important method of design. SiSSYFiGHT 2000 was developed by Marisa Bowe, Ranjit Bhatnagar, Tomas Clarke, Michelle Golden, Lucas Gonze, Lem Jay Ignacio, Jason Mohr, Daron Murphy, Yoshi Sodeka, Wade Tinney, and Eric Zimmerman.

- Staff up with a full team and plan a set of development "sprints" for each of the goals in your plan. Evaluate your game as a team after each sprint to make sure you are still on target with your player experience goals.
- Don't lose sight of the playcentric process during production—test your artwork, gameplay, characters, and so forth as you move along. As you continue to perform iterative cycles throughout the production phase, the problems you find and the changes you make should get smaller and smaller. This is because you resolved your major issues during the prototyping phases.
- Unfortunately, this is the time when most game designers actually wind up designing their games, and this can lead to numerous problems related to time, money, and frustration.

Step 7: Quality Assurance

- By the time the project is ready for quality assurance testing, you should be very sure that your gameplay is solid. There can still be some issues, so continue playtesting with an eye to usability. Now is the time to make sure your game is accessible to your entire target audience.

As you can see, the playcentric approach involves player feedback throughout the production process, which means you'll be doing lots of prototyping and playtesting at every stage of your game's development. You can't be the advocate for the player if you don't know what the player is thinking, and playtesting is the best mechanism by which you can elicit feedback and gain insight into your game. I cannot emphasize this fact enough, and I encourage any designer to rigorously build into any production schedule the means to continually isolate and playtest all aspects of their game as thoroughly as possible.

Prototypes and Playtesting in the Industry

In the game industry today, designers often skip the creation of a physical prototype altogether and jump straight from the concept stage to writing up the design. The problem with this method is that the software coding has commenced before anyone has a true sense for the game mechanics. The reason this is possible is because many games are simply variations on standard game mechanics, so the designers have a good idea of how the game

will work because they've played it, or a variation of it, as another game.

It's important to remember that the game industry is just that: an industry. Taking risks and spending a lot of time and money creating new gameplay mechanics are difficult to reconcile with a bottom line. However, the game industry is changing and growing rapidly, with new platforms that demand innovative designs. This means designing for different types of players outside the traditional gaming audience. New platforms like VR, AR, smartphones, tablets, gestural and multitouch interfaces, and breakout hits like *Pokémon Go* have proven that there is demand from new audiences if the right new kind of gameplay is offered.

While the industry as a whole is extremely skilled at maintaining steady technological innovation and cultivating core audience demand for those innovations, the same isn't true when it comes to developing original ideas in player experience. To meet the demands of new players using game devices in wildly different contexts than a traditional game audience, we are seeing the need for breakthroughs in player experience just as surely as there has always been a need for breakthroughs in technology to drive the industry forward. But it is difficult to design an original game if you skip the physical prototyping process. What happens is that you are forced to reference existing games in the design description? This means your game is bound from the outset to be derivative. Breaking away from your references becomes even more difficult as the production takes off. When your team is in place, with programmers coding and artists cranking out graphics, the idea of going back and changing the core gameplay becomes very difficult.

That is why a number of prominent game designers have begun to adopt a playcentric approach. Large companies such as Electronic Arts have created in-house training in preproduction (see sidebar in Chapter 6, page 175) originally run by Chief Visual Officer Glenn Entis. This workshop includes physical prototyping and playtesting as part of the initial development stage. Entis runs development teams through a series of exercises, one of which is coming up with a quick physical prototype. His advice is make it "fast, cheap, public, and physical. If you don't see people on the team arguing," he says, "you can't know if they are sharing ideas. A physical prototype gets the team talking, interacting."[4]

Chris Plummer, an executive producer at Electronic Arts Los Angeles, says, "Paper prototypes can be a great tool for low-cost ideation and playtesting of game features or systems that would otherwise cost a lot more to develop in software. It's much easier to justify spending the

1.9 Angry Birds Star Wars and Pokémon Go—unconventional markets and players

1.10 USC Games students at work at weekend game jam

resources to realize a game in software after the game framework is developed and refined through more cost-effective means, such as analog prototypes."[5]

Smaller companies often engage in "game jams," events where local independents and students come together for a weekend to generate prototypes for new game projects. The Global Game Jam is an annual worldwide event that brings together tens of thousands of participants to develop innovative game prototypes. By leveraging their local community of independent game designers, small groups and companies are able to jump-start their new ideas in a collaborative environment.

Designing for Innovation

As I mentioned earlier, today's game designers have the challenge—and opportunity—to produce breakthroughs in player experience as part of their basic job description. They will have to do this without taking too many risks in terms of time and money. By innovation, I mean:

- Designing games with unique play mechanics—thinking beyond existing genres of play
- Appealing to new players—people who have different tastes and skills than hard-core gamers
- Designing for new platforms such as smartphones, tablets, and gestural and multitouch interfaces
- Creating games that integrate into daily life, real-world spaces, and the systems around us
- Embracing new business models for games such as free-to-play or subscription
- Trying to solve difficult problems in game design such as:
  - ◊ The integration of story and gameplay
  - ◊ Deeper empathy for characters in games
  - ◊ Creating emotionally rich gameplay
  - ◊ Discovering the relationships between games and learning
- Asking difficult questions about what games are, what they can be, and what their impact is on us individually and culturally

The playcentric approach can help foster innovation and give you a solid process within which to explore these provocative, unusual questions about gameplay possibilities, to try ideas that might seem fundamentally unsound but could have within them the seed of a breakthrough idea, and to craft them until they are playable. Real innovation seldom comes from the first spark of an idea; it tends to come from long-term development and experimentation. By interacting with players throughout the design process, experimental ideas have time to develop and mature.

Conclusion

My goal in this book is to help you become a game designer. I want to give you the skills and tools you'll need to take your ideas and craft them into games that aren't mere extensions of games already on the market. I want to enable you to push the envelope on game design, and the key to doing this is process. The approach you will learn here is about internalizing a playcentric method of design that will make you more creative and productive, while helping you to avoid many of the pitfalls that plague game designers.

The following chapters in this first section will lay out a vocabulary of design and help you to think critically about the games you play and the games you want to design. Understanding how games work and why players play them is the next step to becoming a game designer.

Designer Perspective: Christina Norman

Lead Designer, Riot Games

Christina Norman is an experienced game designer whose credits include Mass Effect (2007), League of Legends (2009), Mass Effect 2 (2011) and Mass Effect 3 (2012).

How did you become a game designer?

I would say I became a game designer at age 9. I was playing Dungeons & Dragons with some kids at school, and our dungeon master moved away. I'd already memorized all the rules, so I was a natural to replace him. This was the starting point of a nine-year-long D&D campaign, and the moment I became a game designer.

The story of how I became employed as a game designer is, of course, entirely different. That story starts with...depression. I had a successful career programming e-commerce web sites, but I felt deeply unfulfilled. I didn't care about what I was doing, so I asked myself—what do you care about? What do you really want to do? The answer was: make games.

I had three things going for me: I was a hardcore gamer, I had created several successful Warcraft 3 mods, and I was a programmer. I applied for a game design job at BioWare and...they rejected me. I applied again as a programmer and they said, okay! After I had been there for a few years I was able to convince the lead designer to give me a shot at game design. Since then it's been all flowers, bunny rabbits, and joy!

On games that have inspired her:

Dungeons & Dragons: This, along with other great pen-and-paper role-playing games, taught me the fundamentals of system design. It was my unquenchable thirst for more Dungeons & Dragons that drove me to CRPGs (what we used to call "computer RPGs").

Nethack (honorable mention to Diablo 2): Nethack is one of the early "roguelike" games. In this vast procedurally generated world, I endlessly pursued the fabled amulet of Yendor. As I descended through the seemingly endless dungeon levels, I marveled at the intricate and complex systems and their many interactions. Years later, Diablo 2 was the first mainstream game I played that captured much of Nethack's strengths, improving it with AAA production values and addictive multiplayer.

Baldur's Gate 2: This game taught me that games can be an exceptional storytelling medium that really makes you feel. Through my adventures I came to truly care for my party members—I wanted to help them achieve their goals! On top of all this, BG2 remains a mastery of systems design and in my opinion is the best realization of D&D in a video game to date.

Master of Orion 2 (honorable mention to Civilization): This was the first 4X (explore, expand, exploit, exterminate) game that completely captivated me. The idea of starting at a single planet, developing the technology of space flight, and ultimately ruling the entire universe was mind blowing.

Everquest: I didn't just play Everquest, I was transformed by it. I entered the virtual world of Norath a role player. I left it a hardcore raider who would eventually achieve world-first boss kills in World of Warcraft. More importantly, through Everquest I developed an appreciation for how deep, strong, and real online social relationships can be.

What is the most exciting development in the recent game industry?

This is an invigorating time to be a game designer. We're experiencing a renaissance in which small games are dominating the creative landscape. The rise of mobile gaming, self-publishing, and fresh game models has created opportunities for small developers to create innovative games that can also be financially successful. League of Legends started as a small game and benefited from these industry dynamics where scrappy challenges really have a shot!

Disruption rocks!

On her design process:

I don't build games for myself. It's easy to build games that you want to play; it's much harder to truly understand the needs of others. Building games so a diverse audience can enjoy them requires a commitment to understanding how others enjoy games.

The first thing I do when I'm designing a game, or a system, is listen to the people I'm building it for. I try to understand what kind of experience will please them. I then relentlessly pursue delivering that experience without compromise.

Do you use prototypes?

I'm a programmer, so code is my paintbrush. When I want to try an idea out, I code it fast and dirty. From there it's test, iterate, test, iterate, test…and when the design works…build it properly. When I do code-based prototyping, I use whatever tools will let me test ideas the quickest.

I'm also a big fan of building physical prototypes. Sometimes it's just faster to build something as a card game, or board game, than to code it.

On a particularly difficult design problem:

Mass Effect was essentially a hardcore RPG dressed as a shooter. Whether you hit enemies or not was determined by an invisible die roll. This meant that even if you aimed perfectly, you could miss, so guns felt weak and unreliable.

For Mass Effect 2 we wanted guns to feel accurate, powerful, and reliable. We disabled the to-hit rolls, but aiming still felt sub-par. This was my unruly introduction to combat design—I learned that making something work a certain way is different than making it feel great. My team studied the great shooters, learned from them, and then we polished our guns until they felt great.

But it wasn't that simple. Making firing guns feel great required adjusting the pacing of gameplay, which required...reinventing pretty much every system in Mass Effect. By the time we were done, we had an entirely different game than the first one, but the results were worth it—ME2 is currently the fourth highest-rated Xbox 360 game of all time on Metacritic.

What are you most proud of in your career?

Reinventing Mass Effect 2's gameplay required more than design. To achieve that goal, I had to achieve buy-in from the team (not an easy task for a designer on her first design project). In the end, I succeeded because I had a strong vision, I communicated it clearly, and I appealed to the team's collective desire to deliver a great experience to our players.

On advice to designers:

Play many games. Play them hardcore. If you get into the game industry, you'll have less time to play games, and so many insights come from your experience as a player.

Go beyond your own insights. Learn to be a better designer by listening to other players. Just watching someone play a game can teach you a great deal about game design.

Listen to your team. Just because someone's title doesn't include the word "designer" doesn't mean they don't have valuable design insights. Some of the best designers I have worked with have producer, programmer, or QA in their title.

Designer Perspective: Warren Spector

Studio Director, OtherSide Entertainment

Warren Spector is a veteran game designer and producer whose credits include Ultima VI (1990), Wing Commander (1990), Martian Dreams (1991), Underworld (1991), Ultima VII (1993), Wings of Glory (1994), System Shock (1994), Deus Ex (2000), Deus Ex: Invisible War (2003), Thief: Deadly Shadows (2004), Disney Epic Mickey (2010), and Disney Epic Mickey 2 (2012).

On getting into the game industry:

I started out, like most folks, as a gamer, back in the day. Back in 1983, I made my hobby my profession, starting out as an editor at Steve Jackson Games, a small board game company in Austin, Texas. There, I worked on TOON: The Cartoon Roleplaying Game, GURPS, several Car Wars, Ogre, and Illuminati games and learned a ton about game design from people like Steve Jackson, Allen Varney, Scott Haring, and others. In 1987, I was lured away by TSR, makers of Dungeons & Dragons and other fine RPGs and board games. 1989 saw me homesick for Austin, Texas, and feeling like paper gaming was a business/art form that had pretty much plateaued. I was playing a lot of early computer and video games at the time, and when the opportunity to work for Origin came up, I jumped at it. I started out there as an associate producer, working with Richard Garriott and Chris Roberts before moving up to full producer. I spent seven years with Origin, shipping about a dozen titles and moving up from associate producer to producer to executive producer.

On game influences:

There have probably been dozens of games that have influenced me, but here are a few of the biggies:

- Ultima IV: This is Richard Garriott's masterpiece. It proved to me (and a lot of other people) that giving players power to make choices enhanced the gameplay experience. And attaching consequences to those choices made the experience even *more* powerful. This was the game that showed me that games could be about more than killing things or solving goofy puzzles. It was also the first game I ever played that made me feel like I was engaged in a dialogue with the game's creator. And that's something I've striven to achieve ever since.
- Super Mario 64: I was stunned at how much gameplay Miyamoto and the Mario team managed to squeeze into this game. And it's all done through a control/interface scheme that's so simple that, as a developer, it shames me. Mario can do maybe ten things, I think, and yet the player never feels constrained—you feel empowered and liberated, encouraged to explore, plan, experiment, fail, and try again, without feeling frustrated. You have to be inspired by the combination of simplicity and depth.
- Star Raiders: This was the first game that made me believe games were more than just a fad or passing fancy, for me and for, well, humanity at large. "Oh, man," I thought, "we can send people places they'll never be able to go in real life." That's not just kid stuff—that's change-the-world stuff. There's an old saying about not judging someone until you've walked a mile in their shoes, you know? Well,

games are like an experiential shoe store for all mankind. We can allow you to walk in the shoes of anyone we can imagine. How powerful is that?

