Introduction to GAME SYSTEMS DESIGN

Dax **GAZAWAY**

Introduction to Game Systems Design

This page intentionally left blank

Introduction to Game Systems Design

Dax Gazaway

Addison-Wesley

Boston • Columbus • New York • San Francisco • Amsterdam • Cape Town
Dubai • London • Madrid • Milan • Munich • Paris • Montreal • Toronto • Delhi • Mexico City
São Paulo • Sydney • Hong Kong • Seoul • Singapore • Taipei • Tokyo

The author and publisher have taken care in the preparation of this book, but make no expressed or implied warranty of any kind and assume no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information or programs contained herein.

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and the publisher was aware of a trademark claim, the designations have been printed with initial capital letters or in all capitals.

Microsoft and/or its respective suppliers make no representations about the suitability of the information contained in the documents and related graphics published as part of the services for any purpose. All such documents and related graphics are provided "as is" without warranty of any kind. Microsoft and/or its respective suppliers hereby disclaim all warranties and conditions with regard to this information, including all warranties and conditions of merchantability, whether express, implied or statutory, fitness for a particular purpose, title and non-infringement. In no event shall Microsoft and/or its respective suppliers be liable for any special, indirect or consequential damages or any damages whatsoever resulting from loss of use, data or profits, whether in an action of contract, negligence or other tortious action, arising out of or in connection with the use or performance of information available from the services. The documents and related graphics contained herein could include technical inaccuracies or typographical errors. Changes are periodically added to the information herein. Microsoft and/or its respective suppliers may make improvements and/or changes in the product(s) and/or the program(s) described herein at any time. Partial screenshots may be viewed in full within the software version specified.

Microsoft® Windows and Microsoft Office® are registered trademarks of the Microsoft Corporation in the U.S.A. and other countries. This book is not sponsored or endorsed by or affiliated with the Microsoft Corporation.

Cover image: Vladimir Vihrev/Shutterstock

For information about buying this title in bulk quantities, or for special sales opportunities (which may include electronic versions; custom cover designs; and content particular to your business, training goals, marketing focus, or branding interests), please contact our corporate sales department at corpsales@pearsoned.com or (800) 382-3419.

For government sales inquiries, please contact governmentsales@pearsoned.com.

For questions about sales outside the U.S., please contact intlcs@pearson.com.

Visit us on the Web: informit.com/aw

Library of Congress Control Number: 2021940285

ISBN-13: 978-0-13-744084-9
ISBN-10: 0-13-744084-7

ScoutAutomatedPrintCode

Editor-in-Chief
Mark Taub

Acquisitions Editor
Malobika Chakraborty

Development Editor
Chris Zahn

Managing Editor
Sandra Schroeder

Senior Project Editor
Lori Lyons

Copy Editor
Kitty Wilson

Production Manager
Vaishnavi Venkatesan/
codeMantra

Indexer
Timothy Wright

Proofreader
Betty Pessagno

Compositor
codeMantra

This book is dedicated to my game family. This includes those who raised me as a gamer, those who have been with me through this journey, and those who took me under their wing as I learned the professional trade. Thank you to everyone.

This page intentionally left blank

Contents at a Glance

This page intentionally left blank

Contents

PREFACE

This book covers the basic aspects of game system design in plain English. It uses numerous examples and analogies to help guide you through topics that might seem intimidating at first but are totally within your reach. The book focuses on learning how to use spreadsheets for system design. It covers the basics and best practices for using spreadsheets to make complex game data more manageable.

Who This Book Is For

The primary audience for this book is aspiring game designers who are new to doing system design and interested in learning more. It is assumed that anyone starting this book already understands basic mathematics. But, beyond that, there are no presumptions for prior game design learning. This book is made to guide someone with a basic high school education from being a complete novice to becoming a practicing system designer.

