EXPLORING THE FOUNDATIONAL PRINCIPLES BEHIND
GOOD GAME DESIGN

A GAME DESIGN VOCABULARY

Anna **ANTHROPY**
Naomi **CLARK**

Praise for *A Game Design Vocabulary*

"*A Game Design Vocabulary* succeeds where many have failed—to provide a broad-strokes overview of videogame design. Utilizing analytic smarts, an encyclopedic knowledge of games, and subcultural attitude, Naomi Clark and Anna Anthropy get to the heart of how games work.

"Why is this book important? Videogames are the defining mass medium of our time, yet even those who make games lack a clear language for understanding their fundamental mechanics. *A Game Design Vocabulary* is essential reading for game creators, students, critics, scholars, and fans who crave insight into how game play becomes meaningful."

—**Eric Zimmerman**, Independent Game Designer and Arts Professor, NYU Game Center

"*A Game Design Vocabulary* marks an important step forward for our discipline. Anna Anthropy and Naomi Clark's extraordinarily lucid explanations give us new ways to unpick the complexities of digital game design. Grounded in practical examples and bursting with original thinking, you need this book in your game design library."

—**Richard Lemarchand**, Associate Professor, USC, Lead Designer, *Uncharted*

"Anthropy and Clark have done it! Created an intuitive vocabulary and introduction to game design in a concise, clear, and fun-to-read package. The exercises alone are a great set of limbering-up tools for those new to making games and seasoned designers, both."

—**Colleen Macklin**, Game Designer and Professor, Parsons The New School for Design

"Two of my favorite game design minds sharing a powerful set of tools for designing meaningful games? I'm so excited for this book. *A Game Design Vocabulary* may very well be the best thing to happen to game design education in more than a decade. I can't wait to put this book in the hands of my students and dev friends alike."

—**John Sharp**, Associate Professor of Games and Learning, Parsons The New School for Design

"Some of the greatest challenges to the intelligent advancement of game-making can be found in the ways we conceptualize and discuss them. This simple yet profound new vocabulary is long-overdue and accessible enough to help new creators work within a meaningful framework for games."

—**Leigh Alexander**, Game Journalist and Critic

This page intentionally left blank

A Game Design Vocabulary

This page intentionally left blank

A Game Design Vocabulary

Exploring the Foundational Principles Behind Good Game Design

Anna Anthropy
Naomi Clark

Addison-Wesley

Upper Saddle River, NJ • Boston • Indianapolis • San Francisco
New York • Toronto • Montreal • London • Munich • Paris • Madrid
Capetown • Sydney • Tokyo • Singapore • Mexico City

Library of Congress Control Number: 2013956696

ISBN-13: 978-0-321-88692-7
ISBN-10: 0-321-88692-5

Text printed in the United States on recycled paper at RR Donnelley in Crawfordsville, Indiana.

First printing, March 2014

Editor-in-Chief
Mark Taub

Executive Editor
Laura Lewin

Development Editor
Michael Thurston

Managing Editor
Kristy Hart

Project Editor
Elaine Wiley

Copy Editor
Gill Editorial Services

Indexer
Erika Millen

Proofreader
Anne Goebel

Technical Reviewers
Colleen Macklin
Sarah Schoemann
John Sharp

Publishing Coordinator
Olivia Basegio

Cover Designer
Chuti Prasersith

Book Designer
Bumpy Design

Compositor
Nonie Ratcliff

To Brenda Romero, whose first game project was the first digital game Naomi ever played, and who has always stood up for better design and community in games;

and to Greg Costikyan, whose bold words on independent development and finding vocabulary to design with have been an inspiration to a generation.

Contents at a Glance

This page intentionally left blank

Contents

FOREWORD

In case you haven't noticed, something is happening in the world of video games, something that is changing the way we think about how they're made, how they're played, and what they mean. The authors of this book are part of a new generation of game creators for whom video games interface fully with all the complex machinery of contemporary culture. For Anna and Naomi, video games are not merely sleek consumer appliances dispensing entertaining power fantasies, they are fragments of shattered machines out of which new identities can be constructed; sites where disorderly crowds can assemble for subversive purposes; platforms from which to examine the status quo; windows into the turbulent flow of power and progress; unruly machines that call into question their own means of production; smart machines that allow us to say new things; and, when correctly operated, beautiful machines that kill fascists.

We are used to other kinds of culture interfacing with all of these dimensions—music, film, literature; these things have long been understood as a domain of identity construction and political struggle. But it's still something of a novelty to understand video games the same way, to pay close attention to not just their form and content, but to their context, to think about the personal voices of the individual creators, the communities that gather around them, and the deeper currents they illuminate.

Having earned a reputation for conservatism, for doggedly clinging to the safety blanket of childishness, for being unwilling or unable to confront the ambiguous complexities of all the meanings they generate, video games are suddenly shocked to find themselves holding a live wire. Coiling, sparking, hazardous, yes, but it's also more than a little bit exciting to discover that what we thought was just a bit of old rope is in fact writhing with dangerous energy. And it is people like the authors of this book, the most progressive members of this new generation, who are plugging it in.

Which is exactly what makes it so important that this is a book about the fundamentals of game design as a craft. This is not a wild-eyed manifesto about the political meaning of video games; it is a patient explanation of how they work—breaking them down to their essential elements and carefully demonstrating how those elements fit together. This is a book about moving and jumping, about pressing and releasing buttons, about color and shape, enemies and hit points, challenges and goals.

The book is organized in two parts. In Part One Anna lays out the basic building blocks of video game design, and in Part Two Naomi shows the different ways these ingredients can be

combined to express an infinite variety of game ideas. But throughout the book there is a careful attention to the most fundamental aspects of game design.

This focus on the fundamentals makes *A Game Design Vocabulary* a very good book for new designers. Basic concepts are illustrated with concrete examples, demystifying what can be a very complex and intimidating process. And this demystification reveals the radical agenda beneath the sober surface of this book, because it's about lowering the barrier of entry into this world, welcoming new hands, new eyes, new voices, and showing them that video games are not mysterious cultural objects to be consumed, they are mysterious cultural objects *you make yourself*. They belong to you, and the first step of owing them is to look at them carefully and understand how they function.

At the same time, I believe this book will be equally valuable for experienced designers. There is no better way for a veteran developer to sharpen the blade of her creative practice than by meditating on the design fundamentals outlined in this book.

Ultimately, I think *A Game Design Vocabulary*'s commitment to the fundamentals of form is itself the book's most radical idea. Some people see a conflict between the revolutionary power of games as a means of expression and a more traditional focus on their formal details, but Anna and Naomi refuse to recognize this division. For them it is obvious that the expressive power of video games flows *through* their formal qualities, that attention to the nuts and bolts of video game design is not a way to avoid confronting all the subtleties of their layered meanings, but a way to trace them, highlight them, and illuminate them.

This most radical idea could simply be put: *the aesthetic is political*. Video games matter and they matter not just in what they are, but in what they say, and not just in what they say, but how they say it.

—Frank Lantz, Director, NYU Game Center

ACKNOWLEDGMENTS

Thanks to Phoebe Elefante, Colleen Macklin, John Sharp, Laura Lewin, Michael Thurston, Olivia Basegio, Sarah Schoemann, and Toni Pizza for assistance in editing and writing; to Emily Short, Eric Zimmerman, Frank Lantz, Ian Bogost, Mattie Brice, Steve Swink, and Mary Flanagan for inspiration on design and the shape of games; and to Keith Burgun for sparring partnership and sword-sharpening.

ABOUT THE AUTHORS

Anna Anthropy is an artist, author, and game creatrix working in the East Bay area. As an ambassador for game creation, she works to empower marginalized voices to gain access to game creation. Her first book, *Rise of the Videogame Zinesters*, is an autobiography/manifesto/DIY guide. She's radical.

Naomi Clark has been designing and producing games for more than two decades, ever since she started creating text-based virtual worlds as a teenager. She has worked on multiplayer web games (*Sissyfight 2000*), casual downloadable games (*Miss Management*), Flash games for kids (*LEGO Junkbot*), and Facebook games (*Dreamland*) while working with companies like Gamelab, LEGO, Rebel Monkey, and Fresh Planet. Naomi has also taught classes and workshops at Parsons School of Design, the NYU Game Center, and the New York Film Academy, and she has written game analysis and feminist critique for *Feministe*. She is currently developing an independent game with the Brooklyn Game Ensemble.

This page intentionally left blank

PART I

ELEMENTS OF VOCABULARY

By Anna Anthropy

This page intentionally left blank

CHAPTER 1

LANGUAGE

This is a book about game design—videogame design, specifically. In 2014? Why? We've been making digital games for more than 50 years, if you take *Tennis For Two* (1958) as an arbitrary starting point. You'd think 50 years would give game creators a solid foundation to draw from. You'd think in 50 years there'd be a significant body of writing on not just games, but the craft of design. You'd think so, but you'd be disappointed. Every day, playing contemporary videogames or reading about them, I see evidence that what both creators and critics desperately need is a basic vocabulary of game design.

Signs Versus Design

New Super Mario Bros. Wii, released by Nintendo in 2009 (see Figure 1.1), is a sequel or a remake of *Super Mario Bros.* from 1985. Though the newer game diverges pretty quickly in design from its progenitor, the first few screens of the first level of *New* are arranged in deliberate mimicry of the same screens from the 1985 version. The player (or players, in the case of *New Super Mario*) starts on the left side of the screen; to the right, there's an enticing, flashing block with a question mark on it, floating just above the ground. Then the game's most basic enemy trundles toward the player to the left. After that, you see two parallel platforms made of hovering blocks, some breakable, some that contain rewards, one of which contains power-up items for the players. After that, there's a tall obstacle that the player has to jump over to progress further: a big green pipe in the 1985 game, the side of a cliff in the 2009 one.

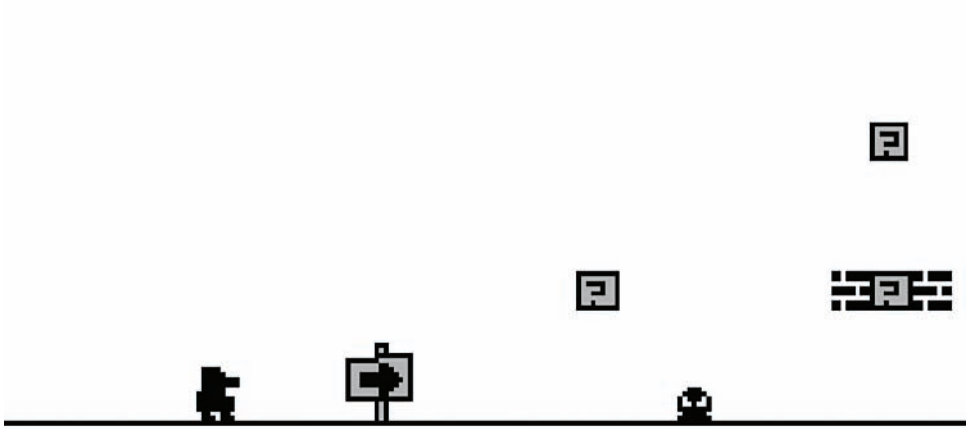


Figure 1.1 *New Super Mario Bros. Wii* starts with an arrow pointing to the right.

Super Mario Bros. was many people's first videogame; there were almost no games similar to it at the time. *New Super Mario Bros.*, in contrast, has almost twenty years of related games as precedent. Despite that, the 1985 game leaves one thing out that's present in the 2009 game: a big sign with an arrow telling the player which direction to go.

What happened between 1985 and 2009 to cause game creators to lose that much trust in the player? The player of *New Super Mario Bros. Wii* gets off easy, in fact, as far as "tutorials" go. Lots of contemporary games feel the need to explain to the player, via game-interrupting exposition and big stupid dumps of instruction text, how they are played. Many games even keep the player from starting the game until she's proven she knows how the buttons work, making her jump in place, in a contextless situation, like a trained pet.

This is shockingly popular. I see it not just in the big-budget commercial games that have the economic incentive to keep as few players from getting confused as possible, but also in smaller games, in freeware games, in games created by one or two people working out of their bedrooms. When I met Pietro Righi Riva, one of the creators of the downloadable game *Fotonica*, at the 2012 Game Developers Conference (GDC), the first thing he said to me referred to my take on *New Super Mario Bros. Wii:* "You were right. That game didn't need a tutorial." This kind of blunt instruction speaks not just to a disrespect for the player's intelligence—and one that influences how she feels about the game, make no mistake—but also to a lack of confidence on the part of the creator.

Super Mario Bros., 1985, didn't need a tutorial. It used design, a communicative visual vocabulary, and an understanding of player psychology—gained from watching players play the game, changing it, and watching them again—to guide the player to understanding the basics of the game. Those first screens teach everything the player needs to know: Mario starts on the left of an empty screen, facing right. The floating, shining reward object and the slow but menacing monster—set in opposition to Mario by walking in the opposite direction—give the player an incentive to jump. The platforms are a kind of jungle gym where the player can experiment with jumping, discover the properties of various kinds of blocks, and encounter her first power-up. Even if the player's not sure whether the power-up is dangerous, it moves too quickly and in too confined a space to be avoided. When the power-up turns out to benefit Mario by making him grow, the player has learned something about how monsters and power-ups look and behave in this game. Then the final pipe barring access to the rest of the game makes sure she knows that the height of her jump is dependent on how long she holds down the button.

You can argue that coding a game in 8605 Assembly for the Nintendo Entertainment System in 1985 was much more demanding, and building a dedicated "tutorial" into the game would have been harder. People like to point to technological justifications for things in digital games because most videogame fans are sold on the idea that the history of games is a history of technology. If there were technological reasons that dissuaded the designers of *Super Mario*—Shigeru Miyamoto and Takashi Tezuka—from training the player through instruction text and encouraged them to use design to teach the player, then God bless the limitations of 1980s game machines. Design is not technology. The printed manual packaged with the game contained more information about how to play, but perhaps keeping in mind how often manuals go unread or get lost far before the software they accompany, Miyamoto and Tezuka made sure that the game itself could convey understanding through playing.

Someone in 2009 looked at the opening screens of the original *Super Mario Bros.*—someone had to, to copy these screens note for note into the first level of *New Super Mario Bros. Wii*—but didn't understand what they meant or why they were so effective. Why are game creators unable to understand and learn from their own history? Why are they bumbling over problems that were solved almost 30 years ago?

Digital games have exploded commercially since 1985—in fact, *Super Mario Bros.* was proceeded by more than a decade of successful videogames—and we've consequently learned a lot of new words with which to talk about and describe videogames. Unfortunately, those words come from marketers, brand-loyalty Internet arguments, and magazines that exist as extensions of publishers' PR departments. The language that exists to describe videogames is facile when applied to the very real problem of discussing design.

Most designers, lacking the vocabulary with which to discuss, analyze, and criticize game design, operate largely by intuition and instinct. And there's a lot to be said for intuition and instinct: A lot of radical decisions are made by instinct and then only understood in hindsight. But what if a designer is working in a team? What if someone else is drawing the characters that will appear in a game? What do they need to convey, and what does the designer need to tell them? What if a designer is working with another designer? How will the two communicate about the needs and direction of the game?

I'm not the first person to notice this problem. Back in 1994, game designer Greg Costikyan wrote an essay all about it, called "I Have No Words & I Must Design." At the beginning, he says, "We need a critical language. And since this is basically a new form, despite its tremendous growth and staggering diversity, we need to invent one." He was right then, and he still is.

Consider that we're all in a team—difficult in light of the practices of most contemporary publishers, I know—and that we all have access to this tremendous, growing resource of game design solutions: every videogame that has ever been made. By understanding those games—how they work or don't work, what they're doing and why—we get better at making our own games. We don't repeat problems that were long ago solved, like how to convince the player to go right. But how can we understand those games if we don't have a language with which to talk about them? How can we have a discussion?

Once upon a time, I studied creative writing. Someone would submit a story, everyone else would read it, and then we'd sit in a circle and people would offer their critiques, with the goal of allowing the author to improve the story and, in the process, improve her own writing ability. This was called "workshopping" a story. We would talk about things like how a story was paced, how certain passages or phrases helped—or failed—to characterize the characters of the story, which parts were weak, and which succeeded.

No game creator wants to put a tutorial into her game, to make the player press the jump button five times before being allowed to press the shoot button five times. A game creator puts a tutorial into a game because she lacks confidence in her ability to teach the player the rules of her game without explicitly stating them upfront. In a board or card game, it makes sense that the players should be aware of the rules upfront because they're the ones keeping the rules. But the great strength of digital games is that, because the computer is performing the task of enforcing the rules and tracking the numbers, the game can withhold some of the complexities

of the rules from the player. When the player discovers those complexities later, it feels like a story is developing.

How do we lead the player to those discoveries? That's called "design." And, frankly, I don't think we, as designers, are doing enough of it.

What game designers need is a workshop—the means to design, have their design critiqued, and improve their craft. We need to be able to discuss design as a craft. And if we're going to discuss game design, the first thing we need is a vocabulary.

Failures of Language

We're not lacking for words to use to describe videogames. But those words were created to sell videogames, not to describe the process of creating and understanding them. Our games vocabulary is peppered with buzzwords, invented by someone in marketing for a press release and regurgitated into a games magazine. Next the words are on the Internet, slung back and forth by forum posters, and then, finally, I hear an otherwise intelligent game developer use a meaningless word to describe a game.

Here's a brief glossary of some of the words I hear a lot and what they might mean:

- **Immersive**—Game takes place underwater
- **Fluid**—Game is actually made of water
- **Flow**—Current of the liquefied game

These words don't have to be nonsensical. In fact, we'll be talking about meaningful ways to talk about "flow" later in this book. When buzzwords are used without context or nuance to promote a game, as part of a press release or blurb, they might as well be meaningless.

When we use meaningless words to talk about games, our ability to describe them becomes more confused; our language for describing them becomes less concrete. But we've bought into this sort of thing in a big way, the same way we've bought into the idea that a game is composed of "graphics," "audio," and "replayability." We're used to thinking of games in those terms, but who gave us those terms?

It was the games press. The terms we think about videogames in are taken from *Consumer Reports*–style reviews of games. *GamePro* magazine would divide games into "graphics," "sound," "control," "fun factor," and "challenge" and then give the game a score of one to five in each of these categories. Doesn't the way a game looks have a relationship to how it plays, though? Don't the way things move in a game tell you a lot about how the game controls? Don't sounds characterize the interactions that they accompany? Doesn't the challenge of the game affect what the experience of the game is—the "fun factor"?

The fact is that although these categories may seem dated, we nonetheless allow them to inform the way we think about games. Instead of considering a game holistically, we mentally divide games into categories. It's especially easy to do within a bigger group or studio, where all these categories may be separate jobs performed by separate people. But what something in a game looks like, for example, tells the player what to think about it, what expectations to have. "Graphics" are part of design. So is sound, and how the game controls, and every part of the experience of a game. We're trained to think of all these parts of a game in isolation.

Our language limits us in other ways. We've bought into the established "genres" of video-games: the shooter! The strategy game! The platformer! These categories make it hard to describe, to pitch, to even imagine games outside of the ideas that are already established. When I created *dys4ia* in March 2012, an autobiographical game about my own experiences with hormone therapy, many players and critics, though they admired the game, questioned whether it actually was a game after all, because it didn't fit their genre-influenced preconceptions of how games should work and what "ought to" happen when you play them.

The language that we use to talk about games constrains the way we think about them. We don't have a vocabulary that can fit games that are as diverse as, say, a game about hormone replacement therapy that relates events that really happened to me and isn't a struggle for victory or dominance. And so the language of games is a language of exclusion. Game culture's vocabulary frames discussions in such a way as to perpetuate the existing values and ideas of that culture, which is problematic when that culture is so insular to begin with.

dys4ia is a traditional game in many ways. It borrows a lot of established game vocabulary to tell its story. Most scenes involve guiding some player-controlled character around the screen to perform a given task. The reason both players and creators fail to recognize it as a game is superficial—we lack the design vocabulary to connect a game about hormone replacement with related games that have more traditional themes.

When I mention "story" in a game to most players and developers, what they think of is cutscenes: an interruption of a game to show a five-minute movie, directed in obvious imitation of a Hollywood production. Or they think of a wall of expository text that the player has to stop and read or, more likely, skip annoyedly past. This is just another symptom of designers' fear of design. The truth is that we already have all the tools we need to tell stories in games—to tell real stories, not exposition—but we don't understand those tools.

Until we learn how to tell real stories in games, "story" is always going to mean "cutscene." Until we learn how to design holistically, games are always going to be broken into "graphics" and "sound" and "control." Until we have a language that can describe games in all their diversity, we will only design "shooters," "strategy games," and "platformers."

By equipping ourselves with a language for talking about design, we are giving ourselves the ability to design.

A Voice Needs Words

When I was little, game development was mystifying to me. I couldn't imagine how any human being could create a game and had no idea where one would even start. By creating a real discourse on game design, we're not only helping existing game creators become sharper, but empowering new game makers with a vocabulary with which to start thinking about and planning design. We're actually giving established creators a means of communicating ideas about design to a newer generation. We're enabling all creators to communicate with and improve each other.

And though people who create games naturally have the most to gain from a real conversation about design, they're not the only ones who would benefit. I'm thinking of critics of games, but not just journalists. We would all become better critics of games—better able to understand them, to analyze them, to communicate about them—if we could cultivate an environment where real talk about games and what they're doing and why was commonplace.

We could have a culture that better appreciates and values games. It may seem ridiculous to suggest that games are undervalued in a culture where tens of thousands of fans flock to conventions like the Penny Arcade Expo to reinforce the great myth that developers and publishers are greater than human. But this isn't appreciation; it's fetishization. Because the myth that game developers are something other than human is just that: a falsehood. But it was this falsehood that kept me, as a child, from realizing that game design was something that I could do and even earn a living doing.

Imagine an audience of players equipped with the understanding to follow and appreciate what game developers are doing rather than merely idolizing them. Certainly there's a "magician's bag of secret tricks" brand of appeal to designing games. After all, we're designing experiences that manipulate players' mental and emotional states (consensually and nondestructively, I would hope). There might be a fear that once players can see the smoke and mirrors, they'll lose a sense of wonder at the trick.

Discussing pacing and expository and characterization techniques in writing has not diminished my appreciation for the written word and admiration for those who can use it well. In contrast, my respect for writing has only deepened with my understanding of technique. I think the average reader is more literate than the average player—not "literate" in the dumb, obvious sense of having read more books, but in the sense of having a wider understanding of the craft that goes into the form they enjoy. It's not surprising that readers might have a better understanding of what they're reading than players have of what they're playing. Not only have the novel and short story been around longer, but writers, being writers, are much better equipped to write *about* the craft of writing and have done so at length.

A "literate" player wouldn't necessarily be a more jaded and dismissive one (we have plenty of those already) but could be a more attentive one, one who was more receptive to weirder ideas.

In my experience as a designer and creator of games, I've had only a precious few experiences where a critic really impressed me with her insight into and attention to one of my creations. Those experiences remind me why I create—to have someone connect with and understand the thing I have designed.

They were also experiences that gave me a better understanding of my own work. What a critic does is articulate an idea that's at work in a game, puts it in a context with other games, with other schools. They help explain the work to others; they start a discussion.

That's what we do when we talk about design and our design decisions: we start a discussion. And we allow others to join in that discussion, to participate in the dialogue, to contribute. Why is this subject important enough to warrant a book? It's not just so that a handful of industry developers can consider themselves a little more savvy. It's because shattering the silence around game design creates a conversation that everyone can learn from, whether they want to become game creators, whether they didn't realize they wanted to make games until they learned that developers are just as human as they are, whether they want to be informed critics, or whether they're content just to be better-educated players. An open conversation about game design demystifies this form that we care about and empowers us with the means to better understand, think about, and, if we wish, to make digital games.

A Beginning

What is this book? It's my attempt at furthering the discussion of design that we need so badly. We need more books that can kick off this conversation and give it places to start. For a while I was attending a game school called The Guildhall at Southern Methodist University, majoring in level design (I got kicked out after a few months), and it was pretty clear to me that my instructors didn't know where to begin teaching design. We watched videos about parallax scrolling in Disney movies, and we took a test on *The Hero's Journey*.

Now, I'd be the first to admit that game design is "interdisciplinary"—that game designers benefit from having a lot of different skills, from understanding things like how to animate depth to what kind of stories players expect—but I still saw this wild grasping for subjects as a symptom of the lack of a foundation from which to teach game and level design.

I also vaguely remember the level design textbook we had to read, which was biased toward a single kind of game. Remember what I said about games discourse reproducing the same kinds of games over and over? The book was clearly written with first-person shooters in mind; I remember a whole chapter on lighting. And while the principles of using lighting to create a mood are interesting and definitely of use to a level designer, we should save the specifics for after we have a grasp of the basics.

Since Greg Costikyan pointed out how badly we needed a vocabulary, many books on game design and development have been written. Some revolve around a particular kind of game; others talk about how to work on big teams with programmers, artists, and project managers, which is great if you're going to work at a huge company, but it's not quite as useful if you're part of the growing number of game creators working in really small groups. We've got game design books that focus on theoretical questions about games and fun, or on how to study games like the cultural artifacts they are. There are even books that have made strides toward establishing a new vocabulary to talk about games. We still have very few books that are meant to serve as a beginner text for game design—especially books that are applicable to games in all their dazzling diversity.

It's my hope that this book can be as universal as possible, that the framework described within can fit as wide a body of games as my perspective can manage. But I'm not unbiased. This book began life as a guide to designing platform games like *Super Mario Bros*.—or my own *Mighty Jill Off* (2008) and *REDDER* (2010)—before it became something else. If my tendency toward a certain kind of game in this text shows, I apologize.

This book is also specifically about digital games, or videogames. This isn't because board games, card games, folk games, or other nondigital games aren't worthy of interest or design. In fact, videogames share a history with this vast continuum of games, and we have a lot to learn from them. (In fact, many design ideas in digital games are borrowed from nondigital ones.) Because the human players of nondigital games are the ones required to keep, and internalize, the rules, there's a stronger existing discourse about design among board game players and authors than digital games have ever possessed.

What makes videogames so worthy of discussion is their capacity for ambiguity and, hence, storytelling. The computer keeps the rules in a digital game, so a player on level one might not know what level three looks like, that her character is going to lose her legs before the end, or that there's some playing technique she will have to become aware of and master in time but is unaware of this early. The ability to withhold information from the player, and to give her the liberty to discover rules and complexities of those rules on her own, makes the design of digital games so interesting. Plus, their capacity for using visual art, animation, and sound, while not completely unique to digital games, is a facet of design that warrants more discussion.

What *isn't* this book? It's not a guide to any single tool or technology. This book won't help you learn how to edit *Unreal* maps. There are resources for that and for any other game-making tool or editor you're called upon to use. To write this book with any one technology in mind would be to write a more limited book. This book is about design. Design is not technology.

This book can't be the perfect tome that covers all games and all aspects of design. It can't be the ultimate book on game design—the last and only book you'll ever need on your

shelf—because it's one of the first. So this book will have a few holes. If this book has the intended effect, readers like you will step forward and write the words that are missing.

This book is intended above all to start a discussion, to be a starting place for a necessary talk about design that hopefully will continue long after. Once you break a silence, it's impossible to get folks to shut up. Criticize this book and tear it apart—as long as we keep talking about what design is.

Here is a book on digital game design. May many more follow.

CHAPTER 2

VERBS AND OBJECTS

Every game in the world is made up of rules. In this chapter, we talk about designing those rules, which I divide into the categories of "verbs" and "objects," and the relationships between them that create the experience, the dynamics, of the game for its player or players. Those rules are the characters in our story and, like any characters, our stories are most effective when we develop them and their relationships fully.

Rules

Games are made of rules. Surround stones of the opposite color with stones of your own color to capture it. Complete a line of blocks to make it disappear. Reduce the opponent's health to zero to eliminate her. It's the interplay of these rules, the interactions between them, that creates an experience for our players.

As game creators, we want to design the rules that will make for the strongest experience. We want to design rules that have relationships to each other. We want to design rules that have the opportunity to develop as the game goes on and avoid rules that we won't be able to develop. I don't mean "develop" in the sense of "game development"—a rule develops as the player's relationship with it grows deeper, more complex, and more refined, as she finds new ways to work with the rule and understands the nuance of how it affects her experience of play.

John Newcomer and Bill Pfutzenreuter's *Joust*, a 1982 coin-operated arcade game, has rules that support and strengthen each other. *Joust* is a game about ostrich-back gladiators who joust with spears in a desolate arena. Here's one rule of *Joust*: when two gladiators collide, the one who is highest defeats the other. Another rule: pressing the button makes the player's ostrich flap its wings and gain a little height. A third rule: the constant pull of gravity causes all the ostriches to fall downward, toward the bottom of the screen (see Figure 2.1).

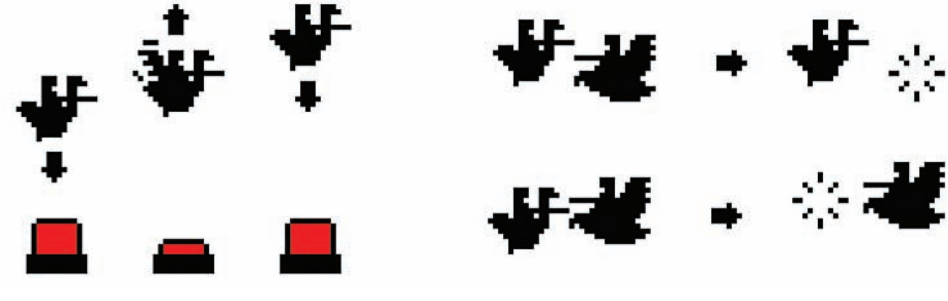


Figure 2.1 Basic rules of *Joust*.

So you can see how these rules work together to create an experience that demands skillful play with attention to one of the main themes of *Joust*: height is important! Flapping to maintain height is critical because gravity keeps lowering your height, and the higher gladiator will always win in a collision. There's also a shrieking pterodactyl that sometimes flies through the arena, devouring player and enemy alike. To slay the pterodactyl, a player has to strike it *directly* on the nose. So the pterodactyl develops the rules about height even more: slaying a pterodactyl demands accuracy and control and represents a significant moment in the game.

Having established the rules of the game, we're then interested in communicating those rules to the player as succinctly as possible and developing those rules through design. Later

chapters in this book are about that. This chapter is about establishing a basic vocabulary with which to understand and discuss rules and how they function in a game—a grammar.

Writing this book, I'm using an established grammar to structure the words and sentences you're reading—hopefully to the end of communicating my ideas as clearly as possible. Writing is a creative work, though, so sometimes I ignore certain rules or conventions when I think it means the results will be more expressive. So let that be a caveat: none of the "rules" I'm going to discuss are law, are immutable. The purpose of this text isn't to constrain your design but to suggest ways to start thinking and talking about it.

Games are made of rules. We're going to think of rules as the characters in our games that are going to be developed over the course of the game. When we talk about story, that's what we're talking about: the development, conflict, climax, and resolution of the rules. Rules are characters. Got it?

Who then are the most important characters—the main characters? You might be tempted to say the protagonist, the hero, the gladiators in *Joust*. The nouns, right? Because in linear stories, we're used to nouns—people—being the ones who develop. But the protagonists of our games, the nouns, aren't rules.

Verbs are a kind of rule; they're the most important rules of a game. By a "verb," I'm referring to any rule that gives the player liberty to act within the rules of the game. Any rule that lets the player change the game state. Any rule that lets the player *do something*. Verbs are the rules that allow the player to interact with the other rules in the game: "jump," "shoot," "fall," or "flap" in the case of *Joust*. Without verbs we have a simulation, not a collaborative story-telling system.

Is it hard to think about verbs as main characters in a story? It's easy to think of the hero as the main character, because verbs characterize the hero. Maybe climbing the sloping foothills at the beginning of the game is easy, but climbing the precarious precipices near the end is much harder. And maybe that suggests both that the hero's journey (not to be confused with the story structure my Guildhall teacher insisted we base every one of our games on) is getting more arduous as she approaches her goal and that the hero is being tested and becoming stronger to meet these challenges.

But we can't design the player or her behavior. We design the rules that shape her experience, her choices, her performance. Rules are how we communicate. Verbs are the rules that allow her to communicate back. The game is a dialogue between game and player, and the rules we design are the vocabulary with which this conversation takes place.

When a game's creator relies exclusively on animated cutscenes or text dumps to tell a story—when she *only* uses noninteractive means of telling a story in an interactive game—it's because she has misunderstood how to think of the story in terms of how the player is allowed to actually act within it.

Creating Choices

Visualize Venus. An endless green sky, purple mountains piled on the horizons like clouds, a yawning cleft like a mouth, leading into the smoking bowels of the planet. The Robot Mines. Not mines staffed by robots—mines where humans dig for technology abandoned by a much older, much more extinct, spacefaring race. Or where they normally do, except that today at 481,900 hours—Venus' days are much longer than Earth's—the unearthed robots suddenly, simultaneously, blinked to life. Immediately and as one, operating under orders hard-coded into their circuits millennia ago, they seized control of the mine and took the human workers hostage—those that didn't manage to escape.

Now it's up to Janet Jumpjet, space-hero-for-hire, to explore the mines, incapacitate the robots, and rescue the human hostages. She's armed only with her wits and the Megablaster 3000 Laser Pistol—shooting it is her primary verb.

What does it mean that Janet's main verb is "shoot"? Probably that there's going to be a lot of gunfire in this game. Venus is a violent place. That's the future for you.

Janet's Megablaster fires a single laser bolt at a time—enough to melt one of those menacing robots. POW! But firing the Megablaster generates a tremendous amount of heat—it takes half a second to cool down between shots. This is a rule that we've designed. We probably spent a lot of time playing with exactly how long the duration between shots is, tweaking the number, playing the game, and trying to decide which made for the most interesting choices. We'll probably tweak it a lot more before we consider the game done.

The duration between shots is part of the "shoot" verb. The rule: pressing the button fires the Megablaster ahead of Janet at a rate of one laser bolt every half-second. Why is this important? Because we can use rules to set up choices for players. A choice can be whether to shoot Janet's Megablaster, or when, or where. If there's a half-second duration between shots—maybe that doesn't sound like a very long time, but it's ages when you've got a crazed robot clanking toward you—what choices does that create? Janet's pinned in a dark corridor, one that looks a lot like Figure 2.2—there are two robots clambering toward her from two different directions. Which one does she shoot first?

Figure 2.2 Rules offer choices: shoot the left robot or the right one?

In fact, *Space Invaders*, the 1975 game of using a moving gun turret to destroy invading aliens, presents the player with similar choices. In that game, the player can have only one bullet on

the screen at a time—she has to wait for a shot to strike an enemy, or leave the back of the screen, before she can fire another one. How does that affect the player's choices?

For one thing, a miss means that the player has to wait a while before she can fire again, but a hit allows her to fire again more quickly. So she's given an incentive to pick her shots more carefully. Shooting invaders who are closer—who are a more immediate threat—affords the player a greater rate of fire than ones who are far away. There's some balance here: the player has more ammunition to use against enemies who are more dangerous. A miss against a close enemy means a longer interval before the player can shoot again—but nearer enemies are also easier to aim at.

Verb Relationships

In fact, there are a few different verbs that are interacting in *Space Invaders*. How does the player interact with the game other than shooting? She moves left and right. There's a relationship between these verbs: when she presses the shoot button, the shot is fired from her current position. Lining up shots with an invader means moving into the range of enemy fire. Dodging enemy fire means moving behind an obstacle, where the player can't return fire. There's an ongoing dialogue between moving left and right and firing shots, and the left/right verbs are more developed as a result. You use them to aim your shots *and* to avoid being hit.

By establishing relationships between verbs, we give ourselves more opportunities to design choices. The relationship between those verbs is also something that we can develop over the course of the game, the same as with any two characters in a story.

At the heart of *Super Mario Bros.* is a strong relationship between Mario's ability to move horizontally—to walk left or right—and his ability to move vertically—to jump. But notice that that relationship becomes stronger over time. At the beginning of the game, the player isn't expected to coordinate these verbs in a very complicated way.

The first jump the game requires is over an enemy that moves of its own volition. Mario is safe if he lands on the enemy—he only has to avoid being touched by the side. This jump doesn't even require horizontal motion: Mario can jump straight up while the monster walks underneath him. The next jump the game requires is onto a stationary, solid obstacle from the ground next to it. This requires a minimum of horizontal motion. It isn't until much later in the game that the player is expected to perform really complicated jumps, with careful management and coordination of both horizontal and vertical movement (see Figure 2.3). By that point, the player understands how to use horizontal and vertical movement as a pair of verbs that work together in harmony.

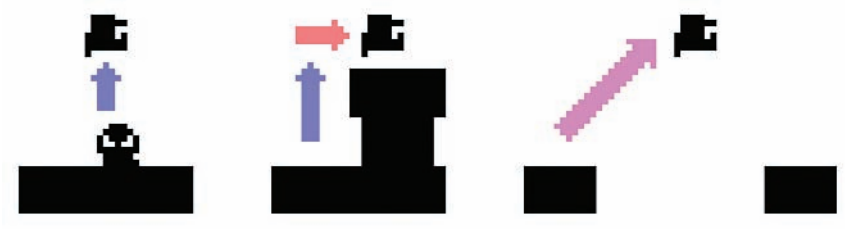


Figure 2.3 Development of the relationship between horizontal and vertical movement in *Super Mario Bros.*

What we as creators want to avoid are orphaned verbs. An orphaned verb has no relationship to the other verbs, so the other verbs don't reinforce it, it doesn't grow, and the player has forgotten about it by the time she reaches the one situation that demands it. Imagine a game where the player finishes each level by using the "open" verb on a door that exits into the next level. If there's only a single door in each level, this verb is an orphan: it never gets used for anything else and doesn't have the opportunity to develop in relation to other verbs and varying situations.

How do we avoid having vestigial verbs? Design the game so that the verbs you've given the player are sufficient to perform everything you ask of her. Increase their utility by giving them more interactions. If you play a bunch of videogames, you might be surprised how many ways there are to open doors.

We're back on Venus. Janet Jumpjet is squinting at a door, an ancient metal bulkhead, in the darkness of a subterranean mine, Megablaster smoking. How is our hero going to get to the other side of this door? Is she going to knock politely and wait for someone to let her in? Is she going to put down her Megablaster and turn the door handle? Or is she going to point her Megablaster at the door and pull the trigger?

Give the important verbs in the game as much to do as possible, so you won't be forced to fill the void with a bunch of secondary verbs that never get developed.

Making Verbs Robust

We want our verbs to be as developed as possible. We want them to be well-rounded characters. That doesn't just mean that they interact with as many other rules of the game as possible, but also that every interaction the player expects to have a reaction does. Verbs are the rules the player uses to learn the rest of the game's rules. If she uses a verb some way and is given no feedback, she doesn't learn anything about the verb or about the rules of the game. We want robust verbs that communicate with the player, even if just to say, "No, you can't do that." That seemingly negative statement can be just as important as showing the player what she can do.

Here's an example from my own work. In 2009, I made a game called *Tombed* about an archaeologist named Danger Jane. Investigating a deep crypt, she's pursued downward, through layers of fragile earth, by a descending spike wall—the quintessential tomb trap. She's armed with a shovel, which she can use to dig through the soft clay blocks that she finds underfoot. "Digging" is a critical verb, established as such as early as the title screen, which shows Jane on diggable ground with a shovel in her hand and the instruction "Press Shift to Start." When the player does so, Jane digs through the floor—every touching, like-colored block (there are three colors) is considered the same piece of clay for purposes of digging—and plummets off the title screen and into the game.

So the player now knows what Jane's important verb is ("dig"), what key to press to trigger that verb (Shift), and the effect of digging on soft clay blocks, many of which she will encounter in the game. Jane's other verbs are "walking" left or right when on stable ground. When she has no ground beneath her feet, she falls until she lands on some.

In addition to the diggable clay blocks, though, there are solid, metal blocks that Jane can't dig through. These are used to constrain Jane's movement: to create choices for the player. Maybe she has to run around a metal obstacle instead of digging through it, allowing the spike ceiling to close in on her. Maybe she has to wait for the spike ceiling to drop far enough to destroy the metal blocks for her. That's another rule: the spikes destroy any blocks they touch—even metal ones. So here the metal blocks work as a pacing mechanism, a solution to Jane getting so far ahead of the spikes that she's off-screen, where the player can't see and make decisions for her.

But how does the player know this? How does she learn that Jane's shovel—which she has been taught can destroy obstacles—can't break through metal? I'll tell you how. She strikes that first metal obstacle with her shovel, and the game provides her with feedback to show what happens.

Here's what happens when Jane's shovel hits an unbreakable metal block: it bounces off with a metal *"ting"* sound (see Figure 2.4). Even when the player is unable to use a verb to break through an obstacle, there's still an observable effect that gives the player information about the relationship between the verb and the obstacle. The rule "Jane can't dig through this" is taught or reinforced when the player uses her verb.

Figure 2.4 Jane attempting to dig through clay and through metal.

In fact, the way the game introduces the rule is this: the player begins by having to dig through three different layers of clay blocks—one of each color in the game. Figure 2.5 shows the first scene of the game. Each color, when struck, crumbles every touching block of the same color, but not ones of different colors. These are the most basic rules of the game and the things the game teaches first.

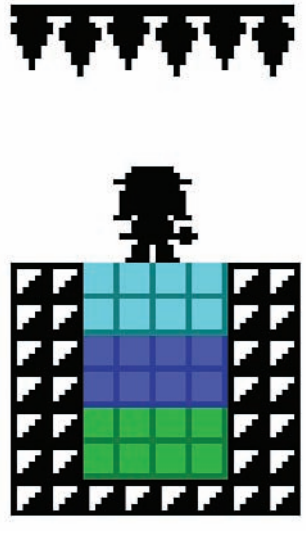


Figure 2.5 Opening scene of *Tombed*.

When Jane has dug through all three colors, she's at the bottom of a well of metal with raised sides. She can strike the metal with her shovel, but it'll just bounce off with a *ting*. In a few moments the descending spike wall will reach the top of the well's raised sides, shattering the whole piece of metal and freeing Jane, who falls to the ground below. So now the player has most likely observed the "Jane can't dig through metal" rule and the "spike ceiling *can* dig through metal" rule.

Every interaction that the player could reasonably expect to have an effect should have one, even if it's negative—that's what I mean by a robust verb. If the player sunk her shovel into the metal and nothing happened, the objects didn't seem to touch, or they just passed right through one another, the player might still figure out that she can't dig through these blocks, but we haven't sold the rule as strongly or effectively. Maybe it takes the player a little longer to figure out, and while she's doing so, the spike ceiling comes down and crushes her. Now she has to go back and repeat the whole thing. She has wasted time and maybe not even learned anything.

That's bad design. As creators, we want to teach and reinforce rules wherever and whenever we have the opportunity.

Explaining with Context

Back on Venus, Janet is ready for adventure, Megablaster in hand. Although she's a fictional character in a story, the world she lives in is a simulation. In the computer that's running the simulation, Jane exists as a bunch of numbers: her horizontal and vertical position on the screen, which direction she's facing, and the speed at which she's moving. The laser bolts she fires from her gun are just signals: they have a speed and a direction. When the computer detects that one of these signals is overlapping an appropriate recipient—a robot, another x and y position with a direction and a speed—the simulation resolves the collision by removing both objects. In the abstraction of the rules, the math of the game, this is all we see.

We explain these rules to the player by giving them a context that she's familiar with, one that helps her understand them. The context of a game is composed of many pieces: the images that represent Janet and her Megablaster, the words that appear to describe these things, the way the images animate, the sound effects that accompany play, and even the timing that brings them all together. We're used to thinking of these elements as parts of the narrative of a game—the story of Janet Jumpjet rescuing hostages in the mines of Venus—and they do arise from that story and help tell it. But in a game, contextual elements do something more: they illustrate and help make sense of what's happening in the *other* story, where our rules are the main characters.

Janet has a gun, so she can fire laser bolts. This is a robot; laser bolts explode it into pieces. These are metaphors that serve to help the player grasp the rules, and we communicate them to her with images, sound, animation, and feedback. If we tell the player the Megablaster needs to cool down after discharging a mega-hot laser, we're selling her a justification for the half-second reload time. If the player can see her Megablaster heat up white for a half-second before fading to normal, we've made a visual metaphor to reinforce the rule.

The more cohesive a game's context is—the more things behave according to the metaphors we've assigned them—the more easily the player can build expectations and anticipate and understand the rules of the game.

In the sub-Venus darkness, Janet is stepping through the blasted remains of robots, keeping her eyes peeled for human hostages. We, the game's creators, have decided that to avoid introducing a new, underdeveloped verb, we want the player to rescue hostages using her "shoot" verb—as an extension of a verb she's already familiar with.

When Janet has a hostage in her sights, staring down the barrel of her Megablaster at a harrowed human captive, will she be able to pull the trigger? Is the player likely to shoot someone she's been tasked with rescuing, now that she's observed how shooting robots wrecks them? Most likely, she'll hesitate, confused.

Maybe once she's done it once, and she understands that shooting a hostage teleports the hostage to safety instead of splattering her on the cave walls, she'll be able to shoot hostages. But that initial doubt is a serious hurdle to the player's process of learning the game. This is the

first time she's encountered this rule, and as creators, we want her to grasp the rule with as little prodding as possible. If we need a commanding Sargent Tutorial to radio Janet and instruct her that shooting hostages tags them for teleportation, we've failed to communicate the rule. Resist Sargent Tutorial.

Shooting a hostage with a deadly weapon to rescue her introduces a dissonance between the player's expectations and the rules of the game that will haunt the rest of her experience and make her doubt her understanding of the game.

What if, instead of shooting the hostage to rescue her, the hostage is in a cage—a robot cage, made of and obviously resembling the same metal as the robots Janet's been blasting. When the player shoots the hostage now, she's blowing open the cage and freeing the hostage. Freed from the robot cage's teleport dampening field, the hostage can now beam to safety. Thank you, Janet! My hero!

This is a lot easier to buy than having to shoot a hostage with a deadly weapon. If the player sees something she can identity visually as a hostage trapped in a cage, she can put two and two together—the other "two" being her knowledge that her Megablaster destroys robot-looking things—and figure out that maybe she should shoot the cage. When she does, the game reacts by communicating that she's freed the hostage. Good job!

The easier we make the rules of our game to understand, the more easily and effectively the player can internalize those rules and begin anticipating them.

Objects

I'm not referring to "objects" in the simple sense—nouns, detritus, inanimate objects. I mean objects for our verbs: the objects that complete their sentences. "Jane digs through a block of green clay." "Jane tries to dig through a metal block but is repelled." We will draw from our palette of objects to set up choices for our player and to tell a story.

The right selection of objects goes a long way. The downward path that Jane navigates in *Tombed* is made up of those four objects: clay blocks in three different colors that connect to blocks of the same color, and metal blocks. As with verbs, we get a lot of utility over choosing the right ones and should try to avoid introducing ones that will be underdeveloped because their uses overlap too much with that of other objects.

One color of clay blocks would be too little. I could use them as pacing, but not to set up interesting choices about where to dig. Two would let me set up choices, but not ones that link into each other very well. Because there are only two colors, two diverging paths can't touch. Four colors—at least in the limited space that I've given the game, for the sake of it being fast—don't give me very many options that I can't achieve with the three colors I already have. See Figure 2.6 for an example.

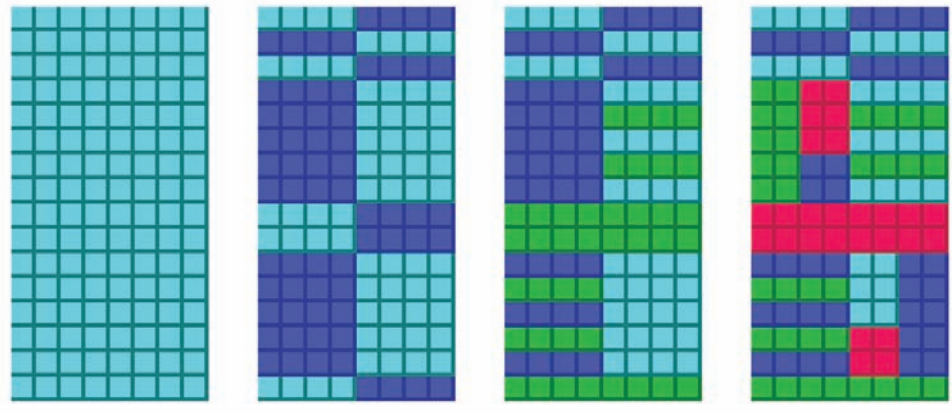


Figure 2.6 Possibilities in *Tombed* with one, two, three, or four colors. Note how the difference between two and three colors adds more possibilities than the difference between three and four colors..

Did you ever do that exercise where you try to color in a map of a bunch of different regions without having any regions of the same color touch, using as few colors as possible? Four, it turns out, is the maximum amount that we would need. (This is known as "four-color theorem.") But if you look at any completed map—for example, a map of the United States, using four colors—you'll notice the fourth color appears infrequently, only in very hairy combinatorial situations. A fourth color in this game would be underused. It would mostly make the visual pattern of the game more complex and potentially confusing, forcing the player to spread her understanding of the game rules that much thinner.

Tombed was made for a two-day game design competition—Ludum Dare 14—so I released its source along with the finished game. (It was made in Game Maker 8.) Using that source, my friend Leon Arnott created a spin-off/sequel called *Tombed II: Twombed Off*. In his sequel, he adds only a single new diggable object to the four that are already there: a red "bomb" that, when struck with Jane's shovel, destroys every piece touching it, regardless of its color, even if it's an undiggable metal block.

This is a good addition because it adds a lot to the game. It gives the player the ability to destroy metal blocks, but only when given the opportunity. And it has a relationship with the other blocks: because it destroys everything connected to it, sometimes the player wants to dig through other blocks to *sever* connections with a bomb block. This is strengthened by another new rule Leon added: now the bottom of the screen is also lined with spikes, so the player wants to pace her descent carefully. Bomb blocks potentially wipe out a lot of terrain at once, which sometimes means Jane falls into the spikes before ground appears to catch her (see Figure 2.7).

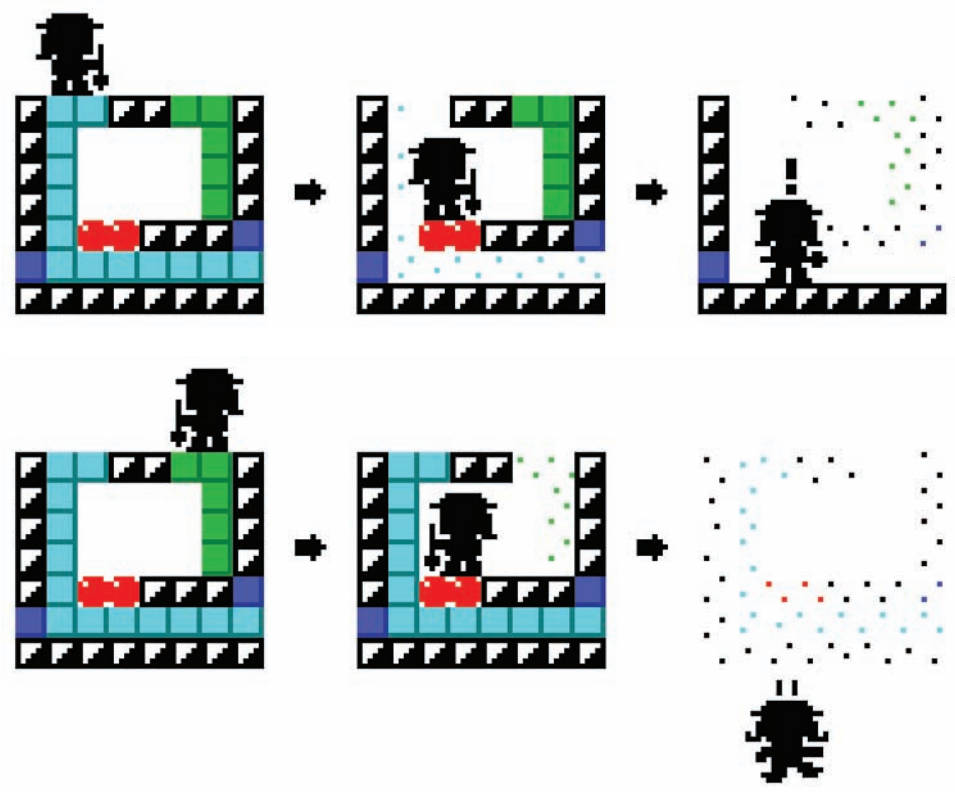


Figure 2.7 Bomb block predicament in *Tombed II*.

