

Introduction to **GAME SYSTEMS DESIGN**



Dax **GAZAWAY**

FREE SAMPLE CHAPTER

SHARE WITH OTHERS



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

Copyright © 2022 Pearson Education, Inc.

All rights reserved. This publication is protected by copyright, and permission must be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise. For information regarding permissions, request forms and the appropriate contacts within the Pearson Education Global Rights & Permissions Department, please visit www.pearson.com/permissions.

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

Preface	xx
Acknowledgments	xxv
About the Author	xxvii
1 Games and Players: Defined	1
2 Roles in the Game Industry	23
3 Asking Questions	31
4 System Design Tools	43
5 Spreadsheet Basics	51
6 Spreadsheet Functions	89
7 Distilling Life into Systems	109
8 Coming Up with Ideas	119
9 Attributes: Creating and Quantifying Life	133
10 Organizing Data in Spreadsheets	145
11 Attribute Numbers	157
12 System Design Foundations	169
13 Range Balancing, Data Fulcrums, and Hierarchical Design	195
14 Exponential Growth and Diminishing Returns	215
15 Analyzing Game Data	229
16 Macrosystems and Player Engagement	245
17 Fine-Tuning Balance, Testing, and Problem Solving	257
18 Systems Communication and Psychology	279
19 Probability	291
20 Next Steps	341
Index	345

This page intentionally left blank

Contents

Prefacexx
Acknowledgmentsxxv
About the Author	xxvii
1 Games and Players: Defined	1
Defining <i>Game</i>	2
Agreed Upon, Artificial Rules	2
Players Have an Impact on the Outcome	3
People Can Opt Out	4
Game Sessions Are Finite	4
Intrinsic Rewards	4
Game Attributes Summary	5
Finding the Target Audience for a Game: Player Attributes	6
Age	6
Gender	7
Tolerance for Learning Rules	7
Interest in Challenge	9
Desired Time Investment	10
Pace Preference	11
Competitiveness	11
Platform Preference	12
Skill Level	12
Genre/Art/Setting/Narrative Preference	13
Value Gained from Players	13
Payment	13
Other Forms of Value	16
Target Audience Value	17
Target Audience Composite	18
Chess	18
<i>Galaga</i>	18

	<i>Mario Kart</i>	19
	<i>The Battle for Wesnoth</i>	20
	<i>Bejeweled</i>	20
	What to Do with a Target Audience Profile	21
	Further Steps	22
2	Roles in the Game Industry	23
	Core Management Team	24
	Vision Holder	24
	Lead Engineer	25
	Lead Artist	25
	Lead Designer	25
	Producer	25
	Lead Sound Designer	25
	Team Subdisciplines	26
	Art	26
	Engineering	27
	Production	28
	Design	28
	Sound Team	29
	QA Team	29
	Narrative Designer	30
	Additional Roles	30
	Further Steps	30
3	Asking Questions	31
	How to Ask a Theoretical Question	32
	Steps of the Scientific Method	32
	Defining a Question for Data Analysis	35
	How to Ask for Help with a Problem	36
	Why How You Ask Matters	36
	Steps to Writing a Good Question	37
	Further Steps	41

4	System Design Tools	43
	What Is Data?	44
	Game Industry Tools	44
	Documentation Tools	45
	Image Editing Tools	45
	3D Modeling Tools	46
	Flowchart Tools	47
	Databases	48
	Bug-Tracking Software	49
	Game Engines	49
	Further Steps	50
5	Spreadsheet Basics	51
	Why Spreadsheets?	52
	What Is a Spreadsheet?	54
	Spreadsheet Cells: The Building Blocks of Data	54
	Cells	54
	The Formula Bar	55
	Spreadsheet Symbols	56
	Data Containers in Spreadsheets	60
	Columns and Rows	60
	Sheets	61
	Workbooks	61
	Spreadsheet Operations	63
	Referencing a Separate Sheet	64
	Hiding Data	65
	Freezing Part of a Sheet	66
	Using Comments and Notes	68
	Using Formfill	71
	Using Filters	77
	Data Validation	80
	The Data Validation Dialog	81
	Time Validation	83

List Validation	84
Named Ranges	84
Further Steps	88
6 Spreadsheet Functions	89
Grouping Arguments	90
Function Structure	90
More Complex Functions	93
Functions for System Designers	96
SUM	96
AVERAGE	97
MEDIAN	97
MODE	98
MAX and MIN	99
RANK	99
COUNT, COUNTA, and COUNTUNIQUE	100
LEN	100
IF	101
COUNTIF	101
VLOOKUP	102
FIND	102
MID	103
NOW	103
RAND	104
ROUND	105
RANDBETWEEN	105
Learning About More Functions	106
How to Choose the Right Function	106
Further Steps	107
7 Distilling Life into Systems	109
An Abstract Example	114
Throwing	114