- Ico: Ico impressed me because it proved to me how powerfully we can affect players on an emotional level. And I'm not just talking about excitement or fear, the stuff we usually traffic in. Ico, through some stellar animation, graphics, sound, and story elements, explores questions of friendship, loyalty, dread, tension, and exhilaration. The power of a virtual touch—of the player holding the hand of a character he's charged to protect, even though she seems weak and moves with almost maddening slowness—the power of that touch blew me away. I have to find a way to get at some of that power in my own work. Interestingly, some recent games, like Last of Us and The Walking Dead, have exploited the human need to make contact with and protect another. Clearly, this is an idea games can exploit exceptionally well—an idea that allows us to move people, emotionally, in ways many nongamers and even some gamers thought impossible.
- Suikoden: This little PlayStation role-playing game showed me new ways of dealing with conversation. I had never before experienced Suikoden's brand of simple, straightforward, binary-choice approach—little things like "Do you fight your father or not? Y/N" or "Do you leave your best friend to almost certain death so you can escape and complete your critically important quest? Y/N" will blow you away! In addition, the game featured two other critical systems: a castle-building mechanic and a related player-controlled ally system. The castle-building bit showed me the power of allowing players to leave a personal mark on the world—the narcissistic aspect of game playing. The ally system, which affected what information you got before embarking on quests, as well as the forces/abilities available to you in mass battles, revealed some of the power of allowing each player to author his or her own unique experience. It is a terrific game that has a lot to teach even the most experienced RPG designers in the business.
- One recent game that inspired me, though perhaps not in the way I expected or the creators of the game intended, was The Walking Dead. Playing that game, I was drawn into a narrative, into an experience, that felt more emotionally compelling than maybe any other game I've played. As an experience, the game was magnificent. As a game? I'm not so sure. I think The Walking Dead worked as well as it did because it was unabashedly cinematic—the creators of the game knew exactly where every player would be at all times, what each player would do, exactly how they would do it...In a sense, that meant The Walking Dead was "just" a movie—but a movie that gives an incredibly convincing illusion of interactivity. As a player, I was charmed by it. As a developer, I was aghast that anyone would make a game where developers would never be surprised by anything players did and where no player would ever do anything the creators didn't intend, plan for, and implement. I'm still working through the contradiction inherent in the idea of a game I loved as a player but felt disappointed in as a developer. Any game that is as enjoyable and, albeit inadvertently, thought provoking is worth including on a list of influences!

On free-form gameplay:

I guess I'm pretty proud of the fact that free-form gameplay, player-authored experiences, and the like are finally becoming not just common but almost expected these days. From the "middle" Ultimas (4–6), to Underworld, to System Shock, to Thief, to Deus Ex, there's been this small cadre of us arguing, through our work, in favor of less linear, designer-centric games, and, thanks to the efforts of folks at Origin, Looking Glass Studios, Ion Storm, Rockstar/DMA, Bioware, Lionhead, Bethesda, and others, people are finally beginning to take notice. And it isn't just the hardcore gamers—the mass market is waking up, too. That's pretty cool.

I'm hugely proud of having had the privilege of working alongside some amazingly talented people. It's standard practice in all media to give one person credit for the creation of a product, but that's nonsense. Nowhere is it more nonsensical than in games. Game development is the most intensely collaborative endeavor I can imagine. It's been an honor to work with Richard Garriott, Paul Neurath, Doug Church, Harvey Smith, Paul Weaver, and many others (who will now be offended that I didn't single them out here!). I know I've learned a lot from all of them and hope I've taught a little bit in return.

Advice to designers:

Learn to program. You don't have to be an ace, but you should know the basics. In addition to a solid technical foundation, get as broad-based an education as you can. As a designer, you never know what you're going to need to know—behavioral psychology will help you immensely, as will architecture, economics, and history. Get some art/graphics experience, if you can, so you can speak intelligently with artists even if you lack the skills to become one yourself. Do whatever it takes to become an effective communicator in written and verbal modes. And most importantly, make games. Get hold of one of the many free game engines out there and build things. Get yourself on a mods team and build some maps, some missions, anything you can. Heck, make something amazing in Minecraft! You can do all of this on your own or at one of the many institutions of higher learning now (finally!) offering courses, even degrees, in game development and game studies. It doesn't really matter how you get your training and gain some experience—of life as much as game development—just make sure you get it. Oh, and make sure you really, really, really want to make games for a living. It's gruelingly hard work, with long hours and wrecked relationships to prove it. There are a lot of people who want the same job you do. Don't go into it unless you're absolutely certain it's the career for you. There's no room here for dilettantes!

Further Reading

Kelley, Tom. *The Art of Innovation: Lessons in Creativity from IDEO, America's Leading Design Firm*. New York: Random House, 2001.

Laramée, François Dominic, ed. *Game Design Perspectives*. Hingham: Charles River Media, 2002.

Moggridge, Bill. *Designing Interactions*. Cambridge: The MIT Press, 2007.

The Imagineers. *The Imagineering Way*. New York: Disney Editions, 2003.

Tinsman, Brian. *The Game Inventor's Guidebook*. Iola: KP Books, 2003.

End Notes

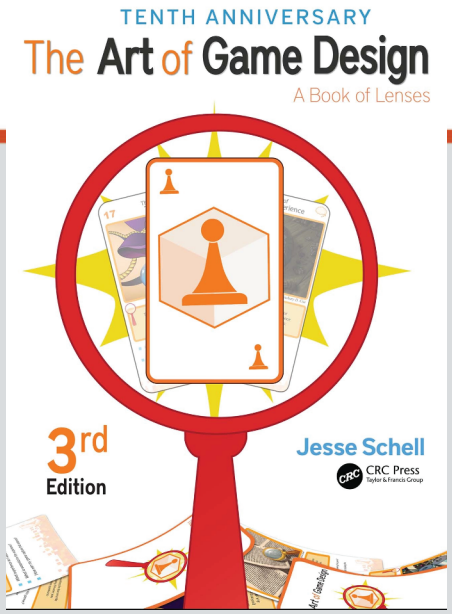
1. Phipps, Keith, "Will Wright Interview by Keith Phipps" A.V. Club. February 2, 2005. https://www.avclub.com/will-wright-1798208435
2. Sheff, David. Game Over: How Nintendo Conquered the World. New York: Vintage Books, 1994, p. 51.
3. Hermida, Alfred. "Katamari Creator Dreams of Playground." BBC News.com November 2005. http://news.bbc.co.uk/2/hi/technology/4392964.stm
4. Entis, Glenn. "Pre-Production Workshop." EA@USC Lecture Series. March 23, 2005.
5. Plummer, Chris. E-mail interview, May 2007.

Taylor & Francis
Taylor & Francis Group
http://taylorandfrancis.com

CHAPTER 2

CHAPTER 2

THE DESIGNER CREATES AND EXPERIENCE



This chapter is excerpted from

The Art of Game Design

A Book of Lenses, Third Edition

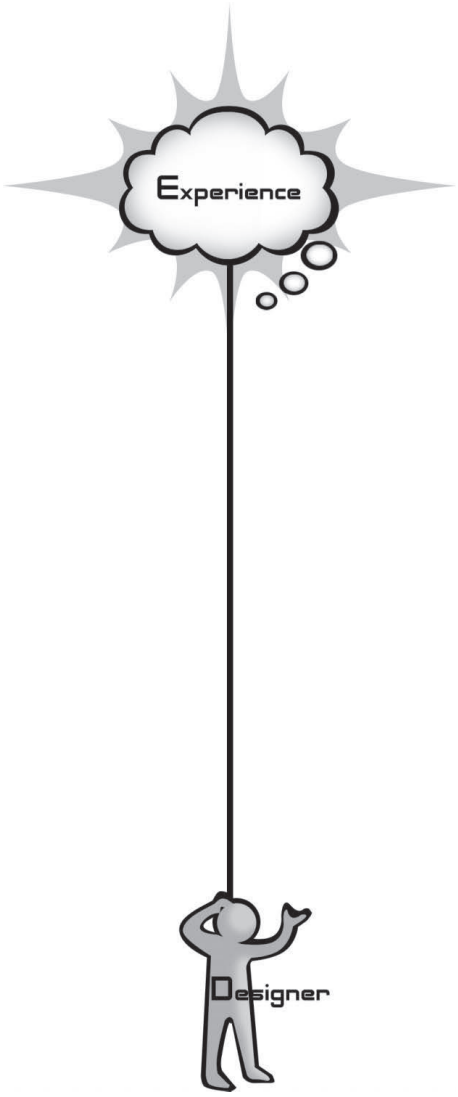
by Jesse Schell

Learn more

CHAPTER TWO

The Designer Creates an *Experience*

FIGURE **2.1**



I already know the ending
it's the part that makes your face implode
I don't know what makes your face implode
but that's the way the movie ends.

—They Might Be Giants, *Experimental Film*

Of the innumerable effects, or impressions, of which the heart, the intellect, or the soul is susceptible, what one shall I, on the present occasion, select?

—Edgar Allen Poe, *The Philosophy of Composition*

In Chapter 1, we established that everything begins with the game designer and that the game designer needs certain skills. Now it is time to begin talking about what a game designer uses those skills for. Put another way, we need to ask, "What is the game designer's goal?" At first, the answer seems obvious: a game designer's goal is to design games.

But this is wrong.

Ultimately, a game designer does not care about games. Games are merely a means to an end. On their own, games are just artifacts—clumps of cardboard or bags of bits. Games are worthless unless people play them. Why is this? What magic happens when games are played?

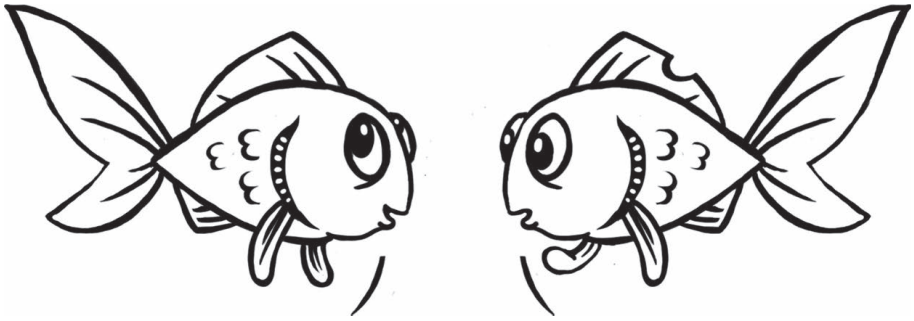
When people play games, they have an experience. It is this experience that the designer cares about. Without the experience, the game is worthless.

I will warn you right now: we are about to enter territory that is very difficult to talk about, not because it is unfamiliar—in fact, quite the opposite. It is hard to talk about because it is too familiar. Everything we've ever seen (look at that sunset!), done (have you ever flown a plane?), thought (why is the sky blue?), or felt (this snow is so cold!) has been an experience. By definition, we can't experience anything that is not an experience. Experiences are so much a part of us; they are hard to think about (even thinking about experiences is an experience). But as familiar as we are with experiences, they are very hard to describe. You can't see them, touch them, or hold them—you can't even really share them. No two people can have identical experiences of the same thing—each person's experience of something is completely unique.

And this is the paradox of experiences. On one level, they are shadowy and nebulous, and on another, they are all we know. But as tricky as experiences can be, creating them is all a game designer really cares about. We cannot shy away from them, retreating into the concreteness of our material game. We must use every means we can muster to comprehend, understand, and master the nature of human experience.

The Game Is Not the Experience

FIGURE
2.2



Nice water today! What's water?

We must be absolutely clear on this point before we can proceed. The game is not the experience. The game enables the experience, but it is not the experience. This is a hard concept for some people to grasp. The ancient Zen question addresses this directly: "If a tree falls in the forest, and no one is there to hear it, does it make a sound?" This has been repeated so often that it sounds hackneyed, but it is exactly what we are talking about. If our definition of "sound" is air molecules vibrating, then yes, the tree makes a sound. If our definition of sound is the experience of hearing a sound, then the answer is no, the tree makes no sound when no one is there. As designers, we don't really care about the tree and how it falls—we care only about the experience of hearing it. The tree is just a means to an end. And if no one is there to hear it, well, we don't care at all.

Game designers only care about what seems to exist. The player and the game are real. The experience is imaginary—but game designers are judged by the quality of this imaginary thing because it is the reason people play games.

If we could, through some high-tech magic, create experiences for people directly, with no underlying media—no game boards, no computers, no screens—we would do it. In a sense, this is the dream of "artificial reality"—to be able to create experiences that are in no way limited by the constraints of the medium that delivers the experiences. It is a beautiful dream, but only a dream. We cannot create experiences directly. Perhaps in the distant future, using technologies hard to imagine, such a thing could happen. Time will tell. For now, we live in the present, where all we can do is create artifacts (rule sets, game boards, computer programs) that are likely to create certain kinds of experiences when a player interacts with them.

And it is this that makes game design so very hard. Like building a ship in a bottle, we are far removed from what we are actually trying to create. We create an artifact that a player interacts with and cross our fingers that the experience that takes place during that interaction is something they will enjoy. We never truly

see the output of our work, since it is an experience had by someone else and, ultimately, unsharable.

This is why deep listening is so essential for game design.

Is This Unique to Games?

You might well ask what is so special about games, compared to other types of experiences, that require us to get into all of this touchy-feely experience stuff. And really, on one level, there is nothing special about games in this regard. Designers of all types of entertainment—books, movies, plays, music, rides, everything—have to cope with the same issue: How can you create something that will generate a certain experience when a person interacts with it?

But the split between artifact and experience is much more obvious for game design than it is for other types of entertainment, for a not-so-obvious reason. Game designers have to cope with much more interaction than the designers of more linear experiences. The author of a book or screenplay is designing a linear experience. There is a fairly direct mapping between what they create and what the reader or viewer experiences. Game designers don't have it so easy. We give the player a great deal of control over the pacing and sequence of events in the experience. We even throw in random events! This makes the distinction between artifact and experience much more obvious than it is for linear entertainment. At the same time, though, it makes it much harder to be certain just what experience is really going to arise in the mind of the player.

So, why do we do it? What is so special about game experiences that we would give up the luxuries of control that linear entertainers enjoy? Are we simply masochists? Do we just do it for the challenge? No. As with everything else game designers do, we do it for the experience it creates. There are certain feelings: feelings of choice, feelings of freedom, feelings of responsibility, feelings of accomplishment, feelings of friendship, and many others, which only game-based experiences seem to offer. This is why we go through all the trouble—to generate experiences that can be had no other way.

Three Practical Approaches to Chasing Rainbows

There ain't no rules around here! We're trying to accomplish something!

—Thomas Edison

So—we've established what we need to do—create games that will somehow generate wonderful, compelling, memorable experiences. To do this, we must embark on a daunting endeavor: to uncover both the mysteries of the human mind and the secrets of the human heart. No one field of study has managed to

perfectly map this territory (Mendeleev, where are you?), but several different fields have managed to map out parts of it. Three, in particular, stand out: psychology, anthropology, and design. Psychologists want to understand the mechanisms that make people tick, anthropologists want to understand people on a human level, and designers just want to make people happy. We will be using approaches borrowed from all three of these fields, so let's consider what each one has to offer us.

Psychology

Who better for us to learn the nature of human experience from than psychologists, the scientists who study the mechanisms that govern the human mind? And truly, they have made some discoveries about the mind that are incredibly useful, some of which will be covered in this book. In fact, you might expect that our quest for understanding how to create great human experiences might end right here and that the psychologists should have all the answers. Sadly, this is not the case. Because they are scientists, they are forced to work in the realm of what is real and provable. Early in the twentieth century, a schism in psychology developed. On one side of the battle were the behaviorists who focused only on measurable behavior, taking a "black box" approach to the study of the mind. Their primary tool was objective, controlled experimentation. On the other side were the phenomenologists, who study what game designers care about most—the nature of human experience and "the feeling of what happens." Their primary tool was introspection—the act of examining your experiences as they happen.