The bomb block adds the opportunity for a lot of new choices—choices that aren't possible with the existing blocks because they only interact with similar-colored blocks, not different-colored ones. But it still fits with the existing objects because it's about digging and caring about which blocks are adjacent to the block Jane's digging.

It's easy to add game objects that have tenuous relationships to existing objects because they're quick fixes, sloppy patches. For instance, if I wanted Jane to run over *here* before having to quickly run back over *there*, the most dumb-simple way to do that would be to put a switch *here* that, when Jane strikes it, opens a door *there*. (See the left side of Figure 2.8, with the switch and door in red.) But that doesn't have anything to do with digging—with any of the rules we've been teaching the player. And the fact is that we can accomplish the same thing using the objects we already have and *design*, instead of having to introduce a new, under-developed rule. The better way to create a "switch and door," using the same objects and verbs that *Tombed* is already based on, is shown on the right side of Figure 2.8. (Keep in mind that Jane is two blocks wide, so she can step over gaps that are only one block wide.)

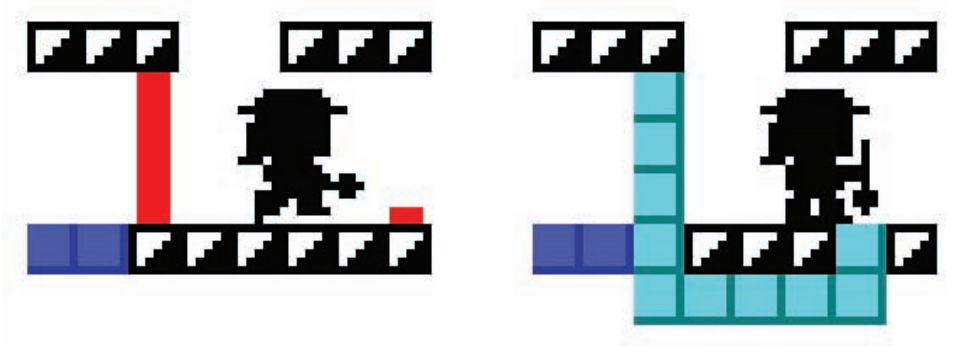


Figure 2.8 Two ways to add a door to *Tombed II*, built by introducing extra objects and an equivalent using existing objects.

The Physical Layer

Digital games, even if we watch their rules play out on a TV screen, have a physical layer. Jane may be digging through blocks of clay, but what the player is doing is tapping the Shift key. The verbs the player is using to interact with the other rules of the game—to make Jane do stuff—are connected to physical actions the player is taking, though they're small ones.

Our goal as creators is to communicate the rules of the game as clearly and succinctly as possible, yes? This is why it's important to make the connection between the verb the player is acting upon and the physical action the player is taking to do it as close as possible. I don't mean that digging in *Tombed* should be physically grueling for the player, because that's a game of quick thinking, not endurance.

Grueling controls might suit a different game. Take the original arcade version of *Track & Field* (1983). Running in that game involves alternately mashing two different buttons. This helps characterize the action of running because you can imagine two feet stepping in sequence: right leg, left leg, right, left. It's also physically arduous, which increases the player's connection to the athlete on the screen and the actions he's taking. In a game where running is less important, it's safe for the mechanics of running to be abstracted. You can walk from one side of the room to the other without having to focus on the motion of your legs. But for a game about competitive distance running, this connection is important.

Of course, if you've played many videogames, you know that we don't need to make a verb's physical performance resemble the verb. There's not much about pressing a button that resembles jumping, and that's part of the interesting quality of videogames—we can explore the distance between the physical act of using the interface and what it represents inside the game. Sometimes we can close the distance. There are platforms that can sense a broader

range of human motions and allow us to force the player to perform a scooping motion to dig through a floor. But even if we explore a wider distance between physical interaction and game representation—if we've restricted the input to a single button—there's a range of physical action we can connect to our game verb.

The player makes Jane dig in *Tombed* by tapping a key. How does Jane dig? A single, fast tap, a poke of the earth with her shovel. A quick action that mirrors the player's key-press. What if the player *tapped* the button and Jane slowly dug her shovel into the earth, pressing it down with her foot, and then threw the dirt over her shoulder? There'd be a dissonance there that would make the player feel less of a connection to the verb she was performing.

That's not to say you would never make a game where digging was slow and time-consuming: digging *is* a slow task. But if you designed a "dig" verb that functioned like that, you would also want to design a physical layer to the verb that would better describe it. Maybe the player has to hold the button down while Jane is pressing the shovel down into the earth. Maybe she throws the dirt over her shoulder when the player releases the button (see Figure 2.9).

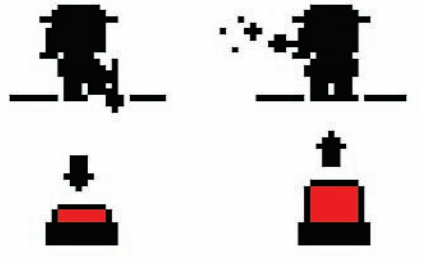


Figure 2.9 Verbs mirroring physical button activity.

In fact, the latter is especially compelling because it's dense: we've mapped two parts of a verb to a single button. We've mapped the different steps of a sequence—jam the shovel in, and then lift it over your shoulder—to two different physical actions that can be performed with the same button: holding it down and letting it go. By compressing as many verbs as possible into as small a physical layer as possible—in this case, as few buttons as possible—we're potentially avoiding a lot of player confusion and fumbliness.

When I play a game that uses three or four different, adjacent keyboard keys, my fingers have a hard time remembering which one is which, resulting in a lot of mispresses—which usually means spending crucial resources by accident or misperforming an important choice. It means a player who feels cheated. The only exception to this multikey problem is with a keyboard's arrow keys, which, like the multidirectional joypads of game controllers, have been designed and arranged with a physical metaphor in mind already! It's no wonder so many games use the arrow keys as controls—it's a handy shortcut.

Some "fighting games," like the *Street Fighter* series, map many overlapping verbs onto many buttons—traditionally, six buttons and an eight-directional joystick. Here, learning to physically perform verbs, and being able to perform them successfully when the time is right, is an important part of the game. The discipline needed to learn and perform verbs is here intended as a parallel to the discipline that learning and performing a martial art requires.

Reinforce a verb's physical layer wherever possible. If a game has a title screen and menu, try to connect the important verbs and input to it. In *Tombed*, as I've mentioned, the player starts by making Jane dig through the floor of the title screen using the same button she'll use to dig in the rest of the game. At the very least, allow the player to navigate the menu with the same input that she uses for the rest of the game. There are a number of Flash games where the player's verbs are tied to keyboard keys—say, the arrow keys and the spacebar—but the menus ask the player to point and click with the mouse.

Imagine if, after every level of the game, the player has to move her hands off the important keys, take her mouse, point at and click the Next Level button, and then return her fingers to the arrow keys and spacebar. That's tedious, and what's more, it weakens the player's focus on the game's verbs and the keys that correspond to them. Games that successfully use both the keyboard and the mouse usually assume that the player has only one hand on a limited set of keyboard keys and the other hand on the mouse, to avoid this kind of fumbling and switching of the hand positions.

I recently discovered an iPad game called *Mini Mix Mayhem* (2012), the premise of which is that the player or players have to manage up to four games, with different rules and goals, that are sharing the screen at the same time. There are around 20 games in all that can show up in the four you have to juggle at once, each controlled in a different way that suits the game's visual metaphor. Unwinding a nut from a screw involves physically flicking the nut to the left; you think of actually using your finger to spin a nut loose. Solving a which-cup-is-it-under puzzle requires just tapping the right cup; you think of pointing at the cup for the barker, who then reveals what's under it. Collecting drops of rain in a cup involves holding your finger on the cup and dragging it left and right; you think of holding a cup and moving it around.

Many of us will never design for a touchscreen, but regardless of what physical actions we're giving the player to perform the verbs in our games, we can design actions that call to mind the properties of our verbs—that communicate and reinforce the rules of our game.

Degrees of Control

Different kinds of input allow for different paces of input. Mouse input, for example, is highly nuanced. It allows for slow, gentle movements, or fast, sweeping movements, along two axes simultaneously. (A player can move a mouse *up* and to the *left*, for example, at rates that are independent of each other.) Often, games use mouse motion for verbs like looking around: your eyes pan slowly along the sprawl of the mysterious city in the distance. What secrets does

it hold? Suddenly in the corner of your vision, you spot a bird made of skulls and swing your eyes quickly after it.

The mouse offers many more degrees of movement than, say, a keyboard key that is simply in a *pressed* or *not pressed* position. But binary keys have their moments, too. Take a game like *Tetris* (1986). The goal of *Tetris* is to position shapes, each made of four square blocks, such that they fill the horizontal space of the playing area with no holes. As such, movement in this game—the player moves one piece at a time—is in the length of a single block. Movement happens on a grid. If it didn't, it would be hard to fit new pieces into the holes between existing pieces. Binary keys make sense here: one tap of a key can correspond to one block of movement.

You could design a way to control *Tetris* with a mouse—maybe the shape would always snap to the closest grid position to the player's horizontal mouse position. But what about the mouse's vertical axis? In this example, it has nothing to do (see Figure 2.10). *Tetris* wasn't designed with a mouse in mind.

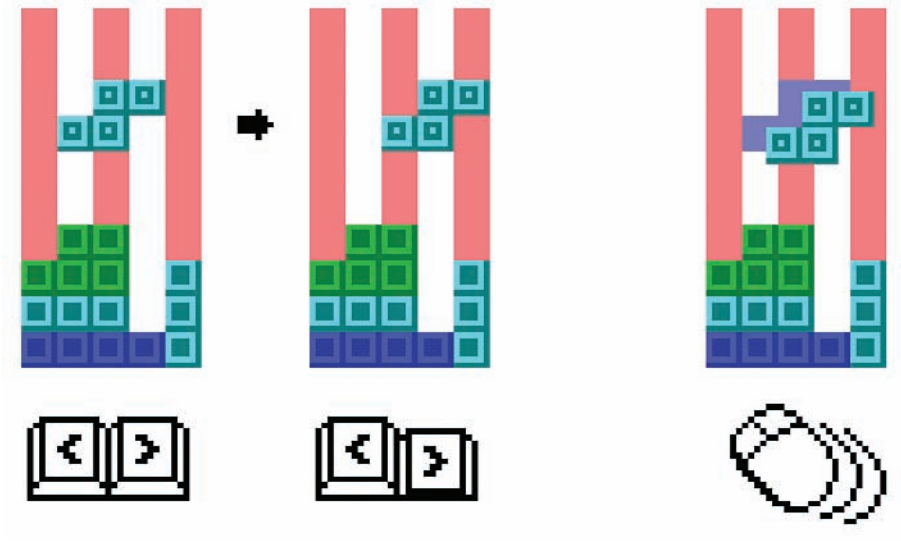


Figure 2.10 Grid-based movement in *Tetris*, with keyboard keys and with a mouse.

A game that was designed for a mouse instead of grafted to one might look very different from a grid-based game like *Tetris*. As an example, take Christophe Andreani's 1987 Atari ST game *Bubble Ghost*. Andreani made the protagonist of his game a ghost—insubstantial, it can travel anywhere a mouse cursor can without worry of colliding with walls or obstacles. However, the ghost is trying to maneuver a fragile bubble (by blowing on it) through a mansion full of things that, while harmless to the ghost, will pop the bubble on contact. So you can see that this game was designed with the mouse foremost in mind.

(And interestingly, *Bubble Ghost* was itself grafted back onto a four-way D-pad in a 1990 version for the Game Boy. Unsurprisingly, it loses something: allowing the player to move at a fixed speed means that lining up and aiming is both easier and less nuanced.)

As a creator, you should always consider the properties of any method of input you're designing for, even if, for whatever reason, the choice isn't yours. I've played a million attempts at *Super Mario Bros.*–style games on the iPad touchscreen that try to reproduce the same controls as the game they're imitating: left and right movement "buttons" on the left, and two "action" buttons on the right. There's no textural distinction between the two action "buttons," as there would be if they were two keys, so it's hard to hit "the left action button" or "the right action button" with any degree of consistency. And any motion that moves the player's fingers too far out of the corners obscures the screen—the information the player is relying on.

The most effective games of this kind pare down the player's verb set until only two binary inputs are needed: either the player's left thumb is on the screen or it isn't, and either the player's right thumb is on the screen or it isn't. In *1-Bit Ninja* (2012), the protagonist moves forward when the left side of the screen is touched and jumps when the right side of the screen is touched (see Figure 2.11). The player doesn't use the touchscreen to choose the protagonist's direction of horizontal motion: rather, there are in-game objects that flip the protagonist around if she touches them. Design for a game's means of input, not against it.

Figure 2.11 Two different models of touchscreen input, one designed a little more consciously of being on a touchscreen.

Be aware of the properties of a form of input before designing a physical layer for your verb. A computer's arrow keys and the plus-shaped "directional pad" of a Nintendo controller might seem the same, but there's a fundamental difference. The keys are four independent binary states, whereas the "d-pad" is a single piece of plastic—it can be depressed left or right and up or down (two axes of movement), but the player can't depress left *and* right at the same time. There's an enforced exclusion in the d-pad that makes sense in games about spatial navigation in two dimensions—after all, what does left and right simultaneously *mean* in terms of that movement? They cancel each other out.

But other contexts have a use for them. *Dance Dance Revolution* (*DDR*, 1998), a *Simon Says* game that players play with their feet on giant d-pads, occasionally gives players a Left and Right command or an Up and Down command simultaneously. These require the player to jump into

a position where their left and right feet are opposite each other. It's important here that *DDR*'s giant d-pad is made up of binary buttons rather than a single piece of molded plastic (which is designed for a single thumb, not two feet).

This may seem obvious—it seems impractical that someone would build a physical input device that size out of a single piece of plastic—but the hidden pitfall here is that, used to the properties of a single type of input, you fail to recognize and design for the less obvious properties of another mode of input—like the developers who tried to re-create Nintendo pads on touchscreens.

Character Development

In the Robot Mines of Venus, Janet Jumpjet stalks the catacombs, her Megablaster 3000 held in front of her, trained at the unseen hordes of deranged robots no doubt lurking somewhere in the darkness ahead of her. Then she sees it—the glint of metal up ahead, the unmistakable gleam of a robot! She stops dead in her tracks, finger tensing on the trigger. But the metal form doesn't move. She takes a tentative step forward and sees that it's not a robot—it's another robot cage! Has she found another prisoner? Peering through the viewscreen in the cage, Janet sees the face not of a human prisoner, but a robot in storage! This is a cage Janet wisely decided to leave unopened.

Suddenly a robot, taking advantage of Janet's momentary distraction, lunges out of the darkness. Taken by surprise, she fires wildly—blasting open the cage and freeing its robot inhabitant, who immediately blinks to life and begins clambering toward Janet, alongside its compatriot. Cursing, Janet blasts the robots to smithereens. She wonders if she'll be seeing more of these robot containers and reminds herself that she'll have to be way more careful with her shots.

Not long after, Janet emerges into a room patrolled by two robots. Janet raises her Megablaster, but the hairs on the back of her neck are trying to tell her something. Taking a breath, she looks around. The walls of the room are covered in caged robots, waiting to be released (see Figure 2.12).

She'd better aim carefully, all right.

Janet is the main character of the narrative that unfolds in this game, but don't forget: the main character of the game itself is still the verb "shoot"—a verb the player has to understand how to use wisely. If verbs are the main characters of our game stories, we develop them as we would characters in any other form: we challenge them, we give them new responsibilities or burdens, we let them show new sides of themselves, we let them grow—or force them to.

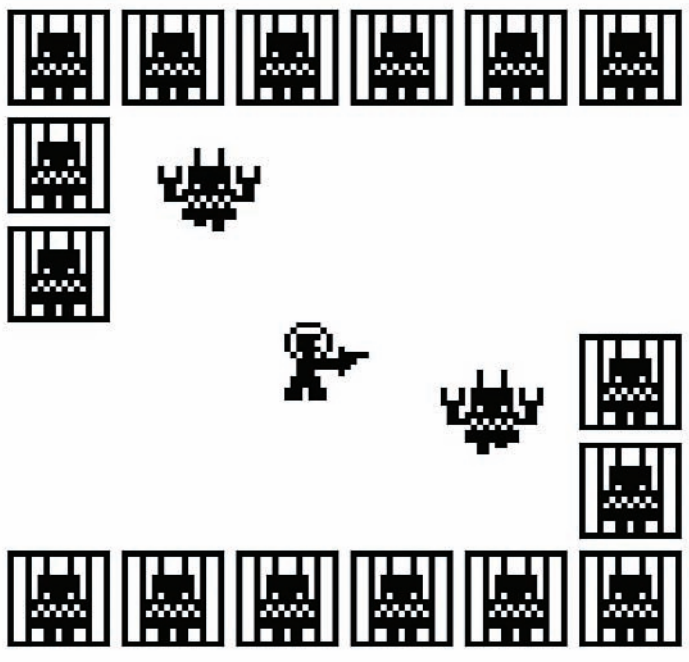


Figure 2.12 Janet had better think before she shoots in this room.

Our rules and our objects are the vocabulary we use to develop our verbs. Again, we want to avoid introducing so many rules or objects that some of them won't be allowed to develop fully—that they won't see enough use. We also want to be thrifty because all these new rules are things that the player has to learn and keep in her head. Every new rule takes a little of the player's attention from every existing rule. The more rules, the more conditions, the thinner the player's understanding of the game is spread. The thinner her understanding, the more she'll forget things, become confused, and fail to predict what some new object will do or how it will combine with existing objects.

We can get a lot of development of our verbs out of the objects we have if we use them well. On Venus, we introduced an object that functions like the earlier "robot cages" but releases a new robot when shot. We can get a lot of interesting situations out of this object alone. There's the room where the walls are covered in caged robots: the player fights mobile robots, but a missed shot means more robots can potentially escape. Or imagine a tunnel, the exit of which is blocked by a caged robot. To get through, the player has to free, and then destroy, the robot. Maybe there are robots chasing from the other side of the tunnel!

The "caged robot" object is in keeping with our existing understanding of the rules: robots chase Janet, and cages release what's inside them when shot. And the preceding scenarios

develop Janet's "aim" and "shoot" verbs. The cage room forces the player to shoot more accurately; the tunnel gives her a better understanding of the Megablaster's rate of fire: a half-second, remember? She chooses when to let out the robot at one end of the tunnel; she has to time it so she won't be overwhelmed by enemies from both sides.

These sorts of arrangements are what we call "level design," and we talk more about that in Chapter 3, "Scenes." It's important to develop the skill of recognizing objects that will support a verb rather than merely add clutter to a game. It's also good to choose ones that have a lot of utility—ones that can be developed themselves, rather than jumping on the stage with a shout and then standing there awkwardly, having nothing more to say.

Elegance

We've been discussing an approach to elegant design through a careful cultivation of verbs, objects, and rules: the idea being that rules that are too specific or narrow to add much to the game will nonetheless occupy a portion of the player's headspace and thus dilute her focus and understanding of the game. We've been discussing the value of being concise.

There are some traditions of games, though, that emphasize having as vast a population of verbs as possible. These kinds of games have their root in text adventure games—games where the player engages with the rules of the game by typing in commands at a text prompt. These games are in turn inspired by role-playing games between human players, where a player in a "Game Master" role narrates an adventure for the other players and responds to the choices they make.

Most digital adventure games don't have a human overseer—the computer responds to the player's input based on a list of permissible verbs defined earlier by the human creator. In practice, this means that a vocabulary of frequently available verbs develop—"north," "south," "get," "examine." The challenge to the player comes in recognizing those moments in which an unusual verb is called for, and figuring out which of those verbs is appropriate.

Many graphical adventure games, inspired by those text adventures, did away with the text prompt but still challenge the player by hiding the solution to any given obstacle in a forest of possible verbs. 1990's *Secret of Monkey Island*, for example, gives the player 12 verbs at outset—"open," "close," "push"—and then gives her steadily more as the game goes on. (See Figure 2.13 for an example.) The protagonist accumulates objects—a mug, a banana, an inkwell—that unlock even more verbs. You can "drink" from the mug, "eat" the banana, "use" the inkwell, adding to the list of options available to the player. This doesn't really make the puzzles more challenging as brain-teasers or riddles—it just makes the solution more tedious to find as the player sifts through all the possible options and combinations, trying to intuit which of them might be correct.

Figure 2.13 Too many verbs makes for a lot of guesswork.

The appeal of this system is its potential for invention. In *Zork*, a 1980 text adventure, for example, there's a room that's described as being "very noisy," which contains a platinum bar. If the player attempts to take the platinum bar like she would any other valuable object in the game—by typing "GET BAR," maybe—the game just repeats her command: "GET BAR." Anything the player types is echoed by the acoustics of the room. The solution is to type "ECHO." That will break the spell, allowing the player to collect the bar.

In a graphical adventure game, some near-useless item the player collects early in the game and allows to pass from her consciousness can have an important role at a critical moment late in the game—a kind of Chekhov's gun. These moments provide a kind of narrative closure and feel clever—when the player gets the joke.

The problem in these cases is that when the player *doesn't* immediately get the joke, the game breaks down, because experimentation is slow. How does the player gain information about the rules of any game? She uses her verbs. In a game that gives the player a hundred verbs, experimentation is a slow and tedious process. The player spends an hour trying each of her verbs on every object in her reach, learning nothing until she finds the correct combination.

Are moments of invention possible in games with a concise vocabulary of verbs? Consider *Portal* (2007) to be in this same tradition: puzzle-solving games with overt, expository storytelling. The game establishes a concise set of verbs very early—the player can join any two points in space by placing a blue portal and an orange portal. These doorways are considered connected no matter where there are. There's also a pretty concise cast of objects: walls that accept portals, walls (a darker color) that don't accept portals, blocks that can travel through portals like the protagonist, switches that are opened by the weight of a player or block, gun turrets that fire at the protagonist when they see her but have the same properties as a block.

Realizing one can position a portal under a turret to remove it from a problematic spot is a moment of inventiveness. So is realizing one can make a portal above a turret—it's on a piece of dark, portal-resistant floor in this example—through which to drop a block onto it and knock it over. In this case, the player is inventing from her existing knowledge of her verbs and the

objects they act on. Here she can experiment, observe, and make connections. But the solutions nevertheless have the potential to be eurekas—real moments of breakthrough development for a verb, like an unexpected "plot twist" where the player has found out something exciting and new about the potential of our main character.

You don't have to avoid moments where solutions come from unexpected directions—but you shouldn't build a game out of them. The player's verbs are the means by which she comes to understand the rules of the game—give her too many verbs that don't interact with each other, and you give her a weak understanding of the rules of the game.

Real Talk

Concurrently with writing this book, I am working on a game with Loren Schmidt. In this game, the player guides a succession of expendable slave miners harvesting precious crystals from the caverns of strange planets for unseen alien overlords. We're only now beginning to design levels—the caverns that the player will explore. Most of the time we've spent on the game thus far has gone toward designing our verbs and choosing objects that reinforce those verbs and each other.

The player digs up the crystals by using radium bombs. She starts every cavern with a limited stock of them, and more can usually be gathered from within the cavern. Pressing the spacebar causes the miner to drop a bomb. After pulsing for a few moments, the bomb goes off, exploding in a small circle that eats through adjacent ground and kills any creature within the radius of the explosion—including the miner. So there's already a tension between planting bombs and giving oneself enough space to avoid them, one that becomes exacerbated by the presence of hostile creatures and other threats.

The player can also dig with her hands. When the player steers the miner into a wall, she starts to slowly carve a path for herself through it. This is slow and inefficient but serves a couple purposes. First, because the walls the player's blowing up are made of tiny grains, and the player plants her bombs wherever she chooses, it's possible to leave thin shells and obstacles between otherwise-open spaces. These really aren't worth eating up an extra stock of the player's bombs, so the player can "clear out" that debris simply by digging through it by hand.

The other purpose is tactical bomb conservation. In some circumstances, with few bombs remaining, she might be willing to trade time for the ability to hang on to one of her precious bombs. The player has a limited amount of oxygen, represented by a meter at the top of the screen, that slowly drains, so trading time for bombs can be a critical choice. There are also situations in which the player runs out of bombs and is forced to rely on digging. If digging by hand wasn't an option, the player might be left with a half-completed level she has no means to finish, which is a situation we want to avoid. Figure 2.14 illustrates bombing versus digging.

Figure 2.14 Choosing between bombing and digging.

When to plant a bomb and when not to is an important choice. "Bomb" is that player's primary verb and means of interacting with the game world. So we wanted a cast of creatures that interact with that verb—that are affected by, and complicate, the player's decision to plant a bomb. We also wanted them to interact with each other, so that they form relationships.

Here are some of the inhabitants of our alien caverns. First, there is a simple, bouncing creature that moves at 45-degree angles (diagonally), slightly slower than the player, dumbly bouncing off any wall that it touches. Because its movements are predictable, it's easy to avoid, except when it's in great number. We can create situations in which the player wants to be careful where she bombs, for fear of letting a bouncing creature out. There's also a rocklike creature that doesn't move, remaining dormant, until the ground underneath it is removed. Then it falls, killing anything it touches. So the player can set off this rocklike creature, sometimes to her detriment, sometimes to her benefit. Set off at the right time, the player can use this creature as a means of clearing out a tunnel full of bouncing enemies, for example.

Another inhabitant is a gun turret that, when the player is nearby, awakens and shoots bullets toward the player. These bullets are lethal to the player, but also to other creatures, like the bouncing things. They also chip away at the ground where they hit, so a resourceful player can sometimes use them to dig paths in lieu of bombs. Or a player can set off one of the falling rock monsters by removing the dirt that's keeping it in place. See Figure 2.15 for an example of all these objects interacting.

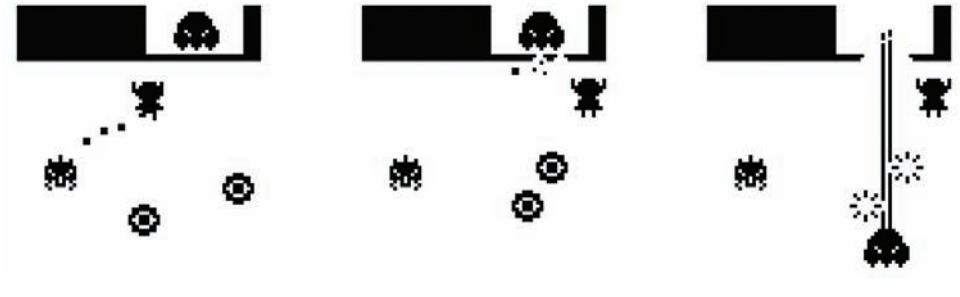


Figure 2.15 Game objects interacting.

We wanted to ensure that even nonintelligent objects in the game react to the player's bombs. Wild radium—radium that the player can collect to increase her stock of bombs—will itself ignite and explode if touched by one of the player's explosions. Sometimes the player will want to set off a chain reaction; other times the player will want to be careful to dig out wild radium without setting it off. The crystals the player is trying to collect will crack and shatter after being hit by explosions. The first explosion causes the crystal to crack. This usually happens while using a bomb to unearth the crystal and is a warning not to do it again. The second explosion shatters the crystal. Your explosions have consequences.

All these things reinforce the player's primary verb: her ability to cause explosions and to dig through the ground with those explosions. And we've tried to design objects that allow us to develop that verb in interesting ways: like giving a player the opportunity to rely on an enemy to dig a path for her rather than spending a bomb, for example. We've tried to avoid objects that would be cul-de-sacs—that would have a single use and then fail to develop and provide meaningful choices after that.

Review

- Games are made of rules. Verbs—"jump," "climb," "shoot," "place a piece," "rotate a block"—are the rules that give the player liberty to interact with the other rules of the game.
- We can use verbs to set up choices for the player. Often, to make for interesting choices, we want to give the player several verbs—"move horizontally" *and* "shoot vertically," for example. We want these verbs to have a relationship to each other.
- We want to be careful about choosing verbs and relationships that we can develop. If we add too many verbs, not only will the player's understanding of the game be less focused, but many of those verbs are likely to be left underdeveloped.
- We "develop" a verb the same way we develop a character in a story. We give them more responsibilities, we ask them to perform together more closely, we give them difficult choices to make.
- Verbs have objects—things they act upon—to reinforce them, develop them, and give them choices. As with the verbs themselves, we want to avoid introducing objects that will be underdeveloped, and we want to design objects with relationships to each other.
- We want objects to have relationships with the player's important verbs. Verbs are what the player uses to gain information about the other rules of the game, so whenever the player uses a verb in a way that she expects a reaction from, she should get one. We want our verbs to be robust.
- Every rule has a context that helps the player relate to it, to understand it. This context can be reinforced by the way the game looks and sounds: something that can hurt the protagonist is covered in spikes, something that we want to direct the player toward is valuable-looking.

- A verb has a physical layer: a button the player presses, a gesture with the mouse, a swipe of a touchscreen, the rolling of a trackball. The more closely the physical layer can suggest the verb—in pace and in performance—the stronger the relationship between these two will be.
- We want to reinforce the player's verbs and their relationships whenever possible. The verb's physical layer does this, the context we give to it does this, the objects' reactions to it do this. Think of games in terms of verbs. Think of what relationships those verbs have. Think of what objects can reinforce and complicate those relationships, and how. Verbs are the characters in our game's story, and when we develop them, we tell our story.

Discussion Activities

1. Pick a game. If possible, do this at random: if you have access to games on physical media, jumble them in a pile and grab one without looking. Play this game for no more than five minutes, not counting publisher logos and title menu screens and too-long opening movies. When your five minutes are up, stop.

   Get a sheet of paper. Start writing down the player's (or players') verbs—as many as you noticed. Diagram them, especially the ones that play the most significant roles in the game. Draw connections between the ones that interact and describe their relationships. (For example, the relationship between "move left and right" and "shoot" is "aiming.")

2. Design an object for Janet Jumpjet's adventure in Venus' robot mines. It should develop or complicate Janet's verbs in an interesting way and should relate to at least one of the existing objects in the game: the robots, the caged human hostages, or the caged robots.

   You're encouraged to imagine Janet's game any way you like for this exercise. I've left some details deliberately ambiguous, like the perspective of the game and the physical layer to Janet's verbs.

3. Choose one of the following verbs and design a physical layer for it, using one or—at most—two binary keys. (Like keys on a keyboard, the game reads them either as depressed or unpressed.)

   - Scooping water from a sinking boat, with a bucket
   - Playing catch with a dog
   - Hammering a nail into a wall
   - Unwrapping a gift-wrapped present
   - Firing a slingshot

   Design a physical layer that expresses the verb as neatly as possible. You may wish to break a verb into several verbs or actions. Remember to make the physical layer as dense as possible. Design physical layers that allow for appropriate degrees of input where it makes sense.

Group Activity

Discuss the verbs you've used when playing your favorite games or the games you've played recently. Are these verbs that you find in a lot of games, like "jump" or "shoot"? Can you think of games that have used these verbs in unusual or interesting ways?

Now think of a verb that you often use in regular life or a verb that you're using right now. Maybe you're "sitting" in a chair or "writing" notes on a piece of paper. What kind of game would result if you decided to use an ordinary verb as the basis for a system?

Using one of the verbs that you just discussed, come up with an idea for a game that develops this verb. This could involve special objects that help develop the verb, such as an object that the player can jump on to change the direction of gravity or the entire view of the game world, or multiple verbs in conjunction with each other, such as a gun that changes objects into jumping platforms. Talk about what kind of gameplay might result from these combinations of verbs and objects.

CHAPTER 3

SCENES

Every game is made up of rules, which we've divided into verbs and objects. We've talked about how important it is to develop these game characters, but how, and where, do we do that? We do it with scenes, which is what we call the units of gameplay experience that unfold during and create the pacing of a game as it's played. Our players are performers, and naturally, we can't always anticipate how they will perform every given scene. But when we create a game, we have the capacity to shape those scenes and frame the choices the players get to make.

Rules in Scenes

In the previous chapter, we introduced the main characters in our story: the player's verbs, which are the rules that allow the player to interact with the game. In this chapter, we're going to talk about the ways we develop them. Our verbs and objects—the two elements of our game that comprise our system of rules—are the actors in the story. These actors perform in *scenes*.

Depending on the game, the speaker, and the position of the stars, we might call scenes levels, stages, rounds, waves, boards, missions, or screens. They're not necessarily equivalent in time, content, or presentation, though. A scene is the most basic unit of pacing in a game.

Every game might use completely different sizes and shapes for scenes. In fact, some games might have only a single scene. That's fine—it just means there isn't going to be a lot of designed character development. Character development might be the development of the player's understanding of the rules and how they interact. Consider the most basic games of the 1970s, such as *Breakout* and *Pong*. In both these games, the player moves a paddle back and forth to intercept a ball and send it back to the other side of the screen, toward another player's paddle (in *Pong*) or a wall of bricks that can be broken (in *Breakout*). These games work just fine with only one scene, and the verb—moving the paddle to hit or miss the ball—is developed as the player practices and comes to understand the ball's trajectory.

Often, a game can be broken into scenes that are even smaller than its own self-demarcation, if we accept that a scene is the most basic unit of pacing. *Super Mario Bros.* has a world 1-1, a world 1-2, and a world 2-2. Within world 1-1, we can identify and talk about a number of different scenes: the part with the monsters and pipes, the part where Mario climbs the stairs and jumps over the pit, the part where he jumps and tries to grab the flagpole. In each of these parts, a different kind of development is going on, so it's useful to consider each one a scene of its own. A scene is a more atomic, fundamental unit of gameplay than a level, or a world, or a region in a game world.

How can scenes develop a verb? Let's visualize, in our enormous brains, a game whose protagonist is from the future, wild though the concept may be. In the future, every citizen is equipped with a personal teleporter. Just think about going to a place, and you're there. Naturally, in this golden age of laziness, human legs have atrophied to the point that they more closely resemble the rockers on a rocking chair in appearance and function.

The player's verb is "teleport," and the physical layer is the mouse. Click somewhere, BING! The protagonist is there (see Figure 3.1). It's a simple game.

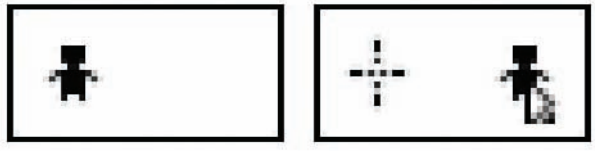


Figure 3.1 Teleportation as a verb: just click and you're there!

What situations—or scenes—could we engineer to develop this verb? Well, we need a reason for teleporting to be important. So now we have our first object. It's an electrified force-field, the kind of thing that appears everywhere in the future. The player doesn't want to touch it because it looks dangerous. We'll talk more about using context to explain the rules of the game in Chapter 4, "Context." Just know that there are sparks coming off this thing; its every inch sizzles with a dangerous green, clearly unfriendly to life.

This force-field is moving toward the protagonist, from the top of the screen to the bottom, and it stretches the entire width of the screen. Now teleportation has purpose: the player has to teleport to get to the other side of the force-field safely (see Figure 3.2). Maybe a food capsule appears on the other side of the force-field, if the player needs a stronger hint. Those futuristic food capsules are delicious! When the player teleports to the food, she's also teleporting past the force-field and to safety.

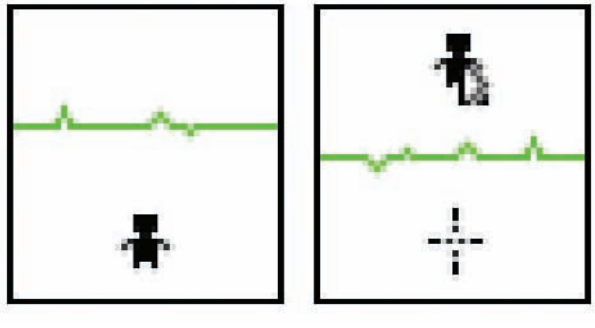


Figure 3.2 Using teleportation to dodge a deadly fence.

That was a first scene in a story about teleportation. What could be a next scene? How do we develop teleportation further? Let's add a stronger element of timing to teleportation. The next deadly fence—we'll have it come from the bottom of the screen this time, since the first

one forced the player to warp to the top—might be really, really thick. In fact, it might be so thick that not all of it fits on the screen at once. The player has to wait until it's all on the screen before she can teleport to the safe space beyond it. Its near side is closing in on the player at the top of the screen when she finally gets her opportunity (see Figure 3.3).

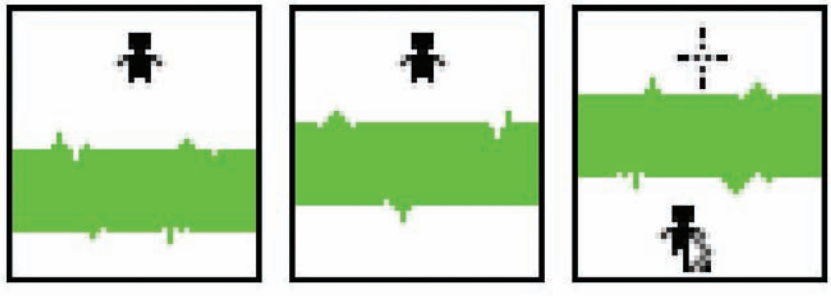


Figure 3.3 Timing the teleportation to dodge a thicker fence.

If we use timing to develop our "teleport" verb further, what does a later scene look like? Imagine an electric fence that covers the entire screen, with holes that appear long enough only for the player to wait for the next hole to appear onscreen (see Figure 3.4). How important is timing to teleportation in that scene?

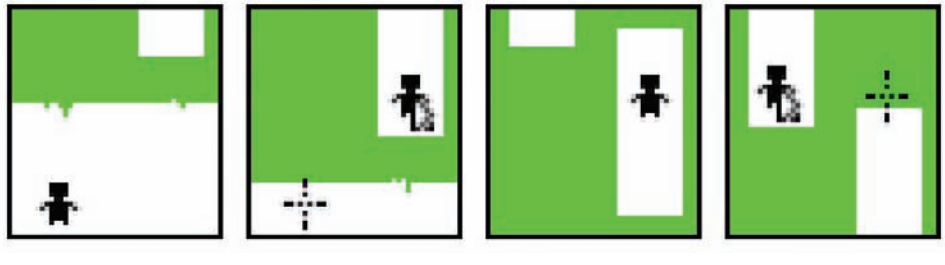


Figure 3.4 Timing is critical when jumping from hole to hole in the electric fence.

Each scene revolves around a particular development of the game's verbs. A scene can introduce a new development, like the previous examples, but it's often useful to revisit how to use a verb, or what aspects of using the verb to focus on (like timing) again and again—just as recurring themes often appear in a written story or a piece of music. As we add scenes to the game, we can also overlap these ways of developing a verb to create a more complex and rich experience for the player.

The ordering of these scenes is crucial, as you might imagine. In our teleporting game, it probably makes sense to show the player how to teleport past a nonmoving force-field first, then a thin force-field, and eventually a thick one. This sequence creates the *pacing* of the game as well as its *resistance*—concepts we'll be discussing later in this book.

The Cast

The purpose of scenes is to introduce or develop rules, to give a chance for the game's cast—its verbs and objects—to shine, and a chance for the player to understand something new about them. Objects are the building blocks of scenes. While verbs are often our main characters throughout much of a game (at least if they're robust!), the selection and arrangement of objects in a scene are often what makes the scene unique. They're our most basic tools for creating choices for the player and setting up encounters between verbs and objects. The timing of an object's appearance on stage is important.

In a game called *Ducks* (www.glorioustrainwrecks.com/node/3833) by Nick Scalzi, the player guides a mother duck around a pond: point the mouse at a position, and hold the left button to move toward it (see Figure 3.5). The verbs are simple: the player can move the mother duck around the pond. Her five ducklings follow in a line behind her, repeating every motion she makes.

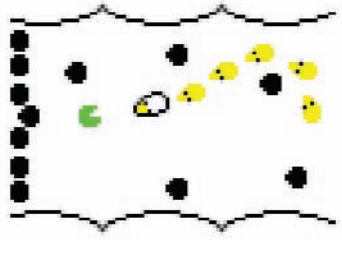


Figure 3.5 Layout of the pond in the game *Ducks*.

In the game's first scene, the mother duck swims around the pond and collects food for her ducklings. The objects here are the rocks in the pond that make impassable objects the ducks have to swim around and the moving lily pads that represent food. You can see the layout of this pond in Figure 3.5. It's a mostly open space with a few scattered rocks. There's a wall of rocks along the left border of the screen, and the banks of the river on the top and bottom.

Food lily pads drift onto the screen from the left and right. They represent moving targets—things that the player wants to catch. They're prompts; they give the player an incentive to "move" and, in doing so, to learn how the mother duck moves. The player also learns how the

baby ducks move behind the mother duck. The player brushes against a rock and learns that she has to plan paths around rocks.

After collecting several pieces of food and gaining a basic familiarity with the main verb and its limitations, something changes. A new scene introduces a sudden new object.

The new objects are bullets, fired downward from the top of the screen, beyond which, we presume, a hunter lurks on the riverbank. Suddenly the movement of the ducklings behind their mother is important—something the player has to pay attention to. As she tries to evade the hunter's bullets, she may also try and keep the ducklings out of the way of those bullets. The player's "move" verb now has an added responsibility: it's become a more developed character in the game's cast of verbs. By stopping at the right time, the player can position her ducklings so that the bullet will pass harmlessly through the space between one duckling and the next. Naturally, this is easier when moving horizontally than vertically, something the layout of the scene is designed around.

Figure 3.6 depicts the getaway scene, the length of the river to the right of the pond. The river to the left of the pond is closed off by rocks, remember. A big arrow flashes on the screen when the bullets start to fire. This book began with a complaint about the use of arrows to indicate where players should go in a game, but given the suddenness of the attack and the panic it induces, it's clear the author felt it was most fair to make absolutely clear where the path of escape lies.

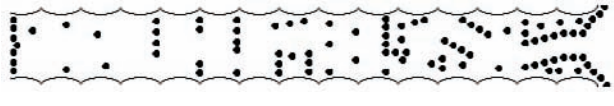


Figure 3.6 The player navigates around objects to escape the scene.

The path is constructed out of rocks, the game's most basic object. You can see from the figure that the first significant obstacle, near the left side of the screen, is a wall with a gap at the bottom. The next wall is fairly wide open for maneuvering before a third wall with a narrow gap that's slightly higher up. Why does the gap start at the bottom and then move up? Because the bullets are firing from the top. The purpose of this sequence is to gradually escalate the tension between moving and keeping your ducks out of the path of the bullets.

The bullets, not just the rocks, are a critical part of this scene. Both objects drive the development of the player's verb forward, and their relationship gets much hairier toward the close of the game. Eventually, at the right end of the river, the space the mother duck has to navigate and dodge bullets in becomes greatly constrained. This progression develops the relationship between the danger of the bullets and the player's main verb: "swim."

Making Introductions

One of the most important responsibilities of a scene is not just to develop a rule but to introduce it. The introduction of any new rule is critical. This is the player's first encounter with that rule or object, and what the player takes away from this encounter will inform every future encounter. It'll dictate her expectations every time she sees that object. What we as creators want to be sure of is that she understands the rule and its implications as completely as possible. If we want the player to be totally intimidated by a scene starring a dangerous object later in the game, we'd better make sure she leaves her first encounter with that object convinced of its danger.

We also want to teach the player this as efficiently as possible. The player's most important resource is time. The player can discover an object is dangerous by having it hurt her, but if that means the player is forced to repeat the last five minutes of the game, we've just wasted five minutes of the player's time. (We'll talk more about repetition and punishment in Chapter 6, "Resistance.") In the space mining game discussed at the end of the previous chapter, one successful enemy attack means that the expendable mining slave has been killed, and the next expendable slave is sent in her place. In other words, the player has to try the scene again from the start. This being the case, we want to avoid teaching the player by death.

You might recall some of the creatures we designed for the space mining game in the previous chapter: a monster that falls when the ground supporting it has been destroyed, and a robot turret that shoots bullets at the player when she's nearby. There were various other interactions we built using these creatures; the turret's bullets could destroy the earth beneath a rock monster, causing it to fall. But the player can't anticipate these interactions if she doesn't understand the implications of these objects in the first place. When designing a possible first scene for our game, how do we introduce these objects and what they mean?

For starters, we have to introduce the player's primary verb: the "bomb." Specifically, what we need the player to know about it is the duration of its fuse and the radius of its explosion: the specific characteristics of the verb that will guide all the player's choices about when and where to place a bomb. The solution for that—at least, for now—could be a round room with thick walls and just enough room for the player to stand outside the explosion of a bomb placed next to the walls (see Figure 3.7). There are no other creatures or items in this room. It exists just for the player to try a bomb and to get a sense of how far away and how long she has to stand to avoid being caught in the explosion.

It's important that the player develops a sense of how objects work, because the introduction of a falling rock creature hinges on this knowledge (see Figure 3.8). It's important that the player sets off this first falling rock guy with her own primary verb: her bomb. That's because the player needs to be conscious that she has the choice to release these creatures by herself (and

eventually to exploit them as weapons). But the creatures will fall quickly. This is a characteristic we decided the falling creatures should have to make it easier to line up other creatures with them and use them as weapons. We want the player to cause this creature to fall, but not to be under the creature when it falls.

Figure 3.7 The player needs a safe place to test how objects work.

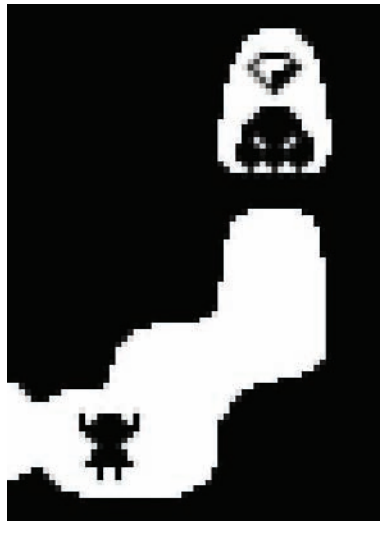


Figure 3.8 The player should be able to use previous knowledge to figure out how to interact with new objects, like falling rock creatures.

And so we exploit the player's existing knowledge of the radius and danger of her own bomb, things she has already learned. We place the first falling rock creature in a small chamber at the end of a winding corridor. The final stretch of the corridor—the part vertically below the rock monster, and thus in range of its falling attack—is too short for the player to share it with

a bomb without being caught in the explosion. So the player has to drop the bomb and then go around a corner to get outside the range of that bomb—also moving her out of reach of the falling creature. When the bomb goes off, she can observe the creature's behavior—that it falls, where it falls, and how quickly it falls—from a safe distance.

Why does the player want to bomb and release the creature in the first place? If you remember our previous discussion of this game, the player's stated goal is to collect crystals. As you can see in Figure 3.8, there is a crystal in the small chamber behind the rock creature.

We introduce the robot turret the same way. We let the player observe its behavior in relative safety, but from a close position, so that she can understand her own relationship to the creature's behavior. Just as with the rock monster, we want the player to understand how her own verbs interact with the turret so that she can eventually exploit its behavior and use it as a weapon or tool. In this case, though, the verb that the turret interacts with isn't the bomb (because the turret itself can dig through the ground, using its bullets) but simply "walking." The turret awakens, visibly unfolding and opening its mouth, when the player walks into its range of fire, and the turret goes back to sleep when the player leaves that range. It fires its bullets toward the player's current position (see Figure 3.9).

Figure 3.9 The player learns about the robot turret from a safe position, since its shots will hit a wall first.

The bait, again, is a crystal. It's at the far end of a short hallway that's within the turret's range, so when the player enters the hallway, the turret awakens. The wall is thick enough that it'll take several rounds of fire from the turret to break through and actually hit the player—giving her ample time to get the crystal and leave, especially if she's in a hurry (which she likely will be, because of the turret).

While the player is grabbing the crystal, she has the opportunity to learn things about the turret. Specifically, she learns how often it fires (currently, three bullets in a quick burst, then a short pause, then another burst), where it fires (at the player's miner), and what the effects

these bursts have on the terrain of the level (each bullet destroys a small piece of dirt that it hits). By the time the player grabs the crystal and gets out, she's learned quite a lot.

Subsequent scenes develop the relationship between these objects and the player's verbs that were introduced here, by allowing the player to actually use these objects as digging tools and weapons against other creatures.

Performance and Expression

Players should be able to make various choices about how to interact and move through a game. But isn't it a contradiction to talk about designing choices for a player while at the same time talking about how a scene should play out in a specific way? To talk about telling a story when we can't predict what our lead performer, the player or players, is going to do? Storytelling in games is like storytelling in theater: that's why we use the word *scene*.

In theater, we write a script, but the actor's performance of it is up to her. But we nevertheless compose the shape of the scene. We give the player (another theater word) liberty to perform a scene; this is where the choices come in. The choices that a player makes come from the verbs we give her, so we have the ability to constrain and design what those choices are. A fictionalized ideal of the videogame is that it gives the player the ability to do anything, to choose anything. There are games that aim for this kind of wish fulfillment and mostly result in weak experiences, lacking cohesion and focus.

We create choices that serve our stories using the verbs the player has access to. *Bioshock* (2007) is a game about rampaging through an underwater city and shooting objectivists; the player's central verb is "shooting." And yet, a critical part of the game's written story involves periodic choices over whether to "murder" or "rescue" orphaned girls. This is a sociopath's idea of a moral crisis: kill a girl to farm her for game resources, or magically transform her from a zombie girl into a ruddy-cheeked white girl.

Aside from the absurdity of having to frame a "moral" choice as being between the furthest possible extremes of human behavior, the deeper awkwardness of these choices is that they have no connection to the rest of the game. They aren't made using the verbs that the player has already been given to communicate with the game. The player is taken away from the game and presented with an arbitrary choice: press A to rescue, press B to murder. What is the relationship between these choices and the rest of the choices the player is making in the game—exploring, shooting? There is none. Although the game acts as if there's a benefit to "doing evil"—murder gives you more resources to spend on powers—it turns out that rescuing them rewards you with even more resources later on in the game. Even the game's supposed moral calculus undercuts itself.

I put a love story into a quick-draw game I made in 2008, *Calamity Annie*. Annie's verbs are "aiming" and "firing" her pistol. When Annie flirts with her love interest, the player lights her cigarette by aiming and firing (see Figure 3.10).

Figure 3.10 In *Calamity Annie,* the player is given a choice to light a cigarette by firing a pistol.

But most of the choices a player makes in a game are much more subtle. Let's go back to the future, to our game of teleportation, fences, and food capsules. When the player teleports to avoid that first electric fence, she's actually making a few choices. First: she decides when to teleport. When the fence is about to touch her? As soon as there's room behind it? When's the safest moment to teleport? Or should she try to get past the fence as quickly as possible? That's a choice.

Second, where does the player teleport? The player has to teleport somewhere behind the electric fence to avoid getting shocked by it; that's her only requirement. She could teleport to the left side of the screen, the right side, right behind the fence, as far away as possible. She could teleport in front of the fence, realize her error, and quickly click behind the fence to teleport there. As long as she ends up somewhere behind it, all that space is equally valuable.

Well, what if we make some spaces more valuable than others? Those food capsules are a way we can do that. Maybe the game keeps track of how many food capsules the player's collected, as a kind of score. Teleport onto a food capsule to collect it. The capsules are small, so the player will have to teleport to specific positions on the screen to collect them.