The following are some of the groups of people who could benefit from the methods described in this book:

- Aspiring professional video game system designers
- Game masters/dungeon masters
- Hobbyist video game designers
- Designers of pen-and-paper RPGs and other analog games
- Experienced level designers who want more system design knowledge
- Programmers/engineers who will be working with system designers
- High school educators who want to connect games with math for students
- Producers/lead designers who want to better understand systems

How To Use This Book

This book is written to be read from beginning to end if you are starting fresh, without much prior knowledge of game systems. It's also made to be a reference book that you can jump around in and pick up useful bits of information, even if you are an experienced system designer. The best method for absorbing the information would be to read through the book

once, working in a spreadsheet as you go, and then come back to the book as you create your next game for guidance on the complex tasks required to fully realize your game.

This book discusses and refers to a number of existing games, and it would be helpful for you to understand these games to some extent. Before you read the rest of this book, familiarize yourself with the following games by at least watching video reviews online or finding free web apps and playing the game a few times:

- Play backgammon, chess, and the Royal Game of Ur. Pay attention to the kinds of dice rolls you make in these games, how pieces are moved, and how the mechanics of each game interact with the game objects.
- Play The Battle for Wesnoth to get a better idea of what a turn-based game is and what an RPG is. Wesnoth has attribute-driven data objects and game mechanics that illustrate many of the concepts covered in this book. Further, it is supported by an active community that keeps the game well documented and up to date.
- Play or at least watch video reviews of *Pac Man*, *Galaga*, and other classic arcade games.

The games used as examples in this book were purposefully chosen because they are easily accessible.

This book describes many methods of working with game systems in great detail. It might seem that the methods in this book are being exclusively recommended, but this is not the case. Game system designers use an infinite number of methods, tricks, and techniques to do their work. They use so many, in fact, that they could not fit into a single book. This book is designed to provide a starting point that shows a small number of sample methods that are useful for all system designers. I expect and encourage you to continue to learn more techniques from other books, colleagues, and you own personal experiences. There are as many different ways to design game systems as there are system designers, and experimenting will help you find your own style.

What This Book Covers

Here is a rundown of what each chapter in this book covers.

- **Chapter 1: Games and Players: Defined**

  This chapter defines some of the important terms used in this book and provides some clarity on some important topics.

- **Chapter 2: Roles in the Game Industry**

  The game industry includes a wide variety of disciplines and subdisciplines that can be confusing to those who are new to game design. This chapter describes the common roles in the industry.

- **Chapter 3: Asking Questions**

  Game designers must ask questions and interpret answers in unique ways, and this chapter helps you rethink how we go about it.

- **Chapter 4: System Design Tools**

  The game industry is, as you would expect, full of computer software tools. This chapter covers the kinds of tools you are likely to use and some of the most popular tools in each category.

- **Chapter 5: Spreadsheet Basics**

  Spreadsheets are ubiquitous in most work, and they are especially useful to game system designers. This chapter covers spreadsheet basics.

- **Chapter 6: Spreadsheet Functions**

  This chapter continues the exploration of the power of spreadsheets by focusing on functions.

- **Chapter 7: Distilling Life into Systems**

  When you really look in detail at the mechanics that compose any game, you find that they are analogs for aspects of real life, even if they are abstracted. This chapter explains how you use those abstractions to create the building blocks of games.

- **Chapter 8: Coming Up with Ideas**

  This chapter helps you develop your skills around being creative, specifically in regard to coming up with new ideas for games.

- **Chapter 9: Attributes: Creating and Quantifying Life**

  One of the most common early tasks system designers perform is creating attributes for game objects. This chapter covers what attributes are and how to get started creating them for a game.

- **Chapter 10: Organizing Data in Spreadsheets**

  Once you have started creating attributes for your game objects, you will need to organize them and eventually analyze them. The best place to do this is in a spreadsheet. This chapter covers how to organize your ideas in a usable format.

- **Chapter 11: Attribute Numbers**

  This chapter discusses how to quantify attributes into numbers, including a scale of numbers and what kind of number granularity best fits a game.

- **Chapter 12: System Design Foundations**

  This chapter covers attribute weights, considerations for intertwined attributes, binary searching for the correct number, and naming conventions.

- **Chapter 13: Range Balancing, Data Fulcrums, and Hierarchical Design**

  This chapter discusses methods of turning a small number of data objects into a fully fledged set of game data.