Unfortunately for us, the behaviorists won out and for very good reasons. The behavioristic focus on objective, repeatable experiments makes for very good science. One behaviorist can do an experiment, publish a paper about it, and other behaviorists can repeat the experiment under the same conditions, almost certainly getting the same results. The phenomenological approach, on the other hand, is necessarily subjective. Experiences themselves cannot be directly measured—only described and described imperfectly. When an experiment takes place in your mind, how can you possibly be sure the experimental conditions are controlled? As fascinating and useful as it might be to study our own internal thoughts and feelings, it makes for shaky science. As a result, for as much progress that has been made by modern psychology, it generally feels obligated to avoid the thing we care about the most—the nature of human experience.

Though psychology does not have all the answers we need, it does provide some very useful ones, as we'll see. More than that, it provides approaches we can use quite effectively. Not bound by the strict responsibilities of good science, game designers can make use of both behavioristic experiments and phenomenological introspection to learn what we need to know, since ultimately, as designers, we are

not concerned with what is definitely true in the world of objective reality but only with what seems to be true in the world of subjective experience.

But perhaps there is another scientific approach that lies somewhere between the two extremes of behaviorism and phenomenology?

Anthropology

> *Anthropology is the most humanistic of the sciences and the most scientific of the humanities.*
>
> —Alfred L. Kroeber

Anthropology is another major branch of study about human beings and what they think and do. It takes a much more holistic approach than psychology, looking at everything about people including their physical, mental, and cultural aspects. It is very concerned with studying the similarities and differences between the various peoples of the world, not just today, but throughout history.

Of particular interest to game designers is the approach of cultural anthropology, which is the study of living peoples' ways of life, mostly through fieldwork. Cultural anthropologists live with their subjects of study and try to immerse themselves completely in the world of the people they are trying to learn about. They strive for objective observation of culture and practices, but at the same time, they engage in introspection and take great pains to put themselves in the place of their subjects. This helps the anthropologist better imagine what it "feels like" to be their subjects.

We can learn a number of important things about human nature from the work of anthropologists—but much more important, by taking a cultural anthropologist's approach to our players, interviewing them, learning everything we can about them, and putting ourselves in their place, we can gain insights that would not have been possible from a more objective point of view.

Design

The third field that has made important study of human experience is, not surprisingly, the field of design. We will be able to learn useful things from almost every kind of designer: musicians, architects, authors, filmmakers, industrial designers, web designers, choreographers, visual designers, and many more. The incredible variety of design "rules of thumb" that comes from these different disciplines does an excellent job of illustrating useful principles about human experience. But unfortunately, these principles can often be hard for us to use. Unlike scientists, designers seldom publish papers about their discoveries. The very best designers in various fields often know little about the workings of other fields of design. The musician may know a lot about rhythm but probably has given little thought to how the principles of rhythm might apply to something nonmusical, such as a novel or stage play, even though they

may have meaningful practical application there, since they are ultimately rooted in the same place—the human mind. So, to use principles from other areas of design, we will need to cast a wide net. Anyone who creates something that people are meant to experience and enjoy has something to teach us, and so we will pull rules and examples from designers of every stripe, being as "xenophilic" as possible.

Ideally, we would find ways to connect all the varied principles of design to each other through the common ground of psychology and anthropology, since ultimately all design principles are rooted in these. In some small ways, we will do that in this book. Perhaps one day these three fields will find a way to unify all their principles. For now, we will need to be content with building a few bridges here and there—this is no small accomplishment, since these are three fields that seldom have much cross-pollination. Further, some of the bridges will prove to be surprisingly useful! The task before us, game design, is so difficult that we cannot afford to be snobbish about where we get our knowledge. None of these approaches can solve all our problems, so we will mix and match them, trying to use them appropriately, like we might use tools from a toolbox. We must be both open-minded and practical—good ideas can come from anywhere, but they are only good for us if they help us create better experiences.

Introspection: Powers, Perils, and Practice

A dedicated scientist never hesitates to experiment on himself.

—Fenton Claypool

We have discussed some of the places to find useful tools for mastering human experience. Let's now focus on one tool that has been used by all three disciplines: introspection. This is the seemingly simple act of examining your own thoughts and feelings—that is, your own experiences. While it is true you can never truly know the experience of another, you certainly can know your own. In one sense, it is all you can know. By deeply listening to your own self, that is, observing, evaluating, and describing your own experiences, you can make rapid, decisive judgments about what is and is not working in your game and why it is or is not working.

"But wait," you might say. "Is introspection really such a good idea? If it isn't good enough for the scientists, why is it good enough for us?" And this is a fair question. There are two main perils associated with using introspection.

Peril #1: Introspection Can Lead to False Conclusions about Reality

This is the scientists' main reason to reject introspection as a valid method of inquiry. Many pseudoscientists over the years have come up with crackpot theories

based mainly on introspection. This happens so often because what seems to be true in our personal experience is not necessarily really true. Socrates, for example, noted that when we learn something new, it often feels like we knew it all along and that in learning it, it feels as if we were just reminded of something we already knew but had forgotten. This is an interesting observation, and most people can remember a learning experience that felt this way. But Socrates then goes too far and forms an elaborate argument that since learning can feel like recollection, we must then be reincarnated souls who are just now remembering what we learned in past lives.

This is the problem with drawing conclusions about reality based on introspection—just because something feels true, it doesn't mean it is true. People very easily fall into the trap of building up structures of questionable logic to back up something that feels like it must be true. Scientists learn to be disciplined about avoiding this trap. Introspection certainly has its place in science—it allows one to examine a problem from points of view that mere logic won't allow. Good scientists use introspection all the time—but they don't draw scientific conclusions from it.

Fortunately for us, game design is not science! While "objective truth about reality" is interesting and sometimes useful to us, we primarily care about what "feels like it is true." Aristotle gives us another classical example that illustrates this perfectly. He wrote a number of works on a variety of topics, such as logic, physics, natural history, and philosophy. He is famous for the depth of his personal introspection, and when we examine his works, we find something interesting. His ideas about physics and natural history are largely discredited today. Why? Because he relied too much on what felt true and not enough on controlled experiments. His introspection led him to all kinds of conclusions we now know to be false, such as the following:

- Heavier objects fall faster than light ones.
- The seat of consciousness is in the heart.
- Life arises by spontaneous generation.

So why do we remember him as a genius and not as a crackpot? Because his other works, about metaphysics, drama, ethics, and the mind, are still useful today. In these areas where what feels true matters more than what is objectively, provably true, most of his conclusions, reached through deep introspection, stand up to scrutiny thousands of years later.

The lesson here is simple: when dealing with the human heart and mind and trying to understand experience and what things feel like, introspection is an incredibly powerful and trustworthy tool. As game designers, we don't need to worry much about this first peril. We care more about how things feel and less about what is really true. Because of this, we can often confidently trust our feelings and instincts when making conclusions about the quality of an experience.

Peril #2: What Is True of My Experiences May Not Be True for Others

This second danger of introspection is the one we must take seriously. With the first peril, we got a "Get Out of Jail Free" card because we are designers, not scientists. But we can't get away from this one so easily. This peril is the peril of subjectivity and a place where many designers fall into a trap: "I like playing this game; therefore, it must be good." And sometimes, this is right. But other times, if the audience has tastes that differ from your own, it is very, very wrong. Some designers take extreme positions on this ranging from "I will only design for people like me, because it is the only way I can be sure my game is good" to "introspection and subjective opinions can't be trusted. Only playtesting can be trusted." Each of these is a "safe" position but also has its limits and problems:

"**I only design for people like me**" has these problems:

- Game designers tend to have unusual tastes. There may not be enough people like you out there to make your game a worthwhile investment.
- You won't be designing or developing alone. If different team members have different ideas about what is best, they can be hard to resolve.
- There are many kinds of games and audiences that will be completely off limits to you.

"**Personal opinions can't be trusted**" has these problems:

- You can't leave every decision to playtesting, especially early in the process, when there is no game yet to playtest. At this point, someone has to exert a personal opinion about what is good and bad.
- Before a game is completely finished, playtesters may reject an unusual idea. They sometimes need to see it completed before they can really appreciate it. If you don't trust your own feelings about what is good and bad, you may, at the advice of your playtesters, throw out an "ugly duckling" that could have grown up to be a beautiful swan.
- Playtesting can only happen occasionally. Important game design decisions must be made on a daily basis.

The way out of this peril, without resorting to such limiting extremes, is again to listen. Introspection for game design is a process of not just listening to yourself but also listening to others. By observing your own experiences, and then observing others, and trying to put yourself in their place, you start to develop a picture of how your experiences differ from theirs. Once you have a clear picture of these differences, you can, like a cultural anthropologist, start to put yourself

in the place of your audience and make predictions about what experiences they will and will not enjoy. It is a delicate art that must be practiced—and with practice, your skill at it will improve.

Dissect Your Feelings

Work in the invisible world at least as hard as you do in the visible.

—Rumi

It is not such a simple thing to know your feelings. It is not enough for a designer to simply have a general sense about whether they like something or not. You must be able to clearly state what you like, what you don't like, and why. A friend of mine in college was notoriously bad at this. We would frequently drive each other crazy with conversations like the following:

Me: What did you eat at the cafeteria today?
Him: Pizza. It was bad.
Me: Bad? What was bad about it?
Him: It was just … bad.
Me: Do you mean it was too cold? Too hard? Too soggy? Too bitter? Too much sauce? Not enough sauce? Too cheesy? What was bad about it?
Him: I don't know—it was just bad!

He was simply unable to clearly dissect his experiences. In the case of the pizza, he knew he didn't like it but was unable to (or didn't bother to) analyze the experience to the point where he could make useful suggestions about how the pizza might improve. This kind of experience dissection is a main goal of your introspection—it is something designers must do. When you play a game, you must be able to analyze how it made you feel, what it made you think of, and what it made you do. You must be able to state this analysis clearly. You must put words to it, for feelings are abstract, but words are concrete, and you will need this concreteness to describe to others the experiences you want your game to produce. You need to do this kind of analysis not only when designing and playing your own games but also when playing games other people have created. In fact, you should be able to analyze any experience you might have. The more you analyze your own experiences, the more clearly you will be able to think about the kinds of experiences your games should create.

We have a special word for the feelings that rise up from within us: emotions. Our logical mind can easily dismiss emotions as unimportant, but they are the foundation of all memorable experience. So that we never forget the importance of emotions for experience design, let's make them our first lens.

#1 The Lens of Emotion

People may forget what you said, but they'll never forget how you made them feel.

—Maya Angelou

To make sure the emotions you create are the right ones, ask yourself these questions:

- What emotions would I like my player to experience? Why?
- What emotions are players (including me) having when they play now? Why?
- How can I bridge the gap between the emotions players are having and the emotions I'd like them to have?

Illustration by Rachel Dorrett

Defeating Heisenberg

But there is still a greater challenge of introspection. How can we observe our own experiences without tainting them, since the act of observation itself is an experience? We face this problem quite often. Try to observe what your fingers are doing as you type at a computer keyboard and you will quickly find yourself typing slowly and making many errors, if you can still type at all. Try to observe yourself enjoying a movie or a game, and the enjoyment can quickly fade away. Some call this "paralysis by analysis," and others refer to it as the Heisenberg principle. This principle, in reference to the Heisenberg uncertainty principle from quantum mechanics, implies that the attributes of a particle cannot be observed without affecting those attributes. Similarly, the nature of an experience cannot be observed without affecting the nature of that experience. This makes introspection sound hopeless. While it is a challenging problem, there are ways around it that are quite effective, though some take practice. Most of us are not in the habit of openly discussing the nature of our thought processes, so some of the following is going to sound a little strange.

Analyze Memories

One good thing about experiences is that we remember them. Analyzing an experience while it is happening can be hard, because the part of your mind used for analysis is normally focused on the experience itself. Analyzing your memory of an experience is much easier. Memory is imperfect, but analyzing a memory is better than nothing. Of course, the more you remember, the better, so working either with memories of powerful experiences (these often make the best inspiration, anyway) or with fresh memories is best. If you have the mental discipline, it also can be very useful to engage in an experience (such as playing a game), with the intention of not analyzing it while you play, but with the intention of analyzing the memory of it immediately after. Just having this intention can help you remember more details of the experience without interfering with the experience itself. This does require you to remember that you are going to analyze it without letting that thought interfere with the experience. Tricky!

Two Passes

A method that builds on analyzing memories is to run through your experience twice. The first time, don't stop to analyze anything—just have the experience. Then, go back and do it again, this time, analyzing everything—maybe even pausing to take notes. You have the untainted experience fresh in your mind, and the second run-through lets you "relive it" but gives you a chance to stop and think, considering how it felt and why.

Sneak Glances

Is it possible to observe your experience without spoiling it? It is, but it takes some practice. It sounds strange to say this, but if you "sneak quick glances" at your experience while it is happening, you can often observe it quite well without degrading or interrupting it significantly. It is kind of like trying to get a good look at a stranger in a public place. Take a few short glances at them, and they won't notice you are observing them. But look too long, and you will catch their attention, and they will notice you staring. Fortunately, you can learn a lot about an experience with a few short "mental glances." Again, this takes some mental discipline or you will get carried away with analysis. If you can make these mental glances habitual, just doing them all the time without thinking about it, they will interrupt things even less. Most people find what really interrupts their train of thought, or train of experience, is interior mental dialog. When you start asking and answering too many questions in your head, your experience is doomed. A "quick glance" is more like "Exciting enough? Yes." Then, you immediately stop analyzing and get back to the experience, until the next glance.

Observe Silently

Ideally, though, you want to observe what is happening to you while it is happening, not just through a few quick glances but through continuous observation. You want it to be as if you were sitting outside yourself, watching yourself, except that you see more than a normal observer. You can hear all of your thoughts and feel all of your feelings. When you enter this state, it is almost as if you have two minds: one moving, engaged in an experience, and one still, silently observing the other. This may sound completely bizarre, but it is quite possible and quite useful. It is a difficult state to achieve, but it can be reached. It seems to be something like the Zen practice of self-observation, and it is not unlike the meditation exercise of trying to observe your own breathing cycle. Normally we breathe without thinking, but at any moment, we may consciously take control of our breathing process—consequently interfering with it. With practice however, you can observe your natural, unconscious breathing without disturbing it. But this takes practice, just as observing your experiences takes practice. Observing your experiences can be practiced anywhere—while watching TV, while working, while playing, or while doing anything at all. You won't get it right at first, but if you keep experimenting and practicing, you will start to get the hang of it. It will take a great deal of practice. But if you truly want to listen to your self and understand the nature of human experience, you will find the practice worthwhile.

Essential Experience

But how does all this talk about experience and observations really fit in with games? If I want to make a game about, say, a snowball fight, does analyzing my memories of a real snowball fight have any bearing on the snowball fight game I want to make? There is no way I can perfectly replicate the experience of a real snowball fight without real snow and real friends outside in the real world—so what is the point?

The point is that you don't need to perfectly replicate real experiences to make a good game. What you need to do is to capture the essence of those experiences for your game. What does "the essence of an experience" really mean? Every memorable experience has some key features that define it and make it special. When you go over your memory of a snowball fight experience, for example, you might think of a lot of things. There are some you might even consider essential to that experience: "There was so much snow, school was canceled." "We played right in the street." "The snow was just right for packing." "It was so cold, but sunny—the sky was so blue." "There were kids everywhere." "We built this huge fort." "Fred threw a snowball really high—when I looked up at it, he chucked one right at my head!" "We couldn't stop laughing." There are also parts of that experience that you don't consider essential: "I was wearing corduroy pants." "I had some mints in my pocket." "A man walking his dog looked at us."