Now the player has the choice between teleporting to a food capsule or teleporting somewhere else. Well, of course the player's going to take the food capsule; that's not really much of a choice. So let's make it a choice: let's put another, smaller fence behind the food capsule. If the player wants the food capsule, she has to teleport not once but twice: she has to teleport to the capsule and then teleport away before this second fence can touch her (see Figure 3.11).

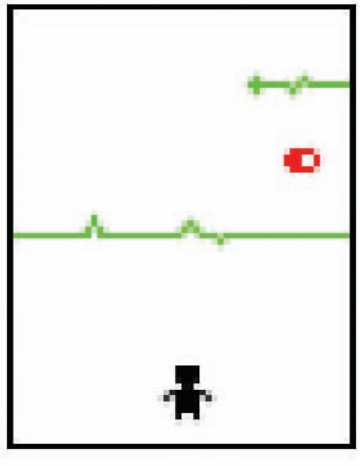


Figure 3.11 The player might have to make dangerous choices to collect some food capsules.

This is an interesting choice: take the easier path, or take a harder path for which you'll be rewarded. Giving the player meaningful choices makes her performance more meaningful. As game creators, we're little Rube Goldbergs, bolting together all sorts of interesting predicaments for the player until we have an interesting system in which she can perform by playing, and in doing so express herself through choices.

Shaping and Pacing

Shape is a word that can have a lot of meaning for us as creators of games. Think about what the shape is of each scene in your game. Remember, we're writing a script for our player, we're setting up the props in such a way as to guide the performance, but ultimately we can't police the details of the player's performance. So we can't always ask, "What happens in this scene?" Instead, we can ask, "What is the shape of this scene?" When we determine that shape, we're also defining the space of possibilities within it, much as an architect creates the space of a room or hallway by deciding where the walls go. The space of possibilities in a scene encompasses all the ways that the player has to move within the game's shape by making choices.

The shape of a scene, accordingly, might look a little like a "floor plan" or a map of the choices available to the player. In the game *Mr. Do!* (1982), the player digs tunnels to get to patches of buried cherries. Monsters then escape into the tunnels and navigate them to catch the player. The tunnels become areas the player must defend, staging areas for booby traps. This pursuit is resolved either by the player collecting all the cherries or by the monsters catching the player (see Figure 3.12).

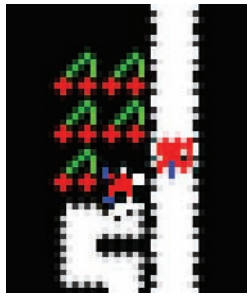


Figure 3.12 *Mr. Do!* is played in a back-and-forth tunnel pattern.

Each of these moments—digging to the cherries and eluding the monsters—has a number of small choices embedded into them that branch into many more choices. Do I dig up? Down? Left or right? Do I build my tunnel in a back-and-forth pattern to ensure that pursuers have to take the slowest path possible to reach me? When they chase me, will I dig under one of the heavy apples in the scene, hoping it will crush them when it drops? When do I drop the apple? Will I wait under it until the perfect moment? Will I dig a vertical tunnel long and straight to maximize the chances of a bunch of monsters being in it when I drop the crushing apple on all of them?

Mapping out the networks of possible player choices in a scene like this would be an impossible task, a combinatorial explosion. But as creators, we can think of the shape of the scene and perceive the relationships between different parts of that shape. We know that the player will dig tunnels and that the monsters will enter those tunnels. Having that shape in mind, we can place objects in such a way as to encourage particular interactions.

Figure 3.13 shows the starting position for the first scene of *Mr. Do!* You can see that every patch of cherries has one or two apples somewhere in relation to it. Apples fall downward if there's nothing under them, crushing monsters and potentially the protagonist, Mr. Do. The lower-left patch is the safest for a starting player to pursue, as Mr. Do starts at the bottom center of the screen. There's space between that patch and the apple above so that the player has room to construct a trap for pursuing monsters after she collects the cherries. In the patch immediately above that one, the apple is directly over the cherries. The player has to take its presence into account *while* she collects those cherries. In designing the rules of *Mr. Do!*, its creators made it

possible for many shapes to emerge—safer and more dangerous juxtapositions of cherries and apples, patterns of tunnels that players can turn to their advantage or be trapped in.

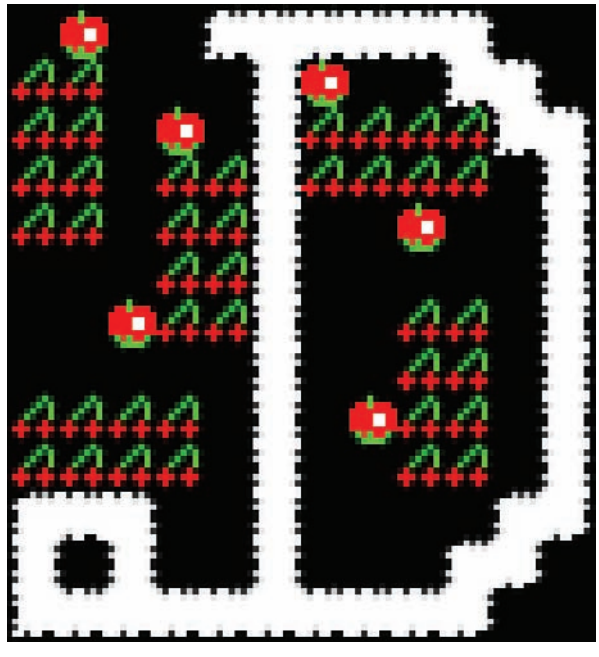


Figure 3.13 The first scene of *Mr. Do!* provides the player with numerous choices.

As we consider the shape of a scene, we can also think how the space of possibilities in that scene changes. Remember the first image in this book, the opening scene of *Super Mario Bros.*, which is shown again in Figure 3.14. The possibility space—the cloud of possible player choices—starts open but small. In this initial open space, a jump in one place or another is inconsequential. Then there's the first monster, whom the player can deal with in one of a few ways—jump on top of it, jump over it, jump while moving, jump straight up and rely on the monster's motion to bring it under Mario—but it must be dealt with. The space narrows.

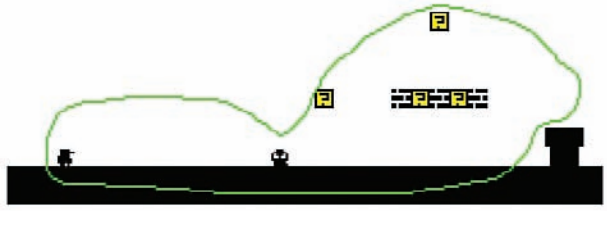


Figure 3.14 The possibility space in the opening scene of *Super Mario Bros*.

Past the monster, the shape opens even wider: there are a lot of tiny choices to make on and in between these hanging platforms. Will the player climb up to the top or the middle? Will the player collect the mushroom? Will she break some of the blocks? Break all the blocks? Just run straight through? And then, after that, it narrows again. Mario has to get over this pipe to see the rest of the game. There are a few different choices again. The player can jump from next to the pipe, from on top of the platforms, she can run and jump, or she can jump from a standing position. But she must prove she understands that she can use her verb, Mario's "jump," to navigate obstacles.

As we change the shape of a scene from beginning to end, alternating wide spaces of choice and narrow spaces, we create the pacing of the game. We can guide the motion of a scene in a particular direction, toward a particular point, without taking away the player's ability to choose. We can open our scenes wide and then slowly narrow them down until the player is performing our script word-for-word. We can choose the shape that is truest to the purpose of a scene. Then we can decide what kind of scene comes next and how the shape of that scene will relate to and continue from the scenes that came before. Will the next scene be more difficult, pushing back against the player's desire to continue? Will we develop new verbs, opening more possibilities?

We can also think of the shape of a scene in terms of how it is presented visually, how it leads the eye around and draws the player's attention to the most important elements of a scene. We talk about that in Chapter 4.

Scenes with Purpose

For every scene in your game, you should be able to answer two questions: what's the purpose of this scene, and how can you accomplish it using established game vocabulary? If you want the player to feel tense in this scene, what rules and objects are in the game that will allow you to create that feeling? Are there objects you can use to add an element of timing, for example? As we saw with the examples of the electric fence at the beginning of this chapter, timing can also help develop a verb. Are there aspects of a verb or other rules that a scene could help the player understand? When we fail to tell the story we want to tell using the vocabulary of our game, remember this: that's when we resort to using devices that have no connection to our game, like movies with no player interaction.

The game *Condensity (*www.newgrounds.com/portal/view/598331) is about guiding a group of water droplets, who all move as one but may be in different positions on the screen, to an exit that they all must occupy at the same time. This is the critical idea of the game: that the water drops may be navigating spaces that are arranged differently, with different configurations of obstacles, using the same instructions from the player. To emphasize this disparity, the droplets can be transformed (by heated or chilling elements) between two different states: the liquid state, which is affected by gravity, and the gas state, which is able to fly (see Figure 3.15).

Naturally, it is possible to have droplets that are in liquid form and droplets that are in gas form at the same time, which makes coordinating them trickier.

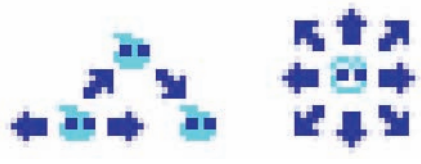


Figure 3.15 Liquid and gas states of the water droplets in *Condensity*; a liquid droplet will fall, but a gas droplet can move in any direction.

Droplets in gas form have properties that are different from the liquid form. There are fan objects that blow them around, pushing them relentlessly in a single direction. Usually these fans function like gates: to pass a fan, a gas droplet has to find a way to become liquid, or a liquid droplet needs to avoid being transformed into gas. But one scene (see Figure 3.16) finds another use for the fan: it propels a gas droplet helplessly along a path while a water droplet below, steered by the player, tries to keep up so that they can meet at the exit (where the gas droplet is transformed to water, and will fall past the exit unless the other droplet is there to catch it).

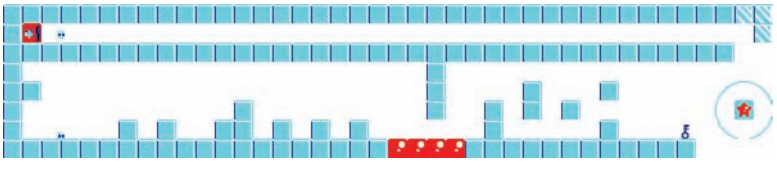


Figure 3.16 Introducing timing in a scene puts pressure on the player.

This introduces an element of timing—of racing, of struggling to keep up—into a game that doesn't normally have that kind of pressure. And it does it using rules that are already established. The player understands the interactions that conspire to create this situation for her; she knows why she's running.

Remember the game *Tombed* we looked at in Chapter 2, "Verbs and Objects"? The one with Danger Jane and the descending spiked wall? Figure 3.17 is an early scene from that game. The purpose of this scene is to make the player aware that she can use the spiked wall's destruction of objects as a strategy. Rules that have been established so far: "soft" blocks, the cyan, blue, and green, can be dug through. Metal blocks cannot—except by the spiked wall. It's the latter rule that we want to develop in this scene to give it a relationship to the player that's nontrivial.

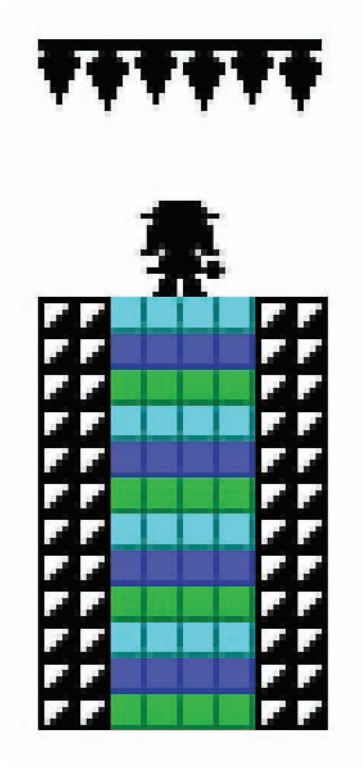


Figure 3.17 An early scene in *Tombed* shows the player how to use spiked walls strategically.

How do we develop a rule like this—one that's not a verb, but which governs the relationship between objects like the spikes and the various kinds of blocks? By giving the player a choice. Chased by the descending spikes, Jane has to dig through the soft blocks in the middle because she can't dig through the metal barriers on the sides. That's not much of a choice.

The choice comes as the spikes grow closer—and destroy the metal barriers. Once the spikes touch any part of a continuous shape, remember, the whole thing crumbles instantly (see Figure 3.18). Jane is likely somewhere in the middle of the soft-block column at this point. There are many thin layers of soft blocks, so digging through them is slow work. In fact, it's so slow that the spikes will almost certainly catch up to Jane should she continue this route. (It might be possible to dig through all the soft blocks, if one's timing is good enough to dig at a rapid pace. But it's not easy.)

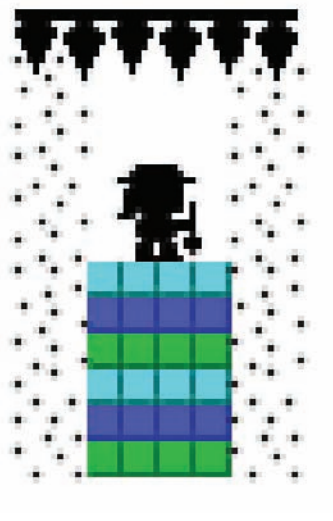


Figure 3.18 The metal blocks crumble as soon as the spikes touch them.

The solution is to walk left or right off the soft blocks into the open columns that have been cleared up by the crumbling of the metal barriers: to become cognizant of the fact that the spikes' transformation of the game world can be exploited by the player. The player's choice here is whether to keep digging or to escape to the side. One of these choices seems right and the other wrong, but making the "correct" choice requires cognition of the fact that destruction by the spikes presents opportunities for the player, and having to make a choice based on that knowledge helps the player internalize the rule more than merely observing the rule would.

Later in the game, other scenes revolve around the player's understanding that some of her options come from the spike wall's destruction of other game objects. Early in the game, this scene's purpose is to develop the "spike wall destroys things" rule to the point where that's clear, to establish the player's understanding of that rule.

Layering Objects

I learned an important lesson in 2005 while tinkering with one of my first games. The game started out being about a pig that projected her astral form into a higher plane to somehow disrupt the machinations of slaughterhouses and ended up being about a squid in a pond being pursued by fish. The game is called *Pond Squid* (see Figure 3.19).

In the space of this small pond, the player has to keep the squid out of the reach of the relentlessly pursuing fish. Eventually (though unpredictably), she gains opportunities to trap fish and use them as projectiles to remove other fish, who by this point have consolidated into a giant lump that is chasing the squid around.

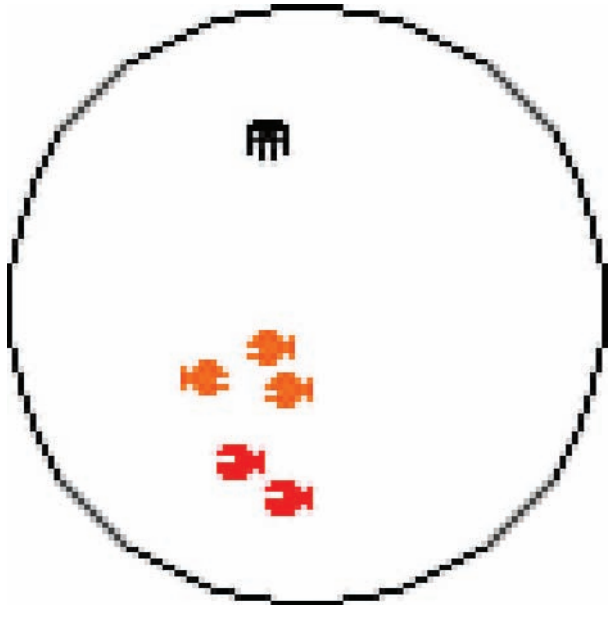


Figure 3.19 Fish pursue a squid in *Pond Squid*.

All the fish follow a really simple rule, you see. They just move directly toward the squid. So though their numbers grow continuously, they all merge into a single cloud that basically moves as one. Now, at that point I wasn't the Baba Yaga of game design I am today, but I realized the solution called for another object, another kind of fish. The fish that I added followed the same rule for pursuit as the first one—move directly toward the squid—but at half the speed.

And that worked, or it would have if I hadn't made the type of fish that appears totally random—meaning the player could see fish that move at the same speed for a really long time without ever seeing a slower fish. But when there's a mix of fast and slow fish, the game is much more interesting to play than when there's just fast fish or slow fish. You can see similar design in "bullet curtain" shooting games—different layers of bullets that are moving at different speeds at the same time are much more complicated to navigate than a field of bullets moving at the same speed.

The lesson is that *layering* is important. Having objects that stack in interesting ways creates more interesting choices. Six years after I made *Pond Squid*, I was confronted with the importance of this rule again while working on *Lesbian Spider-Queens of Mars* (http://games.adultswim.com/lesbian-spider-queens-of-mars-twitchy-online-game.html). In this game, a Martian spider-queen pursues her escaped slaves through a maze, attempting to zap them with a bondage laser and recapture them. The slaves are armed, so if one of them manages to get

the drop on the queen—sneak up on her without getting zapped by the bondage laser—the slaves can turn the tables.

The most basic kind of runaway slave simply picks a new direction at random every time she approaches an intersection (without doubling back—see Figure 3.20). This slave makes up the bulk of the queen's opposition in the game—since she chooses at random, she doesn't always move toward the queen, giving her opportunities to sneak up from behind, and she may avoid opportunities to really corner the queen. She's more difficult in greater numbers—they fill more space, all picking different paths—but she's easily manageable in small numbers.

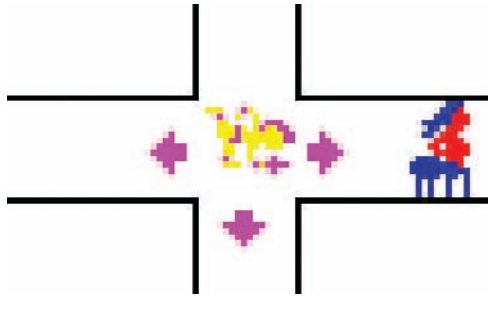


Figure 3.20 Simple slaves pick random directions in *Lesbian Spider-Queens of Mars*.

Midway through the game I wanted to introduce a smarter opponent. This one is called a gladiator. Whenever a gladiator reaches an intersection, she looks at the queen's position in the maze relative to hers and picks the direction that will lead her closer to the queen the quickest (see Figure 3.21).

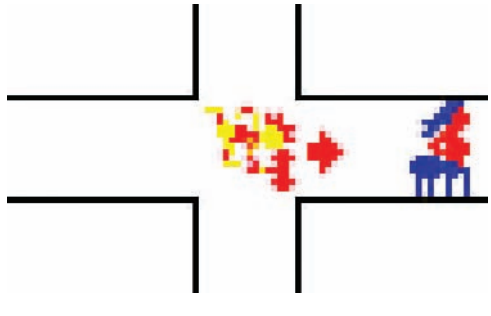


Figure 3.21 The gladiator picks a direction that leads her to the queen.

Originally, I had imagined gladiators becoming the replacement for the original slaves in the later scenes of the game—a harder-to-deal-with version of the same behavior. That's what I thought. In practice, though, it became apparent that they weren't as overwhelming in large numbers, that they didn't gain as much from numbers as the original slaves. You see, they

were predictable: since they always chose to move toward the queen, if a bunch of them were coming from the same place, they'd all march toward her in a line, blundering straight into her bondage laser.

It was much more interesting, I discovered, to mix the random-moving slaves with the queen-chasing gladiators. That made for much more complex situations and much more meaningful choices.

Having objects that complement each other is important to constructing situations. Look at *DOOM* (1993). The game introduces early on a monster called an imp that throws fireballs at the player (which travel in a straight line) and a monster called a pinky demon that attacks by biting and takes circuitous paths to reach the player to try to dodge her fire. Many of the most precarious scenes in the game require the player to manage these two different avenues of attack simultaneously.

For the mining game discussed previously, we wanted to be sure to design objects that could be combined to create interesting scenes and meaningful choices. For example, in Figure 3.22, the turrets potentially set off a chain reaction of falling rock creatures that ultimately destroys the turrets and clears the way to a bunch of crystals, but the player must dodge both the falling rocks and the turrets' bullets. The player could set off the chain reaction with her own bomb. There are many ways, involving conservative and risky choices, to play out this scene.

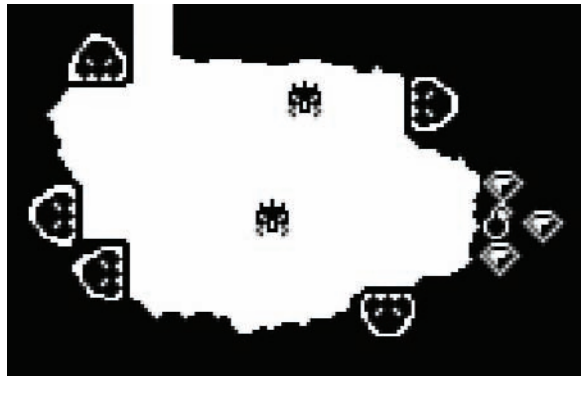


Figure 3.22 A scene in the mining game with a combination of turrets and rock monsters.

This scene doesn't appear until after both the turrets and the rock monsters have been introduced individually (something we did in earlier scenes). The player is unable to make those choices—conservative versus risky, whether to initiate the chain reaction with a bomb, with a bullet, or try to avoid it entirely—unless she first understands how the individual objects—the rock monster and the turret—work. Without that knowledge, based on earlier experiences, this room is a trap, not an arena for player choice.

Moments of Inversion

An encounter with an overwhelming force, a sudden shift in the focus of the game, is a *moment of inversion*. Moments where a rule in the background suddenly comes to the foreground—where a hunter becomes the hunted or vice versa—these are moments of climax in the pacing of a game. Thinking about the shape of a scene, moments of inversion make a scene's shape more dynamic.

Wizard of Wor (1981), the game that inspired *Lesbian Spider-Queens,* is paced by regular moments of inversion. *Wor* is a game in which two human players ("worriors") attempt to survive mazes full of monsters—the arena of the titular villain, the Wizard of Wor. The mazes are symmetrical labyrinths of walls and corridors; a marked door on either side allows a player to "wrap around" to the other side of the maze. A number of monsters patrol each maze, stalking the players. They can fire bullets at the worriors and are invisible when a worrior doesn't occupy the same hallway. There's a radar on the bottom of the screen—a smaller map of the maze—that shows the positions of all the monsters. The players have to rely on it to track hidden monsters. The monsters start much slower than the players and gradually pick up speed until they're much faster.

Fending off the horde of monsters is tricky. The worriors are outnumbered—monsters come from every direction, and, as in *Space Invaders*, only one bullet is allowed on the screen at a time. That means missed shots can be disastrous, and a worrior who acts rashly is easily overwhelmed. Worriors can also shoot one another—by accident or on purpose. (The game awards a big point bonus to a player who shoots the other, to add temptation to the dynamic between the players.) Players often split up, working different parts of the maze, finding advantageous positions and working to defend them from enemies often coming from many different places. Figure 3.23 depicts an average maze in *Wizard of Wor.*

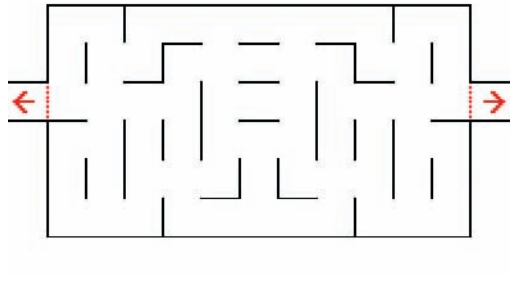


Figure 3.23 Worriors fight in a maze in *Wizard of Wor.*

When the last monster is destroyed, something happens—suddenly. The Worluk, the wizard's pet, appears somewhere in the maze. This creature will not shoot at the worriors. The Worluk

is always visible on the screen. It attempts to reach one of the doors—the wraparound doors on the left and right side of the screen—and exit the maze. Now the players, after fending off a huge mob of aggressive monsters, have the opportunity to be hunters. The Worluk can kill a worrior on contact, but it doesn't pursue him—it just tries to reach an exit door quickly.

There are two exit doors. The Worluk moves much too fast for a single worrior to pursue it. Hunting the Worluk requires teamwork. The reward for catching the Worluk, in fact, benefits both players; it causes all points to be doubled for both players in the next maze. The players outnumber their enemy for the first time and are given an incentive to work together.

Shooting the Worluk sometimes incites the Wizard himself—who spends most of the game taunting the players vocally, unseen—to appear in the maze and attempt to kill a worrior as punishment. Successfully shooting the Wizard is a difficult task, but it's one of the greatest moments of role reversal in the game.

The pace of *Wizard of Wor* is defined by the contrast between these long scenes of the players defending themselves against many enemies and these fast quick ones where they have the opportunity to strike back and be empowered, possibly to humiliate their antagonist. Were it not for these moments—if there was no Wizard or Worluk, only endless mazes and monsters—the game would have no changes in pacing; the space of possibilities would remain the same throughout. It would be a single, unbroken scene, essentially. The inversions that happen in the Worluk scenes not only give a narrative shape to each maze—they mark the end of a scene and the release of the tension that has been building in that scene—but give the game a more dynamic shape (see Figure 3.24). Be slowly hunted for a villain's amusement, have a brief, frantic opportunity to strike back at that villain. We talk more about reasons to give the player a "change of pace," to switch things up in the shape of the game and thus the space of possibilities, in Chapter 4.

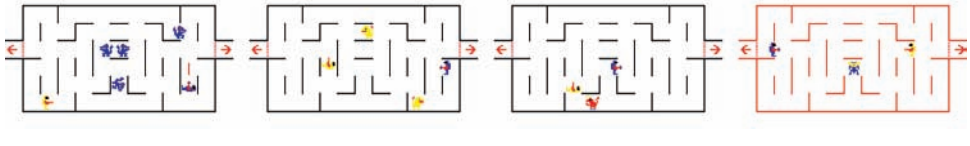


Figure 3.24 Timeline of a single maze in *Wizard of Wor*, with the Worluk battle in the rightmost frame.

Chance

Since we're working with computers, it's easy to incorporate *chance* into many different areas of design. It's hard for a computer to produce a truly random number, but it's capable of generating what are practically random numbers of any size on the fly. A game creator can use these to many different ends.

Chance's usefulness in scene design is to shift the focus of the play onto the rules themselves, not on how they're encountered. Randomness works best when the player is engaged with a complicated network of rules; it can be used to provide myriad combinations of those rules and objects. Randomness isn't a shortcut to design. Its function has much less effect when dealing with more succinct rule sets. Chance is best used for combinatorial ends.

Take the game *Crosstown* (2009), an Xbox Indie Game that's a descendant of *Wizard of Wor*. The players—up to four—explore a maze filled with creatures, attempting to collect four power objects. There are more than 15 creatures in the game's cast, which all interact with the players and with each other in different ways. One builds walls, one destroys them, one only attacks other monsters (not players), one seeks out and defends the objects the players are seeking.

The interactions between these creatures are the center of the play. The layouts of each maze are important only in that they force the creatures—and the four players—to interact. As a result, each maze is a simple, randomly drawn pattern, with left–right symmetry and lots of branching paths (see Figure 3.25), that allows for lots of interaction but also for individual encounters to be isolated to different hallways.

I played an early version of *Crosstown* in which the mazes were much bigger. The creatures and the players didn't interact as much because of the larger space. When the author made the mazes smaller, the game started to really come together—characters interacted much more often.

Crosstown doesn't have much to gain from designed mazes. *Wizard of Wor*, which has a smaller cast of characters that don't interact with each other, only the player, uses designed mazes. Some of those mazes are in fact moments of inversion: the arena scene, which features a large open, wall-less area in the center, and the pit, which is entirely wall-less. Because walls are so important for cover, these scenes represent major upsets for the player.

But chance doesn't have to be all or nothing. Consider the Worluk in *Wizard of Wor*. When it appears in the maze at the end of a scene, it does so at a randomly chosen position (see Figure 3.26).

What would happen if the Worluk appeared at the same position every game? The players could anticipate its appearance and be ready for it each time, defusing its potency as a situation that requires the players to collaborate.

So chance has the capacity to break stagnation in a game. The monsters in *Wor* appear in different locations every time, ensuring players can't simply memorize each scene. This shifts the focus from learning the scenes to dealing with the monsters. The Wizard himself, when he appears, teleports to random positions, making him less like a foe that can be outfoxed than a force of nature to be endured. The rules of interacting with monsters, with staying alive, become more important, not the particulars of any one encounter.

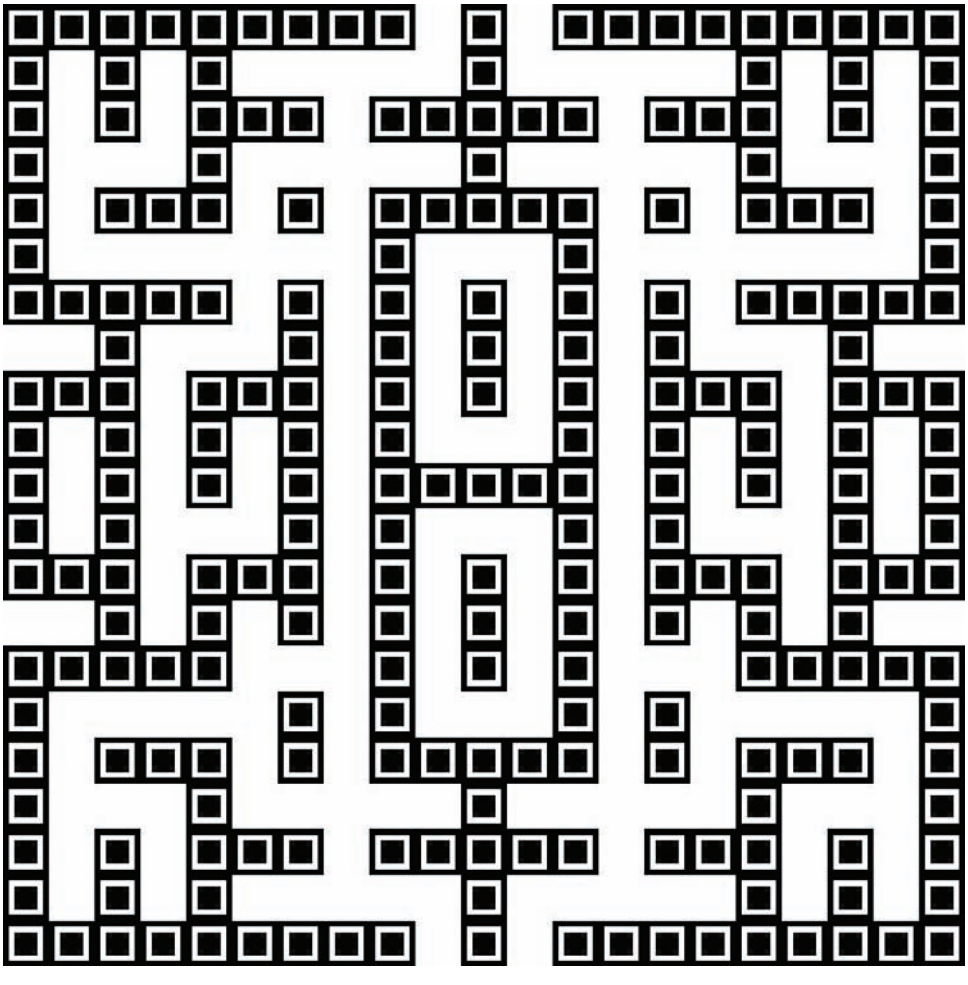


Figure 3.25 A typical *Crosstown* maze layout is a simple random pattern.

Chance is also useful for breaking symmetry. In Michael Brough's *Glitch Tank* (2011) for the iPad, two players attempt to destroy each other by choosing commands for robot tanks. Commands include "turn right," "go forward," and "fire a laser." But the commands available to each player—four at a time—are chosen at random like cards from a shuffled deck (see Figure 3.27). So an ideal move isn't always available, and players often have to choose and incorporate into their strategies less-than-ideal moves.

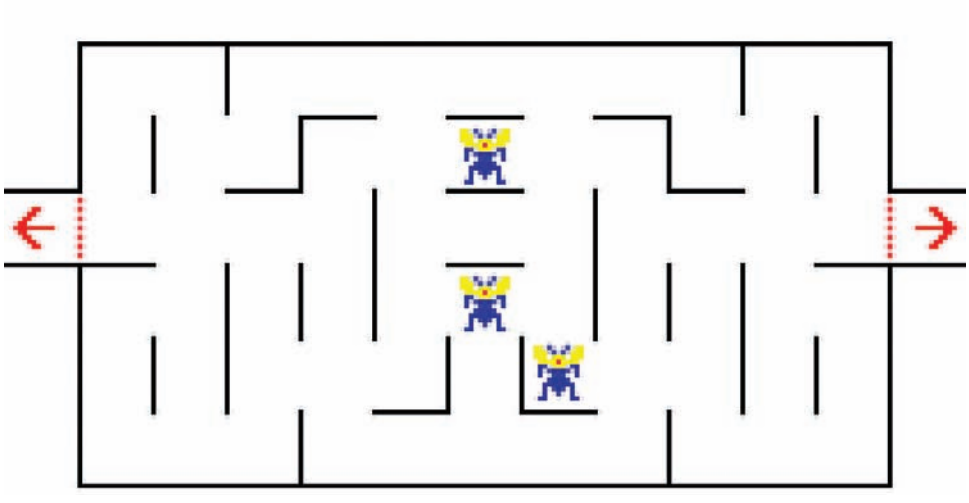


Figure 3.26 The Worluk appears at one of these locations, randomly chosen, in *Wizard of Wor*.

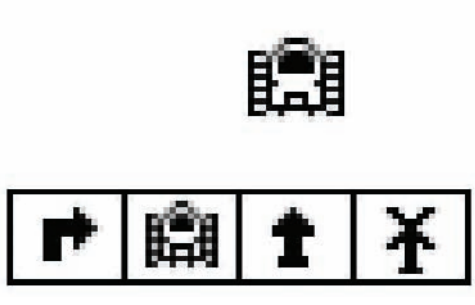


Figure 3.27 The player is given a set of four random commands at a time in *Glitch Tank*.

The effect of this is that it's impossible for both players to play the same series of moves. If that were possible, the result would be stagnation: the player who played fastest would have the advantage. And the game isn't about who can play fastest, although thinking fast is an important skill in *Glitch Tank*. The focus of the game is on outmaneuvering one's opponent with the resources one is given. Chance, in this case, maintains a dynamic between the players by preventing them from performing symmetrically.

Real Talk

Let's talk about how a scene I designed went from conception to its final version. The scene is from the game *REDDER*, which I made in 2010. *REDDER*'s protagonist, Hannah, is an astronaut

who lands on Mars when her spaceship's supply of fuel gems is extinguished. To escape, she must find replacement gems scattered throughout the ruins of an abandoned civilization—abandoned by life, that is, but its electronic defenses are still online—electric fences still sizzle, robot guardians still patrol.

Verbs, primary: Hannah can "move" left and right, on the ground or midair. She can "jump" to four-and-a-half times her height, which means she can "climb" anything up to four blocks—a block being a solid object of varying appearance as tall as Hannah and slightly wider (see Figure 3.28).

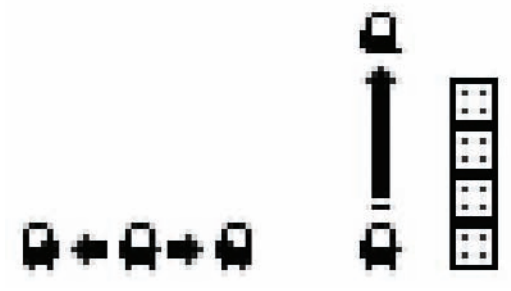


Figure 3.28 The primary verbs of *REDDER*.

Verbs, secondary: Hannah can "trigger" switches that open or close off her way forward. Passage through the old Martian crypts and laboratories is regulated by gates that are more like electronic walls. These things come in two states: extended and receded. When they're extended, they function as solid blocks, serving as wall, floor, ceiling—physical obstacles Hannah cannot pass through. When they've receded into the background, they're as intangible as air, solid space that Hannah can pass right through.

Electronic walls come in two colors: red and green. One color always is extended and one receded. Touching a green switch makes green walls recede and red walls extend: touching a red switch does the opposite. The switches are scanners: they trigger the moment Hannah passes through them, whether she wants them to or not (see Figure 3.29).

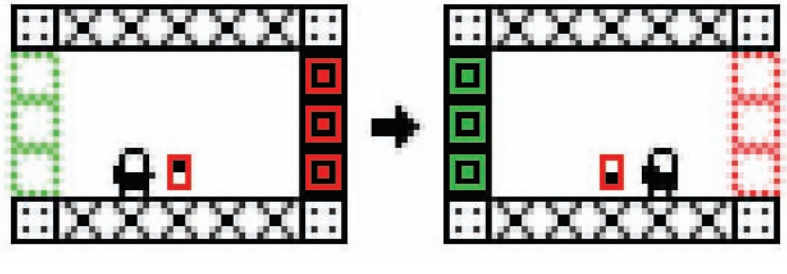


Figure 3.29 "Triggering" switches and gates are the secondary verbs.

The first function of these is to guide the player's exploration of the world: a red wall on one screen might be opened by a red switch a few screens away, requiring the player to go find it, which creates continuity between individual scenes of the game. Switches and walls can also be one-way doors: just passing in front of a switch activates it, regardless of whether it's to the player's benefit. Maybe this switch creates a wall in the hall Hannah just passed through, preventing her immediate return. You can see how switches could even be traps: since extended walls function as solid objects, imagine a midair platform built out of them. If Hannah touched a switch, the floor would open. She would have to avoid switches as she travelled across the platform (see Figure 3.30).

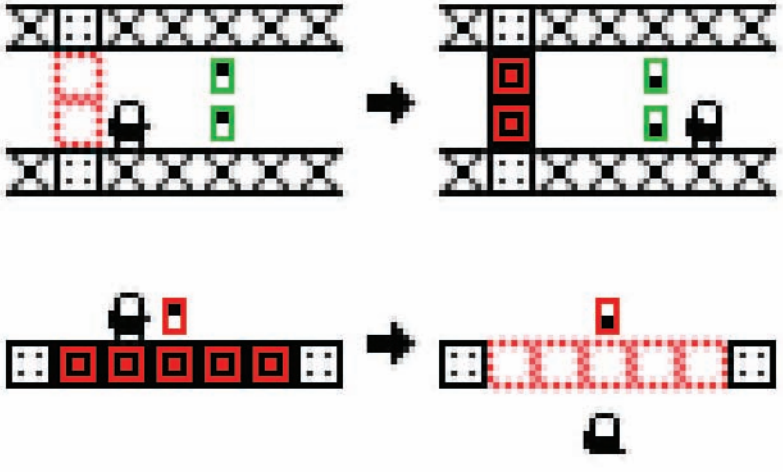


Figure 3.30 One-way door and gate platforms sometimes act as traps in *REDDER*.

REDDER's Mars is divided into distinct areas to help the player focus her exploration. Each area is distinct not only visually (that is, the blocks that make up that area look noticeably different from those of other areas) but in shape. For example, one area is an underground city, divided into two areas, with a gate separating the upper city from the lower city. Another area is a kind of puzzle box, with red and green gates separating each screen. Making progress requires manipulating switches in the right order.

One of the deepest areas is an open cavern consisting of many rooms that can be traversed with a system of rails and catwalks. Passage from room to room, once again, is governed by switches and gates. Whereas the puzzle box area is a maze, this area is straightforward: each room contains a switch opening the entrance to the next room, usually placed in a tricky-to-reach position. When Hannah touches it, she can advance to the next room in the path. There are two paths that the player can follow to navigate the cavern: a clockwise circle and

a counterclockwise circle that join in the lower middle. In the upper-middle room, shown in green in Figure 3.31, the paths divide, giving the player a choice.

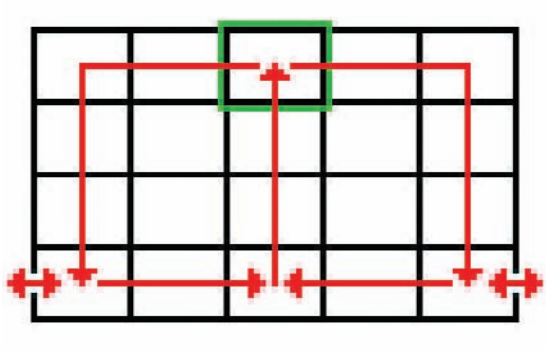


Figure 3.31 Layout of two paths (one clockwise, the other counterclockwise) connecting various rooms.

Figure 3.32 shows that upper-middle room, where the paths split. At least, this is what that room originally looked like, in an early version of the game. It's symmetrical because the scene's role is symmetrical: the player has a choice of entering the left or right path. She enters the room through the bottom. She has to climb up on one of the ledges at the left or right and then jump over the nearest electric fence-topped pillar to touch the switch above it. (The electric fences are dangerous. Hannah will have to return to the previous scene and start again if she lands on one.) The switch opens the left and right exits and closes the path through which she entered, turning it into solid floor.

So this was the room for a while. But I needed to connect this area—the cavern of catwalks and one-way doors—with a different area above it, the aforementioned underground city. In the interest of player mobility, there needed to be a way for the player to pass between these two areas in either direction. This scene, the upper-middle room, seemed the most logical place for the player to enter from above, being an intersection between this area's two paths. A connection here would afford the player the most choices.

I had to create a vertical connection between this room and the one above it. I regretted having to break the visual symmetry of the scene, which perfectly matched the symmetrical left/right choices available to the player. What if the broken symmetry was part of this room's apparent history? A hole in the upper catwalk could be the vertical connection to the above area, and the presence of the hole could signify a structural decay that destroyed the room's ostensible original symmetry.

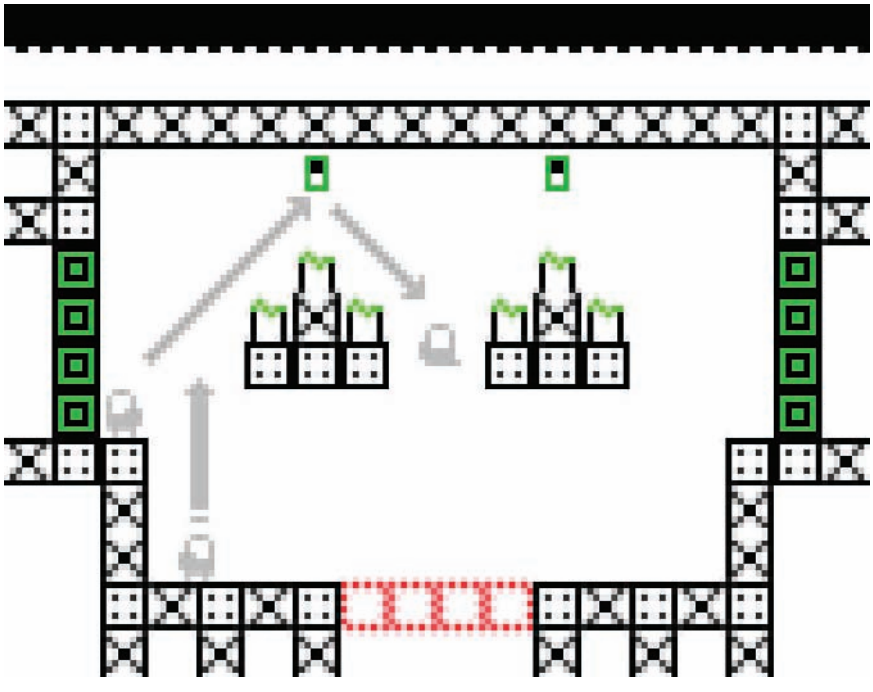


Figure 3.32 Original version of the room where the path splits to the left and right.

Figure 3.33 shows the next iteration of the scene. The piece I took out of the ceiling now rests on top of the platform below, where the electric fences can be found on its mirror platform. The implication is that part of the ceiling collapsed, changing the layout of the room and creating a path upward. This platform-on-top-of-a-platform provides enough height that Hannah can jump through the hole to the top of the above catwalk and travel along it—two rooms to the right is a passage to the city area above.

But this change introduced a new problem. The player can still hit the green switch by jumping from the left ledge over the fences. But because the right platform is so tall, if she tries to jump and hit the switch from the right, she'll hit the ceiling and descend early, landing on the electric fence (see Figure 3.34). This was a bad break in the symmetry of the room that wouldn't do. There's no visual indication that jumping from the right should be less valid or more dangerous than jumping from the left.

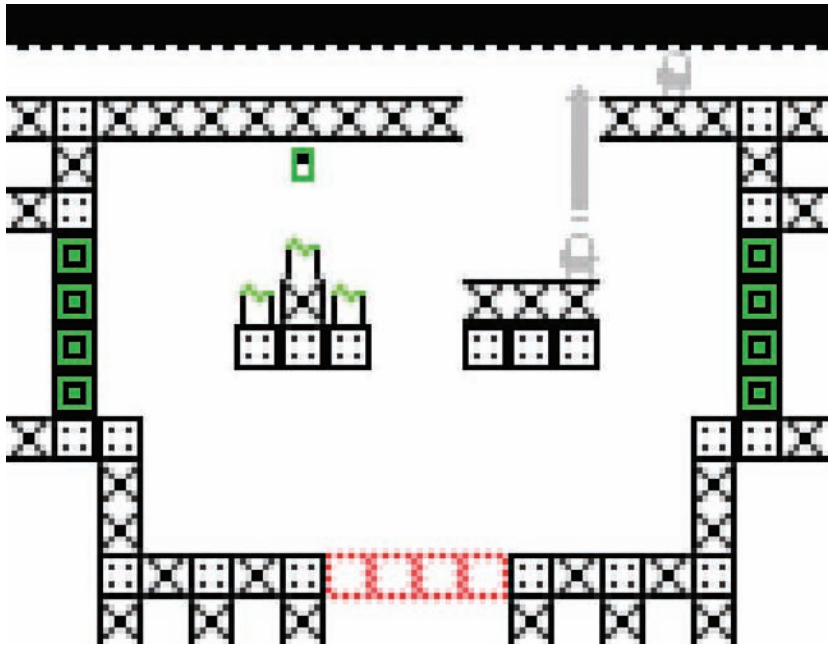


Figure 3.33 Updated scene with fallen platform opening a hole at the top.

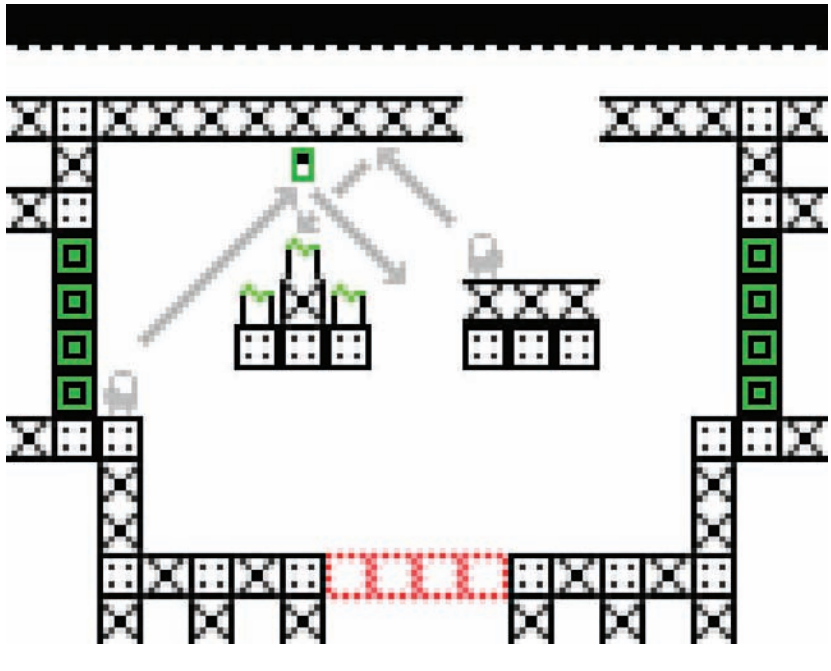


Figure 3.34 A bad break in symmetry makes jumping from the right deadly.

So how could I make the switch accessible from both the left and the right? If I got rid of the fallen catwalk—lowering the height of the right platform—Hannah could make the jump to the switch, but she couldn't reach the upper hole. Well, this is a dislodged chunk of the ceiling, right? There's no reason it needs to be flush with the platform it's lying on. Why would a piece of busted architecture fall in such a neat position?

Figure 3.35 is the finished scene. I shifted the fallen catwalk over one block so it overhangs the platform. As a nice bonus, it's visually even more asymmetrical—a sign that it's an exit from the symmetry of the whole left/right area shown in Figure 3.31. Jumping from the top of the fallen catwalk, the player can reach the hole above and the passage to the city. Jumping from the exposed piece of platform beside it, the player can hit the switch and clear the electric fence. Now the room fills every function I want from it. I created, played, identified problems, changed, played again, discovered new problems, changed again, played again, and solved the problem. That's design.

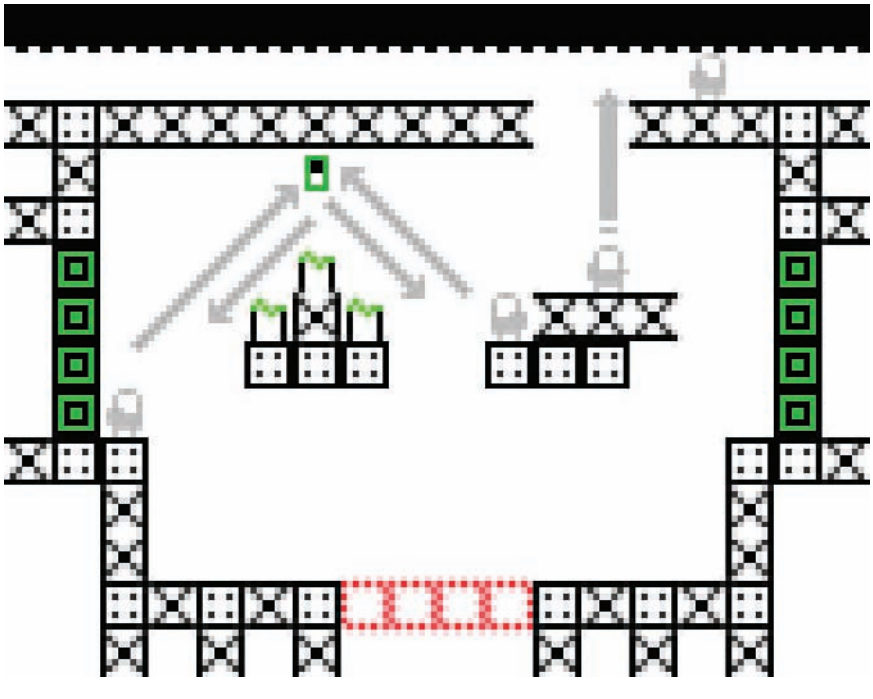


Figure 3.35 The finished scene with all problems fixed.

Review

- A scene is the most basic unit of pacing in any given game. What constitutes a scene might be different in every game.
- The responsibilities of a scene are to introduce and develop rules of the game. Objects are the building blocks of our scenes, what we use to create choices.
- It's important to introduce rules clearly so that the player understands what they mean and represent when she encounters them in successive scenes.
- We should use the rules we've established—verbs, objects, their relationships—to create choices for the player. Otherwise, we're creating choices that have no relationship to the game.
- Scenes are often made of many tiny choices, exclusive to each player's individual performance of a scene. That's why it's often useful to think and design in terms of the overall shape of a scene.
- Each scene should have a purpose that we can identify: to develop a specific rule or to present a specific idea. Design means using the rules the player already understands to communicate that idea.
- Layering—including different rules that work well together in a scene, like making the player track two different kinds of movement simultaneously—can create stronger, more effective scenes.
- Moments of inversion or climax give a scene a more dynamic shape and can draw attention to different aspects of the player's verbs.
- Chance is useful for focusing the game on the interactions between rules rather than how they're presented. Using randomized patterns for scene layouts, for example, is at its most useful when it's the interactions between many rules that are important, not the encounters that can be staged with them.
- Chance allows us to break stagnation by interfering with the player's ability to predict the game and to break symmetry by forcing competing players to play differently.
- Designing a scene often involves several drafts. Between drafts, play, identify problems, and make changes to solve those problems.