- **Chapter 14: Exponential Growth and Diminishing Returns**

  Exponential growth is one of the most powerful methods of balancing modern games. This chapter covers why we use this method and explains a formula you can use to quickly create a nearly infinite number of varieties of exponential growth in games.

- **Chapter 15: Analyzing Game Data**

  An important step in understanding a game as a whole is to evaluate all of its objects together, whether it's a small set of 10 objects or tens of thousands of objects. This chapter covers how to collect data in a spreadsheet and get started doing basic analysis.

- **Chapter 16: Macrosystems and Player Engagement**

  You can use several different styles of difficulty adjustment to make a game harder or easier or to adjust a game to a player's particular needs. This chapter provides a high-level overview of various methods and gives examples of how these methods can be used in a variety of situations to get the proper balance for a game.

- **Chapter 17: Fine-Tuning Balance, Testing, and Problem Solving**

  Much of a game designer's time is not spent designing but balancing, testing, and problem solving. This chapter covers methods of making these important tasks easier and more productive.

- **Chapter 18: Systems Communication and Psychology**

  Games can be delivered to an audience in a variety of ways. A designer must consider how a particular game gives information to players and receives information from them. This chapter covers many of the aspects of communication with players.

- **Chapter 19: Probability**

  Not everything is predictable in the world or in games. However, it is possible to understand some unpredictability. This chapter introduces you to basic methods of calculating and understanding game probability.

- **Chapter 20: Next Steps**

  This final chapter gives you some more direction toward further growth in the world of game system design.

Register your copy of *Introduction to Game Systems Design* on the InformIT site for convenient access to updates and/or corrections as they become available. To start the registration process, go to informit.com/register and log in or create an account. Enter the product ISBN **9780137440849** and click Submit. Look on the Registered Products tab for an Access Bonus Content link next to this product, and follow that link to access any available bonus materials. If you would like to be notified of exclusive offers on new editions and updates, please check the box to receive email from us.

ACKNOWLEDGMENTS

First, I must thank my wife, Melanie Gazaway, who stood by me while I lived this, encouraged me to write it down, and helped me find all my worst typos before I sent in the book for review. Next, I want to thank my children, Mazzy and Jack, who had to put up with an awful lot while I was working in the game industry. From late nights at the office to missed vacations, I was not always able to be there for them when I wanted, but they never made me feel bad about it.

Next, I want to thank my parents, Armen and Michael Gazaway, who raised me as a gamer nerd. I certainly would not be where I am today without them. Michael was my dad and first dungeon master. He was the first person I knew who designed and modified games. He taught me the fundamentals of game design before most kids even knew there was such a thing. My mom, Armen, read me *Lord of the Rings* as a bedtime story and let me skip school to see *Star Wars* on opening day. Even now we discuss games, sci-fi, and fantasy movies as a normal part of conversation.

Beyond my parents, Rick Herrick was a family friend and huge gamer influence on me. Scott Stocklin and Jesse Wise were childhood friends who introduced me to even more games and were the test subjects for some of my earliest and worst attempts at making my own games.

In college, I was in the "crucible of design" where my group of friends were constantly making and playing each other's games. It was in that time that I developed more quickly than at any other time before becoming a professional. I would especially like to thank my gaming group, including Dax Berg, Goose, Todd Meyers, Ron Mertes, Skip, Foz, the Chads, Pig Man, Sarah Lacer, Marie, Glenn, Connor, Evan, and all the Daves.

Once I became a professional, the 3DO team was a tremendous help. Special thanks in particular to the leads of the team, Jason Epps and Howard Scott Warshaw (yes, THE Howard Scott Warshaw). They guided me from being a very fresh rookie into becoming a professional game designer.

I first heard the phrase "game system designer" with the Lucas Arts Team, and once I heard Chris Ross say it, I was hooked for life. In addition, he and Dan Connors were very supportive in letting me explore this new unofficial title to figure out what it meant. I cannot thank the Gladius team enough. They were all great, and I learned a ton of what is written in this book while working on that team. Special thanks go out to the system team of Alex Neuse, Derek Flippo, and Robert Blackadder.