Sticks	115
Running	115
Teamwork	115
Putting Together the Mechanics	115
Story in Games	116
Further Steps	117
8 Coming Up with Ideas	119
Idea Buffet	120
Sample Idea Buffet	120
Running a Brainstorming Session	121
Having Goals	121
Gathering the Troops	122
Giving Yourself a Block of Time	123
Don't Accept the First Answer	123
Avoiding Criticism	124
Keeping on Topic (Kind Of)	124
Capturing the Creativity	125
Keeping Expectations Reasonable	125
Percolating	125
Methods to Force Creativity	126
Bad Storming	126
Jokes	126
Building Blocks	127
Future Past	127
Iterative Stepping	127
Halfway Between	128
Opposite Of	129
Random Connections	130
Stream of Consciousness Writing	130
Further Steps	131
9 Attributes: Creating and Quantifying Life	133
Mechanics Versus Attributes	134

Listing Attributes	134
Initial Brainstorming	135
Blue-Sky Brainstorming	136
Researching Attributes	136
Referring to Your Own Personal Attribute Bank	138
Defining an Attribute	139
Considerations When Defining an Attribute	140
Grouping Attributes	141
Further Steps	143
10 Organizing Data in Spreadsheets	145
Create a Spreadsheet to Be Read by an Outsider	146
Avoid Typing Numbers	146
Label Data	147
Validate Your Data	148
Use Columns for Attributes and Rows for Objects	148
Color Coding	149
Avoid Adding Unneeded Columns or Rows or Blank Cells	151
Separate Data Objects with Sheets	152
Reference Sheet	152
Introduction Sheet	153
Output/Visualization Sheets	154
Scratch Sheet	155
Spreadsheet Example	155
Further Steps	156
11 Attribute Numbers	157
Getting a Feel for Your Attributes	158
Determining the Granularity for Numbers	158
Numbers Should Relate to Probability	158
Some Numbers Need to Relate to Real-World Measurements	159
User Smaller Numbers for Easier Calculations	160
Use Larger Numbers for More Granularity	161

Very Large Numbers Are Confusing	162
Humans Hate Decimals and Fractions, but Computers Don't Mind Them.	163
Numbering Example	163
The Tension Trick.	163
Searching for the Right Numbers	165
Further Steps	167
12 System Design Foundations	169
Attribute Weights	170
DPS and Intertwined Attributes	173
Binary Searching	176
How Binary Searching Works.	176
Lacking a Viable Range	179
Naming Conventions	180
Naming Object Iterations	185
The Problem with "New".	185
Iteration Naming Method 1: Version Number.	186
Iteration Naming Method 2: Version Letter and Number.	186
Special Case Terms	187
Using the Handshake Formula.	188
Further Steps	194
13 Range Balancing, Data Fulcrums, and Hierarchical Design	195
Range Balancing	196
How Range Balancing Works	197
Who Adjusts What	201
Data Fulcrums	203
What Is a Fulcrum?	203
Creating a Fulcrum	204
Testing a Fulcrum	204
Locking a Fulcrum.	206
Using a Fulcrum for Data Creation	206

	Unavoidable Cross-testing	208
	Fulcrum Progression	209
	Hierarchical Design	210
	Starting the Hierarchy	211
	Advantages of Hierarchical Design.	212
	Further Steps	213
14	Exponential Growth and Diminishing Returns	215
	Linear Growth	216
	Exponential Growth	217
	Parts of the Basic Exponential Growth Formula.	218
	Building Blocks of the Exponential Growth Formula	220
	Tweaking the Basic Exponential Growth Formula.	226
	A Note on Iterations.	227
	Exponential Charts and Game Hierarchy	227
	Further Steps	228
15	Analyzing Game Data	229
	Overview Analysis	230
	Next-Level Deep Analysis	238
	Practicing Data Analysis	240
	Comparison Analysis.	240
	Canaries	241
	Further Steps	244
16	Macrosystems and Player Engagement	245
	Macrosystem Difficulty Adjustment.	246
	Flat Balancing	246
	Positive Feedback Loops	247
	Negative Feedback Loop	249
	Dynamic Difficulty Adjustment	251
	Layered Difficulty Adjustment	253
	Cross-Feeding	254