Discussion Activities

1. Choose a game—the game your group used for the previous chapter's discussion, if possible. Give yourself 10 minutes to play. If you're in a group, only one person should play at a time. When the 10 minutes are up, finish the scene the player is currently on. The group can decide what constitutes the end of the scene. Discuss that last scene.

2. Identify any rules introduced in that scene, and any ways in which objects interact in that scene. Most especially, what are the ways that appear in this scene for the first time? What is the focus of the scene? What rules does the scene introduce? What rules does it develop? What's the purpose of this scene?
3. Using graph paper, design a scene for the teleporting/laser game described in this chapter. Assume that six squares wide by eight squares tall is the space the player has to move around in: the scene you're designing will scroll onscreen into that space, from a direction and at a speed that's up to you.

   A line along any grid line is a laser fence and cannot be touched. A box that's filled in is an irradiated laser area and cannot be teleported to. A dot in any box is a delicious food capsule (see Figure 3.36).

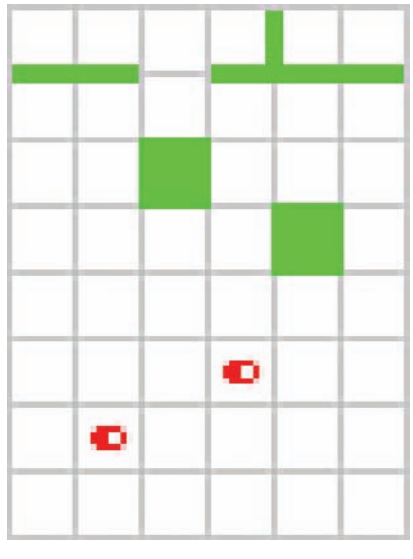


Figure 3.36 Reference for designing a scene in the teleporting/laser fence game.

4. Design a scene around a meaningful choice for the player to make. There will probably be more than one choice the player makes in your scene, but there should be at least one major choice—one the player recognizes as a choice with consequences.

   You may introduce a new object, but you must communicate the rules and ramifications of that object prior to the major choice.
5. Choose one of the following scenarios:
   - A masquerade ball celebrating the protagonist's birthday. She is now old enough to inherit the Imperial throne. At least one of the guests is an assassin and will attempt to strike before the night is through.

- Shipwrecked! The protagonist's vessel has been shattered by a terrible storm, and she must find her way to dry land. Once there, she must somehow find shelter.
- This old house cleaned up so well, no one will even realize it's haunted. Just a few more tasks before the protagonist can put it on the market—but then something goes unexpectedly awry!

Now, imagine that you're going to shape the space of possibilities in the scenario you chose—the space of choices the player can make. Following is a list of a few possible shapes. Each of these shapes changes over time. When the shape is more open, the player has a wider space in which to decide what to do. For a masquerade ball, that might be the informal part of the evening where guests are free to mingle as they choose and talk to all sorts of characters. When the shape is tighter, more closed, the possible choices are more constrained, with fewer options open—perhaps because of tension or danger. In the masquerade ball, this might involve the moment when the assassin strikes, when the player must react immediately.

Here are some possible shapes for your scenario. How might these play out? What would happen at each change in shape?

- A wide-open space in the beginning and for the first half, then narrowing to a tight space for the second half.
- A wide-open space followed by smaller and smaller ones, but ending in a large open space of possibilities.
- A narrow space from the beginning that periodically gets wider and then contracts again at the end.

Defining the rules of the game (and the larger story into which this scene fits, if you so desire) is up to you.

Group Activity

Knytt Stories, created by Nicklas Nygren in 2007, is a platform for creating playable "stories" about a protagonist, named Juni, whose primary verbs are "running," "jumping," and "climbing." You can download it freely at http://nifflas.ni2.se/?page=Knytt+Stories, though it will only run on Windows computers.

Have someone in the group play through the included "Tutorial" story and 10 minutes or so of the other included story, "The Machine." Then, working as a group or breaking into smaller groups, look at the level editor. To understand the level editor, be sure to check out the following posts in the Knytt Stories support forum:

- **Level Editor FAQ:** http://nifflas.lpchip.nl/index.php?topic=28.0
- **Level Editor Manual:** http://nifflas.lpchip.nl/index.php?action=dlattach;topic=18.0;attach=5

Figure 3.37 shows the level editor. The numbers 0–7 on the right represent different layers of the game. Layer 3 is where solid objects go: anything placed here Juni will be able to jump to and touch and climb on. Layers 0, 1, and 2 are the background: anything placed here Juni will pass right over. Click on a layer, and then click on one of the tiles on the bottom of the window. Then you can use the mouse to draw in the scene.

Figure 3.37 The *Knytt Stories* level editor; click the brightly colored numbers to change which layer of the game world you're working with.

Layers 4–7 are for "objects," which in *Knytt Stories* refers to any object more complex than a wall or floor tile. At the far right of the screen, you can select from the many objects in *Knytt Stories*: click on "bank" to cycle through categories of objects and "obj" to cycle through specific objects. Right-clicking cycles in the opposite direction. (You can also cycle tilesets and backgrounds by left- and right-clicking on the appropriate parts of the editor.)

Knytt Stories has a variety of objects. Each group should confer, experiment, and pick three of these. These objects could be a block that's only visible when the player is close, a spiky creature that drives back and forth along the ground, and a button that needs to be held for a while to open a gate.

(Also, don't forget about save points—bank 0, object 1! Juni starts over at these if she touches something dangerous like a spiky creature. Also, be sure to turn on the abilities that you want the player to start with. You can choose these after clicking "set start pos," which allows you to set the story's starting position. I recommend starting the player with the ability to run and climb, at least.)

Create a story about the three objects you've chosen. Start by introducing the first one. Develop it a little in another scene. Then introduce the second object by itself, and create a scene that combines the first and second objects. Introduce the third object by itself. Create a scene that combines the first and third objects, and then one that combines the second and third. Finally, combine all three.

Make sure that your combinations don't merely contain the different objects, but actually interact with each other. Find as many uses for each object—as much utility—as possible.

This page intentionally left blank

CHAPTER 4

CONTEXT

Games are made of rules, and those rules allow us to create choices for the player. But those choices exist only when the player understands those rules. Context is what helps a player to internalize those otherwise-abstract rules that make up our game. Digital games have the capacity to use visual art, animation, music, and sound to shape that context and communicate with the player. These channels of communication can say a lot.

First Impressions

I played an early version of *Super Crate Box* (2010) before it was available online. In this game, boxes appear at random positions on the screen; the player is scored by how many boxes she can collect. Green monsters of various kinds—little and fast, big and slow, monsters that run straight ahead and ones that actively chase the player—drop from a hole in the top of the screen and move toward the bottom. If they make it all the way to the bottom of the screen without being shot, sliced, exploded, or murdered in some way by the player, they fall down a hole in the bottom and reappear from the hole in the top, faster than before and red now instead of green (see Figure 4.1).

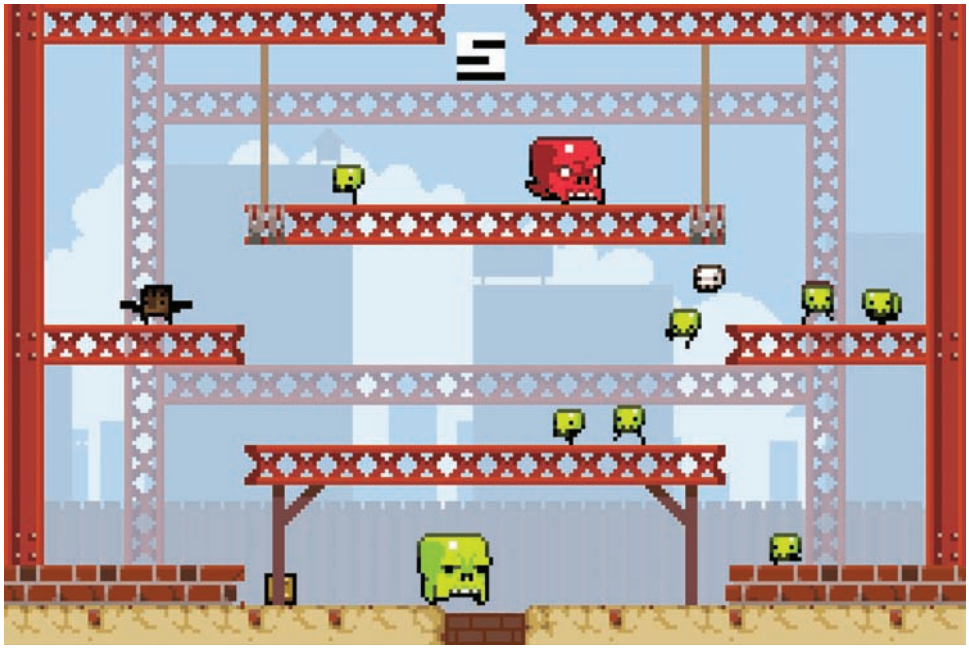


Figure 4.1 Monsters wrapping around the screen and becoming red.

So strategy in this game is a balance between collecting crates and keeping the monster population in check using the weapons available to the player. If the player collects a box, which contains a randomly chosen weapon that she must then use, she is forced to change her strategy. This is the dynamic of the game.

Here's the part that confused me, though: in the early version of the game I played, while the monsters re-emerge at the top more powerful (and red) after they jump into the hole, when I jump my little character into the same hole, it dies. This was, to me, a jarring lack of consistency.

As a designer, I understand why the authors would want to keep the player in a bounded area that emphasizes horizontal motion as much as vertical motion. And I understand why they would want the monsters to travel on a path between introduction and escalation. But as I played, the game failed to communicate an important rule: I can't jump in the pit like the monsters can. And the game didn't help me remember this rule, either; despite dying the first time, I jumped into that deadly hole again and again during the chaotic frenzy of the game.

The solution that the creators of *Super Crate Box* came up with was a simple visual effect: they set the hole on fire (see Figure 4.2). Red flames now dance in the hole at the bottom of the screen. If the player falls in, she's fried. If a monster falls in, it's cooked red hot and released, furiously speeding up as it drops from the top of the screen. I can buy this: it's just a small visual change, but it provides a justification for why the hole kills me but transforms monsters. It provides context.

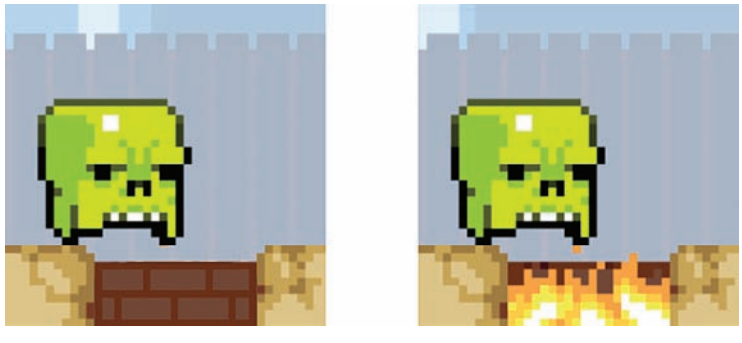


Figure 4.2 Both versions of the hole at the bottom of *Super Crate Box* are deadly, but the one on the right makes that visually obvious.

At a mechanical level, games are about rules and interactions. We introduce things that are dangerous to the player so there can be a conflict, so we can develop the player's verbs: you've got to be good at moving to avoid the hazards hurtling toward you. You've got to understand how to direct your shots to shoot the monster. But in the abstract, our rules are just that: abstractions. Firing a gun, running around and avoiding dangers—these are complex, nuanced activities. We don't want to make a simulation of running (caveat: sometimes, maybe we do). We're trying to tell a story. So we abstract.

How does the player know that the blip is dangerous? How does she know not to touch it? Well, it could look like a dangerous monster, all fangs and claws. What if instead of meandering, it moves toward the player, aggressively? It could move slowly, a shamble of heavy footsteps that shake the screen. How does the player know shooting it is a good thing? Well, when the blip is shot, a melody could play suggesting relief from danger.

Digital games, running on screens, breathing through speakers, are capable of presenting visual and audio information to the player. Images, animation, sounds, music. We can use these channels to communicate the rules of our games, to emphasize and underline the important interactions.

Look at a game like *Plants vs. Zombies* (2009). In this strategy game, the player is trying to defend the left side of the screen by erecting defenses, each with a different function, that cannot be moved once positioned. The player must choose what defenses to build and where to build them. Different kinds of defenses have to be positioned to support other defenses. Enemy forces appear on the right and move slowly toward the left, giving the player time to adjust her strategy and make decisions.

The battlefield is a lawn. At the far left side of the screen is a house, ostensibly the player's house. The defenses are plants, rooted in the earth. The enemy forces are zombies. These images suggest things about the properties of these game objects, about the rules that govern them. It makes sense to a player that when a plant is planted in the earth in a particular place, it can't be moved. Of course, zombies move slowly. The way the game pieces are contextualized gives the player expectations about how they work.

Super Mario Bros. is a game that initially teaches the player a simple response to any opponent: jump on its head. It would be interesting to have a creature that Mario couldn't just jump on top of like all the others, one he had to carefully maneuver around. But when the player is conditioned to resolve all conflicts by jumping on the creature, how do we communicate that jumping, in this one case, is dangerous? Look at the spiny beetle in *Super Mario Bros.* (see Figure 4.3).

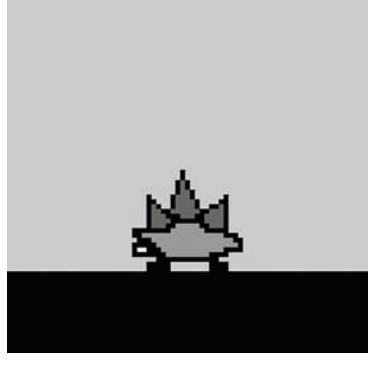


Figure 4.3 The spiny beetle looks a little more dangerous than the other enemies of *Super Mario Bros.* for a reason.

All the other creatures in the game have smooth domes, round noggins, or tortoise shells. Spiny has tall, jagged edges, gleaming and dangerous. This little guy you don't want to jump on.

Compare this creature to another creature in *Mario*'s bestiary you don't want to jump on—the piranha plant that comes in and out of pipes. Its purpose is to create platforms that are temporarily safe and sometimes dangerous. The piranha plant's mouth, full of gnashing teeth, points upward (see Figure 4.4). Mario doesn't want to touch this thing from above.

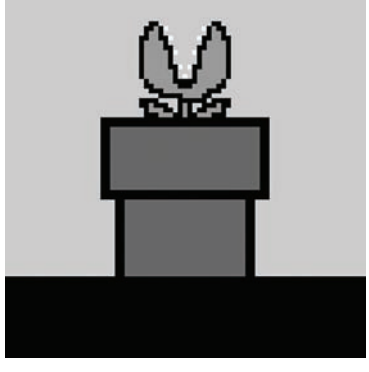


Figure 4.4 The visual context of the piranha plant's gaping maw helps explain why Mario might not want to jump on it.

Interestingly, one of the first creatures the player meets in *Super Mario Bros. 3* (1988) is a Venus Firetrap plant that blows fire toward Mario's current position. Unlike the piranha plant, this plant creature has its head turned to the side, aimed at Mario (see Figure 4.5). As a consequence of where this plant's mouth is facing, the top of the creature is a smooth, round surface. One of the first things I thought to do when playing the game was jump on its head. Mario died, for no clear reason. This is a case of visual miscommunication.

Figure 4.5 In a break with visual conventions of earlier games, the firetrap of *Super Mario Bros. 3* might make you think it's safe to jump on its head. It's not!

Recurring Motifs

Super Mario Bros.' spiny beetle and piranha plant have a similar property: you can't jump on them. They also have a shared visual property: their tops have spikes. This is the kind of symbol that can become part of a game's ongoing visual vocabulary. If the player encounters a distinctive-looking object that obeys one game rule, she'll expect a similar-looking object to obey the same rule later. Visual design in games is all about shaping the player's expectations.

In *Spyro the Dragon* (1998), one of the protagonist's primary verbs is, naturally, to "breathe fire." The player uses this fire to defeat opponents and break open chests full of valuables. But not everything in the game can be defeated by fire. The game uses metal consistently to indicate things that are impervious to fire, such as armor on opponents or certain types of chests. This metal is a visually distinctive gleaming silver. If the player uses fire on it, it heats up red for a moment to acknowledge the interaction between the fire and itself, and then it cools back down to its normal silver. This is usually a signal that the player should try her other verb, a "charging headbutt," on the opponent or container.

It's not just armored opponents and metal boxes that use the visual motif of metal being heated up, however. As the game progresses, other objects appear that react to fire—containers that, when torched by Spyro's flaming breath, expel gems that levitate above them on a geyser of hot air, and switches shaped like fans that turn when heated to open gates. These things are also metal, to indicate that they can't be *destroyed* by fire, but that they *will* change. They heat up just like the metal chests the player first encounters, but the heated-metal motif has developed further; unlike the chests, the gem spouts and fan-switches change in a meaningful way.

Michael Brough's *Zaga-33* (2012) is a strategy game about navigating an alien planet that is generated anew by chance every time you play. The positions of the walls, the terrain, where the monsters appear, the artifacts the players can use—all are random. In fact, the appearance of the useful artifacts is randomized as well. A four-pointed cross can be a laser weapon one play, a healing item the next, or a device that rearranges the walls of the room. Upon using an item, it's identified for the rest of that play—the player knows that the lollipop-shaped thing freezes monsters in place this time. But out of necessity, none of the images for artifacts can convey anything about its use. They're all weird squiggle-shapes.

But there has to be some visual consistency between them, right? The player needs to know that this thing in the corner of the room is an artifact she can pick up and use, albeit one whose purpose she hasn't identified yet. She needs to be able to tell the difference between an abstract-shape artifact and a dangerous monster.

Brough accomplishes this by giving each set of objects a unique, consistent color palette (see Figure 4.6). Artifacts, regardless of shape, are always orange and tan. Monsters are always bright green (with small orange highlights). The walls are green and gray, and the floor is black and

purple. Each element of the game world is identified by its distinct colors. The player may not yet know what the lollipop-shaped thing does, but she knows it's an artifact she can use. Once again, there's a visual motif; in the abstract world of *Zaga-33*, the motif can be as simple as orange versus green.

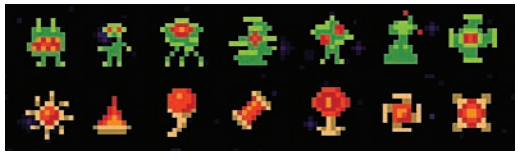


Figure 4.6 Comparison of items and monsters in *Zaga-33*.

Character Design

The way objects in a game appear should tell the player about what they do and what their relationships to each other and to the player are. It should also differentiate objects: objects that follow different rules should be visually distinguishable from one another. An easy way to do this is to pay attention to an object's silhouette, or the shape of the object.

As an exercise, my friend Leon Arnott created his own hacked version of *Super Mario Bros.* called *Silhouette Mario Bros*. He blacked out all the characters in the game, Mario and monsters alike, so that they appear as solid black shapes. And of course, he discovered that everything in the game is still perfectly recognizable. The enemy turtle, which becomes a projectile when Mario jumps on it, is a different shape from the enemy mushroom, which does not. The beetle that is safe to jump on looks different from the beetle that is dangerous to touch. Look at Figure 4.7. Can you recognize these characters?

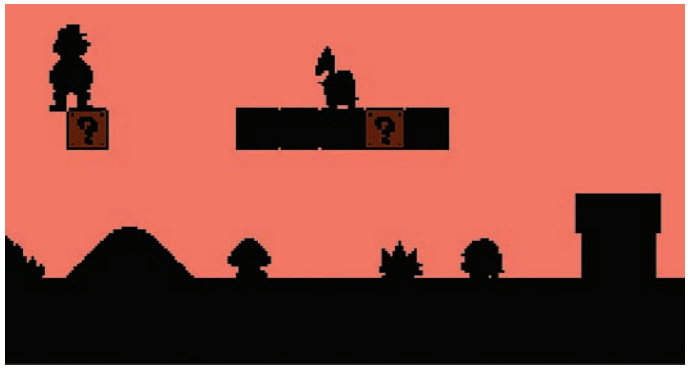


Figure 4.7 Mario characters in silhouette.

The only information the player loses in silhouette is whether a turtle will stop at a ledge or reverse course—the game uses color, green or red, to differentiate that.

Lesbian Spider-Queens of Mars, mentioned in Chapter 3, "Scenes," is a fast-moving game where the queen's renegade slaves attempt to overwhelm her with numbers. It's important that the player can tell at a glance what kind of opposition she's dealing with—whether a slave is an armored slave, who can only be approached from the back or side; a gladiator, who will seek out and pursue the queen; an alchemist, who leaves a trail of fire behind her; or a princess, who's worth a lot of points if the player can catch her before she escapes.

The game uses a few different signifiers to differentiate the silhouettes of the different varieties of slave. First, each has a different hairstyle. The initial, random-moving slaves have mohawks, armored slaves have short bobs, gladiators have spikes, alchemists have straight, neck-length hair, and the princess has twin ponytails that bob when she moves (see Figure 4.8). These ensure that the slaves have different silhouettes even when wrapped in a cocoon by the queen—it's important to the player's strategy to know which threat will return if the queen can't collect the slave before she breaks free.

Figure 4.8 Different slaves of *Lesbian Spider-Queens of Mars*.

Each slave also carries a different weapon—a trivial difference, because they all signify death if the slave manages to sneak up on the queen. The different choices of weaponry just serve to help distinguish the behaviors of the different slaves.

In addition to changing the silhouettes of the characters, the different weapons help to characterize their owners. The mohawk slave's dagger is the most diminutive weapon in the game—the least deadly threat. The gladiator carries an axe: a larger weapon, one that perhaps

suggests that she's a more seasoned fighter and will behave more aggressively. The alchemist carries a sword—longer than a dagger, but less brutal than the axe. She prefers to avoid direct confrontations—her real threat is her fire. And the princess' fencing foil is a class signifier; fencing is about grace and poise before deadly force. The princess can kill but would like to avoid confrontation entirely.

The armored slave carries a long spear and a large shield: the shield provides a context for why she can only be zapped from behind or from the sides. But a shield doesn't suggest harm, it doesn't convince me that she's deadly to touch. As a player, I might expect her to push the queen with her shield before I'd expect her to kill the queen. That's why the long spear reaches past the shield, to ensure that the front of the slave's silhouette is a sharp point, not a flat surface.

Character design also differentiates the slaves from the queen—this is more important than it sounds, because the queen's position is the most important position on the screen. The player needs to be able to identify it immediately. This would be difficult if she wasn't easily distinguishable from all the other moving characters on the screen—and there are usually a lot.

The queen has four legs (when viewed from the side), which makes for a different silhouette compared to the slave with two legs (see Figure 4.9). Also significant is the color. As with *Zaga-33*'s monsters and items, the queen is depicted with a different color palette than the slaves. Slaves are yellow, magenta, and red, sometimes white. (The princess wears white jewelry, and a cocooned slave is wrapped in white.) The queen is red and blue—blue being a much cooler color than any that the slaves wear. The combination of blue and red, in equal parts, also represents the most contrast on the screen—a natural place for the eye to home in on.

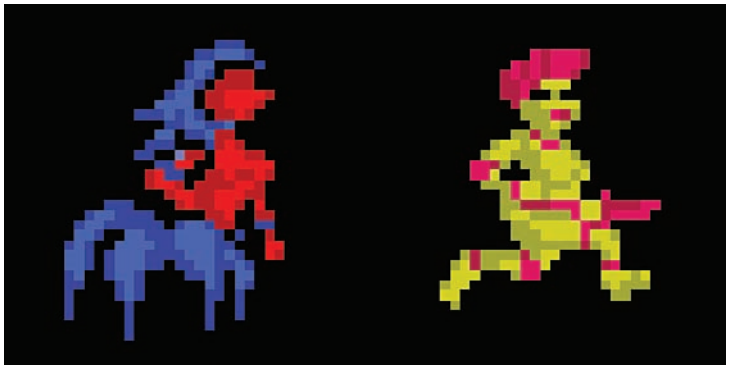


Figure 4.9 Comparing the silhouettes of the queen and slave.

A more subtle difference is that while every slave has a mouth and eyes, the queen's face is blank. She has a nose and shadows that could suggest cheeks, but no eyes or mouth. This sets up a dichotomy that reflects the power dynamics in the game's relationships: the primary use

of a slave's mouth, in this game, is to scream when she's cocooned by the queen. In contrast, the queen's lack of visible emotions suggests control, calm, the visage of a stone bust or some other symbol of a ruler. It's another visual reminder that this character operates very differently from the others.

Animators working outside of games have known these techniques for a long time—to make characters distinct and memorable, you can vary their silhouettes along with other visual cues like color. We have even more reason to use this idea in games, because we need to communicate and reinforce that objects in the game follow different rules.

Animation

Motion can be used to characterize actors, objects, and rules. The armored slave in *Lesbian Spider-Queens* moves more slowly than the others to compensate for the fact that she can only be approached from the back or sides: the player has to do more planning to capture this character. Consequently, she moves differently from the other slaves. Whereas most of the slaves in the game run, waving their weapons threateningly before them, the armored slave moves in a slow, mechanical march (see Figure 4.10). She keeps her shield directed squarely forward to remind the player that she remains unassailable from this direction.

Figure 4.10 Walking animation of armored slave.

Players can tell what's important in *Lesbian Spider-Queens* (see Figure 4.11) because these things, the living things, are moving, while the less important things, the walls and the scenery, are still. Maybe the player's character is moving even when the player's not moving her, rocking on her heels, tapping her foot, or looking at the player expectantly. Make sure the player can find her character at a glance, because her position is the most important piece of information on the screen.

In my game *Tombed*, the player's character Jane is animated even when she isn't actually moving. There's a simple, two-frame animation of Jane when she pushes against a wall (see Figure 4.12), and it serves an important purpose. As covered in Chapter 3, exploiting the crushing spike wall's ongoing destruction of walls and obstacles is important in *Tombed*. There are many situations in the game that require the player to wait until an obstacle is removed by the spikes.

Figure 4.11 A screen from *Lesbian Spider-Queens of Mars*; all the characters are animated, while the backgrounds and the structure of the maze aren't.

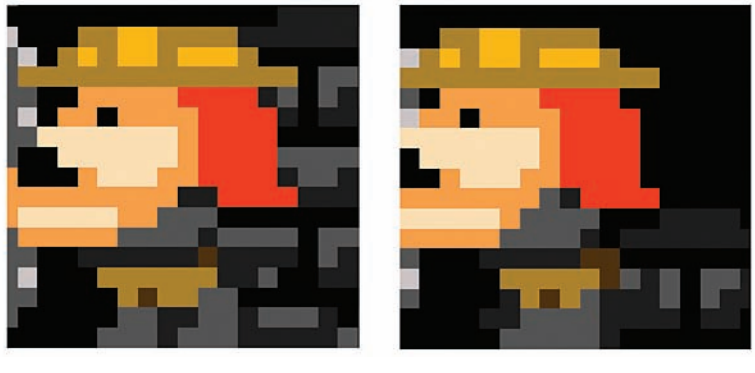


Figure 4.12 Jane animated while pushing on wall.

I also designed *Tombed* to produce a lot of close calls between Jane and the spikes, so there are many situations where, in order to squeak by the spikes, the player will want to start moving as soon as that obstacle disappears. If the player wants to maximize her motion, she'll have to be holding the movement button before the obstacle is destroyed.

If Jane is standing next to a wall, and the player presses the key that moves Jane toward the wall and nothing happens, what does this lack of motion communicate to the player? What she might take from the lack of animation is that the game has not received her input, or that holding the movement key has no effect here, when in fact the time the player gains by holding the key early might be very important.

The pushing animation tells the player that her input *has* been received and *is* having an effect. It communicates.

Motion can be aggressive, cautious, panicked, or controlled. Look at any cartoon and pay attention to the way characters are moving when they're scared, when they're sneaking up on other characters, and when they're giddy or gleeful. Motion can characterize the relationship between objects. In *Berzerk* (1980), the player and a bunch of hulking, broad-shouldered robots try to shoot each other in a maze of electrified walls (see Figure 4.13). Touching the walls means death to either player or robots.

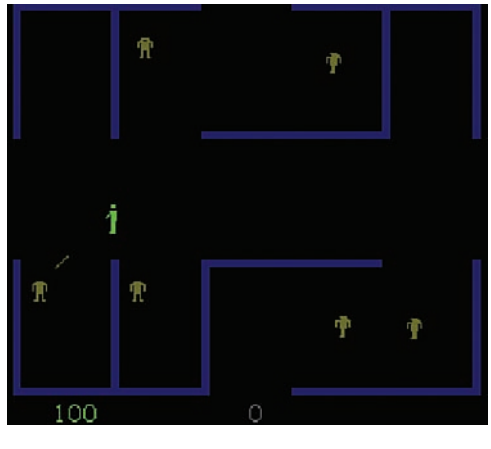


Figure 4.13 *Berzerk* maze, with player and robots.

The mechanism that keeps the player moving through the maze is an invulnerable, pursuing robot named Evil Otto. When the player lingers in a scene too long, Evil Otto enters from the same direction the player did and begins to chase her (see Figure 4.14). Evil Otto is represented in the game by a simple smiley face. But the way Otto moves—a fast, high bounce like a rubber ball as it moves steadily toward the player—tells us a lot about its relationship to the player and to the maze.

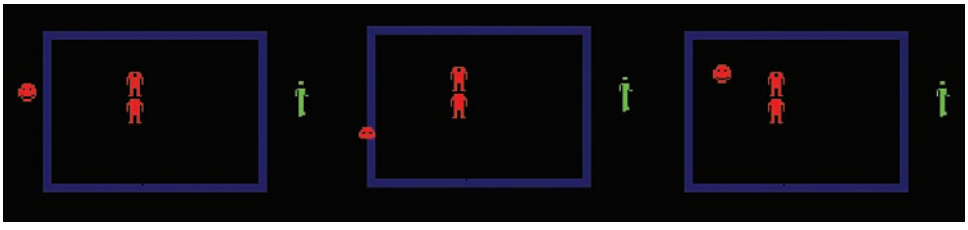


Figure 4.14 Otto chasing the player's character, frame by frame.

Otto's bouncing motion suggests it is above the maze, outranking the robots and the player alike, maybe connected in some way to the maze. More tangibly, its bounce provides an explanation for why the electric walls, fatal to all the game's other characters, don't affect Otto. It is literally above the maze. Otto's slow (though faster as the game accelerates), unwavering pursuit of the player characterizes it as a threat, despite its smiling face.

Simple animations are often enough to convey important changes in state. A common signifier is blinking—making a character invisible every other frame—to convey that the character is temporarily invulnerable, as in a grace period after being hurt. A powerful opponent might flash red when struck to indicate that the player's attack has been successful, but that the opponent hasn't been fully overcome. A weapon glancing off an opponent harmlessly conveys the player's attack has been ineffective. A character meandering randomly might suggest that it's harmless. We have a rich vocabulary of animation to exploit in our games.

Scene Composition

Large images can tell us as much as small images. The composition of a scene can direct our attention to the most important part of a scene. Figure 4.15 shows the final scene of a game called *Labyrinth of Zeux* (1993) by Alexis Jansen. The object of the player's quest, the Silver Staff of Zeux, is present in this scene, and you can see how the entire scene is designed, visually, to push the player toward it.

The T-shaped objects below the Silver Staff are poles the player can ride on. They form an arrow pointing at the Staff, and the poles nearest to the Staff are brighter. The Staff is positioned in the center of an open space, framed by a wide rectangle. The rainbow colors of the walls, the corners of every concentric layer, all point to the Staff. Everything in the screen urges the player upward toward the game's ultimate goal.

The visually striking scene from *l'Abbey des Mortes* (2010) shown in Figure 4.16 is almost entirely portrait: there's nothing for the game's protagonist to do on this screen beyond stand at the window and look out. This tranquil scene, almost empty but for the tops of trees and a sprinkling of stars, represents a moment of peace in the middle of a larger, more hectic adventure.

The protagonist is a Cathar that has been pursued by Crusaders to an abandoned church—death is close at hand. For both the player and the protagonist, this scene represents a break from the rest of the game. The composition suggests just that: a view, a visual reward.

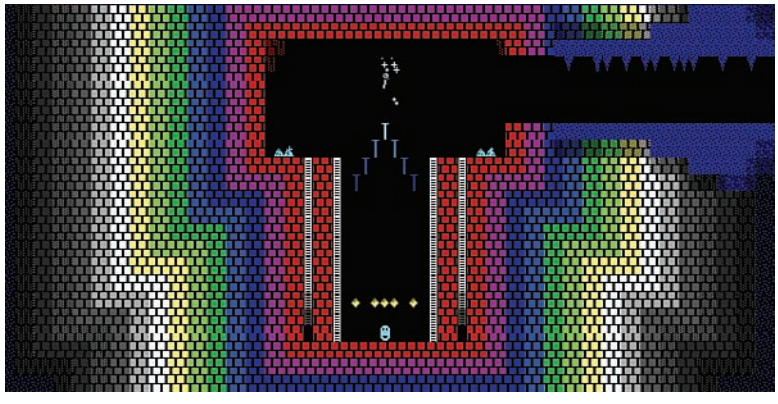


Figure 4.15 Drawing the player's eye to the Silver Staff of Zeux.

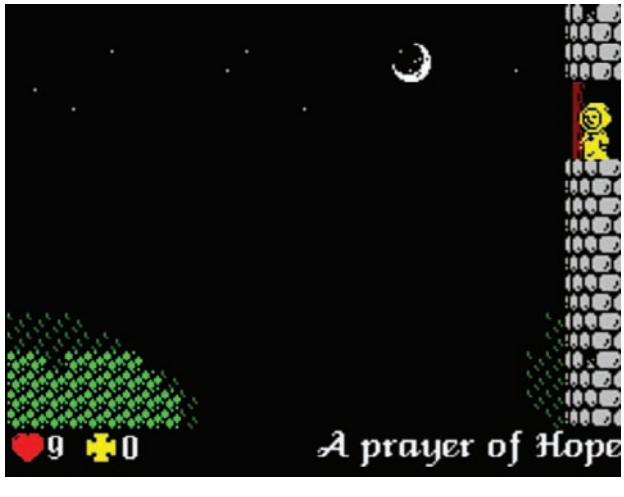


Figure 4.16 Taking a break from danger in *l'Abbey des Mortes*.

The composition of a scene can suggest a lot about its nature. Figure 4.17 shows the first four scenes in Todd Replogle's *Monuments of Mars* (1991). These scenes represent a trip across a barren Martian landscape to the entrance to a strange alien structure, shown at the right side. What marks the transition between Martian crags and industrial construct?

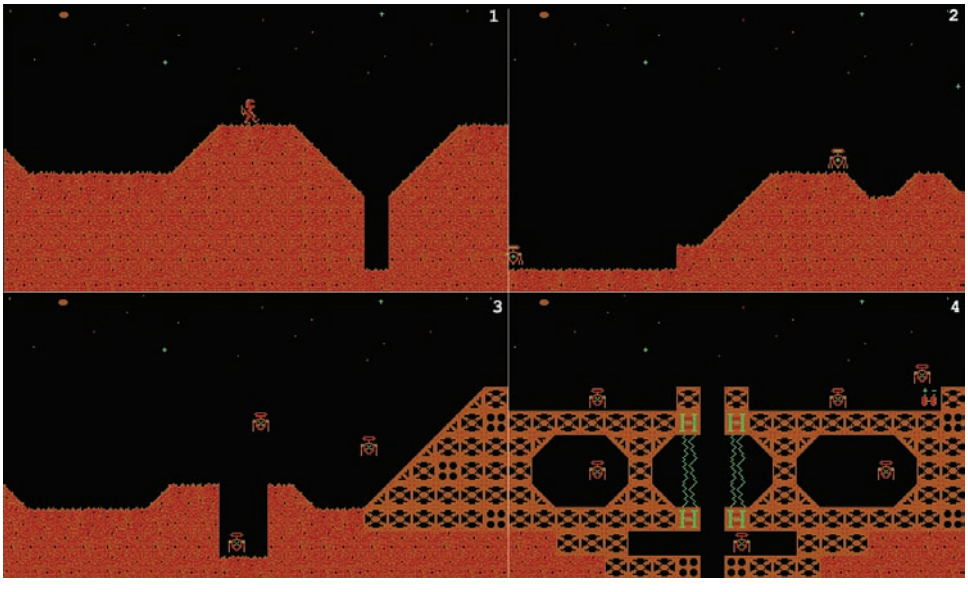


Figure 4.17 First four screens of *Monuments of Mars* show a change in landscape.

Where the Martian landscape ends and the structure begins, there's a shift in the appearance of the shapes, from a rough, pebbly texture to clean, symmetrical girders. But there's also the symmetry of the fourth scene itself, a marked contrast from the irregular surfaces of the previous scenes. Symmetry, here, speaks to the artificiality of the structure: here is something designed.

The robots move back and forth and hurt the player if she comes into contact with them. In the wasteland scenes, they move through open air, through ditches and pits, often more incidental than threatening. At the Martian structure, they move back and forth along the top like soldiers patrolling the walls of a fortress. They're incorporated into the symmetry, moving inside those octagons like pieces of a clock. This is the place they come from, to which they belong.

The horizontal symmetry of the first three scenes points to the entrance to the Martian base, the hole in the center of the fourth scene. The vertical layout of the fourth scene, including two streams of green electricity, guides the player downward into the base. The flat desert scenes suggest lateral motion. This industrial scene, with its symmetry, shouts vertical.

Variations in composition can also say something about a scene. In Loren Schmidt and Mickey Alexander Mouse's collaboration, *Murder Simulator*, the terrain alternates between smooth, straight, symmetrical hallways and rocky, uneven dirt. This suggests an incomplete artificial structure built into the earth, giving a reason for the scene's irregular geometry.

Visual Shape

In the late Fukio Mitsuji's *Bubble Bobble* (1986), players control bubble-spitting dinosaurs (see Figure 4.18). Their goal is to capture every scene's dangerous monsters within bubbles and then to pop the bubbles, destroying the monsters. The dinosaurs can also ride the bubbles, hopping on them as they float through the scene. Bubbles can be popped individually or en masse in one big clump for lots of points.

Figure 4.18 Dinosaurs using bubbles in *Bubble Bobble*.

In a later scene, a new way to destroy monsters is introduced: a bubble of fire. It looks like the other bubbles the dinosaurs spit, but it has a flame trapped inside it. When one of the players pops this bubble, the flame falls vertically until it touches a surface. Then it spreads horizontally, covering that surface with fire that, while it lasts, fries monsters that touch it (see Figure 4.19).

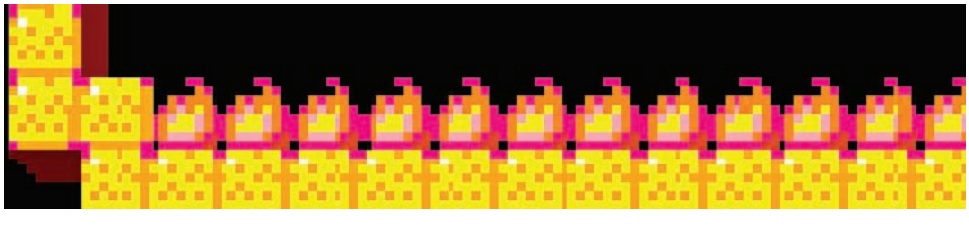


Figure 4.19 The fire bubble spreading flames across a surface.

What is the most effective way to communicate the rules for *Bubble Bobble*'s particular brand of fire? *Bubble Bobble* accomplishes it using an entire scene as a visual metaphor. Figure 4.20 shows a scene of *Bubble Bobble* that has what looks like a giant frying pan. When a player pops a fire bubble, the fire travels down and the horizontal surface of the pan becomes covered in flames—heated, as though the stove had suddenly been turned on.

The monsters that inhabit this scene move diagonally in straight lines, changing direction when they touch a wall. These creatures bounce up and down until they touch the lit surface of the frying pan. Then they pop into the air, as does any monster when it's defeated. That's the source of the scene's inscription, "POPCORN." The visual shape of this scene, composed of the same block objects found in the rest of the scenes, tells the player a lot about the interactions they should expect.

Figure 4.20 A scene acting as a visual metaphor to explain how the fire bubble works.

We've already considered one way to think about a game's shape: in Chapter 3, we discussed how the experience of playing a game changes over time and how each scene gives the player a different range of choices, sometimes wide open, sometimes more narrow. Of course, it's just as important to think about the visual shape of a scene—and what the placement of objects in a scene communicates to the player. These two senses of "shape" can play off of and affect each other in all sorts of ways.

The shapes into which we arrange game objects can determine the way the player thinks about them. In *Chip's Challenge* (1989), Chip the protagonist collects computer chips before finding his way to every scene's exit. He manipulates switches, sliding boxes and moving objects and other pieces to do so.

The "Castle Moat" scene in *Chip's Challenge* doesn't involve searching for chips, but it does have a river that Chip must cross to reach the exit. Crossing the river involves carefully pushing boxes found on the left in the maze. They float when pushed into the water, becoming platforms Chip can walk on. Or Chip can find a pair of flippers hidden in the upper right, allowing him to swim the moat. But how does the scene motivate the player to do all that work?

The important areas of this scene are revealed by the way they are shaped. The area of the scene that contains the exit is shaped like a castle, complete with gate, windows, and battlements. It's a simple arrangement of three objects—a wall, a gate, and an exit tile. But this simple piece of design transforms the body of water around the exit into a moat, an obstacle to overcome.

Chip actually starts this scene on the periphery of the map. To get to the moat and the maze, Chip must walk a path that takes him around the outside of the castle—ensuring that the player sees the exit. While playing the game, the player is only shown a 9x9-square area around Chip at any given time. It's important to have Chip pass the outside of the castle to make sure the player knows where to go. This relates to the idea of camera, which is discussed in the next section.

Camera

Camera in this context refers to what the player can see at any given time. Does she see just part of the world? The whole world? Does the camera move, or stay still? It might move in periodic transitions, or slide over the world freely, or it might follow the protagonist. If the game consists mostly of text, perhaps there is no camera, only the narrator's voice.

The way that the player sees the game characterizes the game world and the player's relationship to it. If she is looking at the game from above, the scenes of the game might feel like a map—maybe one that's being slowly uncovered, as in *Desktop Dungeons* (see Figure 4.21). This camera suggests that strategy will be important. If the camera moves with the protagonist, that provides a closer relationship between player and protagonist. If it doesn't, the player has a closer connection to the world than the protagonist. A camera that lets the player see the entire game from a remote distance or high vantage can suggest that the player is like a god, considering each and every object in the world; in this case, even a protagonist character might be just another small piece that the player can manipulate.

If the camera shows us only what the protagonist sees, from a "first-person" perspective, the protagonist *is* the camera. In this kind of game, the player has a very different relationship to the world. Now everything is not of equal value; what the player's looking at is what's important. And the player cannot look at the protagonist. The protagonist is no longer just a part of the game world; she's the way that the player can perceive the world, the lens she must look through.

Santa Ragione's *Fotonica* (2011) is a game with a first-person camera, but it's a fixed camera (see Figure 4.22). It's a game about running and leaping, and the camera is always fixed on the horizon, even as mountains and hills and other shapes glide by on the periphery of the player's vision. The game provides beautiful things to look at but never lets the player turn her head to look at them directly. The purpose of the game is to run and jump forward, following a track that consistently steers the player's focus toward the center of the screen.

Figure 4.21 *Desktop Dungeons* gives the player an overhead view of each scene in the game.

Figure 4.22 *Fotonica*'s fixed camera moving the player forward.

A common mistake designers make when using a moving camera, most often in a three-dimensional game world, is taking the camera away from the player. The camera flies through the room to show us something the creator has deemed important: the exit, an important thing to collect. This isn't design; it's a failure of design. Stepping in and *forcing* the camera to look at something breaks the relationship between player, camera, and world.

Design has other ways to get a player to look at something. The space should draw the player's attention to the important area. Think about the composition of the player's view when she enters the scene. Is the important thing framed in that view? Does it look important?

There's a story from the "Developer Commentary" to *Half-Life 2: Episode 1* (2006) about a smart solution to the challenge of drawing the player's attention without snatching control of the camera. The scene in question involves the protagonist and his friend trying to escape from a collapsing building. As the characters hurry across a bridge, a helicopter full of enemy troops zooms by below the bridge.

Naturally, the game's creators wanted the player to see this. It develops the game world: the enemy is evacuating its troops because their former headquarters has been shattered. But it comes from a weird direction—to the side of where the player is moving. How did the designers get her to look?

To get the player's attention, the designers put an enemy soldier in front of the take-off site. The game already contains signals to show the player where she's being shot from—the straight, lingering streak of a shot going by or a red glare in the direction from which she's hit. Knowing from which direction one is being shot at is important in a game with so many gunfights. When the player turns to retaliate, she has a great view of the helicopter taking off and zooming away.

Whatever camera our game uses, whatever window the player looks through into our game, design must decide what the player sees. This isn't a place where we're allowed to give up on design.

Sound

So far in this chapter, we've talked exclusively about visual elements. Digital games also have the capacity for audio expression, and it's a powerful tool. We can use sounds to communicate, to underscore the important interactions in our game. A metal "tink" could tell the player her weapon has glanced harmlessly off an opponent. A melody could tell the player the coin she just touched is valuable. Sound is a very different channel than video. It can support visual information, oppose it, clarify it, or confuse it.

Sound as Emphasis

Jeff Minter's *Space Giraffe* (2007) continually bombards the player with visual information, with melting lights and distorting lines. As the player becomes oversaturated with visual chaos, she is forced to depend on the audio. Each enemy makes a different noise when shot, allowing the player to construct a map of the scene and determine what kind of enemy behaviors she'll have to plan for, using a kind of sonar.

Lesbian Spider-Queens of Mars is a fast-moving game. It signals the arrival of each new slave with a different noise. The slow, shielded slave that can only be zapped from the back has a lower-pitched kind of laughing noise, for example. This gives the player an idea of the threats that exist without her having to look away from the danger she's currently concentrating on. *DOOM* is similar: because of its first-person perspective, the player might not be facing a monster when it becomes aware of her presence and begins to move toward her to attack. So each different species of monster has a unique "roar" when it first sees the protagonist and becomes aggressive.

Sound can influence a player's choices. In *Super Mario Bros*., every successive sound made when Mario jumps on an enemy increases in pitch until the player achieves a reward (an extra Mario). A rising pitch creates an expectation: it encourages the player to complete the progression.

Mike Meyer's game *Horse vs Planes* (2012) pulls a similar trick. The player is given a score bonus every time she collects a fruit in quick succession. The first is worth 100 points, the next is worth 200, then 500, to a maximum of 10,000 (see Figure 4.23). There's an element of time pressure, though: if the player takes too long between collecting fruit—two seconds—the bonus resets. The pressure to collect fruit rapidly and maintain the bonus complicates the protagonist's—a horse's—relationship with the antagonists—the planes—that are zooming dangerously through the play area. As in *Super Mario*, the fruit collection noise increases in pitch along the C scale with every increase in bonus points, up to the maximum, at which point the pitch (and the bonus) remain the same until the timer resets (and the pitch along with it).

The pitch of these sounds communicates three things. Most importantly, it tells the player that her actions are having a cumulative positive effect, that she should continue doing what she's doing. It also tells her when the effect has plateaued. Finally, when the bonus times out and resets, so does the pitch of the sound, so when the player hears the pitch drop, she knows the chain has been broken. The sound that accompanies any fruit tells the player, concisely, where in the game's reward structure she currently is. And the expectation of successive notes combined with the disappointment of the pitch resetting creates a strong drive to perform well and not break the chain.

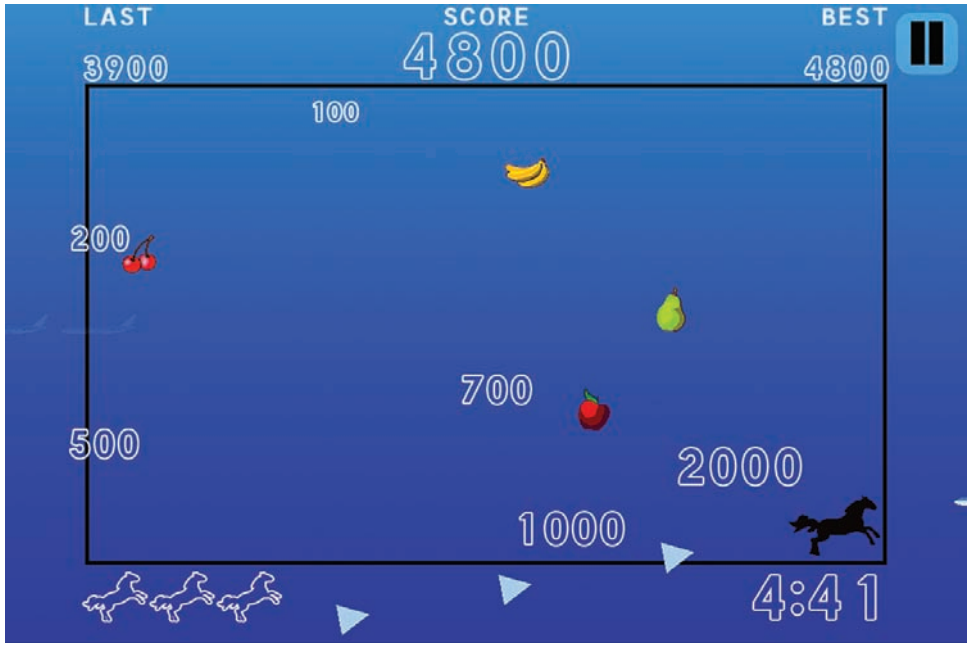


Figure 4.23 A rising point scale for collecting fruit in *Horse vs Planes*.

Sound as Texture

Sound can also be used as texture. In *Dig-Dug* (1982), a little tune plays whenever the player is moving, digging through the dirt. The tune stops as soon as the player stops moving, giving way to silence and the sound of prowling monsters. In *Dig-Dug*, the player is hunting monsters, building passageways to try to goad them into the range of her weapon. If the player isn't moving, it's because she's waiting for a monster to come into range so that she can attack it. But the proximity of a monster is also a dangerous opportunity for that monster to catch the player. The sudden silence, the interruption of the little digging tune, builds the tension. Something similar is done in *Lesbian Spider-Queens of Mars*, also a game where the player alternates between chasing antagonists and lying in wait for them. A little melody plays when the Spider-Queen is moving and goes silent when she's not.

In *Knytt* (2006), the predecessor to *Knytt Stories* (discussed in the next section), a soft scampering sound accompanies any of the protagonist's motions—her running, jumping, and climbing across the strange caverns and crevices of an alien world. This little sound—a gentle rustle in scenes that are otherwise silent save for the sound of wind—does a lot to emphasize the hugeness of the world, the smallness of the protagonist's intrusion upon it. When the player is still, the sound goes away, and the austerity of the planet's near-silence is restored. This sound helps to characterize the relationship between the player and the world.

Fotonica also uses a sudden change in the texture of a scene's sound to great effect. It uses sound to strengthen the player's empathy with the protagonist, to help her better inhabit the game rules. When the player is performing well enough, maintaining a fast enough speed as she runs, a golden haze falls over the screen and the scene's musical score becomes muffled. The running and jumping sounds that accompany the player's movement, always present though normally quiet, now come to the forefront. The effect is something like a runner's high—a feeling that's sometimes called "flow," a concept we return to in Chapter 6, "Resistance." In this condition, the player's speed is greatly increased, and those elements of the game's audio not important to maintaining that speed recede. This change communicates something: the player wants to be in this state for as long as possible, to maintain her speed and to attain this high.