The Vicarious Visions Team brought me on specifically because I was a system designer, and that was the direction they wanted to take the studio. This was a massive responsibility, and I learned an incredible amount while working there. I had more friends at that studio than I can name, so I will say special thanks to my system team of Dan Tanguay, Jonathan Mintz, Alan Kimball (programmer extreme and honorary system guy), Jay Twining, Justin Heisler, Mike Chrzanowski, Brandon Van Slyke, and Jessica Lott. Thanks to Tim Stellmach for introducing me to Bad Storming.

Row Sham Bow was the last professional studio I worked at and easily the best. Every single person there was amazing. The studio set the bar so high for me that I will only ever consider working at a studio this great in the future.

I would like to thank the Full Sail team. I love teaching and sharing my experiences with enthusiastic, motivated students who are at the beginning of their game design journey. In specific, several of my colleagues encouraged me to write this book and provided valuable feedback as I did so. These include Zack Hiwiller, Ricardo Aguiló, Fernando De La Cruz, Christina Kadinger, Andrew O'Connor, Hayden Vinzant, Paul Fix, Derek Marunowski, and Phillip Marunowski. A special thanks also goes to my interns and those wonderful students who kept coming back for game days.

Finally, I want to thank all my wonderful students. Seeing their passion and enthusiasm keeps me feeling young and passionate about this profession. I wrote this book for them specifically. It took me over 20 years to accumulate the knowledge I am presenting here, and now I am passing it along to the next generation. My greatest hope is that I can make their journey easier than mine was, as all my mentors made my journey easier than theirs was.

ABOUT THE AUTHOR

Dax Gazaway was raised in a gamer family. His parents met in a Dungeons & Dragons group, and he was surrounded with games being played and made. From a very early age, Dax was fascinated by the numbers in games. He would pour over monster manuals and board game books, dissecting the rules to figure out how the systems worked.

Dax started in the video game industry in the late 1990s. During his tenure in the industry, Dax pioneered game system design at multiple independent and AAA studios, helping to refine and define the subdiscipline. In recent years, he has become a course director at Full Sail University, specializing in teaching new students the concepts and tools of the system designer. Dax has created new curriculum and multiple classes for system design students, and he teaches introduction to system design courses.

The following is a selection of Dax's game design credits:

- *Star Wars*: *Obi-Wan*, System and level designer
- *Star Wars*: *Jedi Starfighter*, System and level designer and QA liaison
- *Star Wars*: *Bounty Hunter*: System and level designer
- *Gladius*: System designer
- *Syphon Filter* franchise: Lead designer and system designer
- *Spider Man 3*: Lead system designer
- *Marvel Ultimate Alliance 2*: Lead system designer
- *Guitar Hero* franchise: System designer

In addition, Dax has been the studio lead system designer for Row Sham Bow Games and a system design consultant for multiple projects.

This page intentionally left blank

CHAPTER 11

ATTRIBUTE NUMBERS

So far in this book, you have created objects and their attributes. You have also created a spreadsheet to organize all your data. The next step in bringing your game ideas to life is to start putting in numbers for all of those attributes.

Getting a Feel for Your Attributes

Before trying to assign numbers to attributes, you should start by getting a feel for what you want to get from those attributes. For example, if you were making a racing game and wanted to create the speeds and acceleration attributes for three different vehicles, you could start with some descriptions of what the speed and acceleration feel should be:

- **Sports car:** Good acceleration and good top speed
- **Muscle car:** Fastest top speed but with less acceleration than a sports car
- **Motorcycle:** Fastest acceleration but lowest top speed

While you have assigned no numbers to these attributes yet, you now have a guide that will help in determining what numbers fit the feel you want.

Determining the Granularity for Numbers

After you come up with attributes for game objects, you need to assign numbers to the attributes. Because you are making up all the attributes and numbers for your game, technically you could use any numbers you want. The granularity of the numbers you use can have a dramatic impact on how a player perceives the game. The following sections provide some to help you determine the granularity of your numbers.