Balancing Combinations	255
Further Steps	255
17 Fine-Tuning Balance, Testing, and Problem Solving	257
Balance	258
Why Balance Matters	258
General Game Balance	259
Breaking Your Data	261
Problems with Balancing Judged Contests	261
How to Start Balancing Data	263
Performing Playtests	265
Minimum Viability Testing	266
Balance Testing	267
Bug Testing	268
User Testing	269
Beta/Postlaunch Telemetry Testing	273
Solving Problems.	275
Identify the Problem	276
Eliminate Variables	277
Come Up with Solutions	277
Communicate with the Team.	277
Prototype and Test	277
Document the Changes	277
Further Steps	278
18 Systems Communication and Psychology.	279
Games as Conversations.	280
Word Meanings	281
Noise	284
Reciprocity	286
Overstepping Bounds.	286
Shallow Relationship	287
Right Balance	287

Reward Expectations	288
Further Steps	289
19 Probability	291
Basic Probability	292
Probability Notation	292
Calculating One-Dimensional Even-Distribution Probability.	293
Calculating One-Dimensional Uneven-Distribution Probability	299
Calculating Compound Probability	301
Calculating 2D6 “Or Higher” Cumulative Probability	309
Calculating the Probability of Doubles	310
Calculating a Series of Single Events	311
Calculating More Than Two Dimensions	316
Calculating Dependent Event Probability	318
Calculating Mutually Exclusive Event Probability	321
Calculating Enumerated Probability with an Even Distribution	321
Calculating Enumerated Probability with an Uneven Distribution	322
Calculating Attributes Weights Based on Probability	325
Calculating Imperfect Information Probability	327
Perception of Probability	328
Probability Uncertainty	328
Mapping Probability	329
Attributes of a Random Event	329
Mapping Probability Examples	331
Measuring Luck in a Game	334
Testing for Pure Luck	335
Testing for Luck Dominant	335
Testing for Luck Influenced	336
Adjusting the Influence of Luck	336
Chaos Factor.	338
Further Steps	338

20	Next Steps.	341
	Practice	342
	Analyze Existing Games	342
	Play New Games	342
	Modify Existing Games	342
	Work on Your Game	343
	Keep Learning	343
	Index	345

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 your 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

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.

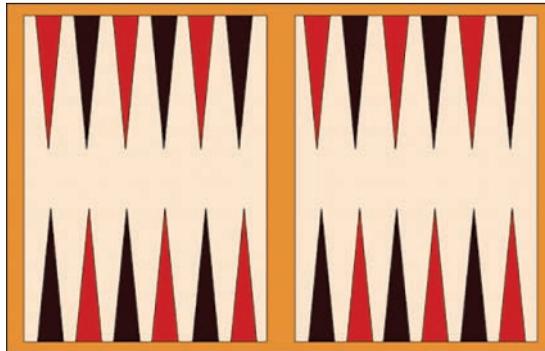


Figure 11.1 Backgammon board

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

- 2D6 “or higher” cumulative probability, calculating, 309-310
- 3D modeling tools, 46-47

A

- absolute referencing, 75
- acquiescence bias, 32
- adjusting influence of luck, 336-338
- advertising, 16
 - social media, 16-17
 - word of mouth, 16
- ampersand (&), 59-60
- analyzing game data, 229-230
 - canaries, 241-244
 - comparison analysis, 240-241
 - existing games, 342
 - next-level deep analysis, 238-240
 - overview analysis, 230-238
 - practicing, 240
- animators, 26
- arguments
 - grouping, 90
 - in more complex functions, 93-95
- asking questions, 36-37. *See also* defining a
 - question for data analysis; scientific method; writing a good question
 - acquiescence bias, 32
 - bad question example, 40
 - good question example, 39
 - for help with a problem, 36
 - theoretical, 32
 - determining numbers to use*, 34
 - form an explanatory hypothesis*, 35
 - test the hypothesis*, 35
- assistants, 28
- attribute(s), 158
 - bank, 138
 - brainstorming, 135-136
 - comparison analysis, 240-241
 - damage per minute, 174
 - data and, 44
 - defining, 139-141
 - DPS (damage per second), 173, 174, 175
 - grouping, 141-143
 - intertwined, 175-176
 - listing, 134-135
 - mechanics and, 134
 - naming conventions, 183-184
 - next-level deep analysis, 238-240
 - numbers
 - accuracy and*, 165-167
 - determining granularity*, 158
 - for easier calculations*, 160-161
 - fractions and decimals*, 163
 - for granularity*, 161-162
 - individual balance*, 201
 - range balancing*, 196-203
 - relating to probability*, 158-159
 - relating to real-world measurements*, 159-160
 - systemic balance*, 201
 - tension trick*, 163-165
 - very large*, 162-163
 - overview analysis, 230-238
 - in past games, researching, 137-138
 - placing in spreadsheets, 148-149
 - of random events
 - computation use*, 329-330
 - outcome dependency*, 331
 - probability distribution*, 331
 - types of randomness*, 330
 - real-world, researching, 137
 - researching, 136-137
 - weights, 170-172
 - balance and*, 172-173
 - calculating based on probability*, 325-327
- audience, 6
- audio engineers, 27
- AVERAGE function, 97