As a game designer trying to design an experience, your goal is to figure out the essential elements that really define the experience you want to create and find ways to make them part of your game design. This way, the players of your game get to experience those essential elements. Much of this book will be about the many ways you can craft a game to get across the experience you want players to have. The key idea here is that the essential experience can often be delivered in a form that is very different from a real experience. To follow up on the snowball fight example, what are some of the ways you could convey the experience "it was so cold" through a snowball fight game? If it is a videogame, you could certainly use artwork: the characters could breathe little puffs of condensation, and they could have a shivering animation. You could use sound effects—perhaps a whistling wind could convey coldness. Maybe there wasn't a cold wind on the day you are imagining, but the sound effect might capture the essence and deliver an experience that seems cold to the player. You could use the rules of the game, too, if cold was really important to you. Maybe players can make better snowballs without gloves, but when their hands get too cold, they have to put gloves on. Again, that might not have really happened, but that game rule helps deliver an experience of coldness that will be an integral part of your game.

Some people find this approach strange—they say, "Just design a game and see what experience comes out of it!" And I suppose it is true—if you don't know what you want, you might not care what you get. But if you do know what you want—if you have a vision of how you would like your game to feel to the players—you need to consider how you are going to deliver the essential experience. And this brings us to our next lens.

#2 The Lens of Essential Experience

To use this lens, you stop thinking about your game and start thinking about the experience of the player. Ask yourself these questions:

- What experience do I want the player to have?
- What is essential to that experience?
- How can my game capture that essence?

Illustration by Zachary D. Coe

If there is a big difference between the experience you want to create and the one you are actually creating, your game needs to change: you need to clearly state the essential experience you desire and find as many ways as possible to instill this essence into your game.

The design of the very successful baseball game in Wii Sports is an excellent example of the Lens of Essential Experience in use. Originally, the designers had intended to make it as much like real baseball as possible with the added bonus that you could swing your controller like a bat. As they proceeded, though, they realized they wouldn't have time to simulate every aspect of baseball as much as they wanted. So they made a big decision—since swinging the controller was the most unique part of this game, they would focus all their attention on getting that part of the baseball experience right—what they felt was the essential part. They decided that other details (nine innings, stealing bases, etc.) were not part of the essential experience they were trying to create.

Designer Chris Klug made masterful use of the Lens of Essential Experience when he created the tabletop role-playing game *James Bond 007*. Klug had been frustrated with previous attempts to create secret agent role-playing games, such as TSR's Top Secret, because they played too much like war games—the essence of what made spy movies exciting just wasn't there. For the Bond game, Klug designed the mechanics to feel like the exciting James Bond films every way he could. One outstanding example was the creation of something called "Hero Points." In traditional RPGs, when players would undertake a risky action, say, jumping out a window onto a moving helicopter, the game master would make some calculation of the probability of it succeeding, the player would roll the dice, and that was that. This gives the game master a difficult problem of balance: if the probability of succeeding at dangerous actions is too low, the players won't risk it. But if the chance is too high, the players will all act like superheroes, attempting and succeeding at all kinds of impossible feats. Klug's solution was to give players a budget of Hero Points, which they could use in risky situations to alter dice rolls to their favor. Since each player only got a small number of points to use on each adventure, players had to be very careful about when to use them—but when they did use them, it was to enact spectacular events that truly captured the essence of the James Bond books and films.

It is true that many designers do not use the Lens of Essential Experience. They just kind of follow their gut instinct and stumble across game structures that happen to enable experiences that people enjoy. The danger with this approach is that it relies on luck to a large extent. To be able to separate the experience from the game is very useful: if you have a clear picture in your mind of the experiences your players are having and what parts of your game enable that experience, you will have a much clearer picture of how to make your game better, because you will know which elements of the game you can safely change and which ones you cannot. The ultimate goal of the game designer is to deliver an experience. When you have a clear picture of your ideal experience and its essential elements, your design has something to aspire to. Without that goal, you are just wandering in the dark.

All That's Real Is What You Feel

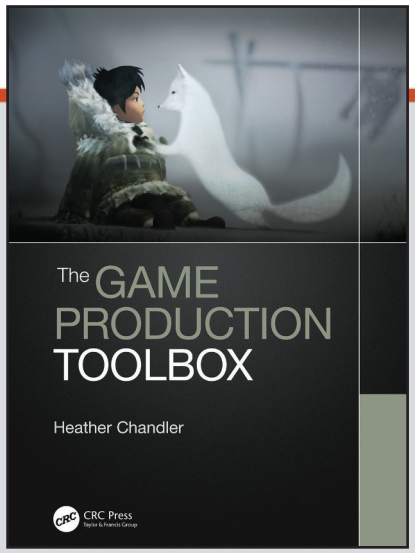
All this talk of experience brings out an idea that is very strange indeed. The only reality that we can know is the reality of the experience. And we know that what we experience is "not really reality." We filter reality through our senses and through our minds, and the consciousness we actually experience is a kind of illusion—not really reality at all. But this illusion is all that can ever be real for us, because it is us. This is a headache for philosophers, but a wonderful thing for game designers, because it means that the designed experiences that are created through our games have a chance of feeling as real and as meaningful (and sometimes more so) than our everyday experiences.

We will explore that further in Chapter 10: *The Player's Mind*, but right now we should take a moment to consider where these experiences actually take place.

CHAPTER

3

DEVELOPER AND PUBLISHER OVERVIEW



This chapter is excerpted from

The Game Production Toolbox

by Heather Maxwell Chandler

Learn more

Developer and Publisher Overview

2.1 Introduction

Developers and publishers each contribute something important to the process of launching a game—developers are responsible for turning a concept into a playable game, and publishers are responsible for the distribution and marketing, and sometimes financing. The functions vary for publishing and development, depending on the type of game, the team size, the budget, and the overall goals. For example, Stardew Valley was developed solely by Eric Barone over the course of 4 years with support from a small publisher. Eric made the game, and the publisher supported him with marketing and distribution. On the other hand, Riot Games, the developer and publisher of League of Legends, has thousands of people focused on the development and publishing effort. With a large team like this, Riot can have a development team focus specifically on creating and launching a single hero for the game, while another team can focus on creating some other game features. From a production standpoint, understanding the functions of all parts of the game team is critical since the producer is responsible for wrangling both development and publishing tasks.

2.2 Function of Developer

A developer is typically focused solely on making the game, which comprises art, design, engineering, audio, UX, and QA testing. Their day-to-day tasks involve designing and implementing game features, and they have little contact with the publishing departments. The exception to this are discipline leads as their input and expertise are relied upon to help publishing build a strong marketing and release pipeline for the game. In a well-integrated game team, the developers work collaboratively and rely on each other's strengths to make the best choices for the game. Engineers, artists, and designers work very closely together to make sure the game is cohesive and fun, and hits a specific quality bar. The input and feedback from all these disciplines working together is invaluable in making the best game possible.

The producer is somewhat of a hybrid since their function sits between the developers and the publishers, and is focused on bridging the communication between the two. For more information on functions in the game industry, and advice on how to learn the necessary skills for these functions, check out *Surviving Game School and the Game Industry After* by Michael Lynch and Adrian Earle (2018).

2.2.1 Art

Artists create the concept art, animations, 3D models, 2D textures, and any other graphic elements in the game. As technology becomes more powerful, artists are able to create anything from simple 2D textures for a platform game to fully immersive virtual worlds. They also create all the amazing special effects that are necessary to create "wow" effects, such as explosions, power-ups, and water droplets. If a game is art intensive and needs a lot of content, the artists on the team might outnumber the other team members by 2 to 1. For example, Fortnite releases new character skins and accessories on a regular basis, and all of these need to go through a full art production cycle before they are ready for the game. This includes creating concept art, 3D models, textures, animations, and special effects, which means the art team needs to be large enough to have multiple art assets in active development at the same time.

If the development team is small, the artists on the team may be generalists, which will mean that they have the ability to create different types of art assets. They might be able to create a character concept and have the skills necessary to take it from an image to a fully animated 3D character. For large development teams, artists may choose to specialize in an area and become very skilled at it. Setting up an art pipeline for a large team is more efficient if different people are focused on different parts of the pipeline. The most common art roles are the following (note that some companies may use different titles for similar roles):

- **Art Director:** They create and manage the artistic vision. They work with other project leads or directors to shape the overall scope and requirements for the game from an artistic standpoint.

- **Lead Artist:** They manage the day-to-day work of the art team. They provide feedback and mentoring, which allows the art director to stay focused on the creative elements. In some cases, the lead artist will also contribute content to the game (if there is time). They work with other project leads or directors to shape the overall scope and requirements for the game from an artistic standpoint.
- **Concept Artist:** They create concepts for game objects, environments, and characters that the other artists use as a reference when making the game assets.
- **World Builder:** They build the game world that the characters and objects inhabit. This is sometimes considered a design position because the level layout has a direct impact on the gameplay (and vice versa).
- **2D Artist or Texture Artist:** They focus on creating all the 2D art or textures for the 3D models. They may specialize in a particular area, such as vehicles, environments, or characters.
- **3D Artist or Modeler:** They build all the 3D models in the game. As with 2D artists, they may specialize in a particular type of area.
- **Animator:** They focus on creating all the animations in the game, including fully rendered animations (highest quality) and in-engine animations (used for interactive sequences). An animator may choose to focus on animation or rigging, and may also focus on a particular type of animation (facial, characters, objects).
- **UI Artist:** They create the art needed for the user interface (UI), including buttons, boxes, and drop-down lists, and other elements that appear in the game.
- **Technical Artist:** They focus on the technical side of asset creation. They work closely with the engineers to push the technology so that more impressive art can be included in the game. They also help the engineers build art tools that can streamline an artist's workflow.
- **Marketing Artist:** They focus on creating assets used by publishing and marketing, including logos, website design, key art, gameplay video, and anything else marketing needs.

2.2.2 Design

The main function of the designer is to create an engaging and immersive gameplay experience. Like artists, there are all types of designers—some are generalists, and some are specialists. They are responsible for creating all the "verbs" in the game, that is, things a player can do and interact with. This includes the control scheme, game systems (combat, trading, leveling up, etc.), narrative and story, character backgrounds and personalities, missions and objectives, level layouts, and so on.

You've heard the saying about too many cooks spoiling the soup. Designers are in the soup pot and receive a lot of feedback while making the game from all corners of the company, including UX Lab, QA, other team members, publishing, and senior management. Designers must be skilled in addressing

feedback and incorporating what works for the game and makes it better. There are several types of designers. Here are some common roles:

- **Creative Director:** They create and manage the overall vision for the player experience. They focus on carrying this vision throughout the entire game. They work closely with the art director. They work with other project leads or directors to shape the overall scope and requirements for the game from a design standpoint.
- **Lead Designer:** They manage the design team and their day-to-day work. They provide feedback and mentoring. If there is time, they may also create art assets. They work with other project leads or directors to shape the overall scope and requirements for the game from a design standpoint.
- **Level Designer:** They create the level layout, mission, and objectives that the player experiences in the game. They work closely with the World Builder to bring all the art and design pieces together.
- **Narrative Designer:** They create the characters, setting, and story. They work closely with the Level Designer on the game missions to ensure that everything is narratively cohesive.
- **Writer:** They specifically write all the dialogues and in-game text. Sometimes, the Narrative Designer is also a writer (and vice versa).
- **Systems Designer:** They design any game-wide systems, such as combat, scoring, character skill progression, and AI design.

2.2.3 Engineering

Engineers (aka programmers) take the designs and turn them into something playable. They are responsible for creating the technology for every aspect of the game, including physics, performance, AI, graphics, scripting tools, audio, lighting, player movement, and so on. They also work closely with the art team to create technical solutions for enhancing the game's artistic style. For example, they create technology that makes it possible for game characters to leave real-time footprints in a snowy environment. Without engineers, a game design is more difficult to translate from an analog format to a digital one. Here are some of the common engineering roles on a development team:

- **Technical Director:** They define and communicate the technical goals and standards for the game. They work with other project leads or directors to shape the overall scope and requirements for the game from a technical standpoint.
- **Lead Engineer:** They direct and manage the engineering team. They work closely with the Technical Director. They work with other project leads or directors to shape the overall scope and requirements for the game from a technical standpoint.
- **Network Engineer:** They focus on networking and multiplayer features.
- **Graphics Engineer:** They specialize in getting the most out of the game graphics from a performance and pipeline creation standpoint. They work closely with technical artists.

- **AI Engineer:** They create the AI behaviors of the NPCs. They work closely with system designers to define this behavior.
- **UI Engineer:** They create the functional UI screens. They work closely with the UI artists and UX designer.
- **Tools Engineer:** They work with the development team to create different tools for the development pipeline: for example, a tool that designers can use to create stats for an in-game character and import these into the game.
- **Build Engineer:** They create the tools and pipeline for checking things into the game and then compile game builds. They will also automate the build creation process as much as possible.

2.2.4 Audio

If you play a game with the sound turned off, you quickly realize how much audio enhances the experience. Audio is one way to provide the player with feedback on what they are doing in the game and is an important component of the player's enjoyment. Large game development teams usually have an in-house audio team that handles all aspects of sound design and implementation. Smaller teams may choose to outsource some of this work if the game doesn't have a lot of audio needs. Common audio positions include:

- **Audio Engineer:** They focus on the technical aspects of sound design and implementation. They work closely with the audio designer.
- **Sound Designer:** They design the sound effects, music, and voiceover for the game. They work closely with the writer and the narrative and level designers. They also process and implement the audio assets in the game.
- **Composer:** They compose the music for the game. They may also be responsible for licensing music instead of creating original content.

2.2.5 User Experience (UX)

UX is mainly about making sure that the design and business intentions are experienced as intended by the target audience of a product, system, or service. UX practices employ knowledge of cognitive science and psychology, as well as user research methodologies (e.g., playtests and analytics), to ensure that the game has good usability and engaging. This encompasses the player's experience with the game and how they interact with the systems and content. It also includes the signs and feedback that occur when the player takes an action in the game. For example, if the player pushes the correct button, they should receive positive feedback on this action, perhaps in the form of a color change, sound, or both. For more details about game UX, please see Chapter 15. Common UX positions include:

- **UX Designer:** They design the layout, flow, and functionality of all the UI screens and player interactions in the game. They work closely with designers and artists.

- **UX Researcher:** They focus on researching how well the game is meeting its UX objectives. They develop research plans and UX tests, and recruit people to come in and play the game. After the UX tests, they analyze the data, extract insights from it, and create a list of recommended UX changes for the game development team.