Numbers Should Relate to Probability

Numbers should have a visible impact on the game. The larger the possible outcome of a random event, the larger the corresponding numbers of the game must be. For example, if a character has 10 HP, it doesn't matter if the character receives 11 damage or 5,000 damage, as either one will be a one-hit kill. Say that you know a character is rolling 1D6 (a single six-sided die) for damage, and you always want the character to survive at least three hits. In this case, the minimum hit point value would be 111.

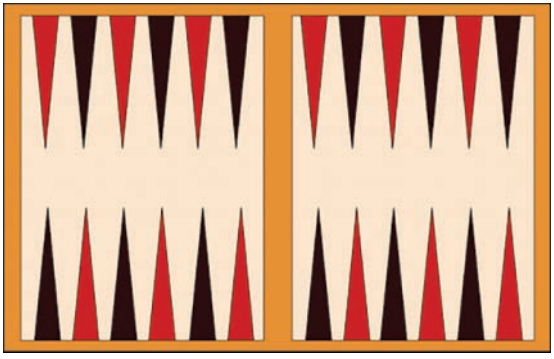

Figure 11.1 Backgammon board

Let's consider backgammon as an example. (Do a search for "official backgammon rules" if you need to familiarize yourself.) In backgammon, the maximum number of moves a piece can take at one time is 24. The maximum is 24 because even the largest roll possible can have a use and not be wasted. In addition, 24 is the number of spaces on the board (see Figure 11.1). The relationships between the number of needed movement spaces and the potential outcomes of the dice are intertwined. If you were to expand the board, you would likely need larger potential rolls to keep the game moving. Conversely, if you were to shrink the board, you would want to reduce the amount of possible movement.

Some Numbers Need to Relate to Real-World Measurements

Some numbers, such as height, weight, and speed, are analogs of the real world. The scale of those numbers has already been decided for you. Even if it is better for your game to use three-digit numbers than to use smaller numbers, you can't decide that every person in your game is going to be measured in hundreds of feet (or meters) in height. Players have incoming knowledge of fixed scales and expect you to play along with the real world. So, if being taller in your game is better, then you will need to adjust your scale. There are a few ways to do this:

- Use a smaller unit of measurement so you get larger numbers.
- Adjust your scale of numbers to fit a fixed attribute.
- Convert the real-world scale to a game scale.

For example, you might list attributes for a basketball player as follows:

Example 1

Strength: 150

Height: 6 (feet)

Speed: 220

Dexterity: 180

This looks odd because the height attribute is a single digit, while the rest of the attributes are triple-digit numbers. In addition to looking odd, this would create the need to use fractions or decimals. Here's another example of attributes for a basketball player:

Example 2

Strength: 150

Height: 182 (centimeters)

Speed: 220

Dexterity: 180

This scale is much better. All the attribute numbers are triple-digit numbers and within a similar range.

Here's another example of attributes for a basketball player:

Example 3

Strength: 50

Height: 72 (inches)

Speed: 73

Dexterity: 60

This scale is also better than the first one. Changing to a more granular measurement of inches and switching all attributes to be two-digit numbers makes them line up nicely.

Now consider this final example of attributes for a basketball player:

Example 4

Strength: 150

Height: 165 (game units)

Speed: 220

Dexterity: 180

This scale also works because you have ditched reality and made your own scale that enables the attributes to all be three-digit numbers in a similar range. Making up your own units may lead to a bit of confusion as a player won't initially know how to picture a height of 165 game units, but you can overcome this difficulty with art.

User Smaller Numbers for Easier Calculations

A player needs clear numbers for each individual calculation and for repeated calculations. If you are asking players to do calculations in their head in the game, then you need to limit the complexity of the numbers. Further, if you are asking players to do many calculations or frequently recurring calculations, you need to further restrict the complexity of those calculations. It is easiest for players to process simple numbers—that is, small whole numbers.

In very old games, attribute numbers are all very small. The number of pieces a player has, the faces of the dice, and total points for a game tend to be no more than two digits. Often they are single digits. Old games use small whole numbers to make the numbers easier for players to remember and use in calculations in their heads. The more frequently a player is required to do calculations, the simpler the calculations tend to be and the smaller the numbers involved are.