B

- bad storming, 126
- balance, 258
 - handshake formula, 188-194
 - applications*, 193-194
 - possibility grid*, 189-192
 - importance of, 258-259
 - indicators of, 259-261
 - judged contests and, 261-262
 - prototyping and, 263
 - reciprocity and, 287-288
 - testing, 267-268
 - weighted attributes and, 172-173
- basic exponential growth formula, 218-220
 - building blocks, 220-226
 - tweaking, 226-227
- Battle for Wesnoth, The*, target audience profile, 20
- Bejeweled*, target audience profile, 20-21
- beta/postlaunch telemetry testing, 273
 - data hooks, 273-274
 - examples, 274-275
- binary searching, 176
 - boss fight example, 178-179
 - jump distance example, 179
 - lacking a viable range, 179-180
 - maximum number of guesses, 178
 - requirements, 176
- blue-sky brainstorming, 136
- brainstorming
 - avoiding criticism, 124
 - blue-sky, 136
 - capturing the creativity, 125
 - don't accept the first answer, 123-124
 - gathering the troops, 122
 - goals and, 121-122
 - keeping expectations reasonable, 125
 - keeping on topic, 124-125
 - listing attributes, 135-136
 - methods to force creativity, 126
 - bad storming*, 126
 - building blocks*, 127
 - future past*, 127
 - halfway between method*, 128-129
 - iterative stepping*, 127-128
 - jokes*, 126
 - opposite of method*, 129

- random connections*, 130
- stream of consciousness writing*, 130

- percolating, 125
- time and, 123
- breaking your data, 261
- bug testing, 268
- bug-tracking software, 49

C

- calculating probability
 - 2D6 "or higher" cumulative, 309-310
 - compound, 301-309
 - dependent event, 318-321
 - of doubles, 310-311
 - enumerated probability with an even distribution, 321-322
 - enumerated probability with an uneven distribution, 322-325
 - multi-dimensional, 316-318
 - mutually exclusive event, 321
 - one-dimensional even-distribution, 293-299
 - one-dimensional
 - uneven-distribution, 299-300
 - of a series of single events, 311-316
- canaries, 241-244
- casual gamers, 7
- cells, 54
 - address, 54-55
 - formula bar, 55-56
 - references, 146-147
 - value, 55
- chaos factor, 338
- character artists, 26
- chess, target audience profile, 18
- choosing, functions, 106-107
- columns, 60-61, 148-149
 - avoiding unnecessary, 151-152
- COMBIN function, 194
- communication, 279
 - language and, 281
 - noise and, 284-286
 - reciprocity, 286
 - balance and*, 287-288
 - overstepping bounds*, 286-287
 - reward expectations*, 288-289
 - shallow relationship*, 287
 - word meanings and, 281-284

comparison analysis, 240-241
 compound probability, calculating, 301-309
 computation use, 329-330
 concept artists, 26
 conversations, games as, 280-281
 coordinators, 28
 core management team, 24
 COUNT function, 100, 233
 COUNTA function, 100
 COUNTIF function, 94-95, 101, 238, 304-305
 COUNTUNIQUE function, 100
 crafting, prototypes, 263-264
 creating, data fulcrums, 204
 creativity
 brainstorming
 avoiding criticism, 124
 bad storming, 126
 blue-sky, 136
 building blocks, 127
 don't accept the first answer, 123-124
 future past, 127
 gathering the troops, 122
 goals and, 121-122
 halfway between method, 128-129
 iterative stepping, 127-128
 jokes and, 126
 keeping expectations reasonable, 125
 keeping on topic, 124-125
 listing attributes, 135-136
 opposite of method, 129
 percolating, 125
 random connections, 130
 stream of consciousness writing, 130
 time and, 123
 capturing, 125
 coming up with ideas, 119-120
 idea buffet, 120-121
 methods to force, 126
 criticism, brainstorming and, 124
 cross-feeding, 254-255
 cross-testing, fulcrums, 208-209