2.2.6 Quality Assurance (QA)

QA testers are the last stronghold between the development team and the players. In an ideal world, nothing is released to the players without being thoroughly tested by the QA team. They focus on finding any bugs, glitches, or errors that would negatively impact the player's experience. They are somewhat unsung heroes, who spend countless hours playing the same parts of the game over and over to ensure everything is working as intended. The size of the QA team depends on the amount of content that needs to be tested and how many platforms need to be tested concurrently. QA teams may be subdivided to focus on specific areas of the game. For example, one QA team might focus specifically on testing the UI, another team might focus on the gameplay in a specific set of levels, and another group might test a specific platform. Chapter 18, "QA Testing," goes into more specifics about how to organize testing as part of the production cycle. Typically, the main QA roles are:

- **QA Lead:** They manage the day-to-day efforts of the QA team. They create the testing plan and organize the QA team to execute it. They work with other project leads or directors to shape the overall scope and requirements for the game from a testing standpoint.
- **QA Tester:** They check the game functionality against the test plan. Also, they do playtesting and ad hoc testing to uncover things that players may discover after the game is released.

2.3 Function of Producer

Since this book focuses on production, understanding what a producer does in more detail is useful. Production, like any other discipline, is a craft that can be learned and improved upon with experience. Some people may be more suited to the role than others because production requires a strong combination of both organizational and leadership skills. The producer works closely with people by helping them get the necessary tools and resources; guiding groups towards consensus; resolving conflict; and, in some cases, addressing performance issues. If you don't enjoy working directly with people, the producer role may not be a good fit for you.

So, what does a producer do? There's no set definition as each team may define the production role and responsibilities differently. At the most basic level, a producer pulls together all the disparate parts needed to develop and launch a game, keeps the team on track, and removes any obstacles that prevent the team from hitting their goals. They are also the information hub for any questions people have about what's going on with the game,

the plan for future content, or how feedback is going to impact the project. A producer's job is never the same on any given day as there are always unexpected fires to be extinguished. Creating a schedule and budget are a small part of the job; these are the tools the producer uses to track progress. While it may be easy to think of the producer as a project manager marking tasks off the list and harassing people about deadlines, if the role is done well, it is much more than that.

Producers are not usually responsible for creating game assets, prototyping features, or writing stories because they are primarily responsible for keeping the team motivated and on track. Oftentimes, their feedback on design, technical, or artistic aspects of the game is a lower priority since they are not deeply involved in some of the creative or technical decisions. However, if a game decision involves something relating to overall scope, schedule, and budget, the producer's informed opinion holds a lot of weight. In situations where parties disagree about what the best decision is, the producer is expected to gather information from stakeholders, synthesize this information, make a recommendation, vet the feasibility of the recommendation with the leads, and then communicate the final decision.

Sometimes, there are hybrid producers who also do design, engineering, or art. This can create some interesting conflicts as it may be hard for a designer to put on a production hat and reduce the scope of the project by cutting a feature. So, if you are a producer with another role on the team, be aware of what biases you could bring to making production decisions.

BEN SMITH, SR. PRODUCER (PARTNER DEVELOPMENT), PUBLISHING PRODUCER CHALLENGES

As the publisher producer, the hardest thing to do is appropriately represent BOTH sides of the project: publisher and developer. Go too far towards repping the publisher's needs and goals, and you're not a value add. Go too far towards repping the developer's needs and goals, and you're going to be accused of going native and lose credibility with the publisher. I'll be honest—I'm not sure I ever managed to completely walk this tightrope. The only advice I can give is that you need to develop and focus on relationships in both organizations to an equal extent and do whatever it takes to add value to them both. Ultimately, you're responsible for the product, even if you're not building it directly, and both the publisher and the developer need your help to make it. Otherwise, your role wouldn't exist.

2.3.1 Background and Training

In general, producers have a particular area in which they are most skilled, such as managing technical projects, managing people, improving processes, managing creative groups, and so on. Some producers started out as engineers, designers, artists, or QA testers, and transitioned over to a production role (likely because someone observed that they were good

with people and getting things organized). Some producers worked in a different industry, usually in a project management role, and were able to transfer their skills to the game industry. So, while they come from a variety of backgrounds, these are some common skills every producer should have:

- **Leadership:** Do you understand the game's vision, and are you able to communicate this to the team, keep them focused on the goals, build consensus, and motivate them to do their best work? Are you empathetic? Do you listen to your team? Are you able to take a risk, make an unpopular choice, or make things better than when you found them? These are the skills a strong leader has.
- **Communication:** Are you able to explain things clearly, diplomatically, and in a timely fashion? Are you able to give constructive feedback and deliver bad news in a transparent fashion? Do you know the different audiences you are communicating with, and do you tailor your message to them?
- **Organization:** Are you able to build consistent processes that are easy for people to follow? Can you break down all the steps needed to accomplish the goal? Are you able to track all the details and actions that are needed to complete each phase of the game? Are you able to create a plan that details what needs to be done, how long it will take, and how much it will cost?
- **Serving Others:** Are you serving your team, or do you expect them to serve you? Do you empower the team to make their own decisions and work autonomously? Are you open to suggestions and feedback, and do you make changes based on this? Do you create a working environment for your team that allows them to be their most productive?

As far as formal training goes, there are very few educational programs geared specifically towards a career in game production. There are a lot of game design, interactive media, and game technology degrees available, and the list of schools offering these degrees is growing. If there isn't a specific game production degree at your chosen institution, studying one of these other areas is useful. The more you can learn about how games are made, the more effective you will be as a producer. If you have a deep understanding of how to program a game, you will be better prepared to solve some of the technical issues your team will face. Oftentimes, degree programs will culminate in a final group project, where you can take on the producer functions to gain valuable experience.

There are other ways to improve your production skills and knowledge besides going to school. Some of these things include:

- **Attending conferences:** There are a lot of game conferences every year. The largest one is the Game Developers Conference (www.gdconf.com), held for one week each spring. Here, hundreds of speakers talk about all aspects of making games. Conferences are also a great place to network and meet fellow producers. They are always ready to swap techniques

and war stories. Refer to the "Resources and Tools" appendix for a list of other conferences.

- **Studying the game industry:** Play games, so you are aware of the latest trends. Become familiar with common game development tools and technologies. Stay up to date with what's happening in the industry, including sales figures, hot topics, cultural shifts, and new platforms and technologies. Check the "Resources and Tools" appendix for a list of websites and other information.
- **Attending management training:** This includes both project management and people management classes. Check with your HR department if you are interested in external training as the company may pay for it if it's relevant to your job. Continuing education programs affiliated with universities may offer classes. There are also several companies that specialize in project management and leadership training. A quick Google search can be helpful in finding them.
- **Improving public speaking ability:** A producer is often talking in front of groups—whether it is at team meetings, in front of senior management for review, or while on a PR tour. Good public speakers exude confidence and leadership. If you are uncomfortable with talking in front of people, you can join a group like Toastmasters International (www.toastmasters.org) to meet other people and practice your public speaking.

2.3.2 Career Progression

Production is a career in which you can start out in an entry-level position and work your way up to a more senior production role. As you progress up the career ladder, you will figure out where your strengths are and choose a particular focus. You may also become skilled or knowledgeable enough in another aspect of game development and transition to a different role on the team, leaving the production role open for someone else.

The following are the basic stepping stones on a production career track. Keep in mind that companies may have different names and different requirements for getting promoted. It's best to check with your manager on what is needed to advance from one position to the next:

- **Production Assistant (PA):** This is an entry-level position. Little production experience is needed, just a willingness to learn, be helpful, and take on a wide variety of tasks as directed. The main focus of this role is to be an extra pair of hands for the production team: someone who can help schedule meetings, take notes, get status updates, organize playtests, and do whatever else is needed. In this role, you get exposed to different areas of game development, so it's a great learning experience. Your work tasks will shift daily as PAs are most effective when focused on short-term tasks and whatever is needed that day.
- **Associate Producer (AP):** This position has more responsibility and works closely with the producer on managing the day-to-day activities of the team. They may be tasked with producing specific parts of the game, such as the localizations or voiceover recordings. They usually

have 1–3 years of production experience. They are mostly focused on what is needed that week or over the next few weeks, so they will have a set of regular responsibilities, such as running daily stand-ups, doing risk analysis, and setting up production pipelines. They may also manage PAs.

- **Producer:** This person usually manages an entire development team and has 3–8 years of experience. Their focus is on executing the plan. If the team is especially large, several producers may split responsibilities across it. For example, one producer might be responsible for all the art content, another might be responsible for design content, and a third might be focused on technical content. They look weeks and months ahead on the project. If the project schedule is organized into sprints, producers focus on the next 2–3 sprints. They are responsible for keeping the team on track, solving problems, ensuring the work hits the quality bar, and facilitating the production pipeline. They also anticipate, define, and mitigate risks. They will manage and mentor APs and PAs.
- **Senior Producer:** The responsibilities for this role vary depending on the studio. Usually, it is a more forward-looking and strategic role that is focused on leading the production team, pulling together the high-level plan and strategies for development, launching the game, improving processes, and mentoring producers and leads. The person also works with departments outside the development team to manage the entire development and release cycle for the game. The senior producer needs to anticipate problems before they happen and get mitigation strategies in place. They likely have at least 8–12 years of experience.
- **Executive Producer (EP):** Responsibilities for this role also vary. Some EPs are focused on managing the multi-year development and release plan for a game franchise, like Call of Duty. They determine the strategies for growing and maintaining the franchise, such as when sequels are released; what new content, in the way of characters and stories, is created; and what new technologies are integrated into the game (such as VR or AR). They may manage multiple products within the franchise. Other EPs might oversee the process of game development and focus on broader development tasks, such as establishing employee training programs, evaluating external vendors, improving processes, determining the needs of the project, and mentoring other producers.

2.3.3 Types of Producers

While a producer's main function is to manage the development team in order to deliver the game on time, on budget, and to a specific quality bar, there are specializations within this role. Generally, these types of roles are more likely to appear on larger teams, which is where a specialized role is more useful. Here's a brief overview of the types of production roles you might encounter:

- **Generalist:** A generalist will work on whatever is needed from a production standpoint, including improving process, managing engineers, or interfacing with publishing.

- **Technical Producer:** If a producer has prior experience as an engineer, they may specialize in technical production. This means they are skilled at working with engineers to scope out project features and can discuss the technical merits of various solutions. They are also good at explaining technical concepts to non-technical people.
- **Art Producer:** An art producer might have previous experience as an artist. As with Technical Producers, they are trained in the discipline they are managing. Art producers are able to work with artists to scope out features, discuss the merits of one approach or another when creating art assets, and help define an art pipeline that is effective.
- **Creative Producer:** This person may have a design background and is especially skilled at working with the various designers on the project.
- **Publishing Producer:** This role focuses on managing the publishing aspects of the project. Sometimes, the person come from a business background and is well-suited to managing the marketing, PR, CS, and other publishing needs of the game.
- **LiveOps Producer:** This person is responsible for maintaining the health of the game after it is released. This position is critical for an online multi-player game because once the game is released, it needs to be actively managed 24/7 to make sure that players can access it and aren't experiencing any game-breaking issues, and that new content is released to keep the game fresh and interesting. The LiveOps producer is responsible for responding quickly to any live issues that impact the player's experience and reviewing the data and analytics to determine what new features and content to add or what types of in-game events will resonant with the players.

2.4 Function of Publisher

The developer and the publisher have a symbiotic relationship. The developer provides a game, and the publisher markets and distributes the game. The publisher is highly invested in the success of the game, especially if they have also provided financing. One way to think of this relationship is as an amusement park; the developer provides the specific games and amusements, while the publisher manages the amusement park (parking, entrance fee, restaurants, restrooms, etc.). They both have a deep investment in the game's success but are supporting it in different ways. The publisher will expect a percentage of the game sales as payment for the services and support offered. This percentage could be 30% or higher, so it may be tempting to forgo working with a publisher in order to avoid this fee. However, a good publisher has a huge impact on the game's success, and the fee more than makes up for itself if the game is able to find a large audience. Don't underestimate the value of a publisher. In addition to financing, there are many other ways in which they support the launch of the game.

2.4.1 Financing

Financing options were discussed in more depth in Chapter 1, "Game Industry Overview." Publishers want to finance games that contribute to the bottom line. If a publisher has internal development teams, they will work with these teams to determine what games to make. The publisher wants to make games that fit within their release calendar and that are going to make a profit. Sequels to successful games are sure bets in a publisher's eyes, so if they have an internal development team who creates a successful game, the publisher finances sequels and updates to this game in the hopes of turning it into a profitable franchise. Sometimes, a publisher might be willing to finance something riskier with an internal team in order to test the waters for a new type of game genre or a new intellectual property (IP). When a publisher finances an internal development team, it can take some of the financial pressure off the team. The team knows they are getting a regular paycheck, and if a milestone slips, it is less likely to have financial implications for them.

If a publisher works with external developers, they still want to finance something that makes money, but they are more open to listening to developer pitches in the hopes of finding an unexpected hit. In cases where the publisher funds an external developer, financial arrangements are focused on milestone payments. The developer and the publisher will negotiate on how much money is needed to make the game and structure a payment plan around when key milestones for the game will be completed. They will also negotiate the royalty rate that the developer receives for each copy of the game sold. After the game is launched, the developer's profits will first be used to pay back the publisher, and then, the developer will start receiving royalty payments.

2.4.2 Distribution

Distribution methods were also covered in more depth in Chapter 1, "Game Industry Overview." Partnering with a publisher makes getting distribution for your game easier. An established publisher is already connected with the major distribution platforms and can support the developer in getting a game launched on a particular platform. Some digital storefronts won't work directly with independent developers and require a publisher to be part of the process. A publisher will also pick up any fees related to getting a game distributed, including creating boxed product and shipping it to retail stores. These costs can add up, and having a publisher foot the bill for them is extremely helpful. If you are an independent developer, the publisher makes up for these costs in how the milestone payments and royalty fees are structured.

2.4.3 Marketing and Public Relations (PR)

One of the biggest responsibilities of a publisher is to get people excited about playing the game, which is done through marketing and PR efforts. Marketing usually focuses on sales and advertising, and PR usually focuses

on managing the public image of the game or company. In this era of social media, the lines between the two are becoming more blurred. In order to create a marketing and PR campaign, the publishers must be directly involved with the development team. This helps the publisher to better understand the game; what features are important; and how the story, setting, and characters are used in the game, which, in turn, helps them to build the best marketing and PR campaign. This also means that they may make suggestions about the game in order to make it easier to market. For example, they may want to partner with a license to raise brand awareness or use a voiceover to capitalize on a celebrity's popularity. PR works hand in hand with marketing to promote the game. Chapter 19, "Getting the Word Out," discusses this in more detail.

2.4.4 Production Support

Production support is another benefit offered by a publisher. This can be helpful if there are multiple things to coordinate with the release and launch of the game. The publisher may also pay for localization and testing efforts, and as part of the package, they may assign a producer from the publishing team to manage these efforts. As discussed earlier, a publishing producer will also manage all the pieces that go into publishing a game. They work directly with their counterparts on the development team to coordinate all the bits and pieces of developing and publishing the game.

2.4.5 Product Management

Product management is gaining in importance with game publishers. A product manager is someone who focuses on growing the game's success within the marketplace. They work with the development team and the publisher to define the game's feature release road map within the context of sales forecasts, launch strategies, target audiences, key competitors, and other market factors. Their goal is to look at the game as a whole to determine how to successfully launch it, grow its audience, and keep it viable with additional content during its life cycle.