Think again about backgammon, for example. Players need to be able to calculate rolls and results in their heads, and complex systems of multiplication or addition would cause unneeded confusion. For each turn in backgammon, a player rolls 2D6 to determine how

much movement their pieces get for that turn. A player gets double that movement with a roll of doubles. (Rolling double 6s, for example, allows the player to move a total of 24 spaces.) On every turn, the player uses the individual rolls of the dice, or adds together the rolls of two six-sided dice, and turns go by in a matter of seconds. Fortunately, adding together the rolls of two six-sided dice is a very easy calculation and does not slow the pace of the game. In addition, the results are all small numbers. The results also tie into the physical space of the game. The board contains only 24 spaces, so any more movement than that would be useless.

Let's now consider scoring in the game spades. Spades has a rather sophisticated scoring system, where players guess their score at the beginning of the game and then, at the end of the game, compare their final results to their initial guess. They then use a scoring system to interpret their results and calculate the final score. This is a somewhat complex calculation, and players often use paper or a calculator to do the scoring—but it is only done once during a game. The numeric results are also much larger than in backgammon, with scores in the hundreds or even up over 1,000. Because this calculation occurs only once a game, it's an event and can even build some tension as a game is calculated, but if it were done every turn, it would completely bog down the game.

Early and even many modern tabletop games and pen-and-paper RPGs continue to use attribute numbers in the single digits and low double digits. For example, a sample fifth edition Dungeons & Dragons character could start with the following attribute scores:

STR 10 DEX 13 CON 14 WIS 19 CHA 14

Note that all of these numbers are in the low two-digit range. Also, while this is a modern, fairly sophisticated game, it is working under the same limitations as backgammon in that the players are needing to do calculations in their head. Whereas in backgammon, players do calculations every few seconds, in an RPG they do calculations every few minutes.

As you can see from these examples, the less frequently calculations are made, the more complex they can be and the larger the numbers involved can be. When assigning numbers to attributes, you should think about how much calculation you expect your players to do in their heads. The more calculations, the smaller the numbers should be for attributes. The more frequent the calculations, the smaller and simpler the calculation and numbers must be.

Use Larger Numbers for More Granularity

If small numbers are easier for players to understand, why not use single-digit numbers for everything? Small numbers do not allow for much granularity or variety. Say that you are assigning strength to five fantasy characters. These are the five characters, and the feeling you want to convey through the strength attribute for each of them:

- **Human:** Middle-of-the-road guy
- **Ogre:** Much stronger than anyone else

- **Ork:** Stronger than humans but significantly weaker than ogres
- **Goblin:** Weakest by far, but not so weak that they can be ignored
- **Dwarf:** Stronger than humans but notably weaker than orks

Here's how you might turn these feelings into numbers if you want to constrain the numbers to 10 and below:

- **Human:** Middle of the road leads you to choose the halfway point, which is 5.
- **Ogre:** Because this is the strongest character, it is 10. Note that there is no longer room on the scale for stronger characters like dragons or giants. While this might be fine within the scope of your game, it does limit your ability to expand the game.
- **Ork:** You might assign an ork a strength of 7 because an ork is much weaker than an ogre but is not that much stronger than a human.
- **Goblin:** A goblin is the weakest character, so you assign it 2, but 2 might be too weak.
- **Dwarf:** You are now stuck. If you assigned a dwarf 6, then this character would be stronger than a human but not notably weaker than an ork.

As you can see, even with just five characters and a few criteria, you start running out of space in the scale to properly translate your feelings about character strengths into numbers. As you add more characters and more criteria, the scale will get even more crowded, and characters will start to feel too similar. To fix this, it is tempting to make all the values considerably larger, allowing more granularity to work with.

Very Large Numbers Are Confusing

Given the problems discussed so far with small numbers, it might seem like a good idea to go to the opposite extreme in a computer game. If you were to use four- or five-digit numbers, you would have plenty of space to make a large variety without ever crowding your range. Further, given that the computer will be doing all the calculations, you don't need to worry about players doing lots of math on big numbers, as they would need to do with a board game. But calculations are not limited to just what a player must do to make the game progress; they also tie in to how well the player can understand what is going on in the game. We humans are, in general, not designed to calculate large numbers in our heads. For example, try to calculate the final hit point score for each of the following scenarios in your head:

- 5 hit points, taking 2 points of damage
- 100 hit points, taking 27 points of damage
- 34863298 hit points, taking 456321 points of damage

It's clear that the smaller the numbers, the easier the calculations.