D

damage per minute, 174
 converting to DPS (damage per second), 174

data, 44
 labelling, 147-148
 validating, 148
 validation, 80-81
 data designer, 29
 data fulcrums, 203. *See also* hierarchical design
 creating, 204
 cross-testing, 208-209
 for data creation, 206-208
 locking, 206
 progression, 209-210
 testing, 204-205
 databases, 48-49
 DDA (dynamic difficulty adjustment), 251-252
 defining
 attributes, 139-141
 questions for data analysis, 35-36
 dependent event probability,
 calculating, 318-321
 difficulty adjustment, 246
 balancing combinations, 255
 cross-feeding, 254-255
 dynamic, 251-252
 flat balancing, 246-247
 layered, 253-254
 negative feedback loops, 249-251
 positive feedback loops, 247-248
 diminishing returns, 215-216
 documentation tools, 45
 doubles, calculating probability of, 310-311
 DPS (damage per second), 173, 174, 175
 calculating, 174

E

enumerated probability
 with an even distribution,
 calculating, 321-322
 with an uneven distribution, calculating,
 322-325
 environmental artists, 26
 equal sign (=), 56-57
 exponential growth, 215, 216, 217-218
 basic exponential growth formula, 218-220
 building blocks, 220-226
 tweaking, 226-227
 iterations and, 227

F

filters, spreadsheet, 77-79
 FIND function, 102-103
 flat balancing, 246-247
 flowchart tools, 47-48
 formfill, 71-76
 formula bar, 55-56

- equal sign (=), 56-57

 formulas. *See also* functions

- ampersand (&), 59-60
- basic exponential growth, 218-220
 - building blocks*, 220-226
 - tweaking*, 226-227
- mathematical symbols, 59
- parentheses, 58

 fulcrums, 203-204. *See also* hierarchical design

- creating, 204
- cross-testing, 208-209
- for data creation, 206-208
- locking, 206
- progression, 209-210
- testing, 204-205

 functions, 89, 106

- arguments, 89
 - grouping*, 90
- AVERAGE, 97
- choosing, 106-107
- COMBIN, 194
- complex, 93-94
- COUNT, 100, 233
- COUNTA, 100
- COUNTIF, 94-95, 101, 238, 304-305
- COUNTUNIQUE, 100
- FIND, 102-103
- IF, 101
- LEN, 100
- MAX, 99
- MEDIAN, 97-98
- MID, 103
- MIN, 99
- MODE, 98, 234
- NOW, 103-104
- RAND, 104
- RANDBETWEEN, 105
- RANK, 99
- ROUND, 105
- SORT, 238

- structure, 90-93
- SUM, 91-93, 96-97
- syntax, 93
- UNIQUE, 238
- VLOOKUP, 102

 future past method, 127

G

Galaga, 134

- target audience profile, 18-19

 game development, 24

- art
 - animation*, 26
 - character art*, 26
 - concept art*, 26
 - environmental art*, 26
 - interface art*, 26
 - technical art*, 27
- core management team, 24
- design, 28
 - data designer*, 29
 - game system designer*, 28-29
 - level designer*, 28
 - scripter*, 29
 - technical designer*, 29
- engineering
 - audio engineer*, 27
 - gameplay engineer*, 27
 - graphics engineer*, 27
 - network engineer*, 27
 - scripter*, 27
 - tools engineer*, 27
- game developer, 24
- hierarchical design, 210-211
 - advantages of*, 212-213
 - exponential charts and*, 227-228
 - starting the hierarchy*, 211-212
- lead designer, 25
- lead engineer, 25
- lead sound designer, 25
- narrative designer, 30
- producer, 25
- production, 28
 - assistants*, 28
 - coordinators*, 28
 - management*, 28