Certain scenes of Shaun McGrath and David Kanaga's game *Dyad* (2012) also use sound to convey the game's rules. A siren begins, faintly at first, and becomes louder as the player's energy approaches exhaustion. Ultimately, it becomes overwhelming, drowning out the original musical score of the scene entirely. The goal is to communicate to the player not only how close the player is to "death," but a feeling of increasing nearness to death, to cause panic. When the player performs the correct action—lancing a target with a special attack, for example—the siren fades somewhat.

Real Talk

Nicklas Nygren's *Knytt Stories* (2008), discussed in Chapter 3's end-of-chapter group activity, is a game-making tool and spiritual sequel to *Knytt*, mentioned in the previous section. I've engaged with *Knytt Stories* both as an author and as a player. Having for several years followed the small community that has arisen around *Knytt Stories*, I've noticed a lot of similar mistakes committed by amateur authors. A lot of these mistakes are of design—scenes that are simply too hard, or built around objects whose behavior is unpredictable—or they're technical—a surface is too jagged, and the protagonist, Juni, gets snagged while trying to climb it. But there are also mistakes of communication: the appearance of a scene or piece of a scene gives the player unclear expectations about the game.

A surface in *Knytt Stories* can look like anything: an author can import her own images into the editor, and those can become walls, floors, ceilings, and surfaces. But the editor comes equipped with 256 sets of graphical tiles, and many amateur authors choose to use images from those tilesets when constructing their worlds. (Handily, a bunch of those tilesets are consistent in style, making it easy to construct a whole world of different terrains that look like they could occupy the same planet.)

Miscommunication can arise when the author uses a tile in a way the artist didn't intend. Often tilesets contain both tiles that are intended as foreground tiles—as walls and floors and surfaces that the player can touch and interact with—and ones that are intended as background

tiles—the texture on the wall of the room the player is in, or a mountain in the distance. In many cases, the artist has drawn the tiles in such a way to distinguish foreground images from background. Foreground tiles are bold with solid black outlines, while background tiles use more muted colors, are partially transparent, or have softer outlines (see Figure 4.24).

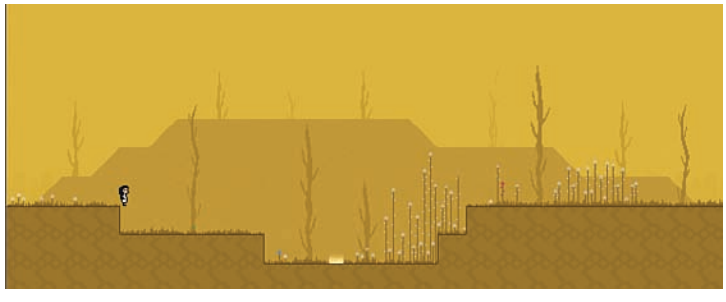


Figure 4.24 Comparing foreground and background tiles in *Knytt Stories*.

A common problem found in games made with *Knytt Stories* is background tiles being repurposed as foreground tiles, as things for Juni to climb on and over. With their muted colors and soft borders, they don't stand out from the background, so the player doesn't always read them as solid, climbable objects. In one particular Knytt Story, I was unable to proceed in a game because a scene required climbing up what I misread as a piece of background scenery. I ran back and searched the entire game world another time, certain that the screen was the back entrance of a one-way passage: you could jump down from the top of the cliff, obviously, but there seemed to be no way to climb up.

Here's another communication problem many authors invent different solutions for: Juni can climb on any surface, so long as she possesses the climbing ability. (The author can give it to her or take it from her at any game transition.) Often it's untenable to allow the player to scale any surface to the very top—it's significantly more work to make lots of extra scenes above the current, for the event that the player chooses to climb up there, and it makes it much harder to constrain the player's movement, to build a path for her, if she can just climb up and over any wall.

The obvious solution is to make each wall meet a ceiling, but it's boring to have each area capped by an artificial ceiling, and it's inconvenient in the case of many scenes. If we want the player to experience this as an outdoor area, how will that come across when there's a ceiling?

The bank of objects in the *Knytt Stories* editor offers another solution: "no climb" blocks that can be placed over any existing game block, which prevent Juni from scaling that block's sides. These blocks are visible in the editor but invisible to the player (see Figure 4.25), meaning that the wall beneath the block can look like anything the author wants. It can look like the rest of the wall, for example.

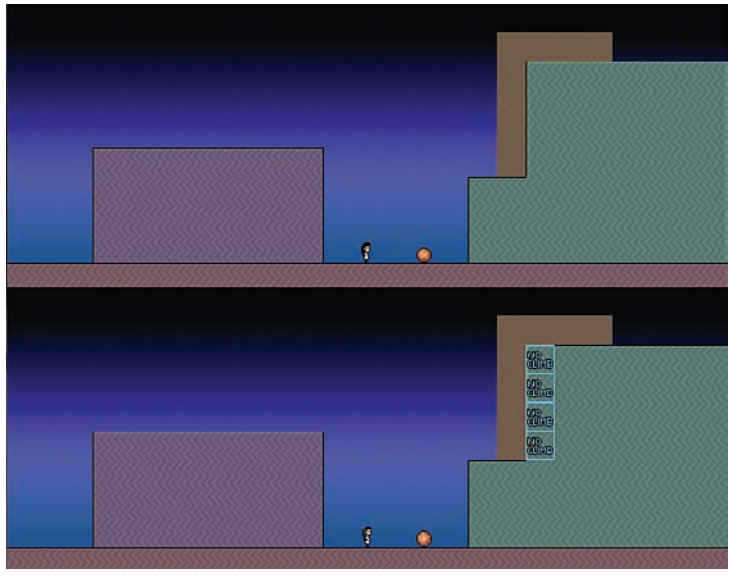


Figure 4.25 "No climb" blocks prevent Juni from going up a wall; they're invisible to the player (top) but can be seen in the *Knytt Stories* editor (bottom).

But it's jarring for a player, scampering up a wall, to suddenly encounter a piece of wall that won't hold her. It can seem every bit as artificial as a sudden ceiling, and, what's more, the player can only discover that a wall is not climbable by trying to climb it. That leads to wasted player time, as the player tries every wall to discover which are climbable and which, seemingly arbitrarily, are not.

What smart authors have done is find a way to visually distinguish climbable from nonclimbable walls. The simplest way involves using another object from the object bank: waterfalls. They come in blocks the size of any other game block and are entirely superficial. They don't affect the rules of the game, but authors have used them to communicate rules. Draw a waterfall over a wall, or over the side that the player might otherwise have expected to climb, and it becomes clear the wall is not climbable: it's too wet (see Figure 4.26). Once the player gets a sense of this logic, she'll be able to recognize, from then on, which walls Juni can climb and which she can't.

Here's a different solution: simply draw different walls, having a wall version that's climbable and a version that's not climbable. One way to do this is by drawing walls that have visible handholds in them and other walls that are smooth. If the player sees handholds, she knows that wall is climbable. If there are no handholds, she expects that the wall is not climbable (see Figure 4.27).

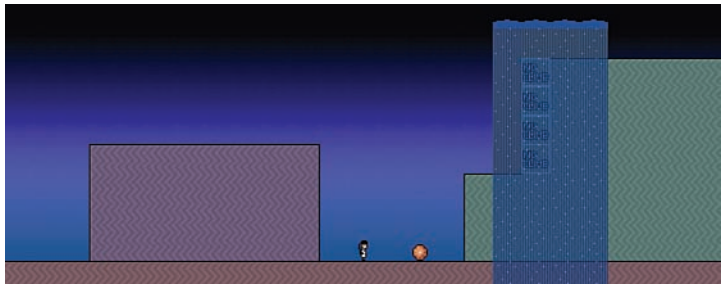


Figure 4.26 Using a waterfall to show a wall is not climbable.

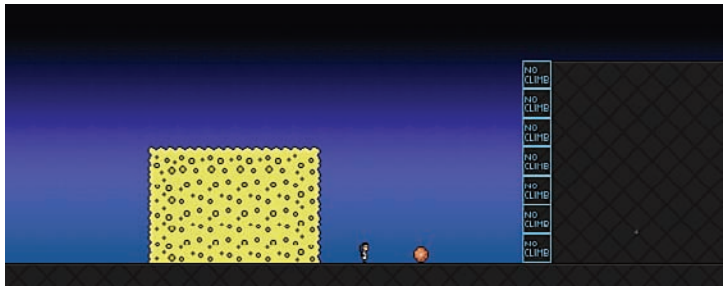


Figure 4.27 Showing walls as climbable with handholds and nonclimbable with no handholds.

There's a similar object in *Knytt Stories*' object bank: the sticky block. If Juni tries to run across a sticky block, she'll be stopped, unable to continue. The only way to move again is to jump off it. Like the no-climb block, this block is invisible to the player, so authors have to use their own methods to communicate to the player which blocks can be walked on and which are sticky. A few authors smear these floors with sticky-looking pink goop (see Figure 4.28). This seems like the best way to describe the rules of these blocks.

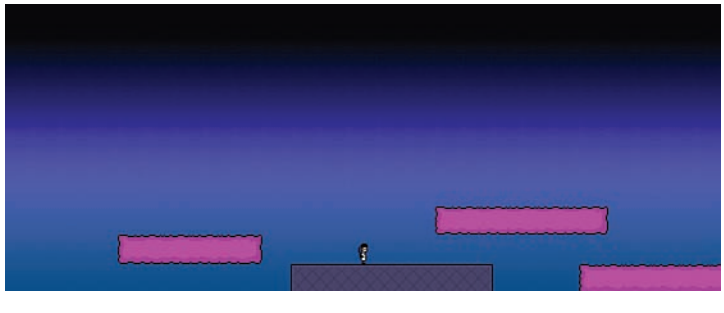


Figure 4.28 Pink goop showing that these blocks are sticky.

Review

- The appearance of game objects can tell the player about their function, their purpose, and their relationship to the player. For example, something covered in spikes is dangerous to touch.
- Recurring visual motifs allow us to develop an ongoing vocabulary of game rules that the player understands. If the player learns that one spiked thing is dangerous, she'll expect other spiked things to be dangerous.
- Character design is useful not just for communicating the rules that govern objects, but also for making them distinct and recognizable. Giving characters unique silhouettes helps the player distinguish them visually.
- The way that things move can be used to differentiate the important from the less important objects in a game. Animation can characterize an object as aggressive, timid, friendly, or dangerous.
- The visual composition of a scene can direct the player's attention to what's important in that scene. Symmetry can suggest importance or artificiality; irregularity can suggest a natural landscape.
- A scene can be an image as well as space. It can be a pretty view or a foreboding visage.
- The way a space is shaped can provide a context for what the player is doing there. For example, battlements create a castle; a circle of water makes a moat. A castle on the other side of a moat can give the player an incentive to cross the moat.
- How the player looks at the game world—the camera—does a lot to characterize the player's relationship to the world and what's in it. Seen from above, the game may look like a map on which the player uses strategy. Seen from the side, it might look like a diorama a character has to navigate.
- A common mistake is to take control of the camera to force the player to see the things you've deemed important. Good design leads the player—and the camera—to the things that are important.
- Sound can communicate information about the game, such as changes in game state (perhaps the presence of characters or monsters). It can indicate success or failure or being on the right track.
- Sound can be a texture, a layer that can be added or taken away to change the player's immediate relationship to the world.

Discussion Activities

1. Think about games you've played recently and find an example of a visual motif: a recurring use of shape, color, or representational imagery that relates to some aspect of how the game works. What did this motif represent, and what did you learn from seeing it appear repeatedly? Were there times when this motif—or a visual that resembled it—was confusing or misused?
2. Can you imagine a scene in this game—or any game—where it might be useful or challenging to the player to disrupt the conventions established by a visual motif? Discuss some scenarios and how they might shape gameplay. As the creator of a game, why might you want to deliberately confuse the player?
3. Name one of your favorite characters from a game you've played or one that you found particularly memorable. This character doesn't necessarily need to be the protagonist of the game; it could be an enemy or another character not controlled by the player. Describe the visual elements of this character's design that stood out for you: the silhouette of the character, use of color, and what various aspects of the visual design represent. How do those elements relate to the role that this character plays—their behavior and the way they affect gameplay?
4. Describe the soundtrack or sound effects of a game that were especially memorable for you. What do those sounds represent, emotionally? What does hearing that sound make you remember, and how do those associations relate to the way the game works or events that happened in the game?
5. Choose a game that you've already discussed, that you used for an exercise from a previous chapter, or that we've talked about in this book. What role does the camera play? Is it fixed or moving? From what point of view does the camera let you see the action? How does this point of view affect your perception of what's going on in the game as a player?

   What would this game be like if it used a different camera with another point of view? If the game's camera shows scenes in the game from an overhead perspective, like the camera in *Pac-Man*, describe what the game would be like if the player saw everything from Pac-Man's point of view. How does the game change?

Group Activity

As a group, come up with an existing game that you're all familiar with. This could be a board game as well as a digital game, but should be something with a recognizable visual theme. *Monopoly*, for example, suggests the blocks of a city, with railroads criss-crossing it.

Design a very different visual theme for this game. Don't change how the game works, but come up with new visual representations for all the elements of the game. Sketch what these new elements might look like. What if *Monopoly* took place in a cemetery or an entire solar

system? You'll probably want to rename the elements and characters in the game as well. When you're done, discuss how the experience of the game has changed. Are there new visual motifs? Are they communicating the same things to the player as they did in the original game?

Now discuss what kinds of changes you might make to the game's system and how the game works. Do you need new rules to help your visual theme make sense? For example, if you modified *Monopoly* to be set in outer space and changed the railroad stations into "spaceports," would it make sense to let players fly to another location from a spaceport? How do you think your rule changes would affect the way the game would be played?

This page intentionally left blank

PART II

CONVERSATIONS

By Naomi Clark

This page intentionally left blank

CHAPTER 5

CREATING DIALOGUE

In any conversation, we need someone to talk with. Without a player, a game is just a set of instructions, whether executed by a computer or human beings who learn what cards to draw on their turn. An unplayed game is like a piece of sheet music: you can see its potential and imagine what it might be like brought to life. You can grasp from notation or rules that it's complex and maybe glimpse its nature. Instructions need someone to carry them out to leap from untapped potential into a living, changing experience. To deepen our practice of playing games, we have to think about our own role in shaping what happens—and understand how our role as game designers intersects and tangles with the choices of players.

Players

My first real player was my little sister. I was around 12 years old when I discovered a digital game that let you design and play your own levels: the Macintosh version of *Lode Runner* (1984). It boasted a straightforward but deep system of climbing up ladders and racing across platforms, collecting bags of gold, running from nebulously defined enemies, and digging holes for them to fall into (see Figure 5.1).

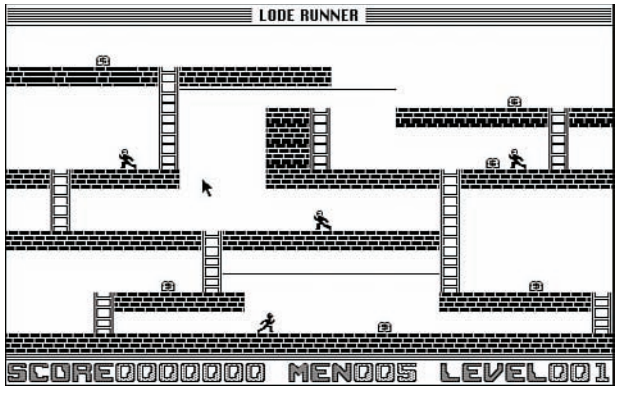


Figure 5.1 A typical level in *Lode Runner*, with the player at the bottom, three enemies, and six bags of gold to collect.

I found the real magic of *Lode Runner* to be in the level-editing mode, which put the dozen or so objects of *Lode Runner* at my disposal. All of a sudden I was experimenting, creating scenes where the hero would be overwhelmed instantly by a horde of implacable enemies, or clamber and fall into a treasure chamber with hundreds of coins. I could create new scenarios that were completely unlike anything that came with the game; I could tell simple stories that played out in a series of twisty, challenging corridors.

When the player has collected every bag of gold in a level of *Lode Runner*, a new object often appears: a ladder that reaches to the top of the screen, allowing exit to the next level. In my own levels, I came up with new ways of using this suddenly appearing ladder. The space of the level would suddenly rearrange, and it would become clear that completing it required getting back across the dangerous level, being chased by enemies, to reach a previously invisible path. Suddenly I was creating plots with turning points!

Even though I could play those levels myself to see how they unfolded, there was something missing: a player, someone else who could experience the dangers and surprises I was crafting. I wanted to express something *to* someone, through this game. I wanted to see how another player would respond and if what I'd done would be clear. So I started using my 10-year-old sister as a guinea pig.

My sister knew how to play *Lode Runner*, and I'd make her sit in my well-warmed chair once I had finished creating a level. I'd tell her, "Go on, see if you can beat it!" She could beat my easier levels without much trouble, and although she had a big smile when she did, I felt disappointed somehow. I could tell that she was smiling in part because she'd beaten me somehow—as if I'd asked her a riddle and she'd managed to outwit me and find the solution with no help.

Before long I started creating fiendishly difficult levels for her to play: they required precise timing and exact knowledge of how to manipulate the movements of each enemy in order to win. These scenarios had lots of hidden trapdoors that looked like ordinary sections of floor but dropped the player right through them into certain death. I orchestrated the behavior of the enemies so that they'd start chasing the player at exactly the moment I wanted.

My sister would insist that these levels were impossible, and I'd smugly show her that they weren't... well, as long as you had exactly the right skill, the correct strategy, if you knew the right path through the scene. As the designer, I possessed all the above, of course. I was thinking more like a player competing with a sibling, though, rather than crafting something for her. I wanted to beat her and see her admit defeat. That's a natural impulse that I've seen play out many times since in games and levels made by kids for each other to play. But creating a system that's practically impossible for anyone but the creator is just a tiny, tantalizing fraction of what we can do when we create games and ask others to play.

I was trying to create a harrowing experience for my sister, something with narrow escapes, unanticipated secrets, and perfect moments where a choice to run left or right made for an instant life-or-death difference. All the pieces were there, but with these fiendishly difficult levels, I hadn't succeeded in engaging my sister, in showing her the magic I was trying to conjure. Eventually, when faced with a level full of tricks that were impossible to understand ahead of time, she rolled her eyes and refused to play.

Creating Conversation

So far, this book has talked extensively about the elements of vocabulary: verbs and objects, the pieces of context that aid in understanding those elements, and the ways those elements combine into scenes that develop verbs and create pacing. In the second part of the book, we look at some broader questions: *why* might you want to pace the development of a particular verb? What kind of story is conveyed when contextual elements, objects, and verbs work together... or against each other? What might you try to say with all that vocabulary? And how might you invite players to say somethivng in response? Do you want to invite players to put their own stamp on your game, or are you trying to convey something that's best understood if a player primarily absorbs and listens to what your game has to say?

We use the vocabulary of written and spoken language to communicate with other people. The vocabulary of games allows us to express ourselves in tremendously powerful ways, saying things with systems in ways that words can't. It lets us create different kinds of dialogue with each other. We're lucky to live in a time when expressive systems—another way of thinking about games—are being explored by creators and players in all sorts of new ways, to converse about and reflect on our every idea.

It's compelling to think of a game as a conversation: players make choices and use verbs within a system. In multiplayer games, these choices can communicate with other players. A single press of a button or move of a chess piece can convey aggression or uncertainty or less obvious concepts that are specific to a particular game. Players who are highly conversant in a system can read the moves of an opponent, whether human- or computer-controlled, and understand what's being said even without words.

As the creator of a game, you also participate in the conversation, but in an unusual and special way. Unlike the times I peered over my sister's shoulder and watched her play *Lode Runner*, you're usually not there to watch your players. Instead, you've facilitated a conversation by deciding many aspects of how it will work beforehand. As a game creator, you craft the particular vocabulary of its conversation, deciding how verbs will develop and shaping the space of possibilities in which the conversation will happen. As creators, we try to shape a space where a good conversation with or between players *could* happen; we hope that players won't throw their hands up in frustration and leave or get bored and drift away.

During a play session of a single-player game—the kind of game that's the primary focus of this book—all the conversation is happening between the creators of the game and the player. It's a tricky kind of conversation to have. As the creator, you have to hope that what you're saying in the conversation—through the rules and shaping of the experience as well as the words, images, or sounds you've added to the mix—gets across and finds a player, somewhere out there, who responds with choices, thoughts, and maybe even interesting strategies and emotional engagement.

This challenge can feel like a gamble, like sealing a letter in a bottle and hoping someone figures out how to open that bottle and understands what you wrote. If you're drawn to creating games—if you've ever felt the spark of excitement that I did when I started making *Lode Runner* levels for my sister—then maybe you have things to say which can't simply be expressed in words, but which could find a compelling form in the systems of a game. Take the gamble! The good news is that in recent decades, many others have gone before you. We've tried, failed, succeeded, and tried again. Despite the fact that we're all still learning exactly how to talk about games, finding words to use and models to think with, creators of games have found a lot of techniques and tricks to get our "letters in a bottle" read.

Iterating to Fun and Beyond

When I first started making levels in *Lode Runner*, I intuitively discovered one of the most pervasively used techniques for refining a game and fine-tuning the conversations that can emerge from it: I got someone to play it, went back and changed it, and made her play it again. Games need players, and as the participants in the conversation who might not be there when our games our played, we need to see people play and hear about their experience. Playtesting and iteration—the process of changing a game based on what you see and hear from the player during play—are the cornerstone of many creators' process. After all, very few composers could create great works of music without ever being able to hear them; Beethoven, who lost his hearing, is the astonishing exception.

We playtest because we want to see a response to determine whether we've succeeded in eliciting the kinds of responses we were hoping for. Usually, the response a game creator is looking for is a smile, a look of intense concentration, the raised hands and lifted eyes that accompany a feeling of victory—all the hallmarks of someone who's really into what's going on and having fun. Playtesting lets us spot the barriers to reaching that place and then think about ways around those barriers. The barriers might include confusion about how to use a verb or pacing that's too difficult for the kinds of players you're hoping will play your game.

"Fun" is the most popular and traditional goal that game designers try to reach, however. Think about the metaphor of conversations again: talking with others, especially your circle of friends or other like-minded people, has often been described as one of the most consistently engaging and pleasurable things in life. That doesn't mean that all conversations are fun. Some are deadly serious, even if they're hard work to stay engaged with, and some conversations are necessary to convey important ideas. More and more, game designers are finding that fun is just the traditional role that games have played in society. We have to remember that it's what most players *expect* of games still, but there's a huge variety of other kinds of system-driven conversations that remain to be explored.

Papers, Please (2013) by Lucas Pope doesn't try to present itself as a straightforwardly fun game. It tells you that you're going to work: you play an immigration inspector, checking and stamping the documents of hundreds of would-be border-crossers (see Figure 5.2). You're employed by a harsh, totalitarian regime that tramples on rights and demands your diligent and detail-oriented assistance in exchange for a meager stipend to keep your family alive. The scenario is grim and mind-numbing, and so is the gameplay: you're literally inspecting paperwork for discrepancy, expiration, and forgery and stamping it APPROVED or REJECTED, over and over. For each mistake you make, you're penalized, which could make a life-or-death difference for your inspector's family.

Figure 5.2 Stamping a seemingly endless series of border documents in *Papers, Please*.

This may not sound fun at all, on the surface—but *Papers, Please* manages to thoroughly express the workings of an unjust system that you find yourself trapped in when you play. You've got to decide whether to prioritize helping mistreated and threatened border-crossers or preserve your own family's health and wealth. The shape of the game—the difficulty and balance of costs and payment—always holds out the possibility that if you're good enough at your job, you can get away with some purposeful "slip-ups" to help people. Just as surely, your power to act is limited by the fact that you're only one cog in the machine.

Lucas Pope playtested *Papers, Please* extensively to fine-tune the workings of the game's fictional injustices. As one of the participants in the web forum where he posted early versions of the game, I took part in that process and saw the game get better at eliciting the kinds of feelings and experiences he was aiming for. Do all games benefit from playtesting, though? There's an argument that can be made that the goal of some games is less about persuading the player to respond, feel particular things, or make certain kinds of choices, and more about expressing something that the creator wants to say—regardless of whether a particular player is willing to hear it.

When we playtest and iterate a game, we make changes that attempt to adapt the game's form and the possible spaces that can emerge from it to the psychology and behavior of players. If we're making a game that's intended for young children, for example, we might change the controls so that they're easier for players with less developed reflexes and motor skills, or we might adjust the difficulty of the game differently than we would for an experienced gamer. We move the game away from purely being about our own expression to adapt it for an audience.

That's not necessarily a bad thing, of course, but it means changing what we're saying or how we're saying it through game systems to attract, retain, or persuade players into hearing and engaging.

Your Conversation

What happens when game creators simply put their thoughts out there in an expressive system and ask players to listen without compromising or adapting? What if a game is trying to express something real about the creator's life? As mentioned in Chapter 1, "Language," Anna's game *dys4ia* (2012) reveals her own experiences of taking hormones through dozens of small systems; it asks players to help unfold that story, piece by piece. *dys4ia* is a game that's less about players choosing what happens or expressing themselves and more about a kind of listening through interaction to understand a kind of life experience that most players don't share.

Telling and listening are part of conversations, too. Sometimes it makes sense to rest our active responses and simply *hear* what the person who's talking is trying to say and understand what they mean in the stories they tell—or the systems they build. Games can present us with overt choices and ask us what we think; they can also show us that in some circumstances and systems, choices are limited or don't necessarily make a difference. For example, as a single immigration inspector in *Papers, Please*, you can't help every single person cross the border. When you play *dys4ia*, you can't change the course of Anna's life or experiment with the system to see what would happen if she stopped taking hormones or reacted differently to emotionally trying circumstances. It's part of the story of her life, and it recounts through its systems what's already happened.

When you go into a conversation, you help shape how it'll evolve and turn out. Conversations can be polite and formal or raucous and free-wheeling; the same is true of games. As the creator of a game, even if you're not present when it's played, you'll make many choices that determine and limit what might happen in the conversation of play. Games can present us with overt choices and ask us what we think—like an interrogator demanding answers or a friend posing questions to help us understand how we feel. What would you do in a difficult situation? What kinds of choices would you make when faced with limited resources? We can also create wider spaces within games where we invite players to come up with their own strategies, reactions, and explorations into territories that we might never have anticipated as the creators of the game's vocabulary. Or we can limit those spaces and ask players to listen—to understand that not every system is open to being changed through the agency of players, not every story can be diverted toward a happy ending, and not every difficult challenge can be mastered and conquered.

These are all different ways of communicating through games, and they raise all sorts of questions. What kind of space do you want to shape? If you have something you want to say, how do

you get that across in a way that feels honest and true to players? How do you decide when to try to adapt to players' expectations and psychology to try to elicit feelings of fun or persuasion, and when do you stop doing that in favor of holding on to your own expressions and just ask players to listen? If you're inviting more open contributions to the conversation from players, how do you help them become conversant enough with our vocabulary to say something interesting in reply? Can we create space for a player to tell their own stories and express themselves in the space of a game, while also conveying what we have to say?

The brightest and most passionate game designers in the world continue to struggle with these questions because it's exciting to explore a space with so much possibility that remains untapped. Although there are no definitive answers, the next few chapters share plenty of ideas about and around these questions. Maybe you'll come up with some of your own answers.

Twenty years after I started experimenting with *Lode Runner*, I had a job designing games *and* another 10-year-old sister in my family. When I went home for the holidays one year, I brought my youngest sister one of the games I'd been working on. She was delighted and played it for weeks, mastering the intricacies of its system. She talked to me about it, asked me for help, and showed me her strategy. Inside the game, around it, and beyond it, we had a conversation.

CHAPTER 6

RESISTANCE

Difficulty is one of the oldest ways to look at a player's journey through a game. A novice player usually starts with simple challenges: learn to jump over this obstacle, understand that pushing the joystick to the right will move her avatar to the right. Even in multiplayer games like chess or golf, a new player will often take on easy opponents: other novice players, or skilled players who are "taking it easy" on the novice by playing with a handicap or deliberately playing below their level of skill. As the player masters some simple verbs, she's faced with more difficult challenges.

Push and Pull

As a beginning designer making *Lode Runner* levels, I had a naive idea of difficulty: *harder is better*, and the ultimate challenge of playing any game is to *master the hardest challenges*. It's an upward narrative of progress and increasing conflict, the same kind of story we find in many heroic narratives of literature or film. At the peak of difficulty, there's an epic battle. On one side, there's the player, with everything she's learned. On the other side, there's the game's system at its utmost, wielding a climactic scene that the designer of the game has made to "throw the kitchen sink" of possible challenges at the player.

Difficulty can be compelling and dramatic: the player starts off easy, learns and deepens her understanding of the game's possibilities, and climbs through increasing challenges to master the system. Overcoming difficulty is deeply appealing to us as human beings for good reason: it can give us confidence in our own ability to learn and even master difficult aspects of our lives.

In earlier chapters, we looked at how verbs can develop in relation to objects and other verbs. These elements of vocabulary are the building blocks of a conversation that players have with games we create, a conversation that we enable and shape by developing the game's vocabulary. In this chapter, we discuss how ideas about pacing and development can be applied to the entire experience of a game, from the start through the middle and toward the end—assuming the game even has an end!

When we think about games as a conversation, we can discover many potential ways of looking at games. After all, not every conversation needs to be about challenging the participants, even if many important conversations *are* challenging. In a conversation, challenge can mix with pauses for reflection, times when we listen quietly, and statements of support and reassurance. Conversations are about *push and pull*: one person says something, and the other person listens and responds. At times we challenge each other, and at other times we allow another's thoughts to explore and develop. A good conversation isn't necessarily led by one person either; some or all the participants have ways to voice their own input about the pace and goals of the conversation.

We can find ways to do all these things with games as well, in the unique ways that conversing through a system can create. As the creators of a game, we can shape the ways that the player can push and pull through the game's system. Verbs are a great example of how a player can take an action and push into a game. We can share decisions with the player about how the push and pull of its conversation evolves—even the purpose of the conversation.

Resistance is another way of thinking about the push and pull of games. When a player uses the verbs at her disposal, she pushes against the game to see what will happen, and the game responds. As discussed in Chapter 3, "Scenes," when the player of *Tombed* uses the "dig" verb against a metal section of floor, causing Danger Jane to hit it with her shovel, the game

responds with Jane's digging animation and a metallic "ting" sound, but nothing else. The metal block does not give way but resists the verb. In that single moment of gameplay, the game has responded to the player's push by pushing back and providing resistance.

In longer stretches of time than a single moment, the player may try many different ways to push into the system of the game: perhaps using the "dig" verb in different circumstances, or combining digging with left or right movement to drop Jane further into the vertical column that comprises the space of *Tombed*. The player may develop strategies to deal with the different scenes that follow in succession, developing her understanding of when and how to use verbs—including the "un-verb" of simply *waiting* for the ceiling to descend and destroy metal objects—so she can keep playing and reach the bottom.

The player of *Tombed* will also think about the goals presented by the game and her own goals in playing—the aspects of the game that pull her forward as she pursues them. She'll have to reconsider how to reach those goals after she finds that Jane gets crushed by the descending spiked ceiling and falls off the bottom of the screen, followed in turn by the game resetting itself to an earlier state. This is a different kind of push from the game, declaring that the player won't be allowed to proceed if the spiked ceiling contacts the top of Jane's hat. The player must decide how to respond and if she wants to keep pushing. Does the player want to win? Then she has to find ways to push when the system pushes her back.

At each turn, the player pushes in different and increasingly complex ways, and *Tombed* pushes back: always applying pressure with the unstoppable descent of the spiked ceiling, but also with the changing objects that make up each scene, pushing the player to find new ways to use verbs and keep descending. Finally, *Tombed* stops pushing when the player reaches the bottom of the shaft. The oppressive ceiling disappears, the player uses the "dig" verb one last time, and the game ends.

Tombed is a straightforward game in many ways. It has a few different verbs and can be played from beginning to end in under three minutes. Even so, the player must find many different ways to use those verbs and push to reach the end. *Tombed* was designed and paced to push back in different ways as well, sometimes giving the player a longer span of time to consider her decisions, and sometimes demanding that she act immediately. Sometimes she's allowed many choices, and sometimes very few.

Flow

Back in Chapter 1, "Language," we made fun of the word "flow." It's a term that's often used by game designers to talk about difficulty, pacing, and challenge in games, but sometimes "flow" is tossed around so freely that it becomes a substitute for "fun" or "quality"—as if flow is a magical substance needed to keep players captivated by your game.

Flow is part of a psychological theory, first proposed by Mihály Csíkszentmihályi; it describes a state of focused motivation where someone's so involved and energized by what she's doing that she becomes completely absorbed and caught up in it. This state of flow is similar to colloquial ideas like "being in the zone." It sounds like a wonderful thing; understandably, many game creators want as much flow as possible in their games. Flow doesn't just come out of nowhere, though. Much has been written about flow, but most of what's useful for making games can be summarized in three elements that Csíkszentmihályi says are necessary for flow to occur.

The first condition for flow is a situation with goals and a participant who can take action to make progress toward those goals. Luckily for us, both these things are fairly common elements of games. The second condition for flow is feedback: the person experiencing flow has to see what happens as she tries to move toward her goal and be able to adjust her actions to respond to changing demands. If this sounds familiar, it's because feedback is exactly what we've been referring to as resistance. Flow is just one way to talk about what happens when the objects, verbs, and resistance of a game develop at a particular pace that encourages a player to stick around for more of the conversation.

It's not enough to simply give players feedback in response to their actions. The third element of flow is that demands on the player's choices and actions *must* change and evolve over time. At first, figuring out how to use a game's verbs to jump over a wall might be an interesting goal with feedback. The player figures out when to jump, and the game shows her that she made it over the wall. Now imagine repeating that action. If she had to jump over the same wall in a modded version of *Super Mario Bros.*, at the same interval, for ten minutes on end, it would become tedious. It would turn into a test of patience more than anything else, and it potentially would feel like a waste of time (see Figure 6.1).

Figure 6.1 What if your avatar had to tediously jump over a long series of walls?

The simplicity and lack of evolution in repetitive, already mastered tasks results in *boredom*, one of the two pitfalls that disrupt flow. On the flip side, flow can also be disrupted if challenges are too difficult before the player has enough understanding and mastery of verbs to overcome them. If the next challenge after jumping over a simple wall involves a highly developed use of the verb that requires a lot of timing, the player may fail over and over again. She may end up feeling like her attempts are futile. This results in *frustration* and, like boredom, it can feel like a waste of time. The player feels stuck "doing nothing" rather than continuing to move through a flow-inducing series of evolving choices, actions, and challenges.

In terms of resistance, boredom is what happens when a player isn't being pushed by the game system to do anything except repeat an action she already knows how to push with. Frustration can be similarly repetitive, such as a player pushing into the conversation of the game and being told, "No, that's not it, try again" over and over again. Although resistance is happening, it's stuck in a loop.

A commonly expressed idea about flow and games is that as designers, we should try to stick carefully to a channel between boredom and frustration, like a shark swimming between dangerous rocks on either side. Also like a shark, the challenge of a game in this model has to keep moving, so that repetition of actions that the player's starting to master doesn't get boring. Get the difficulty exactly right, and the player will stick with your game, developing more and more skill. The game then needs to respond in new ways, pushing back by providing the player with ever greater challenges. This upward ascent resembles a slope toward the maximum possible challenge (see Figure 6.2). It's similar to the narrative of difficulty mentioned earlier, an uphill battle toward an epic conflict. Unlike the simple idea of "the most difficult is the best," however, thinking in terms of flow lets us focus more on the process of this journey. All along the way, the game must keep evolving the system to provide more difficulty so that the player will stay engaged until she reaches that pinnacle.

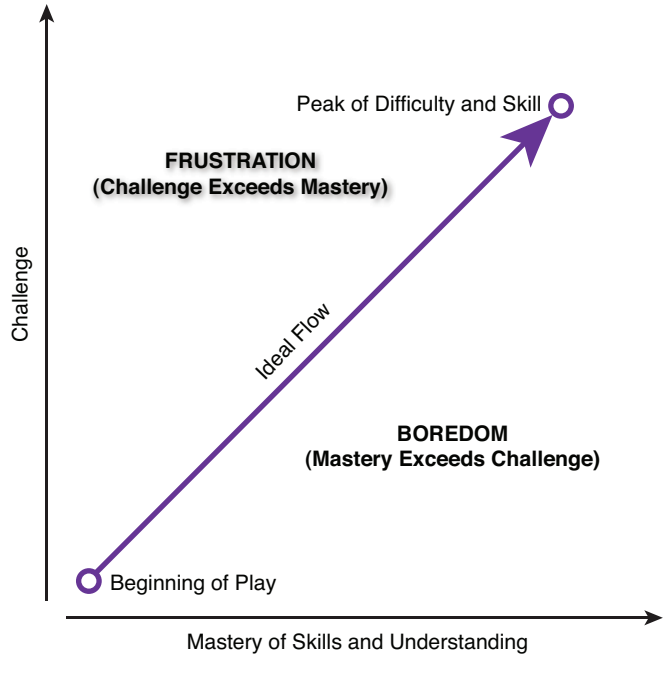


Figure 6.2 For some game creators, the ideal experience involves staying in the zone between boredom and frustration as the player's skills improve.

The channel between boredom and frustration is an ideal path, like a perfect model that many games strive for. In a game with perfect flow, the player would push and be pushed back but would be so engaged in what's going on that it would all feel seamless, natural. Some games are good at finding this channel—even if they don't start there at the beginning of a player's experience!

Super Hexagon (2012) by Terry Cavanagh is an interesting example. To play, you simply use the verbs "rotate clockwise" and "rotate counterclockwise" to keep the arrow you control from colliding with a series of walls closing in from the outside of the screen (see Figure 6.3). The player has to rotate the triangle to go through the gaps. At the beginning, this is an incredibly difficult task, and a player is likely to die by colliding with a wall almost immediately, making game sessions last less than ten seconds. At first, this seems like a clear violation of the "perfect model" of flow, but *Super Hexagon* uses a simple enough system that it doesn't need to start off slow and easy. The player learns what to do by colliding with walls, over and over again. Because these early sessions are so short, it's easy for the player to jump in again, grasp the patterns of walls that close in on her, and hone her reflexes.

Figure 6.3 *Super Hexagon* dares to start off super-challenging.

Before long, many players will improve—and notice that they've improved, since their game sessions (and "longest time" records) will be getting longer. This kind of motivating feedback is essential for flow, but it's worth noting that *Super Hexagon* doesn't start off at the bottom-left

corner of a flow diagram—the kind of very easy, no-skill-required experience that often involves a tutorial that holds your hand or practice levels that go easy on you. Instead, it drops the player into the frustration of the game like a skier descending a steep slope (see Figure 6.4) and lets her figure out through short bursts of intense play that once she starts to get the hang of it, the challenge will become manageable. That steep slope may even be part of why getting better at this game feels so exciting. *Super Hexagon* shows us that not all games have to adhere to or strive for one model of flow. Following the "ideal" channel from bottom left to top right is just an idea that's become traditional for many game creators.

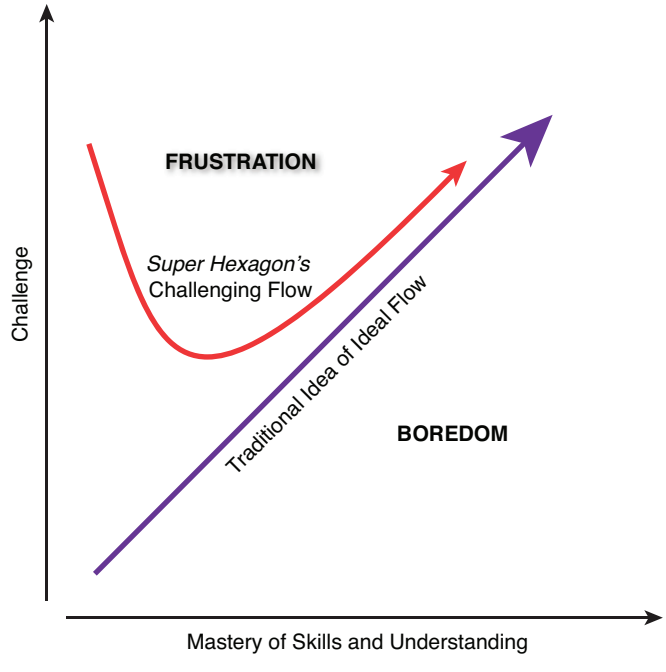


Figure 6.4 If a player isn't put off by the difficult beginning, finding the flow of *Super Hexagon* can be a thrilling ride.

Instead of a straight line running from bottom left to top right on the flow diagram, the experience of many games involves a zigzag path. A game will present a new challenge, like a more difficult kind of jump, a new verb like "shoot" that has to be used in a different way (for instance, timing your shots so that they don't miss), or a combination of verbs, like "jumping" and "shooting "at the same.

The player has to figure out how to master this new challenge. It's a process that often feels frustrating at first as the player learns how to deal with it, especially if she doesn't get it right on

the first try. As she masters the new challenge, the push of frustration lessens. Repeating the same action again and again drifts toward boredom, creating a zigzag. Of course, not all players are the same: some might master a verb or combination of verbs quickly, especially if they have experience from other games, while others may spend longer being frustrated. The purple line in Figure 6.5 shows the traditional idea of ideal flow, with a frustrated player following the red line and a player who masters challenges easily following the blue line.

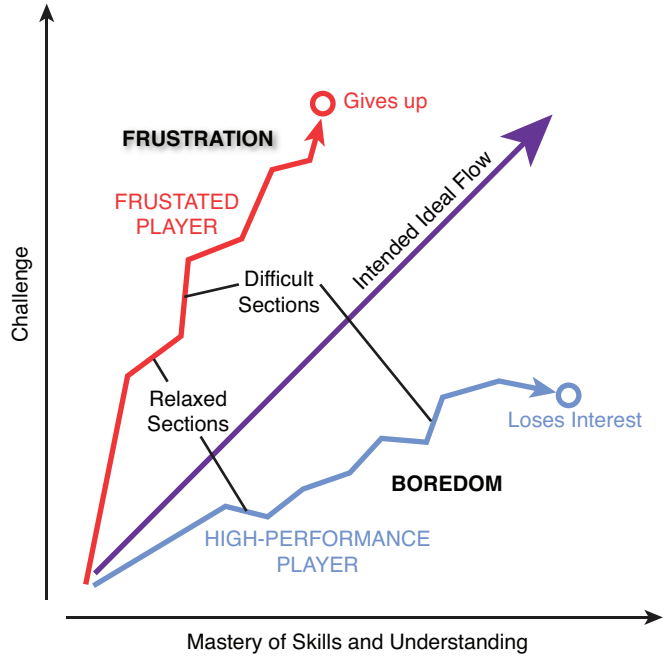


Figure 6.5 Same flow diagram but with zigzag lines for different players.

In shaping the conversation of their game, game designers have figured out how to make this zigzag pattern part of a story that's told through play. A moment of intense challenge that requires the player to use the verbs they've been practicing in previous sections might involve fighting a boss, for example. The visual and audio cues that accompany this moment might include a larger graphic to represent this dangerous obstacle, with music or sound effects that convey an ominous or climactic feel. Before and after this moment, the context isn't as intense, and neither is the challenge: the player can relax and prepare for the next big moment, following an arc that builds up to the next conflict (see Figure 6.6). We discuss more ways to create these kinds of pauses and *plateaus* (where the line of flow becomes more horizontal) throughout this chapter.

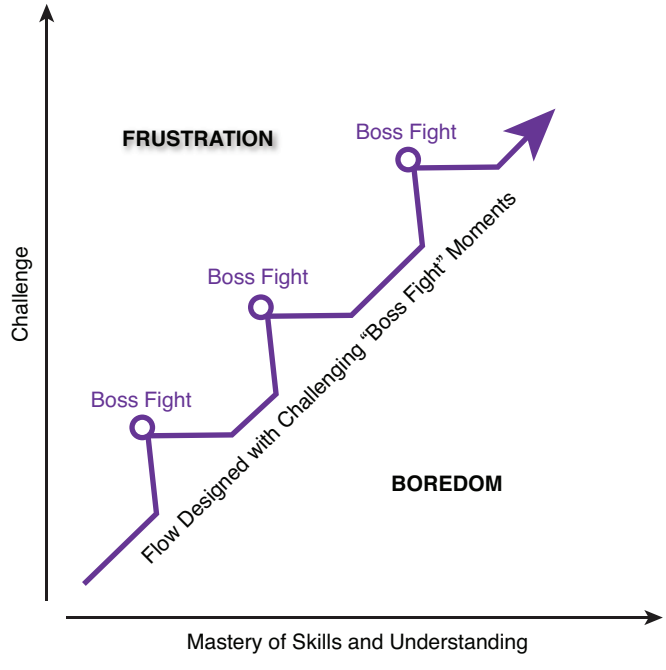


Figure 6.6 A zigzag flow diagram with boss fights dipping into the frustration zone.

Adjusting Difficulty

Games like *Super Hexagon* require the player to deal with frustration and failure and commit the time to overcome hard challenges. Competitive multiplayer games have a long legacy of putting this responsibility in the hands of players—in these games, not just a single player but a pair of competitors, or even a group or community of players who play together. Players who enjoy sport-like digital games such as *Hokra* (2011) or *BaraBariBall* (2012) have to teach newcomers how to master the challenges of the game, growing a community of players so they have more opponents to face. The difficulty in these games comes largely from how good your opponent is. Players can take it easy on beginners or play with deliberate limitations (or handicaps) to help them learn.

Single-player games face a different kind of problem because the player is alone in conversation with a system that can only say as much as its designer has allowed it to. Even so, it's possible for the creators of a game to reveal some of what's going on in the system and give players control over whether it offers more risk of frustration or boredom.

One common way to provide this control is to have the player select a *difficulty mode* at the beginning of the game. The player chooses whether she wants an experience that starts off challenging and evolves to become even more difficult, or one that's easier—potentially to the point of boredom. Creating more than one way to pace the same game system can be difficult, however; one mode often ends up being perceived by players as the "real game." Often this is the most difficult mode, especially for players who value skill and mastery. In addition, giving players this choice at the beginning of the game, before they understand the kind of resistance that the game offers, asks them to guess which difficulty setting will be most satisfying for them. What if a player is good at one aspect of the game but not others? Some games offer detailed controls for adjusting many aspects of difficulty, but there's still a paradox: to understand how all those controls will affect a player's experience, the player first has to learn how to play the game well enough to grasp her options.

In the early decades of digital games, the audience of players was relatively limited. Not only were most self-identified gamers white, male, and well-off enough to have steady access to computer technology (or at least quarters to dump into an arcade machine), but gamer culture and the systems it produced were focused on difficulty, challenge, and mastery. By the turn of the millennium, things had already changed a lot, and the game industry launched a new wave of "casual" games. These games were targeted toward players outside the usual suspects, many of them women, girls, and older people who weren't part of the earlier eras of game enthusiasm. Casual games were known for being much less punishing and intensely difficult than games of earlier decades, and for bringing a much larger segment of the population to gaming. Gamers who had less experience with and fewer preconceptions about a particular kind of system are even less likely to grasp intuitively whether they want to play the "Hard" or "Easy" setting. Ever since the "casual revolution," game creators are more likely to ask, "Who is this game for?" and sometimes, "How can we make this game more fun for more people?"

Game developers have been searching for many years for ways to seamlessly mold the resistance of a game to match each player's abilities rather than create a kind of flow experience that works for some players but frustrates or bores others. Many of these attempts fall under the concept of *dynamic difficulty adjustment* (DDA): methods of adjusting the rules and resources of a game to help players who are struggling with a game's resistance, and increasing the challenge for players who are doing very well. When you're playing the first-person shooter *Half-Life 2* (2004), you'll occasionally come across crates that contain helpful items to replenish your health points or ammunition (see Figure 6.7). If you're well stocked with these resources, you'll find fewer items inside a particular crate, but if you're doing poorly and running low on ammo or health, that same crate is more likely to contain items that will help you replenish those resources. Many players of *Half-Life 2* never notice that they just happened to get a more powerful healing item when they were running low on health; designers who use DDA extensively, helping players who struggle while increasing challenge for others, often try to do so subtly.

Figure 6.7 If you're running low on ammunition in *Half-Life 2*, this crate is likely to contain some to help you out.

Much more egregious examples abound in games. In many racing games, for example, your opponents will actually drive faster if you're in the lead and slower if you're trying to catch up to them. It's not hard to understand how this kind of adjustment keeps the game interesting in the service of flow, but becoming aware of how strangely fluid the behavior of the competition is can be jarring. It might even make the player feel like her own abilities and struggles don't really matter, because the reality of the game world will be adjusted based on the player's situation.

Subtlety is necessary in DDA because of how it changes and manipulates the conversation between player and game. When the player pushes against the game and doesn't manage to make a difference or she meets an expected goal, the game pulls back its own resistance; when the player pushes forward successfully, the game's resistance increases as well. Learning a game through your own ongoing conversation with it is a process of exploration. Exploring a system with an intense amount of DDA is like having a conversation with someone who's changing her mind constantly based on what you're expressing.

Used bluntly, DDA can give the resistance of a game a mushy feeling, as if there's no fixed structure that the player can meaningfully encounter and push against. Used subtly, DDA may go unnoticed by players, but it's still quietly manipulating the shape of resistance to create the

smoothest experience of flow, rather than presenting some unmoving challenges to the player and letting her decide how to overcome it—or simply stop playing. It's no wonder that many creators of smaller games in recent years avoid DDA and simply let their game systems function without constant adjustments and modifications. Would *Super Hexagon* be a better game if Terry Cavanagh had designed it to get easier when the player inevitably fails at its extreme challenge? The shape of that game's relentless resistance to player effort would become different, more malleable—and perhaps less meaningful for players who are willing to throw themselves again and again to build their skills in a hard kind of fun.

DDA doesn't have to leave players' choices about how much resistance they encounter by the wayside. *flOw* (2006) was one of the first games designed by Jenova Chen, who chose that name because part of what he and his collaborators were influenced by and seeking to explore was the idea of flow in games. Rather than adjusting the system's resistance purely based on the player's performance, *flOw* tries to give the player concrete choices about pacing the game. In *flOw*, you control a fish-like creature swimming in an area with other creatures that can be eaten—and that will sometimes try to eat you. If you successfully maneuver your fish's mouth onto one of these creatures, they burst into white food pellets that can be eaten to make your own fish's tail longer and capable of withstanding more bites from other creatures. Each area also has a red food pellet your fish can eat that lets you dive deeper into waters with more dangerous, challenging enemies, and a blue food pellet that takes you in the opposite direction, to safer areas. Every player's journey through the shallows and depths of *flOw* is slightly different because players can retreat and advance based on how much challenge they're seeking. Also, losing all your health doesn't result in the game ending; you instead bump up one level to an easier area.

The organic, player-controlled difficulty of *flOw* is more integrated into the course of playing the game than asking the player to choose "Hard" or "Easy" before the game starts or via a settings control panel. Structurally, it has similarities to early digital games like *NetHack* (1987), where the player learns that travelling deeper into a dungeon via a staircase she's discovered will lead to greater challenge. In *NetHack*, your goal as a player is stated from the beginning: reach level 100 and claim the ultimate prize, the "Amulet of Yendor," before you succumb to various threats and enemies that end the game.

flOw, like Chen's other games, is much less explicit about the player's purpose and whether she should be trying to dive as deep as possible at all. Although reaching the bottom layer of *flOw* does give the player the opportunity to unlock more varieties of fish to play as, it's possible to play and enjoy the game while simply wandering through higher layers and surviving and eating like a simple oceanic organism, content with its lot. Many players, especially those trained to think of games as challenges to overcome, *can* play *flOw* as a game of increasing challenge, much like they might play *NetHack*, but *flOw* avoids stating overtly what a player must do to win.