Before launch, they will research comparable games that have already been released to help determine what minimum features might be critical for launch or what new features could be added to give the game a differentiating factor. After launch, they will review data and analytics to determine which game content the players are most engaged with and use this information to help the teams define what content updates to create and release.

Not every game will have a dedicated product manager. Product managers usually work on games that are part of a franchise or a new IP that has the potential to turn into a franchise. They are usually affiliated with the marketing and publishing departments, and don't necessarily work directly with the team on day-to-day development work. Their role is focused more on strategy and what tactics can be used to execute the strategy.

2.4.6 Live Operations

Live Operations (LiveOps) is a critical function for any online game that is available 24/7. Publishers may have an entire department devoted to LiveOps for a single game. This includes having teams available who can address networking issues, fix game-breaking bugs, release new updates, determine what new content is created, and generally actively manage the health of the game so that it is always available as well as giving players new content to enjoy. Sustaining a LiveOps team can be expensive, especially since it needs to be staffed around the clock. Large publishers are able to provide this type of support by setting up LiveOps teams in various offices around the world in what is known as a follow-the-sun model—as one team ends their workday, a team in another location is beginning theirs. More information on LiveOps is discussed in Chapter 20, "Releasing to Players."

2.4.7 Community Management

Community managers are responsible for the health of the community that forms around a game. At a high level, community management is responsible for creating, growing, listening, and managing the community. They do this by attracting new people or re-engaging inactive members through outreach efforts, such as surveys and promotions. They give active members content to engage with: for example, blog posts and news updates. As part of interacting with the community, they also listen and respond to feedback. Based on this feedback, they work with the development and publishing teams to improve the game, and then let the community know how their feedback was used to make the game better. As the community gets larger, there may be a whole team of community managers—one may focus exclusively on Facebook, while another focuses on forums, and so on. Their goal is to be responsive to the community to demonstrate that the company cares about them and what they think about the game. Richard Millington's book *Buzzing Communities* (2012) provides a good overview of how to build and manage communities.

2.4.8 Customer Support (CS)

Like community management, CS directly interfaces with people who play the game. Their role is to resolve complaints, provide technical assistance, and listen to player feedback. They may also be responsible for enforcing player behavior guidelines: for example, if someone is violating the game's code of conduct, a CS agent may apply a temporary ban to the player and communicate the reason why. If the player receives multiple violations, the CS agent is empowered to permanently ban them according to the guidelines. CS also processes refunds and provides assistance to players, as needed.

2.5 Publishing Your Game

Now that you have a better understanding of what publishers do, you want to consider your publishing options. As discussed in Chapter 1, "Game Industry Overview," there are many distribution options available

to independent developers who want to self-publish their game. To clarify, distribution is merely making the game available for people to purchase. You can distribute a game fairly easily on digital storefronts without putting a lot of publishing effort behind it. However, if you choose to go this route, people won't know that your game is available, and therefore, it won't gain traction in the market. You'll want to weigh the pros and cons of self-publishing versus partnering with an established publisher.

2.5.1 Self-Publishing

If you are an independent developer, self-publishing is something to consider. Its main benefit is that you don't have to give away a percentage of game revenue to a publishing partner, and another is that you have greater control over when the game is published, how much you will charge, when updates are released, how the game is marketed, and so forth. If the game is a niche title that appeals to a small audience with a specific need, self-publishing might be a good option. Some publishers may not be interested in a niche title because of the specialized audience, so they may find the financial return not worth the time and money required to invest on their side.

One of the disadvantages of self-publishing is that, on top of the time, money, and effort you are investing in developing the game, you now need to put additional time, effort, and money into publishing it. This is a huge amount of work for a single team, and it is more difficult if you don't have experience in marketing, sales, advertising, PR, CS, and community management.

You may also find it difficult to get published on hardware platforms, like consoles or VR systems. Companies, such as Microsoft and Sony, who manage the distribution of games on their proprietary platforms prefer to work with established publishers. They know that experienced publishers already have a pipeline in place to market and distribute the game successfully. New developers who are trying to self-publish are not proven entities and thus will require more time and support from the platform holders. This is not to say that small developers can't publish successfully on these platforms, but it is harder to become a certified publisher on these platforms without a prior track record of success.

2.5.2 Publishing Partner

A publishing partner is an attractive option if you want the financial and other support that these partners offer. However, finding the right publisher will take time and effort. It's also possible that you can pitch your game to several publishers and not get any offers to partner with them. Your approach to finding a publisher needs to be methodical and focused on presenting your game and your company as something worthwhile in which to invest.

So, what are publishers looking for in a game? The publisher's motivations are mainly financial—they want to find a game that will make them money. They base their decision on whether they think the game is going to be profitable. Some of the criteria they use to judge this are:

- **Return on Investment (ROI):** A publisher will do what's called due diligence on any game before they fully commit to investing in it. They will do a market analysis to compare the sales figures of comparable games (called comps) with the projected sales figure of the game being pitched to them. This, along with the estimated development costs, is factored into an ROI formula to determine whether there will be a high or low ROI.
- **State of Game:** If the game is in some state of completion, this will also be a factor. If the game is almost complete, and only publishing support is needed, the publisher can invest less, which will boost their ROI. It also gives the publisher an opportunity to play the game and make their own determination about the quality and how well it will do in the market. A partially completed game shows that the developer has some financial stake in the game, which is more attractive to publishers. They are more likely to partner with people who have also made a significant investment.
- **Reputation:** Don't underestimate the value of the developer's reputation. If a developer is well-established as one who makes successful games, a publisher is more likely to provide funding, even if the game is in the concept phase. First-time developers will find it very difficult to procure significant funding from a publisher, even if they have an amazing demo. First-time developers still need to prove themselves and show that they can follow through and make a complete game.

2.5.3 The Publishing Relationship

If you work with a publisher, the relationship needs to be actively managed and maintained. The publisher needs to be informed of the game's progress against the agreed-upon milestones, and the developer needs to know that financial and other supports are available when needed.

If the developer doesn't provide regular status updates and builds of the game, the publisher may perceive this in a negative light. They may think that the developer isn't making progress and pull funding, or, in cases where the developer owns the IP, they may assign the project to another developer. The publisher also needs to provide feedback at appropriate points in the process, usually around milestone deliveries, to ensure that both the developer and the publisher are in alignment on what the game is going to be. Sometimes, a publisher may request a major change after the game is done, so the developer needs to scope out how much this change will cost and ask the publisher to provide the additional funds.

The developer and publisher relationship can be very complex since both parties have a high investment in the game's outcome. But they are also viewing the game from a different perspective and thus have a different set of priorities they are judging against. These different priorities can sometimes lead to major disagreements about decisions made for the game, so time needs to be spent talking through the decisions, weighing the various outcomes, and agreeing on which decisions are the best path forward.

A publisher will assign a producer to work directly with the developer and represent the publisher's interests on the project. As discussed earlier, the developer and the publisher producer will work closely on the game. In order for this production relationship to be effective, the developer and the publisher producer will want to define their areas of responsibilities on the project. Usually, the publisher producer coordinates sales, marketing, PR, community management, and sometimes localization efforts. Meanwhile, the developer producer will manage the day-to-day development of the game. Once the game is launched, a Live Producer may be thrown into the mix as well.

One important role of the publishing producer is reviewing the milestone deliverables to determine if they are at an acceptable level of quality or if they still need more work. Oftentimes, milestone acceptance is tied directly to releasing another funding payment. If the developer consistently misses milestone deadlines or delivers poor-quality work, the publisher may choose to pull funding and assign the project to another developer. In some cases, the project could be canceled outright. Chapter 4, "Laying the Groundwork," discusses milestone deliverables and aligning expectations in more detail.

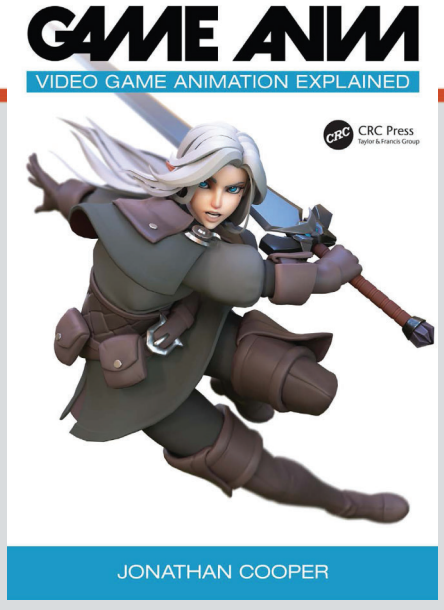
2.6 Conclusion

A strong developer and publisher relationship is good for everyone. The development team, comprising artists, designers, engineers, and testers, is counting on the fact that the game will see the light of day and be enjoyed by players. They have to work closely with the producer to make the game a reality. In turn, the producer plays an integral function in ensuring that the developers are communicating with each other, and the publisher is communicating effectively with the developer. If any of these three entities, developer, producer, or publisher, is out of alignment with the others, it will impact the quality and timeliness of the project. A contractual agreement is useful in making sure that all the parties are aligned on expectations. Chapter 3, "Legal Overview," provides general information on the legal aspects that a producer should be aware of, including publishing contracts and protecting IP.

CHAPTER

4

THE FIVE FUNDAMENTALS OF GAME ANIMATION



This chapter is excerpted from

Game Anim

Video Game Animation Explained

by Jonathan Cooper

Learn more

The Five Fundamentals of Game Animation

The 12 animation principles are a great foundation for any animator to understand, and failure to do so will result in missing some of the underlying fundamentals of animation—visible in many a junior's work. Ultimately, however, they were written with the concept of linear entertainment like TV and film in mind, and the move to 3D kept all of these elements intact due to the purely aesthetic change in the medium. Three-dimensional animated cartoons and visual effects are still part of a linear medium, so they will translate only to certain elements of video game animation—often only if the game is cartoony in style.

As such, it's time to propose an additional set of principles unique to game animation that don't replace but instead complement the originals. These are what I have come to know as the core tenets of our new nonlinear entertainment medium, which, when taken into consideration, form the basis of video game characters that not only look good, but feel good under player control—something the original 12 didn't have to consider. Many elements are essential in order to create great game animation, and they group under five fundamental areas:

1. Feel
2. Fluidity
3. Readability
4. Context
5. Elegance

Feel

The single biggest element that separates video game animation from traditional linear animation is interactivity. The very act of the player controlling and modifying avatars, making second-to-second choices, ensures that the animator must relinquish complete authorship of the experience. As such, any uninterrupted animation that plays start to finish is a period of time the player is essentially locked out of the decision-making process, rendered impotent while waiting for the animation to complete (or reach the desired result, such as landing a punch).

The time taken between a player's input and the desired reaction can make the difference between creating the illusion that the player is embodying the avatar or becoming just a passive viewer on the sidelines. That is why cutscenes are the only element in video games that for years have consistently featured a "skip" option—because they most reflect traditional noninteractive media, which is antithetical to the medium.

Response

Game animation must always consider the response time between player input and response as an intrinsic part of how the character or interaction will "feel" to the player. While generally the desire is to have the response be as quick as possible (fewer frames), that is dependent on the context of the action. For example, heavy/stronger actions are expected to be slower, and enemy attacks must be slow enough to be seen by the player to give enough time to respond.

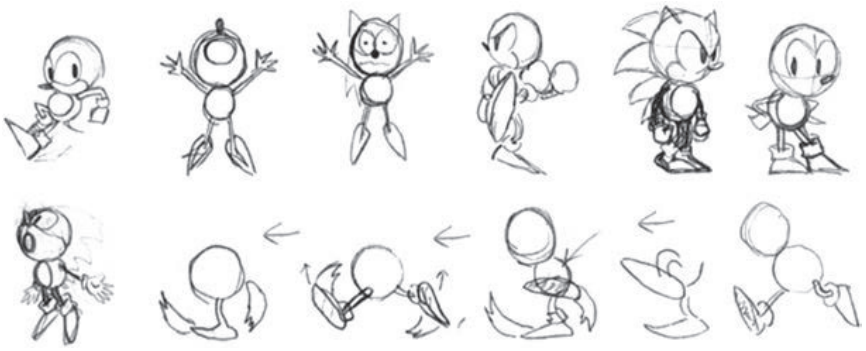
It will be the game animator's challenge, often working in concert with a designer and/or programmer, to offer the correct level of response to provide the best "feel," while also retaining a level of visual fidelity that satisfies all the intentions of the action and the character. It is important not to sacrifice the weight of the character or the force of an action for the desire to make everything as responsive as possible, so a careful balancing act and as many tricks as available must be employed.

Ultimately, though, the best mantra is that "gameplay wins." The most fluid and beautiful animation will always be cut or scaled back if it interferes too much with gameplay, so it is important for the game animator to have a player's eye when creating response-critical animations, and, most importantly, play the game!

Inertia & Momentum

Inertia is a great way to not only provide a sense of feel to player characters, but also to make things fun. While some characters will be required to turn on a dime and immediately hit a run at full speed, driving a car around a track that could do the same would not only feel unrealistic but mean there would be no joy to be had in approaching a corner at the correct speed for the minimum lap time. The little moments when you are nudging an avatar because you understand their controls are where mastery of a game is to be found, and much of this is provided via inertia.

Judging death-defying jumps in a platform game is most fun when the character must be controlled in an analogue manner, whereby they take some time to reach full speed and continue slightly after the input is released. This is as much a design/programming challenge as it is animation, but the animator often controls the initial inertia boost and slowdown in stop/start animations.



Original sketches from Sonic the Hedgehog (circa 1990). The animation frames are already heavily displaying inertia. (Courtesy of SEGA of America.)

Momentum is often conveyed by how long it takes a character to change from current to newly desired directions and headings. The general principle is that the faster a character is moving, the longer it takes to change direction via larger turn-circles at higher speeds or longer plant-and-turn animations in the case of turning 180°.

Larger turn-circles can be made to feel better by immediately showing the intent of the avatar, such as having the character lean into the turn and/or look with his or her head, but ultimately we are again balancing within a very small window of time lest we render our characters unresponsive.

A classic example is the difference between the early Mario and Sonic the Hedgehog series. Both classic Mario and Sonic's run animations rely heavily on inertia and have similar long ramp-ups to full speed. While Mario immediately starts cartoonishly running at full speed as his legs spin on the ground to gain traction, Sonic slowly transitions from a walk to a run to a sprint. While Mario subjectively feels better, this is by design, as Sonic's gameplay centers on high speeds and "flow," so stopping or slowing down is punitive for not maintaining momentum.

Visual Feedback

A key component of the "feel" of any action the player and avatar perform is the visual representation of that action. A simple punch can be made to feel stronger with a variety of techniques related to animation, beginning with the follow-through following the action. A long, lingering held pose will do wonders for telling the player he or she just performed a powerful action. The damage animation on the attacked enemy is a key factor in informing the player just how much damage has been suffered, with exaggeration being a key component here.

In addition, employing extra tricks such as camera-shake will help further sell the impact of landing the punch or gunshot, not to mention visual effects of blood or flashes to further register the impact in the player's mind. Many fighting games employ a technique named "hit-stop" that freezes the

characters for a single frame whenever a hit is registered. This further breaks the flow of clean arcs in the animations and reinforces the frame on which the impact took place.