- QA team, 29-30
 - scientific method and, 32
 - analyze the data*, 35
 - define a question for playtesting*, 32-34
 - form an explanatory hypothesis*, 35
 - gather information and resources*, 34
 - interpret the data, draw conclusions, and publish results*, 35
 - retest*, 35
 - test the hypothesis*, 35
 - sound team, 29
 - tools
 - 3D modeling*, 46-47
 - bug-tracking software*, 49
 - databases*, 48-49
 - documentation*, 45
 - flowchart*, 47-48
 - game engines*, 49-50
 - image editing*, 45-46
 - vision holder, 24-25
 - game engines, 49-50
 - game mechanic(s), 112-114
 - attributes and, 134
 - putting together, 115-116
 - running, 115
 - sticks, 115
 - teamwork, 115
 - throwing, 114-115
 - game system design(er), 28-29, 52, 116-117. *See also* asking questions; functions; spreadsheets; tools
 - attribute weights, 170-172
 - balance, 258
 - attribute weights and*, 172-173
 - importance of*, 258-259
 - indicators of*, 259-261
 - judged contests and*, 261-262
 - prototyping and*, 263
 - binary searching, 176-179
 - boss fight example*, 178-179
 - jump distance example*, 179
 - lacking a viable range*, 179-180
 - maximum number of guesses*, 178
 - requirements*, 176
 - damage per minute, 174
 - converting to DPS (damage per second)*, 174
 - DPS (damage per second), 173, 174, 175
 - calculating*, 174
 - inclusivity and, 13
 - intertwined attributes, 175-176
 - naming conventions, 180-185
 - attribute(s) and*, 183-184
 - object iterations*, 185, 186
 - spaces and*, 183
 - special case words*, 187-188
 - using "new"*, 185-186
 - gameplay engineers, 27
 - games, 2, 342-343
 - adjusting influence of luck, 336-338
 - analyzing, 342
 - attributes, 5
 - finite sessions*, 4
 - intrinsic rewards*, 4-5
 - people can opt out*, 4
 - players have an on the outcome*, 3-4
 - rules*, 2-3
 - chaos factor, 338
 - as conversations, 280-281
 - lead artist, 25
 - luck dominant, 335-336
 - luck-influenced, 336
 - modifying, 342-343
 - new, 342
 - puzzles and, 5
 - session time, 10
 - stories, 116-117
 - total time, 10
 - goals, brainstorming and, 121-122
 - Google Sheets, 62. *See also* spreadsheets
 - graphics engineers, 27
 - grouping, attributes, 141-143
- ## H
- halfway between method, 128-129
 - handshake formula, 188-194
 - applications, 193-194
 - possibility grid, 189-192
 - hardcore gamers, 7
 - hiding, spreadsheet data, 65-66
 - hierarchical design, 210-211
 - advantages of, 212-213
 - exponential charts and, 227-228
 - starting the hierarchy, 211-212

I

idea buffet, 120-121. *See also* creativity
 IF function, 101
 image editing tools, 45-46
 imperfect information probability, 327-328
 inclusivity, 13
 interface artists, 26
 intertwined attributes, 175-176
 introduction sheet, 153-154
 iterative stepping, 127-128

J-K

jokes, creativity and, 126
 judged contests, balancing, 261-262
 jumping, prototypes, 264-265

L

labelling data, 147-148
 language, 281
 layered difficulty adjustment, 253-254
 lead artist, 25
 lead designer, 25
 lead engineer, 25
 lead sound designer, 25
 LEN function, 100
 level designer, 28
 leveling up, prototypes, 264
 linear growth, 216-217. *See also* exponential growth
 list validation, 84
 listing, attributes, 134-135
 locking, fulcrums, 206
 luck
 adjusting influence of, 336-338
 dominant, testing for, 335-336
 influence, 336
 measuring, 334-335
 pure, testing for, 335

M

macrosystem difficulty adjustment. *See* difficulty adjustment
 mapping probability, 329
Mario Kart, target audience profile, 19

mathematical symbols, 59
 MAX function, 99
 measuring luck, 334-335
 mechanic(s), 112-114
 attributes and, 134
 putting together, 115-116
 running, 115
 sticks, 115
 teamwork, 115
 throwing, 114-115
 MEDIAN function, 97-98
 MID function, 103
 MIN function, 99
 minimum viable testing, 266-267
 MODE function, 98, 234
 multi-dimensional probability, 316-318
 mutually exclusive event probability, calculating, 321

N

named ranges, 84-87
 naming conventions, 180-185
 object iterations, 185, 186-187
 spaces and, 183
 special case words
 date or time, 188
 "deleteme", 187
 "deprecated", 187-188
 "test", 187
 using "new", 185-186
 negative feedback loops, 249-251
 network engineers, 27
 next-level deep analysis, 238-240
 noncompetitive games, 11-12
 NOW function, 103-104