The takeaway is that you need to find the right amount of granularity for your game. In general, you want to use numbers that are just large enough to accommodate all needed variety but no larger than absolutely necessary.

Humans Hate Decimals and Fractions, but Computers Don't Mind Them

It is exceedingly rare, outside of educational math games, to ever show a player a decimal score or a fraction. It's not that they aren't valid numbers, but people just don't like seeing or (worse) calculating them. Games typically show players only whole numbers.

However, behind the scenes, computers have absolutely no problem calculating decimals. This means you can feel free to use as many decimal places as you want for computer calculations as long as you can present whole (rounded) numbers to the player in a way that is not confusing.

Numbering Example

Figure 11.2 provides an example in which each column presents a pair of values: one for Attribute A and one for Attribute B. In each pair, the ratio of A to B is the same: 94%. Because each pair has the same ratio, for a computer, they would all work exactly the same way. However, players would be able to comprehend some of these numbers easily and others with great difficulty. If the players are going to see the numbers, you should use just the two-digit numbers, if possible, or the three-digit ones.

Attribute A	1.230769231	16	160	4592
Attribute B	1.307692308	17	170	4879
Ratio	94%	94%	94%	94%

Figure 11.2 Number granularity example

The Tension Trick

There is a trick that systems designers can use to cause a wide variation of tension in a game by manipulating a few related numbers. The basic rules for tension are as follows:

- Using numbers that are not easy to calculate creates dissonance for players.
- Dissonance creates tension, fear, and other heightened negative emotions.

- These emotions can heighten an experience, if used properly.
- Using numbers that are easy to calculate creates calmness for players.
- Use easy-to-calculate numbers to give the players a calm, easygoing experience and use numbers that are difficult to calculate to cause more heightened emotions.

For example, say that a player character (PC) has 20 HP, and an enemy character should kill the PC in 4 hits. You could assign these numbers for the least tension:

> Enemy does 5 damage per hit, so the PC is at 5 HP after 3 hits and at 0 HP after 4 hits.

You could assign these numbers for the most tension:

> Enemy does 6 damage per hit, so the PC is at 2 HP after 3 hits and at 0 HP after 4 hits.

In both of these cases, the PC is alive after 3 hits and killed on the fourth, so functionally they are the same. But they can feel very different to a player. Why?

Let's look at it graphically and then break it down further. Imagine that the PC has taken 3 hits. Figure 11.3 shows two options for the health bar for the PC at this point.

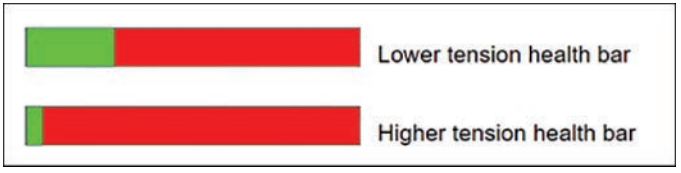

Figure 11.3 Lower- and higher-tension health bars

In both cases, the PC will be killed with the next shot, but which one looks scarier? Players know that more red on a health bar is generally a bad thing. The fact that the lower of the two bars is more red signals to the player, subconsciously, more danger, even though numerically the danger is identical with the two health bars.

Let's look at another example. Say that, in a farming game, the player plants a field that is 20 square meters in 1-square-meter units, so there are 20 total spaces in which to plant. The player has the following resources:

> 5 corn
>
> 10 beans
>
> 5 wheat
>
> 10 rice

In this example, it is fairly easy for a player to calculate the division of crops to plant. All the numbers are easy to grasp and can easily fit in 20, which is also the total number of squares.

Young or inexperienced players should be able to quickly figure out what to do in this scenario, with little stress.