There's no single correct way to shape the difficulty of a game into exactly the right kind of resistance for every player. The right decision for your game depends on its goals and what it's trying to say in a conversation with players: do you want a highly flexible push-and-pull game that changes shape depending on how the player approaches it? Or will you establish a firm structure, making a hard declaration of what your system requires, and let players figure out how to handle it—even if it means some of them may leave the game before finishing it or miss the perfect flow by a wide margin? Do you intend to involve players in deciding how the game's resistance evolves? If games are conversations, they're ones where, as designers, we have to choose what we say carefully and know what we're going to say in advance, even though we're often unable to anticipate how all the unique players will react. When we create spaces for players to make their own choices and determine their own approaches to a system, all sorts of things can happen—but that may mean that our own ideas of how the conversation will unfold have to play less of a role as well.

Alternatives to Flow

So far, our discussion of flow has revolved around the idea that games ought to try to adapt to players, avoiding frustration or boredom for too long, and sometimes including players in decisions about how the games' resistance evolves. Seeking flow states means "meeting players where they are" and ceding some degree of authorial control to foster feelings of engagement and, gradually, mastery through skill building.

Striving for a game with ideal flow that always moves perfectly between frustration and boredom isn't the only way to make a game, however. It's possible to create interesting games that don't seek out a perfect flow state. For example, what would happen if a game didn't start out slow and easy and didn't get harder?

Three Body Problem (2012) by Robin Burkinshaw doesn't change at all as the player continues to interact with it (see Figure 6.8). The system starts off as hard as it's ever going to get, but with simple rules: the player has to maneuver a square to collect points that appear, while two other squares try to collide with and kill it. Just as with *Super Hexagon*, the first time you play *Three Body Problem* you're likely to die very quickly, because the other squares are relentlessly chasing you. It's not an impossibly frustrating problem, however; you can quickly learn to survive longer by watching and learning how the other two squares move.

With practice, a player of *Three Body Problem* can close the gap between her abilities and the challenge, making the game easier. This model puts *all* the responsibility for creating flow into the player's hands: she has to accept that she's a long way from mastery and keep working at it of her own accord. Once she can handle the challenge, the task becomes to survive as long as possible to collect more points, challenging both endurance and skill. If we made a diagram of a player's experience of flow in this game, it would look very different for each player depending

on how each dealt with the challenge of the game's simple system. Rather than trying to meet players where they are, it's up to an individual to decide where to meet *Three Body Problem*.

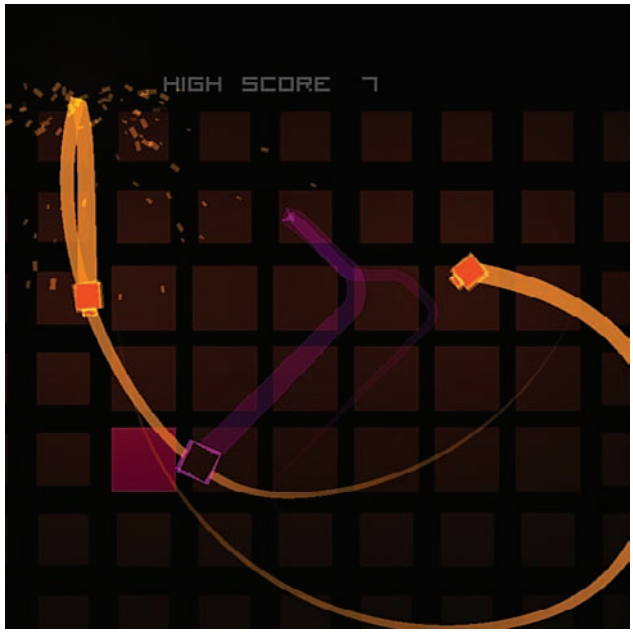


Figure 6.8 *Three Body Problem* is always just as difficult as when it began.

Games like this demand more from a player than games that hold the player's hand, but for players who are willing to *start* in a frustrated place and learn their way out of it, powerful feelings of flow can still emerge.

There's another reason to consider alternatives to traditional flow and require players to meet the game, rather than the other way around: although it's often strategic in a conversation to try to adapt how you speak to your listeners, sometimes that's not enough. Sometimes you have to ask the other participants to hear exactly what you're saying—and as we discussed in the previous chapter, some games are more about asking players to listen. *Gone Home* (2013) is a game in which players enter a house seemingly abandoned by its family, taking on the role of the eldest daughter who's returned from abroad. At first, *Gone Home* seems to play with some conventions of horror games—you explore dark rooms, looking for secrets, and are startled by creepy noises (see Figure 6.9).

The revelation of *Gone Home* is that it's not a game about facing undead horrors or even about a mounting arc of difficulty and mastery. Instead, you find clues as to the recent and long-buried history of the house of the protagonist's family, uncovering the truth about why nobody's home through diary entries, letters, bills, notes hastily left on the kitchen table, and

the mundane details of household life. There's challenge and problem-solving in *Gone Home* as you piece together clues and search for secret passages, but it's not an experience that needs to grow more challenging or gradually build the player's skills. Instead, the player comes to understand the systems at play—the relationships of characters, the ways that different members of the family inhabit and use various parts of the house—by uncovering new information, some of it in the form of words or diagrams, some of it ingrained in the spatial arrangement and visual representation of a home.

Figure 6.9 *Gone Home* subverts expectations with unnerving experiences that can't be conquered with typical game verbs.

Gone Home is set in a world that closely mirrors our own—it could be drawn from the experiences of real people. Of course, some games are overtly autobiographical, like *dys4ia*, which we've already discussed, or *Mainichi* (2012) by Mattie Brice, a game that represents a single day within the author's life. It doesn't necessarily make sense to create a traditional journey of flow through a game that recounts actual events—after all, real people's lives don't always progress from easier to more challenging. They can't necessarily be conquered by building skills and systemic understanding, but they *can* be represented through systems. The shape of resistance in these games plays a role—showing players where the systems represented can or can't be pushed—but the experiences that result aren't necessarily about players overcoming resistance or finding strategies to plant their own flag of victory at the top of a mountain. Instead, they offer players an opportunity to listen and understand systems that they might not otherwise have considered.

Opening Up Space

So far in this chapter, we've discussed resistance in ways very similar to traditional ideas of difficulty. Games that try to create flow require players to push into the system with increasing skill met by increasing challenge, often involving the development of a verb. Other games eschew flow in favor of other kinds of experience and understanding. Let's think about yet another way of looking at resistance: opening and narrowing the space of choice for the player. In a game like *Tombed*, a lot of the resistance comes from the constant "push" of the descending spiked ceiling. The player has to figure out how to act, right now, or face certain death, and the game grows more difficult as the player masters more ways to use its verbs.

Anna's game *REDDER* might seem thematically similar to *Tombed* at first: the player travels deeper and deeper into the ground through a series of tunnels and chambers, albeit as an astronaut exploring a seemingly alien landscape of mysterious and dangerous technology. *REDDER*, however, has a much more open feeling compared to the tight, constrained feeling of *Tombed*, because the player can wander in many different directions (see Figure 6.10). Even though some of the challenges are similar, sometimes involving dangerous force-fields moving toward the player's avatar and requiring quick timing to move past them, the feeling of resistance is very different. Unlike *Tombed*, the player can decide to retreat from a dangerous-looking room and explore in another direction, since *REDDER* allows the player to move off the screen to other areas, traveling left, right, or upward as well as downward. This puts some of the shaping of the game's resistance into the hands of the player.

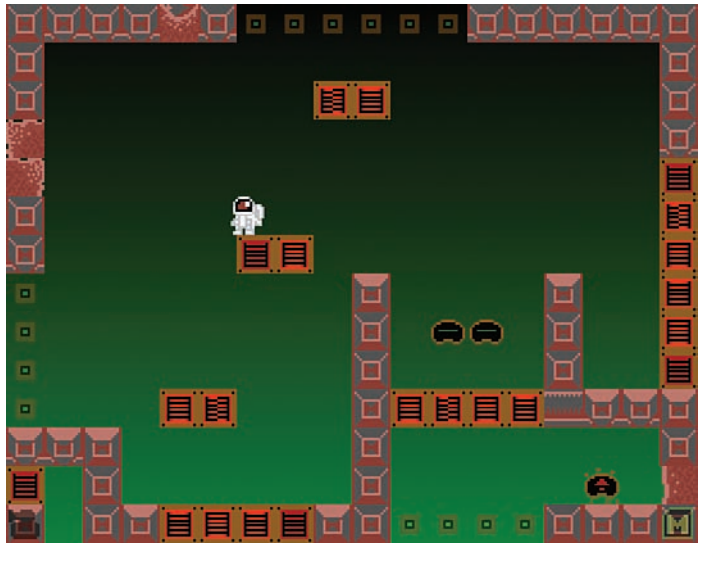


Figure 6.10 A scene from *REDDER* providing the player with different directions to explore and hazards to avoid.

Games also open the shape of resistance to player choice by offering a larger palette of verbs. At certain points in *Super Mario Bros.*, the player finds blocks that sprout fire flowers; if the player chooses to touch the flower, suddenly a new verb is available, and pressing the button which previously allowed Mario to run now *also* shoots a bouncing fireball that can eliminate enemies. It's up to the player whether she wants to find and accept power-ups like the fire flower. Furthermore, there are trade-offs in these choices. For example, when Mario becomes larger after touching a mushroom, he can break bricks with his head, but he can't fit into small spaces. Like the decision to move toward a more difficult or easier area or level of a game, these choices often shape what happens next and what kind of resistance the player encounters—but in these strategic choices, what the player prefers often depends on her way of playing and how she prefers to use verbs. As small Mario, the player is more vulnerable to dying because large Mario can run into an enemy and survive without losing a life. However, many expert players of *Super Mario Bros.* prefer to stay small because it's easier to avoid deadly obstacles and enemies when you're half the size!

Whenever a player decides to use a verb and pushes into the system, we can see choices being made—even the fundamental choice to keep engaging with the game to see what happens, to try to overcome challenges, to accomplish some kind of goal. When the player decides when to use the verb "jump" in *REDDER* to avoid a dangerous force-field, she's made a certain kind of choice: jumping at the right time may allow her to continue, while using that verb at the wrong time could kill her avatar, sending her back to the last checkpoint she passed. Deciding which direction to explore in, or retreating out of a room because it looks difficult, is a different kind of choice, much like the choice in *flOw* to dive to deeper, more difficult waters or surface to easier ones. These kinds of choices affect the resistance that the player encounters, letting her choose her own pace or avoid encountering certain kinds of resistance altogether.

In *Shadow of the Colossus* (2005), the player explores a huge, mountainous landscape, searching for crumbling giants of stone and metal that must be conquered to move on to the next challenge. The player's primary verbs involve "climbing" and "jumping," "hanging" onto the giant's enormous body, and "stabbing" it to gradually defeat it. There's another verb that plays a role: "riding" a horse. Riding can be useful in battles, but the most frequent use of riding in the game involves traveling between various areas of the world, each containing a different colossus.

After each victory over a colossus, the player returns to the central location where the game began: a huge tower in the midst of a large, empty plain surrounded by mountains. To find another colossus, the player must travel across the plain to find another area, and the distances traversed are long enough that it makes sense to ride. There's not much to see on the central plain. *Shadow of the Colossus* takes place in an area with no towns or points of interest, just some crumbling ruins that seem to have been abandoned long ago. Riding across these areas isn't difficult at all and doesn't involve much strategy or decision-making: you simply get on your horse and ride in one direction or another, sometimes urging your horse to gallop faster

with the press of a button. The player can also use an amulet to create a beam of light that points toward the location of the next encounter with a colossus, so there aren't even significant challenges of exploration to figure out which way to go.

Why does *Shadow of the Colossus* require these lengthy travel times, long enough that some players (and reviewers writing about the game) think the ride could even become boring? The contrast between battle and travel feels deliberate: riding across the plains is a relaxation in the resistance of the system, an opportunity for the player to set the pace—or simply enjoy the sights and sounds of the world. For some players, this absence of resistance might even offer a chance to pause and reflect, much as transit times often do in the real world, and perhaps to consider the larger questions of the game, like why you are hunting these gigantic, solitary, often peaceful-seeming creatures. When the player is done with her moment of pause, the goals of the game are right there waiting to be picked up again.

Opening Up Purpose

Games can open up the space of resistance by letting the player decide not only how to play, but toward what end she's striving—that is, what goal she is trying to achieve in the game. A game like *Tombed* has a single, straightforward goal: survive by descending. *REDDER* requires players to pursue a few different goal objects in the form of diamond-shaped objects scattered throughout the game. As a creator, you can add many goals to your game and let players decide which to pursue first—or at all. On the other hand, even with a long list of goals, a player's choice of what to pursue is limited to selecting from what's on that list. In games with complex systems that can produce unexpected possibilities, players can come up with their own goals—things you may never have dreamed of as the creator.

Open world games like *Fallout: New Vegas* (2010) take the first route: they open up the space of play by giving the player many different goals to select from. New Vegas is a huge world, one that's chock-full of things to find, computer-controlled characters to meet, dangerous encounters to overcome. As the player wanders across the deserts and highways of this game, numerous points of interest appear, visible both in the player's view of the world and on a map that fills itself with more and more icons as the player passes nearby or hears about them from other characters. Games like *Fallout: New Vegas* can end up with an almost dizzying array of goals as the player continues to push into and explore the world, accumulating dozens of potential missions to pursue, and spotting even more potentially interesting places to visit from a distance. Each location has its own challenges and particular shape of resistance. In one spot, you might encounter a gas station full of murderous raiders who'll try to kill you, while at another you might meet a friendly merchant who wants to exchange gossip and trade. Exactly what will happen is uncertain for any player who's not playing with a gigantic guidebook to accompany the game, and that's part of the experience of openness and choice: you never know what's going to happen next.

Simulation games, on the other hand, often avoid giving the player any overt goals. They're sometimes described as *sandbox* games because of how they allow a player to explore the game's system, an unfolding array of verbs and objects, to figure out their own objectives. In *The Sims* (2000), the player can build homes for virtual characters (Sims) that can interact with objects in the home, cooking a meal at a stove or using a toilet. The player decides what objects to put in this virtual dollhouse and can command the Sims to interact with the objects, although the Sims have their own simulated needs and will wander around looking for food to eat if they're hungry or someone to talk to if they want social companionship. The needs of the Sims aren't directly under the player's control. They grow hungry or sleepy over time, and the growing needs of each Sim changes the feeling of resistance in the game: the player might be getting two Sims to interact and have a conversation, but a third Sim nearby will be growing frantic if he can't find anything to eat.

The goal of playing *The Sims* is supposedly left up to the player: there's no "game over" or indication of failure if the player chooses to torture her Sims, refusing to place beds or toilets until the Sims are collapsing on the floor in exhaustion, surrounded by pools of their own urine. Like many simulation games, *The Sims* doesn't give the player explicit missions or objectives to check off. It's up to the player to explore what happens if she creates a house full of starving, filthy Sims or a happier home of well-fed virtual money-earners. At the same time, the game still feels like it has *opinions* about how your virtual home turns out. *The Sims* doesn't punish the player directly to convey the idea that dirty, exhausted Sims might not be an ideal state of affairs. The Sims themselves express unhappiness and don't perform as well in generating resources for the player to use when they leave the house to go offscreen to a job and return with money that the player can spend.

The Sims does have *implicit* goals, which should be clear even from reading this description: if the player cares for their Sims well, more money will be available to buy better amenities and furnishings for the home, which in turn provide for the Sims' needs more efficiently. The fanciest-looking beds and computers in the game are also the most effective and most expensive, requiring the player to run an efficient household. Even though it's not explicitly stated, the implicit goal of *The Sims* is to perform well at the cycle of labor, income, and fulfilling basic needs that most of us are familiar with from day-to-day life. This implicit goal is reinforced through rewards, which we discuss later on in this chapter. If you want the best stuff, you need to care for your Sims.

There's no explicit way to "win" *The Sims*, but it's clear that to give your Sims what most of us would consider a nice life, you have to become good at a simple capitalist system of earning and buying, then earning more to buy better stuff. Still, the game doesn't push back with much resistance if the player chooses not to pursue this goal and opts for a life of simplicity or squalor instead. Rather than the hard resistance of losing points or forcing the player to start over, *The Sims* has a softer form of resistance, embodied mostly in the crying faces of unhappy Sims.

Nothing happens to change the system or state of the game if you let your Sims cry, but it's still an emotional signal: you made your little people sad! It's a form of resistance that's up to the player to interpret: maybe she sees herself as a cruel torturer of virtual beings!

Some sandbox games go even further to create a wide-open form of game conversation where players are the ones coming up with the topics at play. Unlike *The Sims*, *Minecraft* (2009) doesn't have an economic system that involves going to a job and earning money. Instead, the player has to harvest raw materials from an enormous landscape composed entirely of cubical blocks in dozens of different types and then figure out not only what kind of objects to populate their sandbox world with, but also how to craft those objects from raw materials. Harvest 16 units of wood with one tool, use another tool to turn the wood into four boards, and you'll finally be able to turn those boards into a chair. As in *The Sims*, *Minecraft* has more expensive and difficult-to-procure items in its huge catalog of objects, although the expense is in rare materials and multistep processes rather than a currency. More interesting still, especially for our discussion of resistance, *Minecraft* allows the player to craft many kinds of components and tools, then put the components together to create new kinds of devices. Enterprising players have built elevators and even calculating machines out of the simple mechanical and electrical components that the game system provides.

Minecraft's system of parts is complex enough that players can explore it to come up with possibilities that the game's creators never dreamed would be possible. Players aren't just making choices about how to deal with a difficult challenge or use a verb; nor are they deciding which goals to pursue in a system or at what pace to pursue them. Instead, they're fashioning their own kinds of resistance—new challenges and objectives made possible through recombining the existing structure of the game such as building an entire house with light switches. Instead of selecting from options preordained by the creator, the possibilities come from their own imagination or from what other players have dreamt up.

It's worth remembering that even an open-ended game like *Minecraft* is not the same thing as a blank canvas upon which participants can draw anything they choose. The shape of possibilities in a game is fashioned out of the structures, rules, and building blocks that creators like you put there, even if there are potential outcomes that you don't anticipate. This is part of what makes creating a game such a uniquely rewarding pursuit. As a creator, you produce structures that serve as the groundwork for all sorts of player choices, including ways that players can make the game their own and surprise you with their ingenuity and choices. The resistance inherent in *Minecraft*'s structure comes from a set of rules that determine what happens. The player can't simply build a platform next to a switch and turn it into an elevator. The system is more complex than that and requires deeper understanding of how its objects interact. Games that are open in this way aren't necessarily better or more enjoyable than games that push the player with tighter forms of resistance, however. They're just different kinds of conversations, and as with verbal conversations, the world would be a less interesting place if everything worked the same way.

By opening up or narrowing the space of player choice in your game, you can let your players affect the shape of resistance. At the narrowest end, players can experiment or choose *when* to use the verbs you've provided them: Is it the right moment to maneuver left or right in *Super Hexagon*? Which is the best spot to dig to descend further in *Tombed*? As you open the shape of resistance, the choices change: *How much* resistance will there be, and *how soon* will it appear? Will the player swim toward more challenging depths in *flOw* or spend time riding through quiet meadows in *Shadow of the Colossus* before the next battle? You may want to open up the choice of *which* verb the player will use, whether that's represented as a selection of weapons in a dueling game or deciding whether to grab a Super Shroom to play as big Mario. Finally, if you pull back your game's shape of resistance to the point where the player's making choices on her own, you can allow her to decide *which* goals to pursue. You can even ask the player to come up with her own goals, leading to the big question of *why* she wants to play and *to what end*.

The Pull of Rewards

The flow of difficulty and challenge is a kind of resistance that pushes the player to varying degrees, requiring her to push back into the system of the game, try to overcome obstacles, and deepen her understanding of the game. A player also pushes into a system and finds resistance when exploring the possibilities of a system, whether it's figuring out what a system can do or coming up with her own goals and strategies within the structured rules of a system. If resistance is a way of looking at the push and pull of the ongoing dialogue between player and game, then we need to find ways to pull players as well as push them, to encourage and lead them forward.

Because the games we create are having these conversations with players in our absence, we set up systems to give that signal to players: "Yes! More of that! That's right!" This kind of positive feedback is often thought of as a reward. The tradition of rewarding players who are "doing the right thing" goes back a long way through the history of various kinds of games: gambling for money, accolades and cash prizes for sports tournaments, and so on.

Chances are that when you a play a game, you're not doing it for the practical value of the rewards you might earn. You're playing the game for its own sake, because hopefully it's enjoyable, meaningful, or rewarding. If games ought to be inherently fulfilling in their own right, do we need rewards? It may make more sense to think of them as feedback in the game's conversation with players. In single-player games, we leave rewards along the way to encourage, to tell players that they're on the right track. Still, the visual language of rewards often has the feeling of a payoff, a prize that has some kind of value. At the surface level, you can probably recall some examples of what these rewards look and sound like: shiny coins, bouncing stars, an overflowing treasure chest, perhaps accompanied by uplifting, victorious music and large text that declares, "You Did It!" *Peggle* (2007) has one of the most well-known examples of a game's contextual elements used to create a rewarding feel. When the player manages to bounce a ball

into the central opening at the bottom of the screen, the game lights up with fireworks, and Beethoven's "Ode to Joy" plays triumphantly. It's a real spectacle.

Just as with other kinds of feedback discussed earlier in the book, it's important to let players know when they've pushed into the system and done something "right." This is part of how players understand the shape of the system when they play. What's the purpose of the conversation? How does the player evaluate whether she is going the right way? Just as Danger Jane has to be shown pushing uselessly against a wall, or hitting an undiggable metal block with a "ting" of the shovel, signifying the places where the game *can't* be pushed, we want to show our players very clearly where the game *can* be pushed. Still, showering players with the audiovisual signals of reward can feel gratuitous or meaningless if we use them everywhere, as if we threw confetti and cheered every time a friend said something we understood in a conversation. Reward, the pull of our system-created conversation, has to come at a meaningful moment—and just as any celebration doesn't feel joyful *just* because people are throwing streamers, the feeling of reward in games is based on what the player's been doing up to that point.

Rewards are great for marking milestones during a player's journey and separating one part of the game from the next. When you manage to navigate past a few minutes' worth of enemies, pits, and navigational hazards in *Super Mario Bros.*, you reach a castle with a flagpole outside of it. This sight, familiar to many gamers, lets you know that it's time for a pause in your journey. It's a short pause, but it's still a break, a release and relief from your efforts. When you enter the castle, fireworks go off. As the pace of activity suddenly slows for a moment, the visual elements of reward appear. All of this happens together to mark the event: you're done with one part; now it's time to take a breath before the next part begins.

Not all games are broken into discrete chunks like this, but these pauses are useful moments to signal players that they're successfully pushing into the game. When a player finishes one section of a game, it could look like one level of *Super Mario Bros.*, leveling up a character in a role-playing game, accomplishing a discrete mission objective, or fully exploring an area of the game world. The relief of having finished an identified section can be a release from the pressure of decision-making and responding to push-back from the game: a moment to stop and celebrate.

Cat Cat Watermelon (2010) from Lexaloffle Games is broken into 20 levels, each one posing a more difficult challenge than the last. The player stacks various objects (including cats, watermelons, beach balls, and more) on top of each other to create a tower that doesn't fall over. If the player manages to stack all the objects for a given level, a sign appears with a fairly standard victory message, telling her that the level's complete. Ironically and intentionally, the sign actually knocks over the tower that the player just built, sending all the carefully placed objects tumbling away. This is a great example of how the moment of success in a game can create a feeling of release, even from what you've just accomplished. Hooray! You did it! And now we sweep everything away. This is a kind of reward, too, even though it doesn't come with

fountains of gold coins and exploding sunbursts. It's there as a marker, an acknowledgment that you've done it. By clearing out the remnants of the past (what you've just accomplished), it also pulls the player toward the next challenge.

Of course, this isn't the only way to mark the occasion of a player successfully pushing into a game up to a particular point. It's easy to imagine games that do the opposite. When you finish a section of the game, a certain set of tasks, or a particularly difficult challenge, you get to keep a memento of your success, such as a medal or souvenir. It all depends on how you want the conversation of your game to mark memories of the past and open the way toward the future, where the player will explore more of the game's possibilities.

Previous chapters have talked a lot about what happens when the player's introduced to a new verb and how verbs develop along with a player's understanding of how to use them. Opening the conversation of a game to include more verbs is an exciting moment, and it's often used to create a moment of reward. When a player has successfully pushed into a game and accomplished something she's been striving for, she's hopefully deepened her understanding of the game's verbs and system. Introducing a new verb—or a new way to use a verb—can help keep the experience flowing interestingly. For similar reasons, introducing new objects to use verbs on is also a common form of reward, whether those objects come in the form of new areas to explore, new opponents to face, or new obstacles that have to be traversed. Unlocking new sections of the game may not seem like a reward in the same way that finding a gun that enables the "shoot" verb is, but both are significant rewards because they let the player push into new parts of the experience. These rewards connect the player's past accomplishments to what's coming next.

Resources

A chest full of gold coins is one of the most traditional forms of reward, signifying wealth. In a game, as in real life, the value of money depends on what you can buy with it. We can think of currency and other spendable resources as enabling particular kinds of verbs: "spend" and "save." Currency is useful for game creators as a flexible kind of reward because it's usually represented as a number. Players can make choices about how much to exchange, perhaps for a new verb or access to new objects, or they can hold onto it in anticipation of being able to afford something else later on. Not all spendable resources look like money, either; *skill points* are another type of currency, usually rewarded after the player reaches a certain point in a game.

You can think about any kind of number that the player can spend or save as a currency reward, the means by which other verbs are enabled. For example, *ammunition* is an expendable resource that enables the "shoot" verb in many games. *Health* is a special kind of expendable resource that the player tries to avoid losing in challenging situations and which she has to save enough of to avoid running out. Running empty on health leads to "death," and although that means something very different in various games, it often involves starting over, dealing with a penalty, or experiencing a setback of some kind.

Players of role-playing games are familiar with *experience points*, yet another kind of resource that doesn't involve much choice. These points are automatically saved for you until you have a certain amount, and then they're spent to reach a new level, at which point many games give the player a reward in the form of spendable resources, new verbs, or access to new areas of the game. Experience points aren't really an expendable currency; they're basically just a marker of progress toward the next part of the game experience.

Dead End Rewards

Creators of games have invented far more ways to entice players into staying engaged with a game, from the bells and whistles of exciting audiovisual feedback to piles of spendable resources—sometimes in such huge amounts that decisions about spending and saving become meaningless. Many games also try to weave a story into the conversation of playing—a subject we talk about more in Chapter 7, "Storytelling." When you're playing a game, your experience of push and pull, tension and release, has a narrative flow of its own.

Game creators who also want to tell a more authored story often do it in bits and pieces at a time, with a prewritten dialogue between two characters onscreen, or a cutscene that involves little to no involvement on your part. These dramatic interludes are often placed between sections of gameplay so they don't disrupt the flow of actual gameplay. As a result, they often happen around the same time as other rewards or pauses in the resistance of a game, because the pleasure involved in watching a story unfold is also a reward. Using story as a reward is a little troublesome, however. If your story is exciting enough that it feels rewarding just to reach the next scene, then experiencing that story could become the real motivation to play, at least for some players. We'll come back to talking about story in the next chapter, but for now it's good to understand how story rewards, divorced from the system and its shape of resistance, are like a dead end: they don't feed back into what the player's doing in the same way as giving the player a new verb does.

Achievements are another kind of dead-end reward, popularized by mass-market online game networks like Xbox Live and PlayStation Online. *Cheevos* (as Anna and other critics like to call them) are deliberately outside the real systems of gameplay. Games can record achievements into an online network's system, but unless a game has its own way of tracking and using achievements, they never affect anything else in the game. Like story rewards that don't function as part of gameplay, cheevos sit outside the conversation; they have to be pursued purely for the sake of reward. Because game creators would have to duplicate the achievement systems inside their own game to avoid this dead end, it's no wonder that many cheevos seem to have little to do with what makes a game interesting.

The least interesting kinds of cheevos are either awarded simply for playing the game, often just duplicating the game's own system of reward and progress, or require the player to do repetitive actions, like killing large numbers of enemies or amassing lots of resources, in ways

that they wouldn't be likely to pursue without the achievement. It's possible to create interesting achievements that can feel genuinely rewarding but, like story, this tends to work best if a game's creators have figured out how to integrate achievements into their own systems. That ends up making the online networks' achievement system a little redundant for most players who don't care about racking up Xbox Gamerscore points that simply show how much time someone's put into playing Xbox games.

Time and Punishment

Rewards involve a shaping of resistance that pulls the player forward instead of pushing her back. Moments of reward encourage the player to push deeper into the game. Difficult challenges, as well as the very rules of the game, are the structures of resistance that the player pushes against. Of course, we can't just talk about rewards and difficulty without discussing another way that games push the player: *punishment*. Moments of punishment happen when the player makes a mistake or fails at the game, whether by being crushed by a spiked ceiling (as in *Tombed*) or losing all health points while fighting an enemy (a system found in countless games). If rewards are the "carrots" that encourage the player to move forward, punishments are the "sticks" that signal when the player does something, or ends up in a situation, that the system doesn't encourage.

Punishments don't just block the player from doing something, as when a player encounters a rule like she can't dig into a metal floor. Instead, they often send the player flying backward through the experience. When the spiked ceiling of *Tombed* reaches the top of Danger Jane's helmet, Jane goes flying off the screen, disrupting the scene. The game then resets itself to an earlier moment, the last time that the game invisibly recorded that Jane had passed a certain depth in her descent. This moment is a *checkpoint*, sometimes marked by a little flag or other landmark, like the small white pillars that the player frequently passes while exploring the Martian landscape of *REDDER* (see Figure 6.11). Returning the player to a checkpoint is like repeating part of a conversation—perhaps to go over something a second or third time that one of the participants didn't understand or that they need to revisit to respond differently.

In many games, players can create their own checkpoints as well, by saving the game. If they encounter a punishment, perhaps one that terminates the experience like "dying" in a game usually does, they can return to one of the moments where they saved. Regardless of whether the system provides a way for players to manage their own checkpoints or includes checkpoints at predetermined points in the experience, the player is sent backward to repeat what she's done. This form of punishment also exists in games that simply punish the player with a "GAME OVER" message, like *Three Body Problem* and *Super Hexagon* do—except in those cases, as with many classic arcade games, the player is returned to the beginning of the experience to start again. The entire system is reset.

Figure 6.11 A player returning to a checkpoint in *REDDER*.

In nondigital games such as sports or board games, a mistake can result in losing the game, which also effectively resets the system. If you want to play again after losing at chess or basketball, you have to start again. More gradual punishments in traditional games also exist. If you move a chess piece to a vulnerable position, your opponents might take it, costing you one of your pieces. In basketball, a punishment can happen if you violate one of the rules of the game, such as walking without dribbling the ball. In this case, the punishment is that the ball is given to the opposing team. This kind of punishment has the effect of tilting the game toward your opponent, possibly bringing the game closer to victory for your opponent and closer to a loss for you.

In single-player games, however, the only opponent is the system itself. Although there are gradual punishments (losing some of your health points), we don't always want to reset the entire game if the player reaches a final punishment. Even if the whole game does reset, we want the player to have the choice to try again. In fact, if your goal is to provide the player with an experience of flow, balancing frustration and boredom, you may want your system to help the player out if she's making mistakes, which is precisely the purpose of the DDA techniques that we discussed earlier.

Instead, single-player games often punish the player with repetition: die or make a significant mistake, and you'll have to repeat a section that you already did, starting from the last checkpoint. Repetition may seem like a boring kind of punishment, since it may involve the player encountering familiar objects and scenes, even using verbs in the same ways that they already did. On the other hand, repetition can be useful in a couple different ways. First, if the space of the game is open enough that the player has a lot of different verbs at her disposal and choices of how to use those verbs, or there are different areas of the game to explore, the player may experience something very different the next time around. Second, even if the player does go through the same motions again, repetition can serve as *practice*: what was challenging the first time becomes slightly easier, and when the player reaches the point where she made a mistake, she has the opportunity to practice overcoming that challenge again.

In these two ways, punishment in the form of repetition has a purpose beyond just saying, "Bad player! No!" in our systematic conversation: it gives the player the opportunity to revisit earlier moments, to push into the game in another way and create a different result. You can imagine the same thing happening in a spoken conversation, if one participant doesn't understand something that's said: we repeat and revisit ideas and probably try to communicate in a slightly different way so that we can continue.

Anna's game *Mighty Jill Off* (2008) gives the player plenty of opportunities to practice because of the way the game develops the "jump" verb, which Jill uses to climb toward the top of a tall tower. The first half of the game consists of a number of sections, each designed to teach the player about a particular way of jumping and color-coded to be easily identifiable. The green section near the beginning involves simple jumps from platform to platform, but the following blue section requires the player to use a more advanced kind of jumping. In *Mighty Jill Off*, rapidly tapping the jump button allows Jill to slow her descent. Combined with moving left or right, this allows her to float sideways to land on platforms. After the blue section, the orange section teaches the player how to run off platforms and jump at the same time, and the lime-green section introduces a new kind of hazard: spiders that chase the player.

Each of these sections starts off with a scene or two that introduces the new kind of jumping or hazard in a straightforward way that's not too challenging or complex; the next part of the section often involves combining what's just been learned with hazards or situations from earlier

in the game, elaborating on the new game mechanic and further developing each use of the verb.

The blue section starts off with a scene that can't be passed without executing a sideways float, followed by a drop, no less than four times (see Figure 6.12). On top of this repetition, it's likely that a new player who's just learning *Mighty Jill Off*'s particular form of jumping will die repeatedly when she touches the torches. When Jill dies, she tumbles back toward the bottom of the tower whence she came—but a few moments later, she reappears at the last checkpoint the player passed, often at the beginning or halfway through one of the colored sections. There's a checkpoint at this first blue scene, letting the player practice, die, and practice some more until she's good enough to clear four jumps in a row.

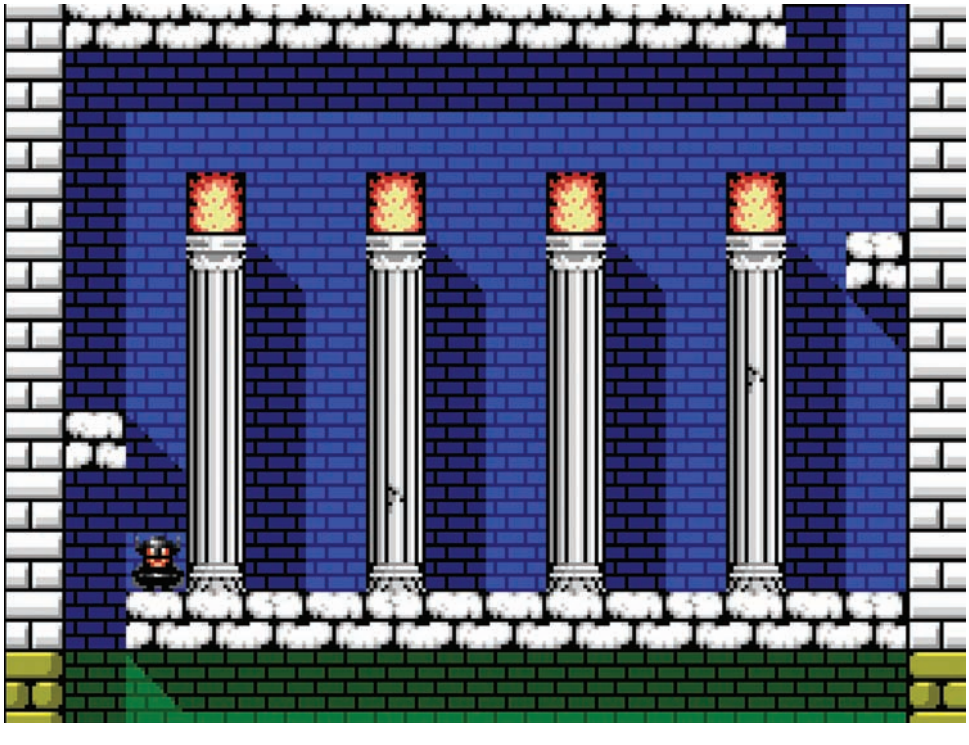


Figure 6.12 The player must master new skills to advance through *Mighty Jill Off*.

Practice is only useful up to a certain point, however. Once a player has thoroughly mastered a certain use of a particular verb, perhaps even multiple times with the same arrangements of objects in a scene, practice becomes rote instead. The player's gains in understanding and skill to use those verbs start to level off, improving only a little bit. Rote activity is closer to second nature, done without a lot of thought and next to no challenge, and rote actions repeated in

the same scenes can rapidly move the player toward the boredom corner of the diagram of flow. Even so, rote plays a huge role in many digital games, and players are willing to put up with a lot of rote punishments, especially if they've become emotionally invested in moving through rote activities to tackle challenges again. Some players might be seeking the satisfaction of overcoming what they couldn't before, while others might be eager to reach the moments of reward that they're hoping lie beyond those challenges, and others still might just be so used to rote actions that they don't mind the boredom.

Many digital games of recent decades use rote activity not just as a form of punishment, but as a basic element of the resistance of the game. Even if the player hasn't made a mistake or died, there's still a lot of rote activity—usually involving verbs that are easily mastered and challenges that are easily overcome—that the player has to push through before reaching a more intense experience of challenge or a moment of reward.

A certain breed of single-player digital role-playing games, exemplified by the many titles of the *Final Fantasy* series, uses rote activities throughout. Although these activities are presented as combat against a variety of monsters and enemies, most of the time the player can defeat those enemies just by pressing the "attack" button over and over again. Even if that simple verb isn't enough, the most the player has to do is use a slightly more advanced verb that involves a limited resource, or understand a system of "weaknesses"—for example, fire-type monsters can usually be defeated by using a "water-type" verb, which uses up an automatically replenishing resource often called *mana*. Although there are verb–object relationships to learn here, they quickly become rote, and the player is often called on to use them again and again.

Players refer to this kind of repeated rote activity as *grinding*: performing the same actions again and again for the sake of reward. One of the hallmarks of this kind of role-playing game is that the player can grind to accumulate resources that make her avatar more powerful—not because the player herself is understanding the system more deeply or learning how and when to use verbs, but simply because the numbers of resources are climbing. In some ways, the choice to grind for more resources can be an interesting way to open up the space of resistance in a game: if the player has a difficult challenge ahead of her in a role-playing game, she can often invest time into grinding to boost her resource numbers, which will make that challenge easier. For example, the player might accumulate enough experience points to level up and increase her health resource, allowing her avatar to take more damage before dying. She might also accumulate enough of a currency resource to purchase a new sword that uses the same "attack" verb, but which can remove more of a monster's health.

By definition, because it involves rote activities, grinding takes time, not skill or understanding. It's a player's choice whether to invest this time, but many grind-based games require a certain amount of grinding just to get enough resources to proceed. Some grind games are all about investing time to get more resources, such as the wildly popular *Farmville* (2009). Like role-playing games, *Farmville* has multiple resources to increase and lets the player level up to access

more kinds of resources and new verbs and objects, but advancement involves planting various crops rather than fighting monsters (see Figure 6.13). Reach level 13, for example, and you can plant strawberries, which, like all the other crops, exist only as a way to invest more time. When you plant a crop, you must pay some of your currency and then wait for a certain length of time—between 30 seconds and 48 hours. When the time's up, you receive an increase in your currency resource by clicking on the crop, which represents harvesting and selling the crop.

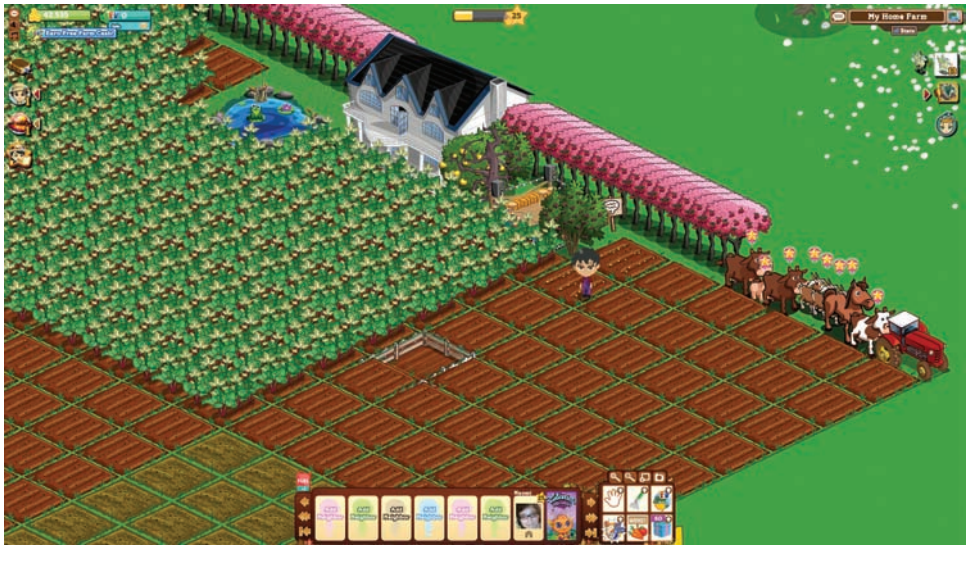


Figure 6.13 Grinding in *Farmville* to earn more resources.

The skill involved in *Farmville* is more akin to deciding how to invest your resources and when; it's still a game with a complex system of verbs and objects that the player has to push into, but nearly all the resistance of the system comes from waiting for your crops to grow. There's punishment in *Farmville* as well, but it happens only if the player makes the mistake of waiting too long to return to the game and harvest—in which case the crops have withered, causing the player to lose her investment.

Whether it appears as a punishment for making a mistake or is simply required to complete actions, grinding and rote repetition are still forms of resistance: it's difficult to keep at a task that feels boring, and it takes patience to wait. This kind of gameplay raises a question for players and creators alike, however: are these interesting forms of resistance that we want to partake in? They involve a smaller number of choices than figuring out where and when to "jump," "shoot," and "dodge" in more action-driven games. Although some players might be more patient or better at enduring rote tasks, it's a different form of skill. Players who enjoy

games where the space of resistance consists of exploring complex rules and choices, pushing against challenges, and dealing with less rote punishments are used as a kind of resistance that comes from the *frustration* side of the flow diagram. Grind-based gameplay, on the other hand, involves resistance from the *boredom* side. And it's not hard to understand why many players simply don't want to invest their time in activities that primarily challenge patience, endurance, and the ability to sink many hours of time into a game.

Many game creators feel that using grinding is disrespectful of players' time. The popularity of grinding in role-playing games, social games like *Farmville*, and increasingly in other game genres might be the result of pressure from business managers and marketing strategies that simply try to deliver more hours of gameplay or keep players around longer so that they'll spend more money in free games where they can make purchases while playing. Along with the lure of rewards, grinding can be abused in this way to create filler that doesn't do much to enhance the conversation of gameplay.

On the other hand, some players genuinely enjoy challenges of patience, efficiency, and optimization; there's strategy involved in figuring out how to grind more effectively, wasting as little time as possible, or even managing your own patience as you push through a boring kind of resistance. If action-driven games sometimes resemble a complex obstacle course to navigate, then grinding games involve something more like the endurance and sustained concentration of a marathon. Different players enjoy different things—so beyond making a choice about what kind of conversation you want to facilitate as a creator, the most important thing is to be honest with players about what kind of resistance your game uses. Players who are looking for difficult challenges or who want to measure their skill against competitors have every right to feel betrayed if they find that a game's been packed with a lot of grind simply to make it longer!

Scoring and Reflection

One more shape of resistance deserves mention: *scoring*. It's neither a punishment nor a reward, but a reflection of the player's experience. A player's score is sometimes revealed throughout a game, as in classic arcade games where the score is displayed in a corner of the screen, often going up steadily as play continues. As we discussed earlier, moments of punishment and reward often happen at significant milestones in the ongoing conversation of a game, after the player pushes through one chunk of the experience—or makes a fatal mistake that ends the game! These moments are often used to reveal the player's score as well or focus the player's attention on a score that's been present all along. It's the time in the conversation where the participants check in about what's been said and determine how things have been going.

A score is an evaluation, and as the creator, it's up to you to decide what's being evaluated. Like rewards, increasing score is a statement to the player that she's done something you consider positive through the system of the game. Many games award higher scores for destroying more enemies, for example. This doesn't shape resistance immediately in the way that in-game rewards like a new verb or the unlocking of new spaces would, but it does send a message: it's good to destroy enemies! If that's what you want your game to say, then it can be useful to reinforce the verb–object structures of your game by reflecting those goals with higher scores. Not every use of a verb or interaction with an object needs to result in a score; it's up to you to decide what's significant for raising or lowering the score.

Because a score is an evaluation, it can feel like either a reward or a punishment, depending on whether the score's presented by the system and interpreted by the player as being high or low. A score usually involves numbers, or sometimes letter grades. In games with complex systems of choice and lots of ways to do well, scores can sometimes be broken down into many different measures of performance. Because scores turn some of the player's actions into numbers or letters, they're useful for comparing performance between players. This is, of course, how scoring has been used for centuries in sports and other competitive games. In more recent decades, scores have appeared on leaderboards that let players compare what they've done in a game to what their friends or vast populations of players online have done.

Scores in single-player games provide a special kind of comparison and feedback: the player can compare her own scores at different times or on successive playthroughs of the game. As a player, being able to track your own score over time provides deeper insight into how you've pushed into a game and how the creators of the game have evaluated your actions. Of course, all this depends on whether the system of scoring is clear enough for players to understand what the score means. If you want a player to actively think about score and how to affect her score, it's useful to be as clear as possible about what's going on. As with other rewards and contextual cues, visual feedback is crucial, but a straightforward explanation of scoring can be just as helpful.

Like some of the systems of reward we talked about earlier, score often exists outside of the system of verbs, objects, and choices that the player pushes through while playing the game. One of the things that makes score more interesting as part of the conversation, and not simply something external, is that it takes its meaning from players' own understanding of how it's important—or unimportant. In a competitive tournament where all players have agreed that the highest-scoring player is the victor, comparisons of score are crucial; the score is imbued with a lot of significance. In a single-player game, it's up to the player herself whether she wants to pursue higher scores to beat her friends or her own past best efforts. Scores are an evaluation and not simply rewards that lure the player on. They're useful as tools to help the player gain her own insights into how and why she's playing.

Although giving higher scores for *desirable* actions and lowering scores for *undesirable* actions in a game is the norm, it's even possible to create scoring systems that don't necessarily favor one style of playing or action over another. In *Wonder City* (2013), a game I helped create, the player guides the actions of a character who's just discovered she has superpowers. Through a series of decisions, the player decides how this superheroine will relate to other characters, protect her secret identity, and deal with tricky situations (see Figure 6.14). There's no single score that determines how well a player did; instead, the game keeps track of the kinds of choices the player makes. Does the heroine use her powers at every opportunity or try to solve problems in other ways? Does she collaborate with her friends, or is she more of a loner? After each chapter of the game, the player sees icons that represent a style of heroism: direct or indirect, powerful or restrained, collaborative or independent, among others.

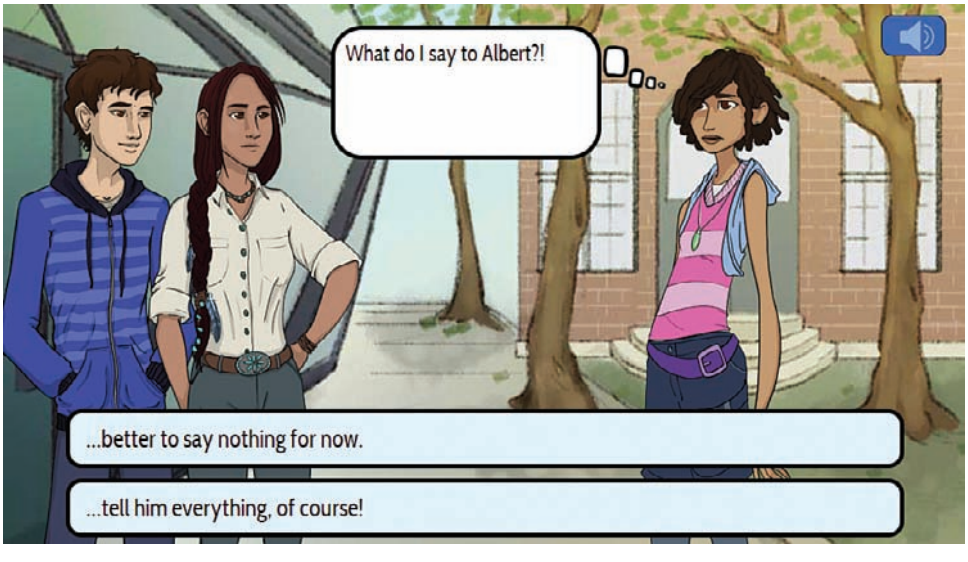


Figure 6.14 Making decisions in *Wonder City*.

Although the game's system keeps track of these styles with a series of numerical scores, we actually opted *not* to show the numbers to the player. In *Wonder City*, we wanted the game's conversation with the player to focus less on trying to affect the score and more on the player's gut feeling about how she wanted her character to behave in each situation that requires a decision. The result is a scoring system that's more like a personality test: there's no *better* score, only different scores that reflect ideas back to the player about the style she's created through a series of cumulative choices (see Figure 6.15). Like the other parts of a game's resistance that we've discussed in this chapter, score can be shaped in many different ways, depending on what kind of conversation you're trying to bring about.

Figure 6.15 Intermission screen in *Wonder City* showing the player's choices so far.

Review

- Difficulty is a traditional way of talking about a player's journey through a game: games start off easy, players deepen their skill and understanding through playing, and then they take on more difficult challenges. In a simplistic version of this journey, the most difficult experiences are the pinnacle of what a game can offer. But this is just one way of looking at the conversation of playing a game.
- When we talk about resistance, we mean the many ways that a player can push into a game by using verbs and making choices, the ways that a game pushes back by presenting challenges to overcome and consequences for actions, as well as the ways that a game can pull a player with rewards or open up wider spaces for the player to push into. Resistance shapes the conversation of a game via a system of rules that the player can push against.
- Flow is another popular way of talking about the way players encounter resistance in a game: an engaging experience that hovers between frustration from high levels of resistance or difficulty on one side, and boredom from the repetition of already mastered tasks on the other. Different games have very different kinds of flow. Some game creators try to create perfect flow, never allowing frustration or boredom for long. There are many ways to make interesting games without perfect flow, such as games that challenge players from the beginning and ask them to overcome frustration.

- Whether you're thinking about games in terms of flow, difficulty, or resistance, you can shape the push and pull of your game's conversation in many ways: by pacing the development of verbs as discussed in Chapter 3, or by opening up the pacing to player input, such as by letting her select a difficulty setting.
- Game designers who strive for perfect flow sometimes use dynamic difficulty adjustment (DDA), techniques that help a player who is struggling with frustration or make things harder for a player who is getting bored. These techniques are tricky, however, because it's hard to guess what a player's experiencing. Even if you can guess, DDA can lead to the game feeling mushy, adjusting itself to everything the player does without providing a firm structure that the player can push against to understand its rules and challenges.
- When you let players have a hand in shaping the resistance of your game's conversation, they can be part of deciding how and when their experience gets more challenging. Games can let players choose how quickly to move to a more challenging space within the game or whether they use the verbs that require more skill. Games with very open spaces of resistance let players choose their own goals as well—or even make up goals to accomplish within a system, as in sandbox games.
- Rewards are a way to shape resistance that pulls players forward, encouraging them to pursue goals or repeat actions. Some rewards operate within the system of a game, opening more opportunities for players to push. For example, rewards can unlock new verbs to use or new areas of the game to push into. Other rewards don't feed back into the system, such as story cutscenes or traditional achievements that exist on a console platform. It's worth thinking about whether the lure of this kind of reward can eclipse the pleasure of playing a game for its own sake.
- Resources are a kind of reward that enables other verbs. For example, ammunition is a resource that lets the player use the verb "shoot." Resources are flexible because they're represented as a number that the player can manage through spending or saving. They can be linked to goals or punishments. In many games, if you run out of a resource called health, you experience a setback or have to start the game over.
- Punishments push hard against the player when they do something the game deems wrong, often involving a penalty that increases resistance or a setback that requires the player to repeat some or all of what she's done before. Repetition can be a useful form of punishment, especially for games that try to maintain flow, where a resistance-increasing penalty would make the game more frustrating. Repetition of a challenge that hasn't been mastered can give the player a chance to practice, to experience the same situation again, make different choices, and seek a different outcome that leads to reward as opposed to punishment.
- Repetition of tasks the player has already mastered is often called grinding and is sometimes used in a game for its own sake. Grinding over long periods of time creates a different kind of resistance that requires endurance and patience to overcome

boredom, rather than challenges to overcome frustration. Some players enjoy this kind of resistance, but many games use it simply to increase the amount of time the player must invest to reach a goal. Many players resent this kind of tactic to keep them playing.