O

objects, 152
 categories, 182
 data fulcrums, 203
 creating, 204
 cross-testing, 208-209
 for data creation, 206-208
 locking, 206
 progression, 209-210
 testing, 204-205

- naming conventions, 180-187
 - special case words*, 187-188
- naming iterations, 185
- placing in spreadsheets, 148-149
- one-dimensional even-distribution probability, calculating, 293-299
- one-dimensional uneven-distribution probability, calculating, 299-300
- one-time purchase(s), 14
- opposite of method, 129
- outcome dependency, 331
- output/visualization sheet, 154-155
- overview analysis, 230-238

P

- parentheses, 58
- payment, 13
 - advertising and, 16
 - expansions, 14-15
 - microtransactions, 15-16
 - one-time purchase, 14
 - other forms of value, 16
 - content creation*, 17
 - market numbers*, 17
 - player interaction*, 17
 - popularity contests*, 17
 - ranking sites*, 17
 - social media*, 16-17
 - word of mouth*, 16
- perception of probability, 328
- platforms, 12
- players
 - attributes, 6
 - age*, 6-7
 - competitiveness*, 11-12
 - desired time investment*, 10
 - gender*, 7
 - genre/art/setting/narrative preference*, 13
 - interest in challenge*, 9-10
 - pace preference*, 11
 - platform preference*, 12
 - skill level*, 12-13
 - tolerance for learning rules*, 7-9
 - value gained from, 13
- playtesting, 265-266
 - balance testing, 267-268
 - beta/postlaunch telemetry testing, 273
 - data hooks*, 273-274
 - examples*, 274-275
- bug testing, 268
- defining a question for, 32-34
- minimum viable testing, 266-267
- user testing, 269-273
- positive feedback loops, 247-248
- possibility grid, 189-192
- practicing data analysis, 240
- probability, 291-292. *See also* calculating; luck
 - 2D6 “or higher” cumulative, 309-310
 - calculating attribute weights based on, 325-327
 - compound, 301-309
 - dependent event, 318-321
 - dice, 293
 - distribution, 331
 - of doubles, 310-311
 - enumerated probability with an even distribution, 321-322
 - enumerated probability with an uneven distribution, 322-325
 - imperfect information, 327-328
 - mapping, 329, 331-334
 - multi-dimensional, 316-318
 - mutually exclusive event, 321
 - notation, 292-293
 - one-dimensional even-distribution, 293-299
 - one-dimensional uneven-distribution, 299-300
 - perception of, 328
 - relating attribute numbers to, 158-159
 - of a series of single events, 311-316
 - uncertainty and, 328-329
- producers, 25
- prototypes, 263
 - crafting, 263-264
 - jumping, 264-265
 - leveling up, 264
 - macrosystems, 265
- pure luck, testing for, 335
- puzzles, 5

Q-R

- QA team, 29-30
- questions, writing, 37-40
- quotation marks, 58-59

- RAND function, 104
 - RANDBETWEEN function, 105
 - random connections, brainstorming and, 130
 - random events
 - computation use, 329-330
 - outcome dependency, 331
 - probability distribution, 331
 - types of randomness, 330
 - randomness, 292
 - range balancing, 196-203
 - individual balance, 201
 - systemic balance, 201
 - RANK function, 99
 - real-world attributes, researching, 137
 - reciprocity, 286
 - balance and, 287-288
 - overstepping bounds, 286-287
 - reward expectations, 288-289
 - shallow relationship, 287
 - ref sheet, 152-153
 - relative referencing, 73, 75
 - researching, attributes, 136-137
 - in past games, 137-138
 - real-world, 137
 - rewards, 288-289
 - ROUND function, 105
 - rows, 60-61, 148-149
 - avoiding unnecessary, 151-152
 - rules, 2-3
 - data and, 44
 - tolerance for learning, 7-9
 - running, 115
- S**
- scientific method
 - analyze the data, 35
 - define a question for playtesting, 32-34
 - form an explanatory hypothesis, 35
 - gather information and resources, 34
 - interpret the data, draw conclusions, and publish results, 35
 - retest, 35
 - test the hypothesis, 35
 - scratch sheet, 155
 - scripters, 27, 29
 - series of probability events, calculating, 311-316
 - session time, 10
 - sheets
 - data objects and, 152
 - introduction, 153-154
 - output/visualization, 154-155
 - ref, 152-153
 - scratch, 155
 - skills, 12-13
 - social media, 16-17
 - solving problems, 275-276
 - canaries and, 241-244
 - come up with solutions, 277
 - communicate with the team, 277
 - document the changes, 277
 - eliminate variables, 277
 - identify the problem, 276
 - prototype and test, 277
 - SORT function, 238
 - sound team, 29
 - special case words
 - date or time, 188
 - "deleteme", 187
 - "test", 187
 - spreadsheets, 49, 52-53, 54, 68, 146. *See also* functions
 - ! operator, 64-65
 - absolute referencing, 75
 - calculating probability
 - 2D6 "or higher" cumulative, 309-310
 - compound, 301-309
 - dependent event, 318-321
 - of doubles, 310-311
 - enumerated probability with an even distribution, 321-322
 - enumerated probability with an uneven distribution, 322-325
 - multi-dimensional, 316-318
 - mutually exclusive event, 321
 - one-dimensional even-distribution, 293-299
 - one-dimensional uneven-distribution, 299-300
 - of a series of single events, 311-316
 - cells, 54
 - address, 54-55
 - formula bar, 55-56
 - value, 55
 - color coding, 149-150
 - columns and rows, 60-61