As many moves are performed quickly so as to be responsive, they might get lost on the player, especially during hectic actions. Attacking actions can be reinforced by additional effects that draw the arc of the punch, kick, or sword-swipe on top of the character in a similar fashion to the "smears" and "multiples" of old. When a sword swipe takes only 2 frames to create its arc, the player benefits mostly from the arcing effect it leaves behind.

Slower actions can be made to feel responsive simply by showing the player that at least part of their character is responding to their commands. A rider avatar on a horse can be seen to immediately turn the horse's head with the reins even if the horse itself takes some time to respond and traces a wide circle as it turns. This visual feedback will feel entirely more responsive than a slowly turning horse alone would following the exact same wide turn.

Much of the delay in visual feedback comes not from the animation alone, but the way different game engines handle inputs from the joypad in the player's hands. Games like the Call of Duty series place an onus on having their characters and weapons instantly respond to the player's inputs with minimal lag and high frame rates, whereas other game engines focused more on graphics postprocessing will have noticeably longer delays (measured in milliseconds) between a jump button-press and the character even beginning the jump animation, for example. This issue is further exacerbated by modern HDTVs that have lag built in and so often feature "Game Mode" settings to minimize the effect. All this said, it is still primarily an animator's goal to make characters as responsive as possible within reason.

Fluidity

Rather than long flowing animations, games are instead made of lots of shorter animations playing in sequence. As such, they are often stopping, starting, overlapping, and moving between them. It is a video game animator's charge to be involved in how these animations flow together so as to maintain the same fluidity put into the individual animations themselves, and there are a variety of techniques to achieve this, with the ultimate goal being to reduce any unsightly movement that can take a player out of the experience by highlighting where one animation starts and another ends.

Blending and Transitions

In classic 2D game sprites, an animation either played or it didn't. This binary approach carried into 3D animation until developers realized that, due to characters essentially being animated by poses recorded as numerical values, they could manipulate those values in a variety of ways. The first such

improvement that arrived was the ability to blend across (essentially cross-fading animations during a transitory stage) every frame, taking an increasing percentage of the next animation's value and a decreasing percentage of the current as one animation ended and another began. While more calculation intensive, this opened up opportunities for increasing the fluidity between individual animations and removing unsightly pops between them.

Prince of Persia: Sands of Time was the first game to really focus on small transitions for fluidity. (Copyright 2003 Ubisoft Entertainment. Based on Prince of Persia®, created by Jordan Mechner. Prince of Persia is a trademark of Waterwheel Licensing LLC in the US and/or other countries used under license.)

A basic example of this would be an idle and a run. Having the idle immediately cancel and the run immediately play on initial player input will cause the character to break into a run at full speed, but the character will unsightly pop as he or she starts and stops due to the potential repeated nature of the player's input. This action can be made more visually appealing by blending between the idle and run over several frames, causing the character to more gradually move between the different poses. Animators should have some degree of control over the length of blends between any two animations to make them as visually appealing as possible, though always with an eye on the gameplay response of the action.

The situation above can be improved further (albeit with more work) by creating brief bespoke animations between idle and run (starting) and back again (stopping), with blends between all of them. What if the player started running in the opposite direction he or she is facing? An animator could create a transition for each direction that turned the character as he or she began running in order to completely control the character's weight-shift as he or she leans into the desired direction and pushes off with his or her feet.

What if the character isn't running but only walking? Again, the animator could also create multiple directional transitions for that speed. As you can see, the number of animations can quickly spiral in number, so a balance must be found among budget, team size, and the desired level of fluidity.

Seamless Cycles

Even within a single animation, it is essential to maintain fluidity of motion, and that includes when a cycling animation stops and restarts. A large percentage of game animations repeat back on themselves, so it is important to again ensure the player cannot detect when this transition occurs. As such, care must be taken to maintain momentum through actions so the end of the animation perfectly matches the start.

It is not simply enough to ensure the last frame of a cycle identically matches the first; the game animator must also preserve momentum on each body part to make the join invisible. This can be achieved by modifying the curves before and after the last frame to ensure they create clean arcs and continue in the same direction. For motion-capture, where curves are mostly unworkable, there are techniques that can automatically provide a preservation of momentum as a cycle restarts that are described later in this book.

Care should also be taken to maintain momentum when creating an animation that transitions into a cycle, such as how the stopping animation should seamlessly match the idle. For maximum fluidity, the best approach in this case is to copy the approved idle animation and stopping transition into the same scene to manually match the curves leading into the idle, exporting only the stopping transition from that scene.

Settling

This kind of approach should generally be employed whenever a pose must be assumed at the end of an animation, time willing. It is rather unsightly to have a large movement like an attack animation end abruptly in the combat idle pose, especially with all of the character's body parts arriving simultaneously. Offsetting individual elements such as the arms and root are key to a more visually pleasing settle.

Notably, however, games often suffer from too quickly resuming the idle pose at the end of an animation in order to return control to the player to promote response, but this can be avoided by animating a long tail on the end of an animation and, importantly, allowing the player to exit out at a predetermined frame before the end if new input is provided. This ability to interrupt an animation before finishing allows the animator to use the desired number of frames required for a smooth and fluid settle into the following animation.

Settling is generally achieved by first copying the desired end pose to the end of an animation but ensuring some elements like limbs (even divided

Uncharted 4: A Thief's End makes heavy use of "abort frames" to exit gameplay animations and cinematics before completion for fluidity. (Courtesy of Sony Interactive Entertainment.)

into shoulder and forearms) arrive at their final position at different times, with earlier elements hitting, then overshooting, their goal, creating overlapping animation. Settling the character's root (perhaps the single most important element, as it moves everything not planted) is best achieved by having it arrive at the final pose with different axes at different times. Perhaps it achieves its desired height (Y-axis) first as it is still moving left to right (X-axis), causing the root to hit, then bounce past the final height and back again. Offsetting in the order of character root, head, and limbs lessens the harshness of a character fully assuming the end pose on a single frame—though care must be taken to not overdo overlap such that it results in limbs appearing weak and floppy.

Readability

After interactivity, the next biggest differentiator between game and traditional animation, in 3D games at least, is that game animations will more often than not be viewed from all angles. This bears similarity to the traditional principle "staging," but animators cannot cheat or animate to the camera, nor can they control the composition of a scene, so actions must be created to be appealing from all angles. What this means is when working on an animation, it is not enough to simply get it right from a front or side view. Game animators must take care to always be rotating and approving their motion from all angles, much like a sculptor walking around a work.

Posing for Game Cameras

To aid the appeal and readability of any given action, it is best to avoid keeping a movement all in one axis. For example, a combo of three punches should not only move the whole character forward as he or she attacks, but also slightly to the left and right, twisting as they do so.

Similarly, the poses the character ends in after every punch should avoid body parts aligning with any axes, such as arms and legs that appear to bend only when viewed from the side. Each pose must be dynamic, with lines of action drawn through the character that are not in line with any axes.

Lines of action are simplified lines that can be drawn through any single pose to clearly illustrate the overall motion for the viewer. Strong poses can be illustrated in this way with a single arcing or straight line, whereas weaker and badly thought-out poses will generally have less-discernible lines that meander and are not instantly readable to the viewer. Lines that contrast greatly between one pose and the next (contrasting actions) promote a more readable motion for the viewer than multiple similar or weak poses.

For the motions themselves, swiping actions always read better than stabbing motions, as they cover an arc that will be seen by the player regardless of camera angle. Even without the aid of a trail effect, a swipe passes through multiple axes (and therefore camera angles), so even if players are viewing from a less-than-ideal angle, they should still have an idea of what happened, especially if the character dramatically changes the line of action during poses throughout the action.

All this said, work to the game being made. If the camera is fixed to the side, such as in a one-on-one fighting game, then actions should be created to be most readable from that angle. Similarly, if you are creating a run animation for a game mostly viewed from the rear, then ensure the cycle looks best from that angle before polishing for others.

League of Legends pushes animation incredibly far due to the far overhead cameras and frenetic onscreen action. (Courtesy of Riot Games.)

Silhouettes

At the character design/concept stage, the animator should get involved in helping guide how a character might look, not just to avoid issues such as hard, armorlike clothing at key versatile joints such as shoulders or waists. The animator should also help guide the design so as to help provide the best silhouettes when posed. A character with an appealing silhouette makes the job of animating far easier when attempting to create appeal than one composed entirely of unimaginative blobs or shapeless tubes for limbs.

Team Fortress 2 uses distinct character silhouettes for gameplay, making the animator's job of bringing appeal much easier. (Used with permission from Valve Corp.)

It is advisable to request "proxy" versions of characters at early stages of development so they can be roughly animated and viewed in the context of the gameplay camera, which, due to wide fields of view (for spatial awareness gameplay purposes), often warps the extremities of character as they reach the screen's edge. Generally, the most appealing characters look chunkier and thicker than they might in real life, due to them being warped and stretched once viewed from the wide-angle game camera.

Collision & Center of Mass/Balance

As with all animation, consideration must be given to the center of mass (COM; or center of balance) of a character at any given frame, especially as multiple animations transition between one another so as to avoid

unnatural movements when blending. The COM is generally found over the leg that is currently taking the full weight of the character's root when in motion or between both feet if they are planted on the ground when static. Understanding this basic concept of balance will not only greatly aid posing but also avoid many instances of motions looking wrong to players without them knowing the exact issue.

This is especially true when considering the character's collision (location) in the game world. This is the single point where a character will pivot when rotated (while moving) and, more importantly, where the character will be considered to exist in the game at any given time. The game animator will always animate the character's position in the world when animating away from the 3D scene origin, though not so if cycles are exported in place. Importantly, animations are always considered to be exported relative to this prescribed location, so characters should end in poses that match others (such as idles) relative to this position. This will be covered in full in the following chapter.

Context

Whereas in linear animation, the context of any given action is defined by the scene in which it plays and what has happened in the story up to that point and afterward, the same is impossible in game animation. Oftentimes, the animator has no idea which action the player performed beforehand or the setting in which the character is currently performing the action. More often than not, the animation is to be used repeatedly throughout the game in a variety of settings, and even on a variety of different characters.

Distinction vs Homogeneity

Due to the unknown setting of most game animations, the animator must look for opportunities to give character to the player and nonplayer characters whenever possible, and must also consider when he or she should avoid it.

If, for example, the animator knows that a particular run cycle is only to be performed on that character being animated, then he or she can imbue it with as much personality as matches the character description. It's even better if the animator can create a variety of run cycles for that character in different situations. Is the character strong and confident initially, but later suffers loss or failure and becomes despondent? Is the character chasing after someone or perhaps running away from a rolling boulder about to crush him or her? The level of distinction the animator should put into the animation depends on how much control he or she has over the context in which it will be seen.

The player character generally moves at a much higher fidelity and with more distinction than NPCs. (Copyright 2007–2017 Ubisoft Entertainment. All Rights Reserved. Assassin's Creed, Ubisoft, and the Ubisoft logo are trademarks of Ubisoft Entertainment in the US and/or other countries.)

If an animation is not designed for the player character but instead to be used on multiple nonplayer characters, then the level of distinction and notability should generally be dialed down so as to not stand out. Walks and runs must instead be created to look much more generic, unless the animation is shared by a group of NPCs only (all soldiers might run differently from all civilians). Almost always, the player character is unique among a game world's inhabitants, so this should be reflected in his or her animations.

Repetition

Similarly, within a cycling animation, if the action is expected to be repeated endlessly, such as an idle or run cycle, then care must be taken to avoid any individual step or arm swing standing out against the rest, lest it render the rhythm of repetition too apparent to the player—such as every fourth step having a noticeably larger bounce for example.

Stand-out personality can instead be added to on-off actions or within cycles via "cycle breakers" such as the character shifting his or her footing after standing still too long, performing a slight stumble to break up a tired run, or even by modifying the underlying animation with additive actions—covered in more detail in the following chapter.

Uncharted: Drake's Fortune utilized additive poses to avoid repetition when in cover.

Onscreen Placement

A key factor in setting the exaggeration of movement is the relative size on the screen of the character as defined by the camera distance and field of view. While cameras have gotten closer and closer as the fidelity of characters has risen, players still need to see a lot of the environment on screen for awareness purposes, so many games may show characters that are quite small. Far cameras require actions to be much larger than life so as to be read by the player.

The same is true of enemy actions that are far off in the distance, such as damage animations to tell the player he or she landed a shot. Conversely, only really close cameras such as those employed in cutscenes afford subtleties like facial expressions—here, overly theatrical gestures will generally look out of place. It is important as a game animator to be aware of the camera for any particular action you are animating and to animate accordingly within the style of the project. The wide field of view of the gameplay camera will even distort the character enough to affect the look of your animation, so, as ever, the best way to evaluate the final look of your animation is in the game.

Elegance

Game animations rarely just play alone, instead requiring underlying systems within which they are triggered, allowing them to flow in and out of one another at the player's input—often blending seamlessly, overlapping one another, and combining multiple actions at once to ensure the player is unaware of the individual animations affording their avatar motion.

If not designing them outright, it is the game animator's duty to work with others to bring these systems and characters to life, and the efficiency of

any system can have a dramatic impact on the production and the team's ability to make changes further down the line toward the end of a project. Just as a well-animated character displays efficiency of movement, a good, clean, and efficient system to play them can work wonders for the end result.

Simplicity of Design

Industrial designer Dieter Rams, as the last of his 10 principles of good design, stated that good design involves "as little design as possible," concentrating only on the essential aspects. A good game animation system should similarly involve no more design than required, as bloated systems can quickly become unworkable as the project scales to the oft-required hundreds or thousands of animations.

Every unique aspect of character-based gameplay will require a system to play back animations, from the navigation around the world to combat to jumping and climbing to conversation and dialogue and many more. Here, the game animator must aid in creating systems to play back all the varied animation required to bring each element of character control to life, and often the desire to create many animations will come into conflict with the realities of production such as project length and budget.

DOOM opted for full-body damage animations over body parts alone for visual control. (DOOM® Copyright 2016 id Software LLC, a ZeniMax Media company. All Rights Reserved.)

Thankfully, there are many tricks that a team can employ to maximize their animation potential, such as reuse and sharing, layering and combining animations to create multiple combinations, or ingenious blending solutions to increase the fluidity without having to account for absolutely every possible transition outcome. While the simplest solution is to do nothing

more than play animations in sequence, this will rarely produce the best and most fluid visuals, so the smartest approach is to manipulate animations at runtime in the game engine to get the most out of the animations the team has the time to create. Again, we'll cover some of the potential systemic solutions in the following chapter.

Bang for the Buck

Just as we look to share animations, being smart about choices at the design stage should create a workable method of combining animations throughout production. This will in turn prevent unique solutions being required for every new system. For example, a well-thought-out system for opening doors in a game could be expanded to interacting with and opening crates if made efficiently. When building any one system, anticipating uses beyond the current requirements should always be considered.