- Score is not inherently a reward or a punishment; it's an evaluation where the player can reflect on what she's done. In creating a score system for a game, you're telling the player what kinds of actions are desirable or undesirable in the conversation. A player can compare a score to her own previous scores or to other players' scores to understand how she's doing. Scores can be open to player interpretation as well. Especially in single-player games, players can choose to disregard the evaluation of score. It's possible to create scoring systems that don't value one type of action or decision more highly than another but simply let the player reflect.

Discussion Activities

1. Think about experiences you've had playing games that were very challenging, frustrating, or difficult. Now think about some experiences that were very easy—even to the point of boredom. Compare your experiences with others. Are there particular kinds of gameplay that you find challenging or easy? Why?
2. Imagine that you've been asked to make a game based on something that happened to you, or some aspect of your life. What would you draw from your own experiences to make a game about? How could you express something about your life using systems as well as contextual elements like words and images?
3. Using the scenario you designed in Discussion Activity 5 of Chapter 3, discuss how you could change the rules of the scenario to increase the resistance and difficulty. Also discuss how you might raise and lower resistance to create different kinds of pacing from this list:
   - Pacing based on the classic idea of flow, where a game starts off easy, introduces the player to new verbs and objects, and builds in challenge to avoid boredom.
   - Pacing that starts off difficult, demanding a lot of the player, and only becomes less frustrating through practice
   - Pacing without any flow at all. Think about other ways that a game's shape might change over time that aren't about building skill and rising challenge.
4. With that same scenario, are there additional rules you could add that let players have a hand in whether the game is harder or easier? How would you open the conversation of your game to include player choices about their own goals in the game?

5. Get other friends or classmates who weren't involved in designing your scenario to play it. Afterward, ask them whether their goals, choices, and scores were clear to them. What did or didn't they understand? How could you make these systems clearer? Alternatively, are there ways in which lack of clarity could make the game more interesting?
6. Split the scenario you've been discussing into multiple parts by inserting moments of reward. For example, after part of the scenario has been completed, give the player or players an additional verb to use if they've completed certain goals or actions. Try experimenting with different kinds of rewards, including combinations of the following:
   - A moment to pause and relax, where the resistance of the game lets up for a while
   - A new verb that they can use in the next section of the game
   - A limited amount of a resource they need to use to enable a verb
   - A purely decorative reward like a medal or a title that recognizes an accomplishment but *doesn't* affect the game system

Group Activity

For this activity, you'll once again be using *Knytt Stories*, the game creation tool described at the end of Chapter 4, "Context." Split into two teams (even a team of one is fine!) and design a level in *Knytt Stories*. When you're done, have the other team play it, and watch them playing. Don't say anything while they're playing, but keep track of your own reactions as they move through the scenes you've created. How does watching other people play make you feel about what you created?

Using the preceding process—creating a level, and then watching someone else play it—try out the following ideas:

- Make an easy game that just introduces a new verb.
- Take the easy game and see if you can make it more challenging by increasing the resistance somehow. Can you increase the resistance by developing verbs? How about just with objects?
- Extend the game so that it has periods of high resistance (for example, a difficult section) and periods of low resistance (a section where the player can pause and relax or just engage in rote activities she's already mastered).
- Make an extremely high-resistance game—not impossible, but one that you're not sure if the other team could actually beat. See what happens!

This page intentionally left blank

CHAPTER 7

STORYTELLING

Human beings like to tell stories. How we describe the world and live our lives are bound in the stories we tell. We're not just good at telling stories, however—we're also good at seeing them. Perceiving stories is part of how we understand our world. Just as we can see patterns in the arrangement of shapes and forms—animals in the clouds or a face in the knots of a tree—we make stories out of things that happen around us. We impose a pattern, a meaningful sequence, on a set of events, and a story emerges.

Pattern Recognition

Stories told through films and comic books have always relied on the human ability to stitch together meaning out of a sequence of images. In film, many images flicker by every second; in comics, they're laid out on a page, where our eyes can move from one to the next. In the 1920s, a Soviet filmmaker named Lev Kuleshov conducted an experiment that demonstrated the ability of film viewers to construct stories: he created three short, wordless sequences, each starting with a different image and then cutting to a shot of a famous actor looking into the camera with a neutral expression (see Figure 7.1). The first sequence started with a bowl of food, the second with a small girl in a coffin, and the third with a woman smiling as she sat on a divan. Each sequence ended with the same shot of the actor looking into the camera, his expression unchanged. The audience, however, interpreted the actor's emotions differently for each shot: they praised his ability to express hunger, grief, or lust, depending on which clip they were shown. They constructed a story in their minds about what was happening.

Words are one of the oldest and most familiar means that we have for telling stories. Whether through spoken dialogue or exposition and narration, words can be used to tell stories in film, comic books, or games, but part of the beauty of these forms of communication is that they can also tell stories *without* words, as in Kuleshov's experiments, or in comic books like Chris Ware's *Acme Novelty Library,* which features many scenes and pages without words. Comics and films can convey elements of stories and allow the audience to construct stories through the juxtaposition of images.

Stories have been part of human culture since the origin of our species. Games are a little bit newer but have been around since the dawn of civilization, for many thousands of years, and add something else into the mix: a system of moving parts, made up of all the elements of vocabulary you've learned about in this book. When a player acts in a game, she pushes into the game's shape of resistance, and the game responds by pushing back or opening more possibilities and spaces for her to push into. In this process, a potential fragment of a story emerges, much as it does when a viewer sees juxtaposed images in a comic book or film. Once again, it's up to the audience—in this case, the player—to interpret what the emerging story might mean. For example, what does it mean that a pawn becomes a queen if it reaches the other side of a chess board? What does it mean that Mario can jump on enemies to squash them? These elements may not sound like the typical building blocks of a story, but they're parts of what the creators of chess and *Super Mario Bros.* (1985) have put into those games for players to find, explore, and interpret.

As we've discussed in previous chapters, this process is a conversation between the player and the game. On one side, the ideas and vocabulary elements built into a game by its creator, and on the other, the experiences and choices the player creates by interacting with it. In some cases, the possible stories that emerge from a game depend largely on what the creator chose to put into the game. These stories often resemble the kinds of stories that are told through

novels, comics, or films. At the same time, just as with all forms of communication, the audience plays a huge role in what the game has to say through their own interpretation—perhaps even more so, since it's up to the player to decide how to push into the system, make choices, and pursue goals. As a creator, it's worth thinking about what you're putting into the game, how players will interpret it, and what they can do with the story.

Figure 7.1 Editing experiment of Lev Kuleshov.

This chapter explores the varied and challenging terrain that lies at the intersection of story and games. Game developers have been deliberately exploring this intersection for decades, trying to understand whether the interactive systems that make games work could be a vehicle for storytelling that's as rich and powerful as novels or films, and which might have unique

qualities of their own to offer the world. On the other hand, "decades" is not a long time in the history of storytelling! There's a lot left to figure out, and many awkward hybrids between traditional storytelling and notions of what games are or could be. Some game designers prefer to avoid overtly trying to tell their own stories through games at all, seeing the mixture of the two as oil and water.

As we continue to investigate the challenges and pitfalls of storytelling in games, we'll use two primary ways to think about the intersection of story and games:

- **Authored story**—First, games can carry stories. The experience of playing a game can involve many different elements that help convey a story: images, animations, words, sounds, and even the rules and processes of playing a game. By shaping these elements, the creator can tell a story. This kind of story, deliberately told through the experience of a game, is sometimes called an *authored story*. Let's take a game we've already discussed as an example. The authored story of Janet Jumpjet is described in Chapter 2, "Verbs and Objects." Mysterious, long-dormant robots left behind by an ancient civilization have awakened in the mines of Venus and taken human workers hostage. It's up to Janet to explore the mines, rescue the hostages, and incapacitate the robots. Janet is clearly the main character of the story: it's about what happens to her and what she does, even if she never says a word (much like some other well-known video game protagonists, such as Mario and *Half-Life*'s Gordon Freeman).
- **Emergent story**—Second, games can generate stories. The experience of playing a game, the push and pull of the player and the system, can generate a story that's worth telling, just like a good conversation can. You can think of this as a story about the playing of a game. These stories are often unique to each player; sometimes this kind of story is called an *emergent story*. The emergent story of a game is what we've spent a lot of this book talking about: it's the experience that the player has while learning the game, exploring and understanding its system and spaces, and perhaps mastering or completing the game. The player's story involves learning how to use verbs, which in Chapter 2 were described as the main characters of the emergent story. An emergent story is about exploring and figuring out systems with goals and rewards, deciding what to do, and often repeating one aspect of the game to understand it better and develop skills. The experience of flow and resistance is part of the player's story, and it may be different for each player. In this chapter we'll look at some different takes on emergence in game stories: *interpreted story*, a kind that hovers between authorship and player experience, and *open story*, which is more purely player-driven.

Which kind of story is better? Once again, there's no right answer to that question—it all depends on your goals as the creator of a game and what drives you to create and experiment. Do you have a story of your own that you want to tell? Or do you want players to be able to discover their own stories through the act of playing? Are you interested in blending the ways we have to tell our own stories with the unique qualities of games—the ways that playing games lets players push into, interpret, act on, and perhaps even change the story they experience?

The rest of this chapter explores the techniques that game creators have used to tell stories, along a broad spectrum ranging from authored stories to emergent stories. You don't need to think about that range as a single, straight line, however. There are almost as many ways to tell stories as there are stories to tell, and the intersection of storytelling and games is still relatively unexplored. Maybe you'll discover an innovative way to tell stories or a way for new stories to emerge from systems at play.

Authored Stories

At the beginning of the authored end of the spectrum, it's easy to understand what we mean by *story*. In its most straightforward, traditional form, a game's story doesn't need to be too different from the kinds of stories we experience through novels, comics, or films. It has a plot with a beginning, middle, and end, with characters who develop, experience conflicts, and perhaps resolve those conflicts.

If you want to tell a story in a way that resembles the forms used in less interactive media, there's an awful lot of material out there already about how to proceed. You can find numerous books written for creators of novels, films, and comics that explain how to create a five-act structure, develop interesting characters, and incorporate concepts like "The Hero's Journey" (a multistage plot structure that's often found in the telling of grand, mythic adventures). All this material can be useful for games as well, in the sense that telling a compelling story can draw on similar techniques regardless of whether that story's intended to be experienced in a movie theatre, while reading a book, or through the playing of a game. These traditional storytelling concepts are beyond the scope of this chapter, however, since we're going to focus primarily on the unique intersection of game systems and story.

As the creator, everything you put into your game informs its story: all the components of context, for example, as discussed in Chapter 4, "Context." The way a game looks and sounds can be the building blocks of the setting of the game, elements that help the player understand what's going on. Elements of context can be simple enough to give a hint of story without being explicit, as in *REDDER* (2010), which uses few words and many simple, pixelated images to create the feeling of exploring an alien planet. The ground looks red and arid, and you have to wear a spacesuit as you explore the world. Your ship is lacking crystals, which can be found deep in the tunnels that teem with dangerous machines (see Figure 7.2).

On the other hand, by pouring in a lot of context, it's possible to create an extremely rich world with a history, nations of different peoples, and political struggles, even if the player doesn't directly interact with or witness them, as in games like *Skyrim* (2011), which feature hundreds of books and artifacts that the player can read and inspect to learn more about the fictional setting of the game. The system of the game, its rules and verbs and objects to interact with, have a huge influence on the story, as we'll discuss soon. First, let's look at some ways that game creators have told authored stories in games.

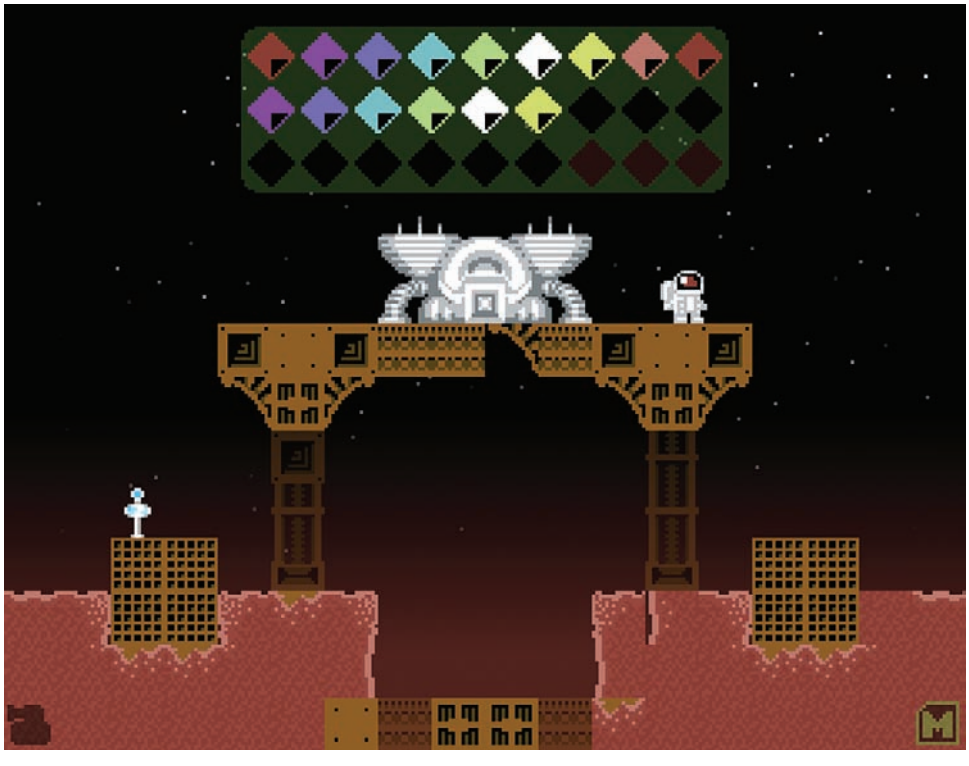


Figure 7.2 Setting the scene in *REDDER*.

Story as Intermission

The term "cutscene" is often used to refer to sections of a game experience that are used solely to show noninteractive elements of a story to the player. The first games that cutscenes appeared in were arcade games like *Pac-Man* (1980), and they were simple animations that appeared after completing certain levels. *Pac-Man*'s cutscenes are short vignettes that show the characters of the story wordlessly acting out moments that the player will recognize from the game: the ghosts chase Pac-Man, who powers up and chases them back.

Interestingly, the original *Pac-Man* cutscenes let the creators express ideas about Pac-Man and the ghosts that never appear in the rest of the game. In one scene, Pac-Man grows enormous to show that he's powered up and able to chase the ghosts; in another, the red ghost gets caught on a nail that tears his red covering, revealing a pink leg and foot underneath! These scenes were often referred to as *intermissions*. After pushing hard to complete a level, the player was given a break with something amusing to watch before continuing on.

The year after the original game came out, *Ms. Pac-Man* (1981) was released as a sequel. Now the intermissions were called *acts*. They told the story of how Pac-Man and Ms. Pac-Man met, fell in love, and had a child. Since then, cutscenes have grown more and more elaborate. In big-budget games, they sometimes add up to hours of video footage with dozens of characters, and they are often written by screenwriters with experience writing film scripts. Ironically, in some cases it's not even necessary to play a game to experience these stories: enterprising players have stitched together all the cutscenes for story-heavy games like *Uncharted* (2007) and its sequels so that they can be watched on sites like YouTube as if they were films. The only sections of the authored story that a viewer ends up missing are the "action sequences" that comprise the system of the game. In the case of *Uncharted*, the action sequences are moments where the protagonist is shooting at enemies, climbing up walls, or solving puzzles to open gates.

The ability and desire to watch the entire story of a game without actually playing it raises a question: is a game the best way to experience an authored story like this? When the story carried by a game is mostly or entirely told through cutscenes, it's almost as if they exist alongside each other; the player takes a break from one to experience the other, bouncing back and forth between playing the game and watching cutscenes. Is there a benefit to letting a game system and a story work in parallel, taking turns in the spotlight?

Some game creators prefer keeping story and game relatively separate, letting each part stand on its own merits precisely because of the difficulty of intermingling the two, as we'll see in the rest of this chapter. There's definitely something that's left untouched with this method, however: the interesting, complicated possibilities that arise if you mix story and game.

Still, a game can benefit from the presence of a story alongside it, if only to provide breaks in resistance. Story can show what kind of imagined world the game exists in and help make sense of what's happening—or even help create a feeling of nonsense or humor, for some games. In one of the first games I designed, *Egg vs. Chicken* (2006), I told a story through a series of comic-book pages that appear before and after each major section of the game. The game itself involves defending a series of fortresses against an oncoming army of chickens, who the player can defeat by flinging groups of eggs at them. The context of this challenge—eggs being used to defend against chickens—was clearly and deliberately surreal and nonsensical, so we decided to tell an equally ridiculous story to help explain it.

The comic-book pages star a group of four revolutionary eggs who refuse to hatch into chickens. Menaced by a chicken police force, they escape in a time machine to try to uncover the answer to the age-old question: which came first, the chicken or the egg? Like the ghost's uncovered leg in *Pac-Man*, the four revolutionaries never appear in the game, but they help explain—in a manner just as wacky as the basic concept—why the enemy chickens the player encounters look like greedy nineteenth century industrialists, then medieval knights and

archbishops, and eventually Egyptian pharaohs. These images and animations, elements of context that appear during the play of the game, tie in with the noninteractive story elements that appear in the comics, creating a simple, silly world. The context isn't strictly necessary for the game system itself. We could have also created a similar game using abstract enemies and defenders rather than chickens and eggs, but together, all the pieces of context and story create an experience with a uniquely ludicrous flavor.

Story as Exertion

Cutscenes like the animations of *Pac-Man* or the comic-book pages of *Egg vs. Chicken* show up when the player's successfully completed a certain level, at a moment of rest and reward. They don't necessarily have much to do with what the player just accomplished, which is why they often feel like an intermission. As short breaks from play, they often just involve watching or reading the story rather than interacting with a system. Because games almost always involve some level of interactivity and choice, it's tempting to get the player more involved in the story. Could the player feel more like they're part of the story? What if the story couldn't proceed without them?

Adventure games, which started to appear in the 1970s and 1980s, usually put the player in control of a character who has to overcome various challenges to progress toward the end of the game. The earliest adventure games, like *Adventure* (1979) and *Zork I* (1980), had little in the way of story; the nameless adventurer was simply trying to find and collect every valuable treasure in the game world. Later adventure games started to provide more context in the form of other characters to talk to, who often give the protagonist quests to undertake. In *King's Quest* (1984), the main character is a knight trying to find three treasures; along the way, he meets a woodcutter whose wife is starving. This dilemma can be solved if the player finds a magic bowl that can create food, which fortunately happens to be lying on the ground not far away. In exchange, the woodcutter gives the knight a fiddle, which turns out to come in handy later for dealing with some angry leprechauns who can be pacified through music. By finding the correct solution to these problems and pushing into the game's resistance, the player participates in the story and moves it forward.

Since then, many kinds of games have given players quests and undertakings to complete and drive the story of the game forward. In massively multiplayer online role-playing games (MMORPG) like *World of Warcraft* (2004), players experience sections of a particular character's story piece by piece as they complete various tasks: locating missing items, slaying dozens of nearby enemies, carrying a message or package from one place in the game world to another. In single-player role-playing games, it's often necessary to grind through many enemy battles in the game's combat system before reaching the next area of the game where the protagonist can talk to new characters and find out what happens next in the overarching plot. Even social games such as *Farmville* (2009) and *Cityville* (2010) have realized the lure of storylines, creating

quest structures that give players a series of tasks and a snippet of conversation with characters between each task (see Figure 7.3).

Figure 7.3 Beginning an early quest in *Cityville*.

Like the intermissions of *Pac-Man*, these pieces of story alternate with periods where the player pushes into the systems of the game and focuses on playing rather than experiencing a story. There's a crucial difference: the core gameplay of *Pac-Man* doesn't present the story as a rationale for why it's necessary for Pac-Man to elude and devour ghosts while collecting every yellow pellet in the level. The intermissions appear as a break in the action rather than as its culmination and fulfillment. In games with elaborate series of quests, on the other hand, story is the driving reason why the player needs to push against the challenges of the game.

Although this creates a tighter integration between story and game systems, there's a risk. As we discussed in Chapter 6, "Resistance," the lure of rewards can come to eclipse the pleasure of the processes of play that lead to those rewards. If the player is more concerned with simply seeing the next part of the story, and if the act of playing and dealing with the challenges along the way isn't an interesting enough conversation in its own right, then playing can become a chore. When playing involves repetition of low-skill, already-mastered activities (grinding), the problem becomes worse. It's no wonder that some players who get bored with grinding, or frustrated by difficult puzzles and challenges, end up looking online for videos of the story, preferring to go directly to the reward.

When we interact with and experience story, there's always a little bit of effort required of us as a member of the audience. At the very least, we have to keep our eyes open and pay attention in a darkened theater, or turn the pages of a book. Games, on the other hand, sometimes require much more labor and time to experience the stories embedded within them. The investment of effort can create emotional attachment because the players are involved in a much more significant way than if they were simply turning pages; they've worked hard to see the story proceed. This investment comes with risk, however. If the story disappoints in the end, players can feel let down, even robbed.

Even simpler forms of interactivity can become onerous when they end up feeling like barriers to the next section of story. Some of the games of the *Resident Evil* series feature *interactive cutscenes* that require the player to act in order for the scene to progress—in theory, bringing the player more directly into the drama. The cutscenes of *Resident Evil 5* (2009) include moments where the player has to press a button to leap over a pit, dodge falling pillars, or avoid enemy attack. If the player doesn't press the correct button within a short time limit, the cutscene shows the protagonist of the game dying a horrible death, and the player's forced to repeat the cutscene. This technique certainly requires the player to be "involved" and pay close attention for the crucial signal to quickly press the correct buttons, but many players express extreme annoyance at having to repeatedly watch these scenes when their timing was slightly off.

Story as Exploration

Another method of weaving story into a game sidesteps some of the issues of cutscenes: rather than alternating between a period of playing in the game system and watching another section of story, many games incorporate story elements that are available to players but not necessary to move the plot of the story forward. Instead, they're optional. We've already mentioned games like *Skyrim* that detail vast fantasy worlds with long histories through books that players can read and objects they can examine. Many role-playing games that try to create the experience of traversing a rich world also include characters that the player can talk to—or at least use a "talk" verb that produces some dialogue giving the player more details about that character or the setting. These story elements are similar to the more open spaces of resistance discussed in Chapter 6. They're there for players to push into and explore, or move past and ignore as they see fit.

This kind of optional story material is sometimes referred to as *lore* because it frequently fills in the backstory of a game world. Lore tends to play a supporting role in the story of many games rather than creating a series of events that unfold into a plot for the player to follow. Pieces of lore provide additional flavor for players who are interested in knowing more about the setting and who want to pursue the background and creative expression that make up the game world.

Many games use exploratory story elements alongside a more traditional plot, expressed through the kinds of techniques described in previous sections. In other games, entire subplots are available as *side-quests*, which are not necessary for completing the game but create an option for a player to find out more about particular characters or aspects of the world. The more important these optional stories are to the narrative of the game, however, the less players are likely to see them as being purely optional. It's hard to pinpoint exactly when an extra side-quest goes from being something you can ignore to a piece of the plot that's vital to the experience.

It's possible to play Benjamin Rivers's horror-mystery game *Home* (2012) without discovering many fragments of story that are scattered along the way. In *Home*, the protagonist is an amnesiac who wakes up in a mysterious house next to a freshly murdered corpse. If the player explores thoroughly while guiding this amnesiac through tunnels, locked gates, and dark forests back to his house, she can piece together dozens of clues that add up to some kind of explanation of who the murderer is and how the protagonist came to be there (see Figure 7.4). On the other hand, the player could just focus on getting home. The decision of how thoroughly to explore, and what to make of all the fragmentary evidence, is left up to her.

Figure 7.4 A mysterious discovery near the beginning of *Home*.

Story as Choice

All the techniques we've described so far in this chapter give players different ways to experience an authored story. Some moments of story feel like rewards for progressing through the game, while others might be optionally available through player choice. Because a game resembles a conversation with push and pull between the player and the system—perhaps more so than other forms of storytelling—it's natural to ask whether the player can make meaningful choices that affect the story itself. If we're trying to blend the craft of storytelling with the craft of game design, why not let the player change the course of the authored story?

The emergent story of her experience would then be affected by the choices she makes, by how she pushes into the game's system.

Even before the beginnings of digital games, the idea of stories with choices was being explored in written literature. The 1941 short story "An Examination of the Work of Herbert Quain," by Jorge Luis Borges, describes an imaginary novel structured like a branching tree. After the first chapter, each of the next three chapters relate a different version of subsequent events, and each of those chapters diverges again into three more, so that the story has nine endings. Whereas a conventional narrative can be read linearly from beginning to end, this kind of work less resembles a straight line than a tree with many branches (see Figure 7.5).

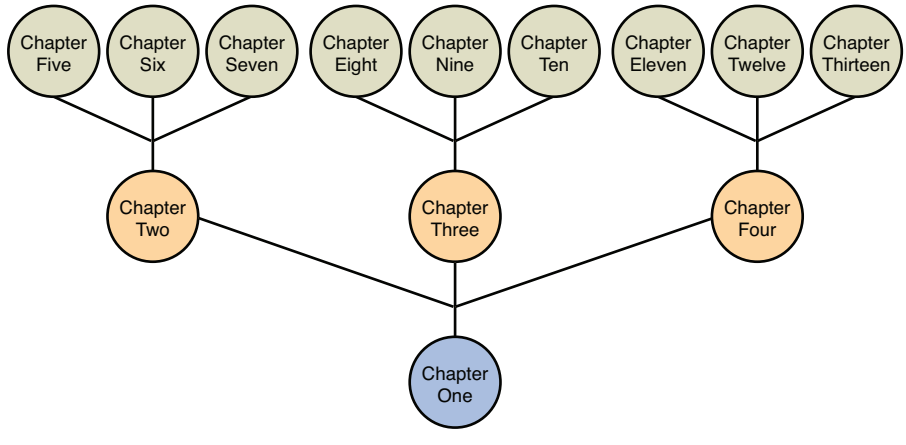


Figure 7.5 The structure of an imaginary novel described in Borges's "An Examination of the Work of Herbert Quain."

A few decades later, this narrative structure had started appearing in actual books—the best known being the *Choose Your Own Adventure* series, marketed to kids. These branching narratives were sometimes called *gamebooks*, for obvious reasons. They're systems that players push into by making choices, often with the aim of reaching a happy ending, at least in the early books. It's no surprise that videogames started to use branching stories as well, and for increasingly adult subject matter. In an inversion of the gamebook label, a game where the primary verb involves "choosing" which branch of the story to pursue next is sometimes called a *story-game*.

Text-only story-games like Zoe Quinn's *Depression Quest* (2013) and Anna's own *Encyclopedia Fuckme and the Case of the Vanishing Entree* (2011) carry on the tradition of gamebooks in digital form. The story is told in the second person, describing how you, the reader, experience scenarios that involve choices. After reading a section, these games ask the player to make a choice. When you visit your girlfriend's house, will you eat dinner or make sexual advances?

When you wake up feeling depressed, will you struggle to get your work done or give up and do something else? In *Depression Quest*, some of the choices presented to the player are visible but unavailable, a way of conveying the limiting feeling of depression (see Figure 7.6). These kinds of choices—and even absence of choices—can feel powerful for players because they can potentially affect the resolution of the story. It's exciting to imagine that, through picking one option or the other or figuring out how to unlock all the possible choices, you can change the course of events, creating a sad ending or a happy one for the characters involved. At the same time, it's worth noting that nearly all branching stories are still authored stories. Although they have many paths that a player can explore through her actions, all the paths have been placed for her to discover.

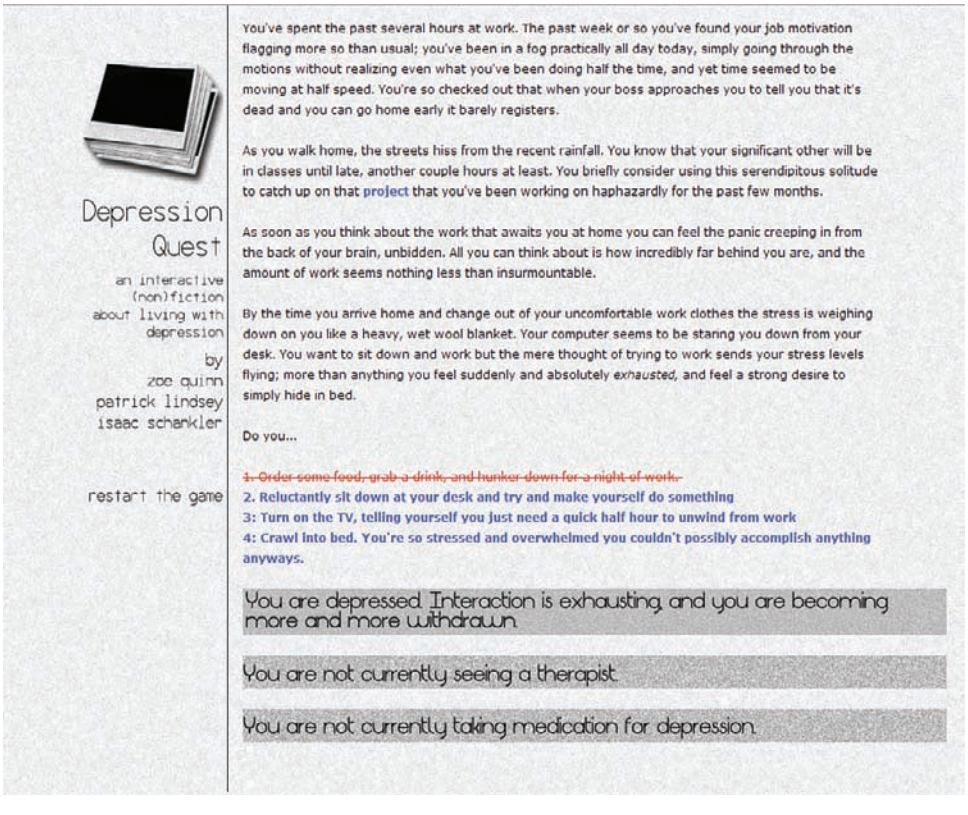


Figure 7.6 Facing a decision in *Depression Quest*, including one unavailable choice.

Game creators sometimes talk about how we give players the *illusion of choice*. The conversation of a game is based on a structure we've created, and even the verbs at the player's disposal, the ways she has to push into the system and mold her own experience, are predetermined.

When you play a story-game and explore every branch, replaying or returning to the same moments to make different choices, you can experience the totality of what the creators have made available to you, but it's still a constrained choice.

These choices are a little bit like choosing what to eat off a restaurant's menu. You can't simply make up your own dish, but the availability of options is still meaningful since it lets you pick your own experience, and not everyone has the same taste. A branching story that lets players express their own preferences allows each person to look for the story *she* wants to pursue, within the branches that you've provided. Many people enjoy sweet, happy endings, just as many of us like sweet food (especially at the end of a meal), but players of games with multiple story endings can choose whether they want a happy ending or not. Some might choose different paths simply to see what happens, or because certain decisions simply work better within the narrative logic they perceive or bring a satisfying resolution to the struggles of the characters.

Sometimes it's nice to choose what kind of story you'll experience, just as it's nice to choose what to eat for dinner. At other times, we can take pleasure in leaving those decisions to someone else, as in a dining experience with a set menu, or a traditional story without branches. Telling a good story is a skill, and crafting a multibranched story is a particularly complicated task for that skill. For one thing, branching stories simply have more story and naturally end up involving many more words, animations, or video to tell the story than a linear story that recounts one version of events. As a result, many creators of story-games try to limit the amount of effort and complexity involved by creating small branches—choices that cause the story to move off in one direction but reconverge into the main plot, or choices that happen near the end of the game, where a change in the plot can feel significant without involving a huge amount of additional storytelling.

Creators of story-games employ many kinds of techniques to create branching structures. One structure, sometimes called a *shrub*, gives players choices at every turn, resulting in many possible variations that continue to branch—and sometimes lead to abrupt or unresolved endings. (If you've read classic *Choose Your Own Adventure* books, many of the dead ends in those books involved your character dying.) A structure with *reconverging branches* tries to solve the issue of endlessly branching shrubs, which also require a lot of creation of plot and writing by having the branches split up and come back together. This sometimes means a relatively linear plot with a single resolution but multiple paths to reach the final outcome that can differ in their details and feel. Finally, many story-games end in a branching set of choices that result in different endings—a practical moment to create multiple branches, since each one ends and doesn't need to continue branching or extending (see Figure 7.7).

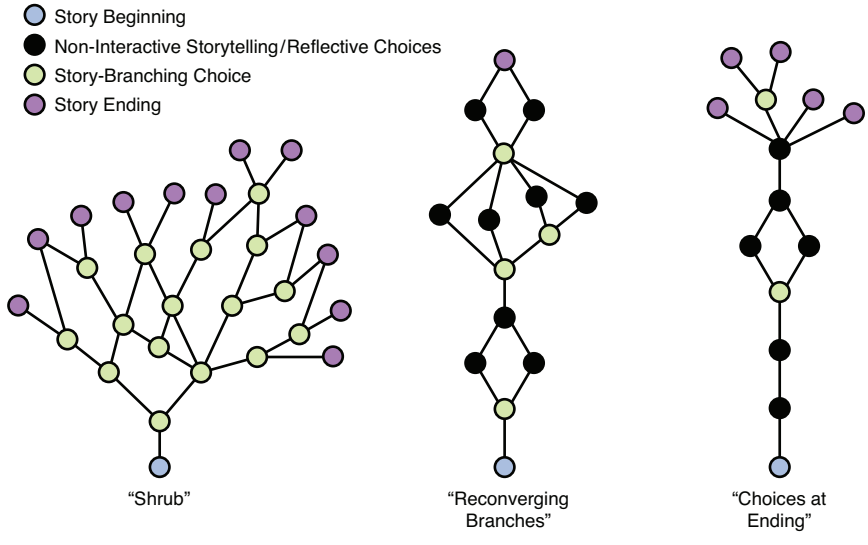


Figure 7.7 Possible structures emerging while assembling a branching story.

Besides the quantity of story, there's the question of quality. No matter the path taken, we want our stories to feel satisfying, with a plot and outcomes that make sense to the player's intellect and imagination. In 2005, I was a game designer at a studio called Gamelab, working on the aforementioned *Egg vs. Chicken.* Another team at Gamelab was developing *Plantasia* (2005), in which the main character is a faerie who helps a gardener bring flowers into bloom. *Plantasia*'s designer, Nick Fortugno, originally intended the game's story to culminate in a branching choice: does the faerie pursue a relationship with the gardener or leave the garden behind? Ultimately, he decided that a choice *wouldn't* make sense in this story; the things the characters had gone through, the way they had developed and the events of the plot, all pointed toward a relationship as the ending that made the most sense. It wouldn't have been hard to create an alternate ending where, despite those cues, the faerie decided to avoid romance, but in Nick's opinion, it would have felt like an awkward, unsupported branch hanging off to the side, a vestigial limb.

By contrast, Emily Short's game *Floatpoint* (2006) has a story constructed to lead up to a crucial moment—again, near the end of the game. The main character is a human diplomat who's an envoy to an alien planet. Through playing, the player discovers that this diplomat must make a single choice that will send a message to the aliens and determine the course of their relationship with humanity. Much of *Floatpoint* is spent exploring an alien city to learn more about

its culture, history, and the current situation. Although it's not a huge world, there are many significant yet optional story elements that can be discovered through exploration and puzzle-solving—or avoided, leaving the player with less understanding and a simpler set of options in the final choice. The striking thing about *Floatpoint* is that Short constructed the plot around a single turning point—a moment in history where one decision, flowing from limited snatches of cross-cultural understanding—makes a huge difference to the course of the future. As a result, the various outcomes of the diplomat's choice (which can be experienced by replaying the pivotal scene) all feel like they flow naturally from the events of the story.

Story as System

So far, we've talked mostly about games that tell stories constructed out of the same materials used to tell stories in other cultural forms. We can make stories out of words, cartoon characters with speech bubbles, moving pictures. We can even use human actors for some kinds of non-digital games experienced at live events! As discussed at the beginning of this chapter, games have something unique to offer beyond traditional storytelling materials: a system of verbs, scenes, and rules that the player can push into and learn. Beyond the straightforward offer of a choice of which path to pursue next, the vocabulary of a game's system can help convey a story through its very structure.

Many of the oldest games in history express something about the world and how it works through systems and mechanics. They often contain stories about overcoming conflict, solving problems, and pushing through difficulty to overcome a challenge. Chess can be seen as a story about two equal and opposing forces, each trying to defend a crucially important figure (the king). Mancala, an early board game from Africa, represents the cycle of planting seeds and harvesting. It's crucial to decide what plot of land to take seeds from to start the next cycle. Games not only describe how certain scenarios work in our world—by creating a system that players can push into through the use of certain verbs—they also say something about what kinds of actions are important in those scenarios and what kinds of decisions make a critical difference.

As discussed in Chapters 1 and 5, Anna's game *dys4ia* (2012) tells a story based on her own experiences as a transgender woman who decides to take hormones and shows how her own feelings, relationships, and ways of moving through the world are affected. *dys4ia* conveys this story in part through words and pictures that explain what's going on in various scenes and facets from Anna's life. At the same time, much of Anna's experience—what it felt like to be her, in those situations—is also expressed through game mechanics (see Figure 7.8). These systems convey something that words wouldn't on their own. In one scene, the player moves a pixelated character toward home while the text explains that medication has made her exhausted. The system shows this as well, creating more and more resistance to the player's movement across the screen, slowing it to a crawl, and making the experience of exhaustion tangible.

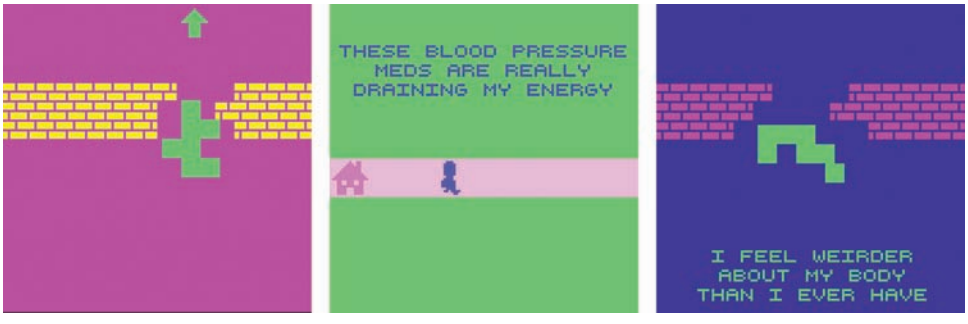


Figure 7.8 Scenes from *dys4ia* that express different aspects of Anna's experiences through system as well as images.

In other scenes, the player controls an object similar to a *Tetris* piece that's trying to pass through a gap in a wall but is strangely shaped and doesn't fit. Even without a more literal representation of a human character, Anna's experience of feeling wrong and awkward comes across clearly, especially as this scene recurs. Eventually, this aspect of the story reaches some measure of resolution. The player gets a chance to struggle against the wall and knock holes in it, and a brief scene at the end of the game holds out the promise that even an ever-changing object can make it to the other side.

Other kinds of messages can be conveyed through game mechanics as well. *The Best Amendment* (2013) is Paolo Pedercini's response to the National Rifle Association's assertion that "the only thing that stops a bad guy with a gun is a good guy with a gun." The game seems simple at first. In each level, you maneuver a white-hooded character to collect stars, sometimes by shooting black-hooded characters (see Figure 7.9). As the levels progress, the black-hooded characters start shooting as well, creating a dangerous playing field. Eventually, something starts to feel familiar about the movements and shooting of the black-hooded "bad guys with guns": their behaviors turn out to be recordings of how the player moved and fired on previous levels! A scene that initially felt like a dangerous, chaotic firefight with a number of dark enemies turns out to be the result of your own actions while pursuing the goal (collecting stars) set forth by the game. *The Best Amendment* doesn't explain its message with words, but in authoring a system that produces certain kinds of experiences when played, Pedercini gets an idea across: who's a "good guy with a gun" depends on your perceptions, and running around shooting apparent "bad guys" leads to a general bloodbath.

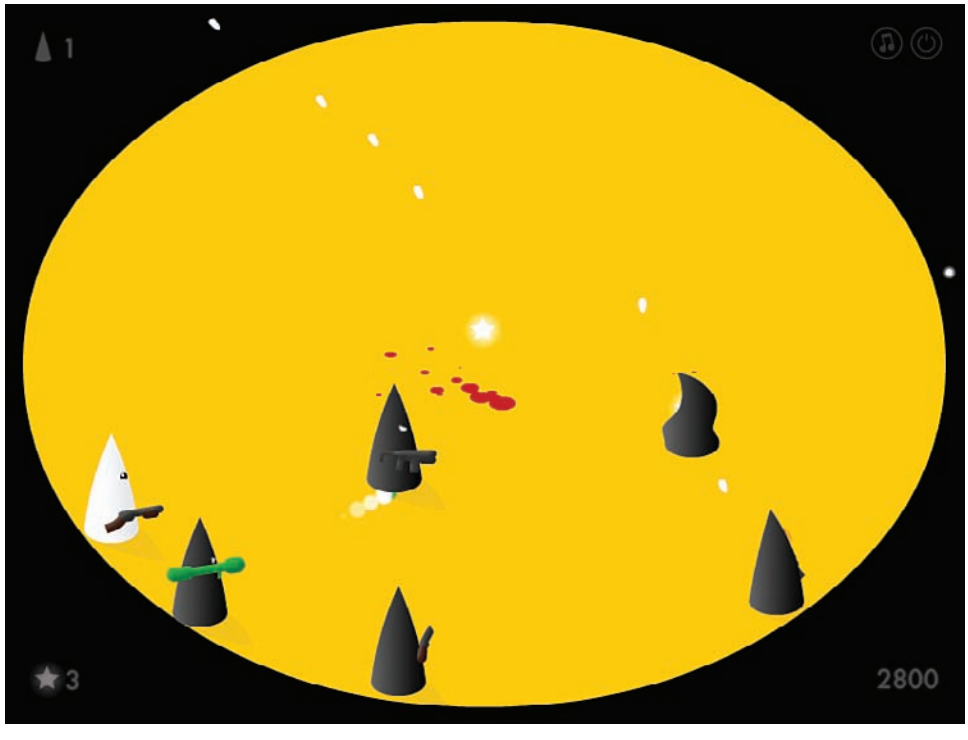


Figure 7.9 *The Best Amendment* conveys a message through game mechanics: who's actually good or bad?

Interpreted Stories

Whether expressed with words, images, and characters or through the vocabulary of a game's system, games let us tell an incredible variety of authored stories. Game systems give us ways of getting a player involved in the progress and evolution of an authored story. The player might simply be pushing the story forward or deciding which branch of the story to explore. She may even discover parts of an authored story embedded in the game's system by figuring out how it works and understanding what it has to say. In the previous chapter, we talked about how games can open up the shape of resistance to allow players to pursue their own strategies, even create their own goals. Can we open up stories in a similar way to get players involved not only as audiences for our authored stories, but as storytellers in their own right?

Each time a game is played, an experience results, generated by the player's own process of pushing into the game. In many games with authored stories, this experience is relatively predictable; it may feel like nearly the same experience if played again. The process of playing

dys4ia is likely to be relatively similar for most players, and intentionally so: *dys4ia* aims to convey the lived experiences of a real person—events and situations that actually happened. It wouldn't make sense for players to insert their own stories into *dys4ia* any more than it would be polite to interrupt an oral storyteller's recounting of events in her life. Like spoken conversations, games can offer players moments to listen and moments to participate more actively; both are valuable in their own way.

Not all games aim to convey personal experiences, of course; some games create open spaces for players to define their own style of play or agenda, and these games can result in a huge variety of experiences as players interact with them in different ways. Open games we discussed in the previous chapter, such as *The Sims* (2000) and *Minecraft* (2009), produce different emergent stories for each player, depending on what those players seek to do in the game system and how they push against and into the rules. Because those rules create a structure that's the same for every player, there are many commonalities. Each player's emergent story is composed of some consistent building blocks that can be arranged in many different patterns.

Interpretation

Even at the authored end of our story spectrum, there are many ways to open a game's story to the creativity and ingenuity of players. Traditional notions of an authored story describe a story as if it's an object that's simply delivered from the author to the audience, but those notions aren't the only way to think about how a story comes into being. The Kuleshov Effect shows how powerful interpretation is. Depending on how elements are arranged, an audience may have wildly different interpretations of the same expression on an actor's face.

Of course, interpretations of a story can vary even if two people experience the same story. That's why we can argue about exactly what happened in a play like Shakespeare's *Hamlet* after seeing it with our friends. Was the ghost of Hamlet's father really a supernatural manifestation or some kind of hallucination or trick? Was Hamlet justified in his plot to murder his uncle, and was he pretending to be mad as his dialogue states, or did he truly become unhinged? Shakespeare may have had his own interpretation of what happened, but the author's views don't need to be the last word on the meaning of the play. Rather than thinking about a story as something produced entirely by an author and handed over to the audience, what if we thought of a story as coming into being at the moment of interpretation?

Games lend themselves well to ambiguity, in part because players can become conscious of the way their actions in the game create a different experience, even slightly so, from those of another player. Using the same techniques and tools as stories told in other cultural forms, game creators can leave elements of the story untold—mysterious and open to interpretation. *Home* is a great example that we've already discussed. The story is a mystery that can be seen in many different ways, and although the authored content stays the same, each player may find

more or fewer pieces of the story depending on how thoroughly she explores. Benjamin Rivers, the game's creator, deliberately left the story of *Home* completely open to interpretation and even prompts players to actively come up with their own ideas. After completing *Home*, players are asked to visit the game's website and fill out a form to submit their own explanation of the story. In the month after the game's release, dozens of possible interpretations accumulated (see Figure 7.10).

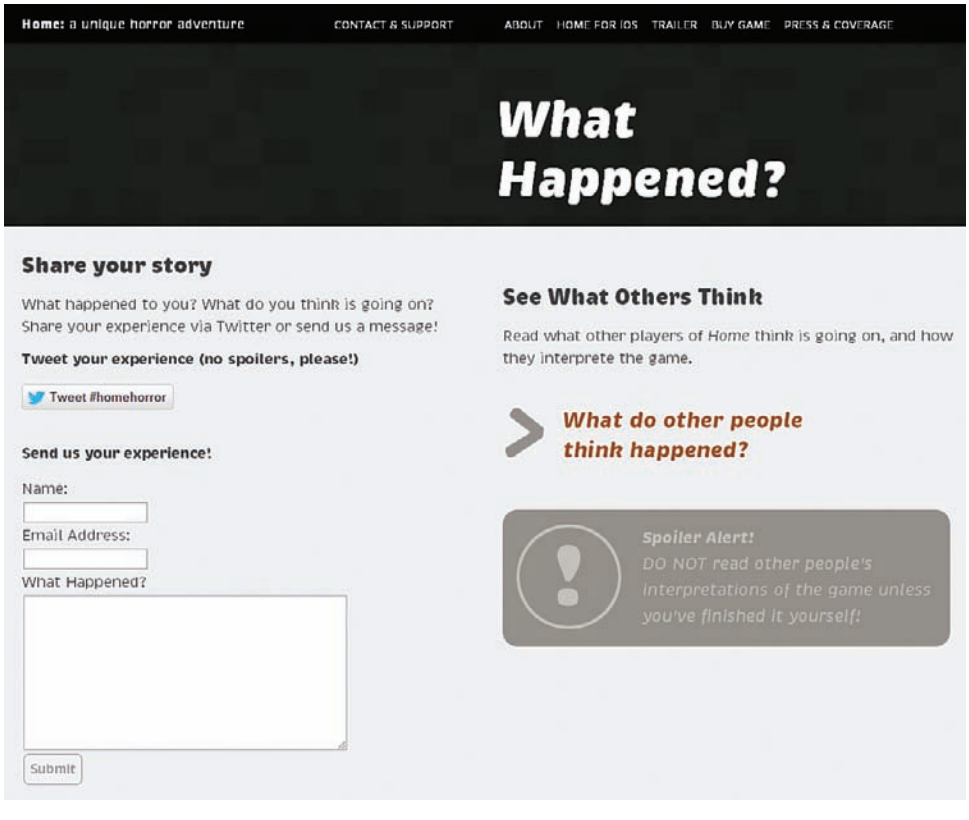


Figure 7.10 *Home*'s website asks players to provide their own interpretation of the story.

Stories told through game systems lend themselves well to interpretation, precisely because they're partly told without words: players experience this aspect of a game's story by pushing into a system, using verbs and seeing what happens, and feeling out the contours of how the game works. Although the mechanics may stay the same, different players may reach different understandings of what the systems are trying to say.

Pipe Trouble (2013) is a game where the player is tasked with laying sections of natural gas pipeline in rural and suburban areas of Canada. On either side of the main playing area, two characters express disapproval when things go wrong and let you know how happy or unhappy

they are with how you completed the pipeline. On the left side, a farmer complains when the pipe runs too close to fields or scares farm animals; on the right, a businessman gets angry if the pipeline isn't completed fast enough or uses too many pipes and goes over budget (see Figure 7.11).

Figure 7.11 Getting feedback after laying pipelines in *Pipe Trouble*.

It's difficult to keep both sides happy, especially considering that other factors appear during play: protesters show up to block your pipeline if you build it over forested areas and may even sabotage your pipeline by blowing it up if you're too reckless with the environment. Although *Pipe Trouble* is clear about whether the farmer and businessman are pleased, it's up to the player to decide on the right way to play. Is the best measure of success to keep everyone happy? To save the most money possible? To let the businessman get upset but make sure not to disrupt farms, homes, and forests? The many possible interpretations of the right way to play are useful for creating a complex systemic story. It's hard to keep everyone happy, and a natural gas pipeline ends up affecting someone negatively, whether it's a forest animal or a tycoon's checkbook.

The branching narratives of story-games can also create interesting forms of ambiguity, especially when the system underlying the impact of the player's choices isn't completely revealed

to the player. In Emily Short's *Bee* (2012), the player makes choices about how to use the time of the main character, a home-schooled girl who's studying to become a spelling-bee champion. At the beginning of the game, the player is shown two attributes that are affected by her choices: study more, and the Spelling Skill attribute will rise, but the Motivation attribute will decrease. If Motivation is too low, some study-related choices become unavailable, but Motivation can be raised by pursuing other kinds of activities (see Figure 7.12).

Figure 7.12 Making choices in *Bee* sometimes affects the character's attributes, at left.

Although Spelling Skill and Motivation seem like the focal points of the game and its stated goal, other parts of the system are affected by the player's choices as well—most notably the relationship of the main character to her parents, sister, and other peers. Interestingly enough, many of the outcomes of the story—some of which describe what happens years later or involve abandoning spelling mastery altogether—are linked to what kind of relationship the main character has with others. *Bee* must be played several times to feel out the contours of this system to interpret what's going on in this girl's life. Although the game presents a straightforward narrative of overcoming difficulty, increasing skill, and mastering problems to become a

champion, the decisions that feel like they change the story suggest that the game may contain an underlying message about human relationships and life beyond the narrow constraints of a simple goal.