comments, 68-70
 data
 labelling, 147-148
 validating, 148
 data validation, 80-81
 data validation dialog, 81-83
 filters, 77-79
 formfill, 71-76
 freezing part of a sheet, 66-67
 functions, 89, 106
 AVERAGE, 97
 choosing, 106-107
 complex, 93-94
 COUNTA, 100
 COUNTIF, 94-95, 101
 COUNTUNIQUE, 100
 FIND, 102-103
 grouping, 90
 IF, 101
 LEN, 100
 MAX, 99
 MEDIAN, 97-98
 MID, 103
 MIN, 99
 MODE, 98
 NOW, 103-104
 RAND, 104
 RANDBETWEEN, 105
 RANK, 99
 ROUND, 105
 structure, 90-93
 SUM, 96-97
 syntax, 93
 VLOOKUP, 102
 hiding data, 65-66
 list validation, 84
 named ranges, 84-87
 notes, 70-71
 possibility grid, 189-192
 references, 146-147
 referencing a separate sheet, 64-65
 relative referencing, 73, 75
 sheets, 61, 152
 introduction, 153-154
 output/visualization, 154-155
 ref, 152-153
 scratch, 155

symbols
 ampersand (&), 59-60
 equal sign (=), 56-57
 mathematical, 59
 parentheses, 58
 quotation marks, 58-59
 time validation, 83-84
 VisiCalc, 53
 workbooks, 60, 61-63
 sticks, 115
 stories, 116-117
 stream of consciousness writing, 130
 SUM function, 91-93, 96-97
 surveys, acquiescence bias, 32
 systems, 109-112

T

target audience
 age and, 6-7
 competitiveness, 11-12
 desired time investment, 10
 gender, 7
 genre/art/setting/narrative preference, 13
 interest in challenge, 9-10
 pace preference, 11
 platform preference, 12
 profiles, 21-22
 The Battle for Wesnoth, 20
 Bejeweled, 20-21
 chess, 18
 Galaga, 18-19
 Mario Kart, 19
 skill level, 12-13
 tolerance for learning rules, 7-9
 value gained from, 13
 value of, 17-18
 teamwork, 115
 technical artists, 27
 technical designer, 29
 telemetry testing, 273
 data hooks, 273-274
 examples, 274-275
 testing
 canaries and, 241-244
 fulcrums, 204-205

- handshake formula, 188-194
 - applications*, 193-194
 - possibility grid*, 189-192
- for luck dominant, 335-336
- for luck influence, 336
- for pure luck, 335
- throwing, 114-115
- time validation, 83-84
- tools. *See also* spreadsheets
 - 3D modeling, 46-47
 - bug-tracking software, 49
 - databases, 48-49
 - documentation, 45
 - flowchart, 47-48
 - game engines, 49-50
 - image editing, 45-46
- tools engineers, 27
- total time, 10
- toys, 5
- troubleshooting. *See* solving problems

U-V

- UNIQUE function, 238
- user testing, 269-273
- validation, 148
- video game industry, 23
- VisiCalc, 53
- vision holder, 24-25
- VLOOKUP function, 102

W-X-Y-Z

- weapons
 - damage per minute, 174
 - DPS (damage per second), 173, 174, 175
- weighted attributes, 170-172
 - balance and, 172-173
 - calculating based on probability, 325-327
 - DPS (damage per second), 173, 174, 175
- word meanings, 281-284
- word of mouth, 16
- workbooks, 60, 61-63
- writing a good question, 37-39