To increase the tension in the same farming game, you can change the units to something more difficult to grasp and also change the amounts to numbers that are more difficult to calculate. This time, say that the player has 2.5 acres to plant and plants in units of 100 square yards. This alone makes the calculations much more difficult for anyone who is not already familiar with converting square yards into acres. In this case, the player would have 121 things to plant. The player has the following resources:

37 corn

63 beans

58 wheat

29 rice

In this revised example, it is very difficult for the player to do the planting calculations in their head. This difficulty will cause a sense of stress and tension. In an action game, this can heighten the player's experience, but in a farming game, it might create stress in what should be a relaxing activity.

There are no universal right or wrong answers about inducing tension in a game through use of numbers, but there are situational rights and wrongs based on the feeling you want the player to have at any given time.

Searching for the Right Numbers

Once you decide on the granularity of the numbers you are going to use, it's time to start plugging in numbers. If you have already described the feel you want with the numbers and determined the number of digits and ratio you want to use, you can do a rough pass immediately.

Keep in mind when doing a first pass at data numbers that they will almost certainly not be what you end up with. This is okay and to be expected. Until a game is tested, it is impossible to know the exact effect numbers will have on the game. Don't think of this as failure; instead realize that you can take the pressure off the first pass. If you approach the first pass knowing that the numbers will be wrong, you don't have the stress of trying to guess right the first time. Instead, you can just get some numbers in there. Use the targeted number of digits and rough ratios for each object and just plug them in.

Let's go back to our racing game example from the beginning of the chapter. Say that you want to make a very simple, new-audience-friendly game, so you want to stick to single-digit

numbers. This is what you came up with earlier for what the speed and acceleration should be:

- **Sports car:** Good acceleration and good top speed
- **Muscle car:** Fastest top speed but with less acceleration than a sports car
- **Motorcycle:** Fastest acceleration but lowest top speed

Based on this list and the fact that you want to use single-digit numbers, you might assign the numbers shown in Table 11.1. Are these numbers right? Almost certainly not. But they're a start.

Table 11.1 Basic data table

Car	Acceleration	Top Speed
Sports car	8	8
Muscle car	6	10
Motorcycle	10	6

When testing numbers, it's a good idea to go beyond reasonable, expected numbers. To find the extents of a range, you must exceed those extents during testing. You want to try making something with too much acceleration or a speed that's too low; for example, you might experiment with your numbers as shown in Table 11.2.

Table 11.2 Experimental data

Car	Acceleration	Top Speed
Sports car	8	8
Muscle car	1	15
Motorcycle	200	10

These numbers are undoubtedly wrong—and, again, that's fine and expected. You are not trying to get the numbers right at this point. Instead, you are trying to understand your game and game engine. Can the engine handle an acceleration of 200? Does this number cause the game to crash? Does collision still work? By testing unreasonable numbers, you can understand the game and engine better, which will make it more likely that you will find interesting and exciting new results.

The great news is that with game data, there is nothing you can do in testing that can't be undone. You can use this aspect of game making to your advantage for wild and interesting tests. Once you have broken the game in interesting ways and understand the mechanical workings better, it's time to home in on the balance you truly want.

The next step is to test and test and test—and then tune and test more and then do more tuning and testing. On this first round of testing, the goal is to get the numbers to emulate what you wrote in your original list of what you feel you want from the numbers. Does that motorcycle feel like it has great acceleration? Does the sports car feel like it has slower acceleration but can eventually top out at the highest speed? Eventually you will find the right balance with the numbers.

Further Steps

After completing this chapter, you should take some time to practice in the real world with the concepts covered here. Try these exercises to further explore the numbers that populate game data:

- Look online for data for your favorite games—in a variety of genres—and analyze the scales used in those games. Take note of the kinds of numbers used for each game and how the games compare with each other in terms of the numbers.
- Take the preceding exercise a step further and redo the values for each of the games by changing their values proportionally. Try doubling them, or multiplying by 10, or multiplying by 0.1 Describe how the feel of the game changes when you change the scale of the data numbers.

INDEX

Numerics

A

B

C

D

E

F

G

H

I

J-K

L

M

N

O

S

T

U-V

W-X-Y-Z