Reflective Choices

In talking about story-games, we've focused so far on choices and options that the player deals with to push the story down one branch or another, potentially even changing the ending of the story. It's understandable that these kinds of plot-altering choices get a lot of attention in discussions of story and games. When we're involved in an unfolding narrative or a system that we can act in, we want to know how we can make a difference. What do we have to do to make sure our favorite character doesn't die? Which choice spells the difference between a good ending and a bad one?

Plot-altering choices aren't the only kind of choice that contributes to a player's emergent experience of a game, however. Another kind of choice, which some story-game creators have started calling a *reflective choice*, has huge potential to involve the imagination, interpretation, and psychology of the player—even though these choices don't affect the plot of the story, or the state of the game, in any way.

Near the beginning of *Choice of Romance* (2010), a story-game about a young noble navigating the social intrigues, politics, and relationships of a royal court, the player is asked to make what sounds like a significant choice. The young noble spots a purple butterfly, said to bring good luck, and the player decides on a wish the noble will make. Will your character wish for money? Adventure? True love? Or to do something amazing that will change the world? If we assumed this choice was plot-altering, we can imagine what kind of impact it could have on the story. Maybe a noble who chose true love would be more likely later on to have an epic romance or even complete the game upon reaching that goal.

The truth, although it isn't revealed to the player, is that this choice doesn't affect anything else in the game or its story in any way, and that's what a reflective choice is: a choice that exists primarily to focus the player's attention on the act of making a choice. In *Choice of Romance*, the player can end up pursuing any or all of those goals, regardless of the initial wish, but that wish serves to make the player think about what's important, what she wants to pursue in this story or at least this particular playthrough of the game.

Reflective choices may seem like a deception or a cop-out, especially because they require far less work to build into a game than a plot-altering choice. That point of view, however, only makes sense if we're starting from the idea that every choice in a game *has* to affect the state of the game and that everything else is meaningless. An attitude like that does make sense for many games, where the focus of choice is on strategies that could create a winning or a losing outcome. In chess, deciding to point the horse-head of your knight forward or backward won't

help you win or lose; it's the opposite of an important choice. When the meanings and interpretations of a story become intertwined with a game, however, the act of choosing becomes more complicated. Not every action a character makes in a traditional story changes the fate of the world or means the difference between life or death, a good ending or a bad one. Instead, many things a fictional character might do or even think are reflections of who that character is, how she reacts, and what kind of person she is.

The Walking Dead (2012), a game based on the comic book series by the same name, includes many examples of reflective choices. The setting of *The Walking Dead* is the United States during and after a zombie apocalypse. The fates of many characters are extremely bleak in the comic book as well as the videogame and television series based on it. In the game, the player controls the choices of Lee Everett, a convicted felon who escaped during the outbreak and joins up with a group of survivors. In the second chapter of the game, Lee meets a woman who's been infected by a zombie bite; she's sadly awaiting the worse-than-death fate of becoming a zombie. The choice to make is awful: she wants you to give her a gun so she can commit suicide. The player must decide whether to give her one or refuse (see Figure 7.13).

Figure 7.13 Making a difficult and memorable choice in *The Walking Dead*.

This difficult choice turns out to be mostly reflective, because even if the player refuses, the woman grabs a gun and commits suicide. The experience of having to make this choice, however, was a memorable one for many players, despite not having any way to create a positive outcome. *The Walking Dead* is full of choices and situations like this, from deciding whether to bury a young boy's corpse to whether to listen to a dying man's final words. Reflective choices may come naturally to a story about survival in such a grim setting where there's no happy ending in sight. In such circumstances, the game seems to suggest, what's important is how you react to the harshness of reality and what kind of attitude you have no matter whether you can change the outcome or not.

If choices don't necessarily need to affect the outcome of a game or the course of its story, we can consider whether many kinds of apparently "small" choices can play a role in affecting the emergent experiences of a game's players. Many games give players the opportunity to decide what the character or avatar they control will look like—whether to play as male or female,

whether to be embodied in a game as someone who looks a bit like you, or an idealized version of yourself, or completely different. In some games, choices about appearance or gender of the main character have an impact on a game's systems. In others, the plot and state of the game are relatively unaffected. Still, the experience that emerges from watching an avatar die may be very different if that avatar looks exactly like you!

The *Mass Effect* series intersperses sections of exploring and shooting with moments where your character (Commander Shepard) converses with others, and the player frequently has to make choices about what to say. Some are reflective choices, while others drastically alter the course of the game's story, result in the death of major characters, or alter your character's attributes. The game's most significant and widely discussed reflective choice, however, may be the choice of the main character's gender, which the player makes as the game begins. Notably, the rest of the choices in the game aren't affected by the player's choice of gender. The character says exactly the same lines, simply voiced by a different actor. The experience of playing a female Shepard has been described by many players as being distinct and novel, however—perhaps because she has the same choices to react in all the same ways to the epic situations and conversations as her male counterpart would.

Emotional Resonance

Diner Dash (2004), another game developed at Gamelab and designed by Nick Fortugno, revolves around keeping customers happy. The busy waitress/manager of a restaurant, the main character, has to juggle seating customers, taking orders, delivering food, and handling customers' bills. The longer customers wait, the more impatient and upset they get, which is represented by a change in facial expression. In a study on *Diner Dash* players, game researcher Nicole Lazzaro found that this simple representation of emotion was effective in changing the feel of the game. Players experienced emotions of their own when the way they played resulted in happy characters or frustrated ones, because as human beings, we naturally respond to the feelings of others. Even a cartoon representation of an angry, impatient person will make us respond differently than an abstract timer.

Miss Management (2007) was a game that Nick and I collaborated on as a follow-up to *Diner Dash*; part of our goal was to further explore the intersection of character-driven emotion, stories in games, and the emergent story of systems. In *Miss Management*, the player directs several unique, authored characters—all coworkers in a busy office. Each character gets stressed out while completing work assignments, and each has her own likes and dislikes. One might want to microwave a snack to relieve stress, while another becomes even *more* stressed out by the smell of food (see Figure 7.14). It's up to the player to juggle these conflicting needs and complete a series of tasks in each level of the game that relate to those needs: make sure that Timothy spends a certain amount of time snacking, but don't let Tara get stressed out, and complete ten work assignments while you're at it!

Figure 7.14 Characters in *Miss Management* complain in cutscenes about their pet peeves (left), which then stress them out during gameplay (right).

Each level's tasks are introduced by scenes that play out in dialogue between the characters. Although these cutscenes are noninteractive, forming the authored story of the game, they also provide context and explanation for the player's challenges during the rest of gameplay. When a player decides to make Tara happier or let Timothy get impatient, those decisions feel more meaningful because of the way they're embedded in a longer story about the conflicts and development of these coworkers. Some of the tasks on many of the levels are optional: the player can earn extra recognition (in the form of a gold star, a simple cosmetic reward) for the difficulty of juggling all the tasks, but it's up to the player to determine how hard she can push and which goals feel important to her. The overarching theme of "Can you really keep everyone happy, and is it worth it?" also becomes the main question for the struggles of the story's main character (Denise, the office manager) by the end of the game.

Of course, not all characters in games have authored plot arcs and character development, but we don't necessarily need to use those traditional elements of story for players to care about the characters whose actions they guide. In war games like *Risk*, chess, or *Axis & Allies*, players make decisions that represent military movements, often sacrificing pieces or sending abstracted soldiers to their deaths. Very few players get emotionally attached to these stylized armies or feel remorse or sadness for their imagined deaths beyond "Argh, that was a stupid move!" This is part of the point of play: there's no real loss or death in failure.

When we experience a well-told story, however, we enjoy being swept up in the lives and emotions of the characters involved, even if they're imaginary. We can feel pride in their struggles, sadness at their experiences of loss or death. What about characters in less authored, more emergent stories, who aren't part of a preordained plot? Grunts and peons are some of the basic units of games in the *Warcraft* series, commanded by the player to build and fight. They die frequently, but they're generic and replaceable, so these deaths carry little emotional weight. Other characters in games like *Warcraft III* have names and personality and speak lines

of dialogue, but when these "hero" characters die, they can be brought back to life with an expenditure of resources and go on to speak their assigned lines during the rest of the story.

In other games, the emergent role of characters who can fight or die becomes a little more complex. In *X-Com: Enemy Unknown* (2012), the soldiers start off with random names, ethnicities, and genders. If they die during one of the game's battles against Earth-invading aliens, they're gone forever but can be replaced by hiring a new soldier. Unlike generic grunts, however, *X-Com*'s soldiers become more individualized over time. As they gain combat experience, they specialize in certain roles and are granted nicknames based on their actions. Before long, a player's squad has unique characters who might have names like Alex "Boom Boom" Cheng, who uses heavy weapons, or Michelle "Banzai" Rodriguez, who's known for running right up to her enemies.

These soldiers aren't the same characters as those commanded by other players of the same game, and they don't need to speak lines of dialogue to feel like personalities. Their character emerges from a combination of purely contextual (and randomized) elements like name, appearance, and events that happen due to the game's mechanics: remember that time when Banzai ran right up to that Berserker alien, and Boom Boom finished it off with a grenade that nearly killed her? Because these soldiers are individuals who can't simply be brought back to life, players have described feeling scared when they're in danger, sad when they die—even refusing to accept a favorite character's death and reloading the game to avoid it. Far more than generic grunts, *X-Com* soldiers and their personalities become part of the stories that gamers tell about what happened when they played: the emergent story of the game.

The experiences and feelings of soldiers in *X-Com*, impatient customers in *Diner Dash*, and Sims in *The Sims* don't need to be conveyed by an authored script. Players will react creatively to fill in the blanks, to imagine that Michelle "Banzai" Rodriguez has a hotheaded personality, or even a backstory that explains her recklessness. Players will project their own experiences of relationships and living situations onto events that arise from the system of *The Sims*; although each Sim is in some sense a stack of numbers running in the code of the game, associated with an arrangement of pixels and polygons on a screen, the human-like quality of how it all comes together is enough for human minds to connect the dots. Just as we can perceive a smiling face out of an arrangement of dots and lines (:-)) or an emotional scene out of Kuleshov's juxtaposition of images, we can perceive a story with someone we can relate to in the situations that arise from games.

Open Stories

When we discuss interpretation of stories and the activation of imagination to perceive a story in a system's arrangement, we're discussing a particular kind of emergent story. It arises out of the combination of authored elements (a character's smiling face, or a nickname that's assigned

to soldiers who charge in) with the player's perceptions and actions. The magic of this kind of emergent story is that it results from the conversation between what we've provided as game creators and the individual experience of play. If we travel even further down our game-story spectrum toward emergence, we find *open stories*, a different kind of storytelling in games with fewer controlled and authored elements, where players create their own stories.

The emergent story of a game is exactly what most of this book discusses: it's the story of a particular conversation that happens as a particular player figures out how to use verbs, when to use them with various objects in the context of a scene, and how to push into a game's resistance toward goals and rewards, whether established by the game or their own motivations. The emergent story is the tale of what happened when that player jumped into the game and started doing things. Some emergent stories are boring and short, even hilariously so: I started playing, ran to the right, and tried to jump over a pit, but I fell in and died, so I quit playing. (Fortunately, most players don't quit quite as easily.) Other stories may emerge from play but are nearly the same every time, regardless of who's playing or how many times they play. These stories emerge from relatively predictable systems.

If you want to tell a particular authored story with a game—or convey a message of your choosing, which may be an important one—a predictable system may serve you well, even if the message it carries is ambiguous and open to interpretation. If, on the other hand, you want the emergent story to be more unique to the player, it's worth considering ways of creating even more openness and unpredictability. Stories that are wide open to player involvement or imagination are inherently difficult to control as a creator. Making a system that can produce them means letting go of the traditional idea of authorship, of the creative goal of creating and delivering a message. Instead, as the game's creator, you become the facilitator of new conversations—ones that you never might have expected.

By their nature, emergent stories are open and unpredictable. They move beyond the limits of what you've created and into the space around a game, where players become creators themselves. That doesn't necessarily mean that emergent stories are better, more moving, meaningful stories than authored stories; in fact, they're often not what anyone would call a "good story" if measuring by the yardstick of traditional storytelling. But why measure that way? What emergent stories are is a different kind of thing entirely: still stories, still part of our human tradition of having experiences and telling tales about it, but as many and varied as there are people who play.

Sharing Authorship

There are many ways to create an open space that a game's stories can grow into. The most straightforward is always available, even to stories told in other media. If you tell a good story, it grows in the telling, is retold and changed by those who tell it. By focusing your creativity and efforts on creating a rich, interesting world with memorable characters and events, you're

creating something that may inspire others to extend it. This is true of the worlds of written works; the *Harry Potter* books have inspired thousands of works of fan fiction. Many of the tools to extend a fictional universe, or continue telling the stories of its characters, are available to audiences already. Fan writers and artists do so with words and images.

As a game creator, you have an additional option: you can open up your code and tools so that your players can become "fan" game designers and programmers. You can even create or refine tools to let players do this, although designing and building tools that make this easier for players can often be as much or more work as creating a game. Many open-world games, including ones we've already discussed like *Skyrim* and *Fallout: New Vegas*, come with tools that help players extend those worlds with mods that include new characters, scenarios and quests, different kinds of weapons, and even changes to the systems of the game. By putting the evolution of game worlds in the hands of players who become modders, those games continue to live on and be played for much longer than they would have if they were limited to only the original authored content.

Since the early days of digital games, creators have sometimes included level editors that let players become designers. *Lode Runner* (1983) is an example we discussed in Chapter 6. Even as a child, I was able to learn how to use its level-editing system, getting inspiration not only from trial and error but from the examples and techniques used in the levels that were built into the game by its creators. *Lode Runner* has a simple story, not much more than a premise. Games that involve worlds full of story and character, and then invite players to extend those worlds, give players sparks for their imaginations to build on.

Games have a rich tradition of creating and extending worlds, going all the way back to nondigital role-playing games like *Dungeons & Dragons*. Played with dice, pen, and paper, the purchased materials of the game provide a set of rules to play by along with some ideas about story and setting, scenes and objects. One of the players in a traditional nondigital role-playing game is more like a game designer: the dungeon master or game master, who often acts like a modder. She extends the materials of the game with her own ideas, perhaps changing the rules along the way, and can dream up her own worlds for a story that's a collaborative creation forged by her and the other players through playing.

In recent years, more and more digital games have been trying to appeal to potential players with their ability to be extended with stories and content made by players, for other players. *Shadowrun Returns* (2013) is a digital role-playing game based on a pen-and-paper roleplaying game. In that tradition, it was released with a single, relatively short piece of authored content, a scenario with quests and objects that can be played like any other game, but which also serve as an example to take apart, study, and use as materials for new scenarios. Beyond the authored content, the promise of *Shadowrun Returns* is largely based on what the creators of the game hope players will do with the game system and the tools they've provided to extend it (see Figure 7.15).

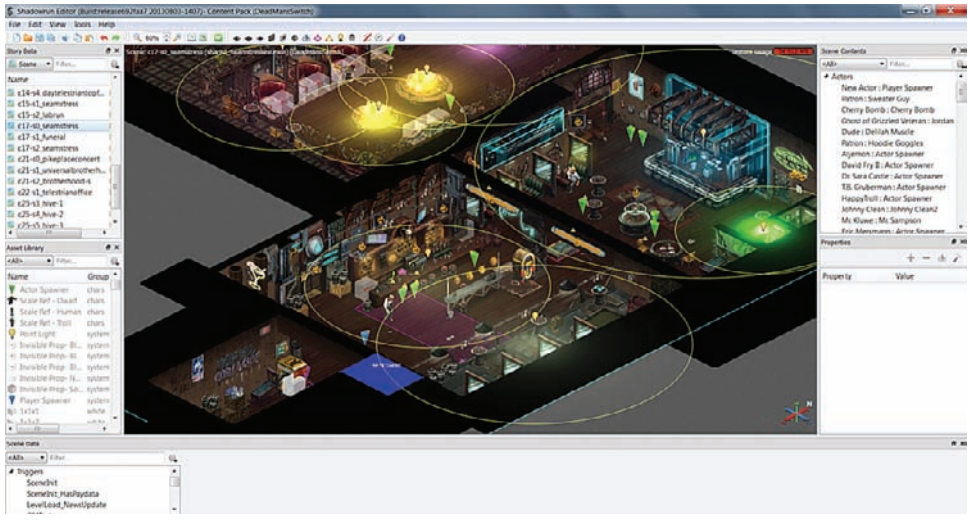


Figure 7.15 The creation tools provided to players for *Shadowrun Returns*.

System Complexity

Even novels and children's books can create imaginary worlds that activate the imagination and invite extension, but once again, we can look to the underlying structure of games to find something more unique: the unpredictable nature of complex systems. If we want many different players to experience their own uniquely emergent stories, a complex system of many parts that interact with each other is invaluable. A complex system often has many verbs, and many objects for them to act on, so that every possible combination can't be easily known or plotted out in advance. This creates a space of possibilities that can be explored, where players can have the thrill of discovering something new.

Sheer quantity of verbs, objects, and combinations isn't the key to complexity, however. What's important is the relationship between them—the fact that different elements in the system influence each other. Although a full discussion of complexity and the design of complex systems is beyond the scope of this chapter, here's a simple example. Imagine that a game of fantasy combat gives its players a verb like "hit" and objects that include the Uberknight, controlled by the player, and a type of enemy called a Bandersnatch. These objects have an attribute called Health that is reduced by "hitting," and which, if reduced to zero, eliminates the object from the game. (This system, which originated with games like *Dungeons & Dragons*, should sound pretty familiar to you if you've played games with combat systems.) Each Bandersnatch will try to "hit" the Uberknight until she's dead—it takes about four "hits"—but can itself be slain in two "hits." This is a simple combat system, with just one verb and a couple objects, but it already includes many rules and relationships!

Now imagine that the Uberknight has another verb, "deathwail," which can kill a Bandersnatch in just one hit. Deathwail has a restriction, however. It can only be used if the Uberknight is close to death, with less than 30% of her Health remaining. All of a sudden, there are a couple ways to play the game—different strategies that players can pursue and, in doing so, create slightly different emergent stories of their time playing the game. The player can "hit" the Bandersnatches she encounters until they're dead, or she can wait until a Bandersnatch has reduced her Health enough to "deathwail"—a more powerful and quicker way to eliminate her enemies, but one that's more risky as well. The differences between these verbs arises from their interaction with other rules of the game (like the one that says you're eliminated if your Health reaches zero) as well as objects in the game and their properties. (Bandersnatches take two "hits" to kill, but Uberknights take five.)

Of course, the exact details of how the emergent story will play out, and what strategies are interesting to pursue, depend on even more rules, some of which you might already be wondering about if you enjoy playing this kind of game. For example, who "hits" first? Is it random? Are there verbs that affect this? How about regaining lost Health: does it return over time, or are there even more verbs? Even in a simple system, it's easy to see how adding "deathwail" makes for a more complex system than one which just included "hit," and we can imagine how adding more rules would open up the space of possibilities even further.

We've already mentioned some examples of very open games in Chapter 6: games like *The Sims* or *Animal Crossing* (2001), which let the players pick their own goals to pursue amidst a system of many verbs and choices, and games like *Minecraft* or *Dwarf Fortress* (2006), which have complex systems for players to explore, and even come up with their own goals.

It's worth noting that complexity isn't the same as simple unpredictability of the kind that's produced by randomness, as in the toss of a die. Roulette is a highly unpredictable, random game; as the ball bounces around, there are dozens of pockets on the spinning wheel where it might land. As a system, it's not complex at all: you pick a pocket or a category of pockets (like red, black, or under 15), and if the ball lands there, you win. Betting and odds are what make roulette minimally interesting, but the system is simple: there aren't that many elements of vocabulary that can interact to create something new.

Multiplayer Complexity

There's one more element that helps a game become highly unpredictable and lends itself well to the emergence of stories that can be told and retold: other players. Multiplayer games have been the rule rather than the exception for most of history. Games have a tradition of social interaction, both in and around play. If you want unique, complex circumstances to arise, it's hard to create something more consistently unpredictable than what happens when you throw two or more people together.

The complexity of a game's system still matters a great deal. Tic-tac-toe can be played by two human beings but has a simple and easily mastered system that becomes very predictable. It produces few if any novel emergent stories for anyone who's played more than a handful of games. When players push into a complex system with many possibilities and potential strategies, however, they can surprise each other and (hopefully) even the game's creators.

The presence of other humans in the space of play can be inspiring for our creativity and ingenuity. When we're motivated to beat a human opponent in a competitive game, we're aware that our own unpredictability can be an asset. Especially in games of head-to-head competition, where our way of playing can negatively or positively affect our competition, the element of surprise can be crucial. As a result, we experiment, coming up with new things to try out and throw at our opponent.

At the same time, we try to envision what our opponent is thinking, what they understand about the game and how they'll act. (This attempt to read the opponent's mind is sometimes described with the Japanese word *yomi*.) On top of that, multiple players engage in this mental maneuvering at the same time; you're trying to guess what the other players are planning while knowing that they know that you know the possibilities they might be planning. Frank Lantz, an eminent game designer and theorist, calls this tangle of complexity *donkey space*, and it creates the most beautifully human aspects of unpredictability in games—qualities that really can't be replicated with computer code.

The playing of multiplayer games, whether cooperative or competitive, is also an inherently social act. We're never *just* making moves in a game system that we share with another human player; we're also *saying* something to the other player, even without words. Skilled players of a complex game can wordlessly express many things in how they use the verbs of game: wariness, mercy, the aggression of a quick onslaught, even ideas like "I'm just toying with you" or "Let's get this over with." The conversational nature of games becomes even clearer when there's more than one live participant in the conversation, using the structure and tools that the creator of the game, the facilitator of the conversation, has provided.

The emergent stories of a great game session can be immensely powerful: stories about an unexpected turnaround, a clever move that wasn't anticipated by the opponent. In these stories, the players are protagonists, and the feelings we see on their faces in the exultation of victory or the agony of defeat are real feelings of other human beings. That may be why they're told again and again and why the authored story of a game like *Tekken* (1994), which features many unusual and memorable characters struggling with their own fictional conflicts in a global fighting championship, is often eclipsed for *Tekken*'s community of players by the emergent stories that are created by one human player struggling against and defeating another.

Review

- Games and stories have both been vital parts of human culture and expression throughout history. As game creators have begun to explore ways of combining the two in recent decades, we've found that games can help us tell stories in unique ways, although the intersection between the two can be tricky to master.
- Games can carry *authored stories*, much like other more traditional ways of telling stories through words and images. If you want to tell an authored story, there's much to learn from techniques used by storytellers who've expressed themselves through forms like novels, comic books, and films.
- Games can also produce *emergent stories*. These are the stories about the playing of a game, often unique to a particular player or a particular time a game was played. They're the stories that happen when players push into the shape of a game system's resistance, make decisions, and understand the system through learning to use its verbs and engage with other elements of a game's vocabulary.
- Neither type of story is necessarily better, and both can lead to interesting conversations that involve the creator of a game and its player. Leaning toward authorship, emergence, or a combination of the two depends on what you hope to accomplish with a game as its creator: do you have your own story to tell, that players can listen to? Do you want to open up a space for players to experience and tell their own stories?
- There are many ways to weave an authored story into a game: you can tell a story alongside a game, in noninteractive cutscenes that can take the form of animations, characters speaking dialogue, or even wordless images. To create a story that feels more integrated with the game, it's possible to frame the story in such a way that the player feels driven to push the story forward and work to reach the next section of story as a reward. This has its pitfalls, however: the lure of the reward could become more compelling than the journey to get there through your game, especially if your game involves a lot of rote grinding!
- Exploratory elements of story can be spread throughout the space of a game in ways that let players choose whether to look for them, experience them, and find out more about a game's setting, backstory, or characters. Because exploratory story elements are optional, they're usually used to add flavor to the game world rather than to advance the story's plot.
- *Branching stories* in game books and story-games give players choices that take them down one of many available paths in the story, perhaps leading to multiple different outcomes. Although it's hard to make every branch feel like a compelling story and it can take a lot of work to create widely divergent branches, this kind of storytelling can give players a way to find and pick a story that suits them best or explore all the possible branches. Although a branching story is still an authored story, it involves the player in a deeper way by asking her which version she likes the best.

- Games can tell stories in a unique way through their mechanics. As a player pushes into the resistance of a game, she comes to understand how a system works and where she can use verbs to affect it. By shaping a game's system, its creator can express something about how the world works—even how it feels to be in a particular situation.
- *Interpretation* is a vital part of how stories are conveyed and how meaning is produced out of stories—not only by the author of a story, but by the way a reader, viewer, or player understands the story. Ambiguity in storytelling leaves some aspects of the story open, so that the potential meanings of the story, even the nature of what happened, is left partly up to the audience's imagination.
- *Emotional hooks* in storytelling, even expressed in simple ways such as the names and facial expressions of characters, give players the chance to engage their own empathy and connect the dots into a story that has meaning for them.
- *Reflective choices* show that decisions in a game don't need to change the game's state or outcome to affect the experience and make it meaningful. Reflection can pose deeply important questions to the player. What kind of person are you in this story? How will you react in this situation? What's important, and what are your goals? Even if the answers don't alter the outcome, the process of asking and answering changes the player's relationship to what happens in the story.
- There are many ways to open up a game so that it can produce a variety of emergent stories, in addition to or instead of an authored story. Most obviously, games and tools that accompany games can help players become creators in their own right, extending an existing authored story-world or coming up with their own.
- Games can produce many different emergent stories through the workings of complex systems as well. When verbs and other vocabulary elements of games can intersect in many different ways, the space of possible experiences in the game, and the number of stories that emerge from those experiences grows larger.
- Some of the richest complexity and uncertainty available to us through games can come from the interaction of human players. When multiple minds engage in and have a conversation through a game system, whether it involves conflict or cooperation, the experience becomes social and can produce many compelling, emergent stories that are worth telling.

Discussion Activities

1. Think of and retell some memorable stories that you've experienced in a game—ones that really stick with you. Are these authored or emergent stories? Think of both kinds, and discuss what makes each kind of story memorable and distinct.

2. Do stories need to affect the outcome of the game to feel meaningful? Think about games you've played and a situation where it felt difficult to decide what to do. What hinged on the outcome? Is it meaningful for you to make reflective choices, such as ones that change what your game's avatar looks like but don't affect the story? Why or why not?
3. Discuss a story that you and someone else you know, like a friend or relative, both experienced. Did you interpret the meaning of the story, or what happened in the story, differently? Why?
4. Have you had experiences while playing a game where the story felt irrelevant to what you were doing? How about a choice that didn't feel significant, or was just annoying or distracting to what you wanted to be doing? Why was this, and how do you think the experience could have been improved?

Group Activity

In Chapter 2, we described the story of Janet Jumpjet, involving ancient robots that come to life deep in the mines of Venus and start kidnapping miners. Work in a group to figure out ways that this story could be told in that particular game. You could use cutscenes, create characters in the story who interact with Janet, find ways of including exploratory story elements, or create branching story choices. Which way of storytelling in this game feels the most satisfying to you, and why?

Now come up with an entirely different setting for Janet Jumpjet, using the same game mechanics that have been described for that game in earlier chapters or that you added to it in an earlier exercise. Instead of a space hero who fights robots in the mines of Venus, can you tell a story where Janet is a pirate queen, a wolf, a high school student, or even an ancient Venusian robot? What would you need to change about the way the game's system works to go better with this story? What storytelling techniques would make sense for your new story?

This page intentionally left blank

APPENDIX A

FURTHER PLAYING

The best way to learn more about games is by playing them to understand how they work and what makes them interesting or average, more effective or less effective. To that end, the games listed in this appendix are recommended if you'd like to explore more examples of the ideas in this book. Each game has been selected because it's relevant to one or more of the preceding chapters and is accompanied by notes on why you might want to play it, information about what technologies or platforms you may need to play the game, and price as of Fall 2013.

Achievement Unlocked (John Cooney, 2008)

Resistance

This simple platforming game also serves as a commentary on the proliferation of nearly meaningless rewards in games: you earn your first achievement simply by loading the game, and you earn three more for watching and clicking the opening screens. Once play begins, you unlock dozens more just by using the game's controls, moving and jumping around its single scene, and dying by colliding with its spiked obstacles. There are 99 achievements in all, displayed in a list on one side of the screen; completing them all requires a little bit of platforming skill as well as guesswork and math to solve what the names of the still-locked achievements signify. The resulting experience hovers somewhere between the satisfaction of solving every last task on a list and the awareness that these rewards are simply being handed out like candy.

Platform: Web

Price: Free

http://armorgames.com/play/2893/achievement-unlocked

American Dream (Stephen Lavelle, Terry Cavanagh, Tom Morgan-Jones, and Jasper Byrne, 2011)

Scenes, Resistance

The player assumes the role of a novice stock trader who lives in a modestly furnished apartment and aims to make a million dollars. In each of the game's days, the player can spend her money on upgraded furnishings or go to work and invest her money on various stocks that can be bought and sold. One of the game's few instructions is "Buy low, sell high," and the stock-trading gameplay hinges on seemingly random price fluctuations that the player can predict to some degree by watching the patterns of highs and lows. After finishing trades, the player is returned to a brief moment of pause (and perhaps furniture-buying) in her apartment.

American Dream lets the player return to stock trading immediately, however, and the temptation to see the latest price changes and cash-in can mean that players spend very little time in the apartment. Because the simple, pixellated graphics of the apartment don't feel drastically different even with the most expensive furniture, the ostensible rewards of the gameplay are eclipsed by the drive to gamble and earn. Unexpectedly, a maximally furnished apartment turns out to be critical for optimal success in the stock market because having the best furniture at later stages of the game allows the player to get an "insider trading" tip that guarantees huge

profits. As the player becomes immensely rich, the difficulty of making money becomes trivial and the game's resistance becomes slack, possibly leading players to wonder what the point of so much money actually is.

Platform: Flash

Price: Free

http://ded.increpare.com/~locus/american_dream/

Analogue: A Hate Story (Christine Love, 2012)

Verbs, Story

At the beginning of *Analogue*, the player enters an empty spaceship that was built to carry a civilization over many generations to a new world. Rather than exploring the ship by walking around it, the player explores the history of the vanished civilization by reading documents that were left behind, now made accessible by interacting with two digital intelligences that still remain in the ship. Much of *Analogue* involves piecing together and interpreting these fragments, letters, and diary entries to understand the social structure and personal intrigues of the ship's passengers, which are based in part on changes in the society of ancient Korea, making the game's themes a fascinating blend of history and sci-fi. At certain moments, the verbs available to the player and the visual context of the game become entirely different as you're moved to a stark, monochrome, command-line interface that allows you to control the ship's systems directly. It's in these moments that some of the crucial branching points determining the outcome of the story occur.

Platform: Windows, Mac, Linux

Price: US $10. Free trial also available.

http://ahatestory.com/

The Banner Saga (Stoic, 2014)

Verbs, Scenes, Resistance, Story

At the heart of *The Banner Saga* is a turn-based game of tactics played with teams of characters who move and attack each other on a grid. This game in turn revolves around a constant choice between two verbs: will you "wound" an enemy unit's ability to wound you back, or "break" their armor, making it easier to wound them back? The choice between these two options remains surprisingly rich throughout each battle, especially as your own units become

wounded or unable to fight, narrowing the space of possibilities and increasing the feel of resistance. *The Banner Saga*'s story complements this grueling feeling, set in a frozen northland where food is scarce and communities must constantly stay on the move to avoid extinction. Between battles, the player is faced with other kinds of choices—during dialogue with characters who may support or abandon you in future battles, as well as in events that happen along the constant journey and affect your supply of food and able-bodied warriors.

Platform: Windows, Mac, Linux

Price: US $10. Free trial also available.

http://stoicstudio.com/

Candy Box (aniwey, 2013)

Scenes, Resistance, Story

Candy Box paces the player's experience in an interesting way: it starts off by showing the player that she has a pile of candies that increases at the rate of one per second and gives a single verb: "eating" the candies. If the player waits, more and more choices start to unfold, and the game is revealed as a mixture of "farming" candies to produce more resources (candies and lollipops) and going on adventures to fight enemies. All the resources needed to progress are generated over long periods of real time, much as in games like *Farmville* that rely on patience and rote grinding. Interestingly, *Candy Box* is entirely free and doesn't offer players paid shortcuts past grinding; instead, it gives players ways to optimize and improve their pace of resource generation, keeping a steady pace through more and more surreal adventure levels. Eventually, the story becomes a meta-narrative in which the player challenges a "developer" for control of the code that underlies the game itself.

Platform: Web

Price: Free

http://candies.aniwey.net/

Consensual Torture Simulator (Merritt Kopas, 2013)

Verbs, Resistance, Story

What does it mean to choose violence as part of a game, or to make a choice at all? *Consensual Torture Simulator* presents a very different kind of violence than that suggested by the conventional-wisdom notion of "violent videogames." In this game, the player chooses

what kind of violence to inflict on a consenting sexual partner—who says early on that her goal is to be brought to tears. As the player, the choice of how and whether to go this far, stop short of the goal, or continue even further is entirely up to you. *Consensual Torture Simulator* has three primary verbs, and a system that responds with vivid feedback to the player's actions, but which doesn't frame the outcome in terms of winning or losing—perhaps because those sport-like notions don't make as much sense in the context of loving sexual interaction. Instead, the meaning of the decisions the player makes must reside as much within her own head—a particularly foregrounded kind of reflective choice that asks, "What does this form of violence mean to you? How does it make you feel?"

Platform: Web

Price: US $2 (suggested price)

https://gumroad.com/l/consensualtorturesimulator

Corrypt (Michael Brough, 2012)

Verbs, Scenes, Resistance

For the first half of the player's experience, *Corrypt* presents a mentally challenging series of puzzles. Drawing on the tradition of puzzles like *Sokoban*, which involves pushing crates around each other in a constrained, grid-based space, the player has to explore a series of caves to collect mushrooms, keys, and other items. The various scenes of the game develop the verbs of pushing and pulling in various ways, but in the second half of the game, a new verb is introduced: the player can exchange mushrooms for the limited ability to "glitch" a single tile from one of the game's rooms. The glitched tile then shows up at the same position in every other room, replacing what was already there; this allows the player to replace a wall with an empty space from another room so she can pass through.

All of a sudden, the space of possibilities and resistance in the game becomes dramatically wider, requiring the player to decide how to spend a limited stock of mushrooms and acquire more. A constrained verb that lets you edit the world turns out to be a much bigger challenge, pairing an open space of choice and possibility with rising challenge. The visual context of the game, combining intricate pixel-patterns and garish colors with a world whose logic often feels counterintuitive, complements the anxiety that arises from the dizzying freedom of tearing apart the structure of the game.

Platform: Windows, Mac, iOS

Price: Free (Windows, Mac) or US $2 (iOS)

http://mightyvision.blogspot.co.nz/2012/12/corrypt.html

Crypt of the Necrodancer (Ryan Clark, 2013)

Verbs, Resistance

Like several other games discussed in this appendix and throughout this book, *Crypt of the Necrodancer* is a rogue-like game—the player explores a dungeon that's randomly generated each time she enters it, and moves around on a grid of tiles while battling or avoiding enemies, finding treasures and stores to shop at, and discovering secrets. *Crypt of the Necrodancer*'s chief innovation lies in adding music and rhythm on top of these actions: any song can be played while you explore the dungeon, and your character has to move and fight to the beats of the music. The key to success in *Necrodancer* is to continue acting on each beat of the game—as long as you keep making moves, a "groove chain" bonus will build up, allowing you to earn extra resources. The need to think and react quickly transforms the rogue-like genre into a game that's difficult in a different way, but the player can drop the resistance of this challenge at any time by giving up on the groove chain and pausing to consider her next move.

Platform: Windows, Mac, Linux

Price: US $10. Free trial also available.

http://necrodancer.com/

Dwarf Fortress (Tarn Adams, 2006)

Resistance, Story

One of the most complex and open-ended simulation games ever created, *Dwarf Fortress* puts the player in charge of building and maintaining an underground community of dwarves. Dwarves must be directed to build, gather resources, and perform dozens of other jobs, from tending the wounded to crafting weapons; the player's main concern is to keep dwarves from dying or becoming so unhappy that they go insane. Each dwarf is a detailed simulation of its own, with moods and abilities affected by the game world's weather, his environment, his tiredness, and so forth. Dwarves can get married, have children, and even hold political office or act as law enforcement for other dwarves. The huge world around the player's fortress is equally complicated and generated randomly for each new playthrough in the manner of a roguelike game, with specific geological and weather patterns, nations of other species to war or trade with, and unexplored areas to discover. *Dwarf Fortress* is extremely challenging to learn due to the sheer complexity and number of interlocking systems, but this complexity also produces an incredible range of emergent stories as the lives of your dwarves proceed differently every time.

Platform: Windows, Mac, Linux

Price: Free

http://www.bay12games.com/dwarves/

English Country Tune (Stephen Lavelle, 2011)

Verbs, Scenes, Resistance

The control scheme of *English Country Tune* seems basic: you "move" a flat tile across a three-dimensional structure of cubes, flopping from one side to the other as you move each space. The complexity in the use of this simple verb arises from the various and unusual properties of the other abstract objects that are affected by your movements: spherical "larva" that obey different pulls of gravity depending on which side you approach them from; cubical "whales" that must be pushed by moving into beams of light that extend from their six faces; spaces that punch patterns of holes in your tile or cover one of its sides in ink that can be painted onto other spaces; switches that freeze the whales and larva in place, letting you climb on them; and much more.

Each of *English Country Tune*'s worlds examines one or two of these objects in depth, challenging the player with complex puzzles that explore the space of possibilities arising from the objects' associated mechanics. The final worlds start to combine many types of objects at once, creating complex dynamics that are, by that stage of the game, oddly intuitive to the player despite coming from a universe of simple but strange behaviors. In a couple of the game's worlds, the player is even asked to take on the role of designer and given a different set of tools and verbs to complete the challenge of building an *English Country Tune* level that has certain properties.

Platform: Windows, Mac, Linux, iOS

Price: US $5 (Windows, Mac Linux), US $3 (iOS). Free trial also available.

http://www.englishcountrytune.com/

Even Cowgirls Bleed (Christine Love, 2013)

Verbs, Story

What happens to a story when the protagonist has only one verb? *Even Cowgirls Bleed* is a short story-game that reveals more of the story when the player "moves crosshairs" over highlighted words in the text that makes up the game. Putting a word or phrase in the crosshairs causes the main character (a would-be cowgirl and sharpshooter) to shoot whatever's mentioned. There's one other possibility: a button labeled "Holster" appears to the side of the story text, switching back and forth as the story proceeds. To keep your gun holstered, you must move your mouse across the text, sometimes leading to an inadvertent discharge of the gun. This simple mechanic conveys the feeling of an itchy trigger finger and results in a story in which

the cowgirl simply can't keep from shooting everything; when you have a verb right at your fingertips, it's hard not to use it. When a game—like so many action games—affords you only one meaningful way to interact with the world, it's hardly surprising that stories are shaped by the structure of the game and end in blood.

Platform: Web

Price: Free

http://scoutshonour.com/cowgirl/

Gone Home (The Fullbright Company, 2013)

Scenes, Story

Gone Home communicates its narrative entirely through exploratory story elements: the eldest daughter spent a year abroad and then arrives back home to find her large family house empty. To solve the mystery of where everyone's gone, it's necessary to look through the house for scraps of paper, notes, letters, and other bits and pieces of information that slowly reveal the story of what happened during the time she was gone. *Gone Home*'s story isn't just communicated with words; the game also uses the environment of the house to convey what kind of people lived there and what their daily thoughts and preoccupations were. It even contains mundane items like boxes of books and bottles of hair dye so you can interpret past events and the state of the family.

Platform: Windows, Mac, Linux

Price: US $20

http://www.gonehomegame.com/

Mighty Jill Off (Anna Anthropy, 2008)

Verbs, Scenes, Resistance

Like many other platform games, the primary verb of *Mighty Jill Off* is "jump," but this game explores a specific variation of that verb: when the player jumps, her avatar can float slowly down while drifting left or right. The player jumps up a tall tower through a series of rooms, each of which has been carefully crafted to develop jumping in a different and increasingly challenging way. *Mighty Jill Off* also expresses a theme that's of personal importance to the creator, since the protagonist (Jill) is a lesbian submissive trying to please her queen, who kicked her down to the bottom of the tower and waits for her to ascend to the top. When the

player passes the final room, the queen briefly expresses pleasure and then kicks Jill down to start the process over again. If the player manages to ascend the tower in less than 12 minutes, she's rewarded with a different and even more challenging tower. The experience of playing the game, pushing against the resistance of the tower's dangerous rooms only to repeat the process, mirrors the game's concise story and reflects on the nature of games and their resistance: as players, we jump through hoops and submit ourselves to moments of frustration and challenge for the sheer pleasure of it.

Platform: Windows, Mac

Price: Free

http://auntiepixelante.com/jilloff/

NetHack (NetHack Dev Team, 1987)

Verbs, Scenes, Resistance, Story

NetHack is one of the earliest games in the roguelike genre, a type of game that often involves exploring and fighting enemies in a series of levels that are semi-randomly generated and different every time you play. It's also a great example of a game with a huge number of verbs. Players can "use" or "consume" hundreds of different items, "flee" when they spot enemies or "attack" from a distance, and even "pray" to dozens of different gods. Because of the rich variety of objects, opponents, and situations in the game, these verbs don't feel underdeveloped; they can all be used in a lot of different ways, with overlapping effects and strategies resulting from the combinations. *NetHack* has a rich emergent story that's unique every time and requires few authored elements. The stories that arise from the game don't even need much in the way of visual context, since the entire game world is represented by letters, numbers, and punctuation arranged in a grid.

Platform: Windows, DOS, Mac, Linux, Amiga, Atari

Price: Free

http://www.nethack.org/

Papers, Please (Lucas Pope, 2013)

Verbs, Resistance, Story

As an immigration inspector on the border of a fictional Soviet-bloc-themed country in the 1980s, the player is given the rote task of examining the paperwork of hundreds of would-be border crossers and then approving or denying their entry. Although the grind of this gameplay effectively communicates the mechanical feeling of being a cog in the wheels of bureaucracy,

Papers, Please deepens the experience with a variety of story elements that appear as the processing of paperwork becomes increasingly complex and challenging. Most of the people you examine are randomly generated, and some have incorrect or forged paperwork, but all want to get across the border.

It's up to the player to decide whose sob story is worth breaking the rules for, from immigrants with apparently misspelled names and smugglers offering bribes to families trying to stay together and a revolutionary cell trying to bring down the oppressive government. Your pay is docked for making too many broken rules or mistakes, and at the end of each day, you see if there's enough money left to keep your family from starving and getting sick. The game uses a series of nonrandom characters to weave in an authored story structure that branches depending on who the player chooses to help, leading to many different outcomes. *Papers, Please* accomplishes all this with a simple and unusual set of verbs: "stamp" paperwork with APPROVE or DENY, "give" items on your desk to the person being inspected, and sometimes "fingerprint," "strip-search," or "detain" them for questioning.

Platform: Windows, Mac

Price: US $10 (Limited beta version available for free at http://dukope.com/)

http://papersplea.se/

Persist (AdventureIslands, 2013)

Verbs, Scenes, Resistance

Many games open up possibilities and increase complexity by giving the player more verbs to experiment and push into the game with. As the player masters verbs, she finds more verbs to try out, creating a feeling of progression and accomplishment. *Persist* turns this idea around, increasing resistance by taking the player's abilities away one at a time. You start off with verbs common to many platform games: the avatar can "run," "jump," and "swim" through water. In each of the game's sections, the avatar loses something. First, water becomes a dangerous hazard that can no longer be crossed; next, the "jump" verb is removed. Interestingly, the lack of jumping focuses the player's attention on and develops the "move" verb, as timing left and right movements are all that remains. Finally, the player loses the ability to see the avatar, except for a small "spark" when it moves, and must navigate levels under a kind of blindness.

Platform: Flash

Price: Free

http://www.kongregate.com/games/AdventureIslands/persist

QWOP (Bennett Foddy, 2008) and *GIRP* (Bennett Foddy, 2011)

Verbs, Resistance

These two games from the same creator are worth mentioning together because they both explore the relationship of the player's controls to what games often systematically depict as a simple, intuitive verb: "running" in *QWOP* and "climbing" a wall in *GIRP*. In *QWOP*, rather than pressing a single button to run, the player must use the four keyboard keys represented in the title to control the left and right calf and thigh of a runner. Despite simulating the muscles used in actual running, this process turns out to be vastly more difficult than intuitive control of your own legs. In *GIRP*, each handhold in a vertical cliff-face is represented by a different alphabetic key, requiring the player to hold one key down while choosing another, higher handhold to reach for. This often means stretching your fingers across the keyboard in a manner that feels akin to the strain of real climbing.

In both games, losing control over the precise pressing of keys means falling down and starting over. *GIRP* is made even more frustrating by the presence of a bird that starts to perch on some of the handholds, making them inaccessible until the player can flail their climber's limbs across that area of the screen and scare it away. Despite the simple representation and behavior of the bird, the role it plays in *GIRP*'s system makes it one of the most frustrating "enemies" ever encountered in a game.

Platform: Web (both games), iOS (*QWOP*), Android (*GIRP*)

Price: Free (Web) or US $1 (iOS, Android)

QWOP: http://www.foddy.net/Athletics.html

GIRP: http://www.foddy.net/GIRP.html

Spelunky (Derek Yu, 2008)

Verbs, Resistance, Story

A platform game that draws on many of the techniques used by roguelike games, *Spelunky* puts the player in the role of an archaeologist exploring an ancient temple full of dangers. Unlike traditional roguelikes (see the entry on *NetHack*), verbs and objects in *Spelunky* interact with the game's real-time verbs of "running," "jumping," "climbing," and "predicting" enemy movement to "avoid" or "eliminate" them, making a complex and challenging system of emergent possibilities. Each level in *Spelunky* is created in a semi-random process that results in a different layout every time but also follows certain rules to shape the player's experience consistently. The player starts at the top and must eventually make her way to an exit somewhere at the bottom.

There's always a path that the player can follow from top to bottom, but the player can also reshape the structure of the level by blowing up walls and floors or attaching ropes to climb into inaccessible areas. Each set of four levels has a distinct theme, with different enemies and hazards as well as special areas that always appear, such as a room with a golden idol that triggers a huge rolling boulder when taken. These semi-fixed elements create a reliable but varying structure that weaves an authored story into the emergent experiences that are unique for every playthrough.

Platform: Windows, Mac, Xbox 360, PlayStation 3

Price: Free (Classic Version for PC, Mac) or $15 (HD Version for PC, Xbox 360, PlayStation)

http://spelunkyworld.com/

Triple Town (Spry Fox, 2011)

Verbs, Resistance

Triple Town starts with a premise that will sound familiar to players of "tile-matching" games such as *Bejeweled*, *Dr. Mario*, and *Puzz Loop*. You're tasked with making groups of three or more matching objects. The creators of *Triple Town*, however, intended to create a new genre rather than a variation on existing games. In this game, three or more adjacent matching objects combine into a new object at the spot where the player placed the final object. The new object is an "evolution" of the earlier ones: three grasses combine into a bush, three bushes into a tree, three trees into a wooden hut, and so forth. The goal of the game is to reach as high a score as possible without running out of space in the tightly constrained playing area. Players have to plan well in advance to place the 27 grasses needed to create a house, which only represents the fourth of nine tiers of objects!

The game's resistance is shaped not only by the limited playing area, but by the fact that the player is given a random new object to place each turn and sometimes must place a "bear" object that can move around, disrupting her plans. *Triple Town* takes the simple action of placing a randomly dealt object onto a board and a few rules to create a complex system with many possibilities and strategies—even player-driven goals. The game offers many ways to "build up a town" that all result in a different arrangement of buildings, flora, and mischievous bears.

Platform: Windows, Mac (unlimited version). Web, iOS, Android, Kindle Fire (free version with premium purchases available)

Price: Free (Web, iOS, Android, Kindle Fire) or US $10 (Windows, Mac)

http://spryfox.com/our-games/tripletown/

INDEX

D

E

F

G

H

I

J-K

L

M

N

O

P

Q-R

S

T

U-V

W

X-Y-Z

This page intentionally left blank

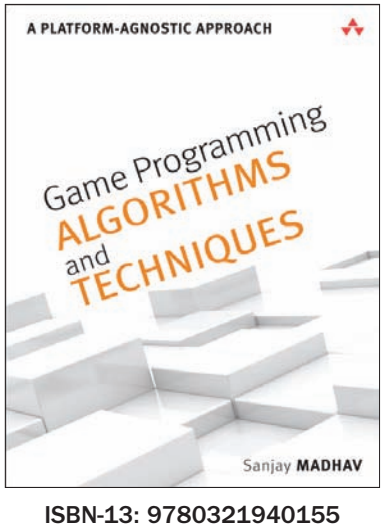
Addison Wesley

Practical Resources for Game Programmers

ISBN-13: 9780321898388

Start Building Web Games with HTML5 and JavaScript

Evan Burchard walks you step-by-step through quickly building 10 popular types of games accompanied with full JavaScript source code listings. Each game recipe uses tested and well-proven patterns that address the development challenges unique to that genre and shows how to use existing tools and engines to build complete substantial game projects in just hours. Need a quick JavaScript primer? Evan Burchard provides that, too!



ISBN-13: 9780321940155

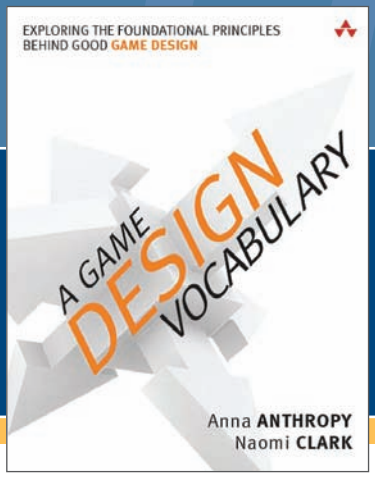
The Fundamental Techniques for Working with 2D and 3D Graphics, Physics, Artificial Intelligence, Cameras, and Much More

Game Programming Algorithms and Techniques is a detailed overview of many of the important algorithms and techniques used in video game programming today. This book focuses on practical concepts that see actual use in the game industry and each concept is illuminated with pseudocode that has been refined and proven in Madhav's game programming courses at the University of Southern California. Sanjay Madhav takes a unique platform- and framework-agnostic approach that will help develop virtually any game, in any genre, with any language or framework.

informIT

Safari Books Online

For more information and sample content visit informit.com.
eBook and print formats available.



Safari Books Online

FREE Online Edition

Your purchase of ***A Game Design Vocabulary*** includes access to a free online edition for 45 days through the **Safari Books Online** subscription service. Nearly every Addison-Wesley Professional book is available online through **Safari Books Online**, along with over thousands of books and videos from publishers such as Cisco Press, Exam Cram, IBM Press, O'Reilly Media, Prentice Hall, Que, Sams, and VMware Press.

Safari Books Online is a digital library providing searchable, on-demand access to thousands of technology, digital media, and professional development books and videos from leading publishers. With one monthly or yearly subscription price, you get unlimited access to learning tools and information on topics including mobile app and software development, tips and tricks on using your favorite gadgets, networking, project management, graphic design, and much more.

Activate your FREE Online Edition at informit.com/safarifree

STEP 1: Enter the coupon code: NUHEHBI.

STEP 2: New Safari users, complete the brief registration form. Safari subscribers, just log in.

If you have difficulty registering on Safari or accessing the online edition, please e-mail customer-service@safaribooksonline.com

Addison Wesley · Adobe Press · Alpha · Cisco Press · FT Press · IBM Press · Microsoft Press · New Riders · O'Reilly · Peachpit Press · Prentice Hall · Que · Redbooks · Sams · SAS Publishing · VMware Press · Wiley · Wrox