A good approach to system design will produce the maximum quality of motion for the minimum amount of overhead (work). It must be stressed that every new animation required not only involves the initial creation but later modification over multiple iterations, as well as debugging toward the end of the project. Every stage of development is multiplied by every asset created, so avoiding adding 20 new animations for each object type is not only cost effective but allows more objects to be added to the game. (All that said, sometimes the solution to a system is just to brute-force create lots of animations if your budget allows it.)

Sharing & Standardization

As mentioned earlier, it is important to know when to keep animations generic and when to make unique ones for each example. If the game requires the player character interact with many objects in a game, then it would be wise to standardize the objects' sizes so one animation accommodates all objects of a particular size.

The same goes for world dimensions, where if a character can vault over objects throughout the game, then it makes sense to standardize the height of vaultable objects in the environment so the same animation will work anywhere—not least so the player can better read the level layout and know where the character can and cannot vault.

Gears of War featured high and low cover heights supported by different sets of animations. (Copyright Microsoft. All rights reserved. Used with permission from Microsoft Corporation.)

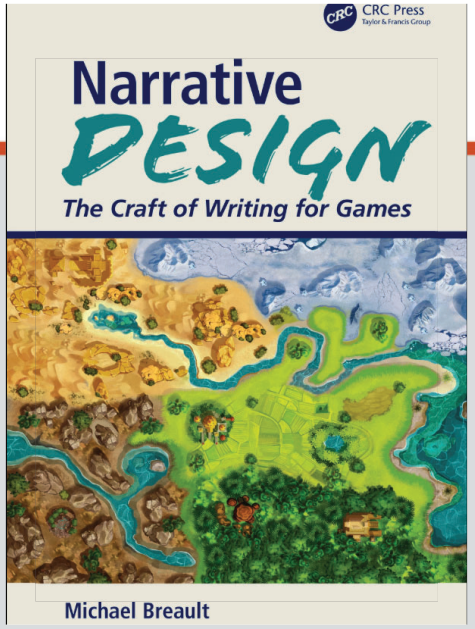
That said, if your gameplay is primarily about picking up objects or vaulting over things, then it may be worth creating more unique animations to really highlight that area and spend less effort elsewhere. This, again, feeds back into the idea of bang for the buck and knowing what is important to your particular game.

All these decisions must come into play when designing systems for your game, as very few teams can afford unique and bespoke animations for each and every situation. Nevertheless, beautiful game animation can come from even single-person teams that focus on one thing and do it very very well. This is the crux of what good design is, and every aspect of game development benefits from clever and elegant design, regardless of game type.

CHAPTER

5

STORY IN GAMES



This chapter is excerpted from

Narrative Design

The Craft of Writing for Games

by Michael Breault

Learn more

CHAPTER 3

Story in Games

WHAT IS A STORY?

Every game has a story. That story is the foundation that gives context to the players' actions. Good game stories empower players with agency, enabling them to drive the story forward. It lets them know where they fit into the game world and what their goals are in the game.

A game's story doesn't have to involve any writing, dialogue, or text. Chess is an abstract strategy game with a story. Two kingdoms meet on a battlefield. Each king has troops to fight for him, and each side's goal is to capture the opposing king. The names and shapes of the pieces—knights, pawns, bishops, king, queen—are evocative of the historical origins of chess. That story gives context for chess players; the goal conveyed by that story informs every action players take in that game, whether they realize it or not.

Courtesy of libreshot.com.

The story for *Monopoly* is equally simple. Each player is a wheeler-dealer in the real estate world of Atlantic City. Players strive to corner the market on the city's properties and bankrupt all their rivals. Last player standing is the city's new real estate mogul. That story and end goal drive every player's activities in *Monopoly*.

A game's story isn't something that gets in the way of gameplay. And it shouldn't be set up with a wall of text from some second-string character, like the village elder at the start of a fantasy adventure. At its most basic, a game's story forms a background for gameplay and gives context to players that helps motivate them to continue to play. In a well-developed game, story enhances gameplay and never gets in the way of players' enjoyment of the game.

Think of *Shadow of the Colossus*. It's an action game about a young man fighting 16 giant monsters in a forbidden land. The script is short on dialogue—large stretches of the game, including the first five minutes, are wordless. Yet players and critics view it as one of gaming's greatest adventures and fondly remember the protagonist and his horse. By limiting dialogue and explanation, *Shadow of the Colossus* establishes a deep sense of mystery and foreboding. Its script does more with less.

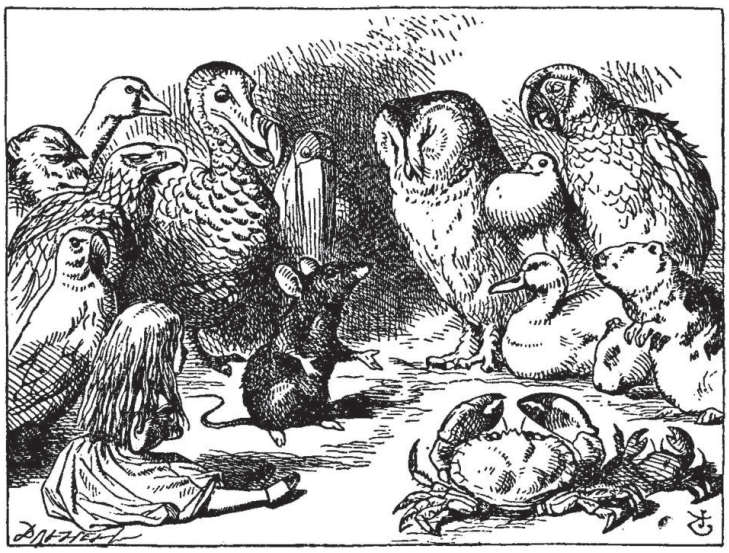
STORY IN GAMES

In a video game, stories serve several purposes:

Stories give context for players, grounding them in the game's fiction and world. They tell players what the game is about, what part they play in the story, and what they are supposed to do in the game. This helps engage players in the game world.

Stories motivate players to continue by intriguing them with what's around the next corner. One of the primary goals of a game's story is to lead players on with mysteries, subplots, and engaging characters. Games with more complex stories will have a rolling collection of subplots, with new ones opening up as old ones are resolved, to enliven the game's main story. The quests in an role-playing game (RPG) like *Skyrim*, for instance, involve characters with stories and troubles that represent subplots scattered liberally throughout the game.

Finally, stories can keep players playing even when gameplay gets stale. There are a limited number of mechanics and unique areas that can be added to a game. Those assets are expensive, both in terms of developer time and memory requirements. There's always a risk of players getting



Courtesy of Pixabay, at https://pixabay.com/vectors/alice-in-wonderland-animals-tale-30130/.

bored after lopping off the head of the one-thousandth orc. But what might that orc have been guarding? What twist will players discover as they explore his lair? Adding story elements is much faster and cheaper (in development costs) than motivating players with new progression systems ("let's add fishing!") or rewards. An evolving story gives players a reason to keep plowing through all those orcs.

One thing to remember about game stories—in almost all big-budget games, story is secondary to gameplay. Story exists to support the gameplay experience the team wants players to receive. Every aspect of the story should work to enhance that experience. Players are in a game to enjoy *gameplay*, not read masses of text, listen to endless spiels from NPCs (non-player characters), or watch interminable cinematics.

As much as possible, put yourself in the player's shoes. Does the story you're adding support gameplay and motivate the player, or does it add friction at a time when the player wants to rush forward?

WHO CREATES A GAME'S STORY?

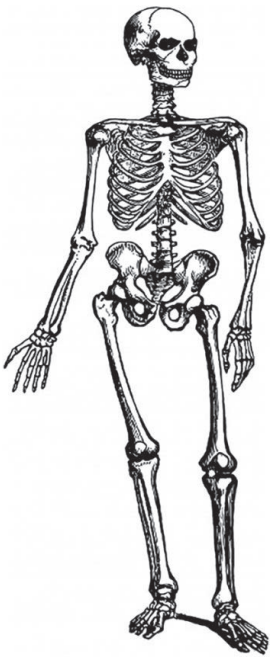
The game's story is the narrative designer's responsibility, but the project's creative director (CD) and game design lead usually determine the overall direction of the story, which needs to complement the intended gameplay experience. The actual story details are left to the narrative designer to flesh out, with the aid of other game designers and scripters who help implement it and add their own creative touches.

The CD may supply the overarching vision, but it's the narrative designer's job to transform that vision into concrete design documents and a written script. The narrative designer oversees the day-to-day efforts to bring the CD-approved story to life. Think of the narrative designer as the CD's point person in charge of story integrity.

PLOT VS. STORY

Many people use the words "plot" and "story" interchangeably, but they have very different meanings. Plot is the skeleton of the story, the series of events that unfolds from the game's start to its finish. Story is created from that plot when you add characters to the mix; how the plot's events affect the characters and how they react to those events are the truly compelling aspects of a story.

PLOT



Courtesy of FreeSVG.org

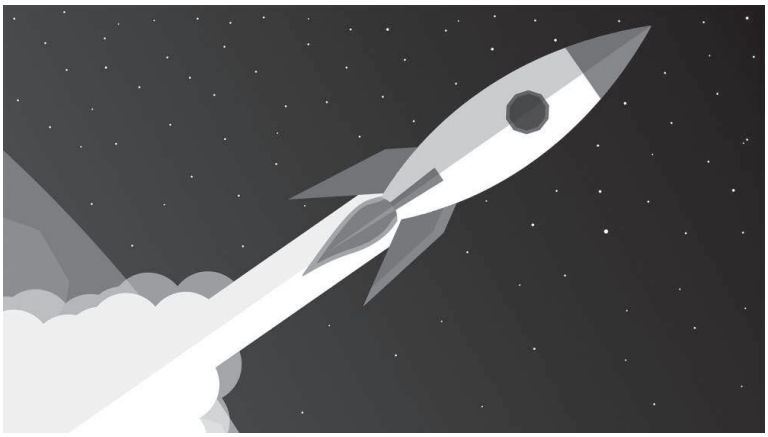
STORY

Courtesy of Eric J. Cutright, Wikimedia Commons. https://commons.wikimedia.org/wiki/File:US_Navy_070504-N-0995C-072_Chief_Mineman_Kevin_Sperling_appears_as_the_guest_body_builder_at_an_Armed_Forces_body_building_competition_held_at_Sharkey%27s_Theatre_at_Naval_Station_Pearl_Harbor.jpg

People are fascinated by people. A dry recounting of a sequence of events leaves readers unmoved, no matter how important the events might

be. But show readers the human impact of those events, and you have a story worth reading. Characters make a story. Designers often forget this, getting caught up in the excitement of building a world, mistaking history and events for a story that will engage players.

When I talk to my narrative design students about plot vs. story, I give them an assignment to help them see how an intriguing story can develop from a simple plot. This assignment (the "Arcade Game Short Story" assignment in the Narrative Design I class in Appendix C) asks students to choose an arcade game that currently has no real story and to write a short story about that game. The story needs to be set in the world depicted in the game and it needs to focus on a character in that game. Good choices for this assignment are games like *Frogger*, *Asteroids*, *Pac-Man*, *Space Invaders*, *Galaga*, *Dig-Dug*, and so on.



Courtesy of GoodFreePhotos.com.

These arcade games contain a simple plot (frog crosses road, spaceship pilot navigates an asteroid field, etc.). Students are asked to create a story from that plot. Those stories need to revolve around the characters in the arcade games. The stories need to tell the reader who those characters are, how they got into that situation, what they're doing there, and so on. They need to turn a sequence of events into an interesting story.

Students have come up with some great stories. Did you know, for example, that Pac-Man is a drug addict, running around the corridors of a psychiatric hospital, popping pills, and hallucinating he's being chased by ghosts? Or that Frogger is a college kid in a frog costume, drunkenly staggering home from a Halloween party? One student brought the world of Missile Command to life by setting it within the context of the Cuban Missile Crisis of 1962.

These kinds of adaptations happen all the time outside the classroom. Several of the games I've worked on have been game adaptations of someone else's IP (intellectual property). While I was still at TSR back in the mid-1980s, working on Dungeons & Dragons (D&D) and Advanced Dungeons & Dragons (AD&D) games, I also worked on a pen-and-paper Conan RPG and an Indiana Jones RPG; those were adaptations from books and movies, respectively. In the video game industry, I've worked on a *Jeopardy* game, a game based on the *Robin Hood: Prince of Thieves* movie, one based on the *Punisher* comic books, the *Carmen Sandiego* TV series, the *Top Gun* movie, and more.

Learning to take an existing IP and convert it into a game is a valuable skill for game designers and narrative designers. People who have never done this before often think of the existing IP as a straitjacket that limits a game's tone and setting. In my experience, it's actually very freeing. The IP gives you the basic setting, background story and characters, and a solid foundation for your game's story. This frees you to create a unique story for players set against a background they already know and love.

We work on game adaptations in my narrative classes, first with the arcade game short story assignment and later with a concept document assignment aimed at envisioning a game made from an IP that already exists in another medium. The instructions for both assignments, plus the template for the game adaptation concept doc, can be found in Appendix C.

TELLING STORIES WITHOUT WRITING

One of the key lessons narrative designers learn is how to tell stories without words. Visual storytelling in games is always a joint effort between narrative designers, artists, and level designers. Working together, you build an environment that tells the story to the player. When players work out what's happening (or learn what has already happened) by exploring the world around them, they engage more strongly with the story and world and feel like they're discovering these details rather than simply being talked at.

Imagine walking into a village. A resident runs up to you and immediately launches into a long, boring story about how a dragon attacked the village five years ago and they've been slowly rebuilding ever since.

Courtesy of Planet-Science.com.

That's how you force-feed a story to players. Now imagine that you walk into the same village and are greeted by the same person. This time, they walk you through the village, talking about some problem the village is currently suffering from. As you walk along, you notice a few toppled structures and faded scorch marks nearby. Ivy has grown up along the walls and partially covered up the damage. There are also newer buildings in town, all of them made of stone instead of wood. Without the villager saying a word about it, you've likely pieced together the story: A fire-breathing dragon attacked the village, but it happened long enough ago that plant growth has covered up some of the damage. Maybe later you can find someone to ask about this.

That's visual (or environmental) storytelling. It allows the player to piece together the story and leaves room for them to make up their own version of events. This gets the player more engaged with your game and its world.

Narrative designers also work with sound designers to tell stories via audio cues. Screams off in the distance or monstrous roars from unseen sources will get your players' minds racing with possibilities (and tension).

When I worked on the *Elder Scrolls Online*, most of my time on that project was spent in the PvP (player vs. player) province of Cyrodiil. I worked with the artists, level designers, and our Quality Assurance (QA) team to build visual storytelling into that vast landscape. I asked the rest of the team to search for areas that didn't have much going on and to think

of interesting objects to add. In a cave along a stream I added three skeletons, one of them clutching a note from their commander ordering them to spy on the enemy camp. A level designer added an excavated hole with an open casket at the bottom; I added a note from a nephew apologizing to his uncle's ghost for disturbing his rest and stealing a family heirloom. Our QA team sent in reports of dull areas; artists and level designers would take a look and add something interesting to it. Visual storytelling takes effort and coordination among different disciplines, but it's almost always worth it.