FROM CONCEPT TO PLAYABLE GAME

WITH UNITY™ AND C#

Introduction to GAME DESIGN, PROTOTYPING, and DEVELOPMENT

Jeremy **GIBSON**

FREE SAMPLE CHAPTER

SHARE WITH OTHERS

Praise for *Introduction to Game Design, Prototyping, and Development*

"*Introduction to Game Design, Prototyping, and Development* combines a solid grounding in evolving game design theory with a wealth of detailed examples of prototypes for digital games. Together these provide an excellent introduction to game design and development that culminates in making working games with Unity. This book will be useful for both introductory courses and as a reference for expert designers. I will be using this book in my game design classes, and it will be among those few to which I often refer."

—Michael Sellers
Professor of Practice in Game Design, Indiana University, former Creative Director at Rumble Entertainment, and General Manager at Kabam

"Prototyping and play-testing are often the most misunderstood and/or underutilized steps in the game design and development process. Iterative cycles of testing and refining are key to the early stages of making a good game. Novices will often believe that they need to know everything about a language or build every asset of the game before they can really get started. Gibson's new book prepares readers to go ahead and dive in to the actual design and prototyping process right away; providing the basics of process and technology with excellent "starter kits" for different types of games to jumpstart their entry into the practice."

—Stephen Jacobs
Associate Director, RIT Center for Media, Art, Games, Interaction and Creativity (MAGIC) and Professor, School of Interactive Games and Media

"Jeremy Gibson's *Introduction to Game Design, Prototyping, and Development* deftly combines the necessary philosophical and practical concepts for anyone looking to become a Game Designer. This book will take you on a journey from high-level design theories, through game development concepts and programming foundations in order to make your own playable video games. Jeremy uses his years of experience as a professor to teach the reader how to think with vital game design mindsets so that you can create a game with all the right tools at hand. A must-read for someone who wants to dive right into making their first game and a great refresher for industry veterans."

—Michelle Pun
Senior Game Designer, Zynga

This page intentionally left blank

Introduction to Game Design, Prototyping, and Development

The Addison-Wesley Game Design and Development Series

USING JAVASCRIPT AND HTML5 TO DEVELOP GAMES
THE WEB GAME DEVELOPER'S COOKBOOK
Evan BURCHARD
A PLATFORM-AGNOSTIC APPROACH
Game Programming ALGORITHMS and TECHNIQUES
Sanjay MADHAV
EXPLORING THE FOUNDATIONAL PRINCIPLES BEHIND GOOD GAME DESIGN
A GAME DESIGN VOCABULARY
Anna ANTHROPY
Naomi CLARK
A PRACTICAL GUIDE TO GRAPHICS PROGRAMMING
REAL-TIME 3D RENDERING with DIRECTX and HLSL
Paul VARCHOLIK
Addison-Wesley

Visit **informit.com/series/gamedesign** for a complete list of available publications.

Essential References for Game Designers and Developers

These practical guides, written by distinguished professors and industry gurus, cover basic tenets of game design and development using a straightforward, common-sense approach. The books encourage readers to try things on their own and think for themselves, making it easier for anyone to learn how to design and develop digital games for both computers and mobile devices.

Make sure to connect with us!
informit.com/socialconnect

informIT.com the trusted technology learning source | Addison-Wesley | Safari Books Online

ALWAYS LEARNING PEARSON

Introduction to Game Design, Prototyping, and Development

From Concept to Playable Game—with Unity® and C#

Jeremy Gibson

Addison-Wesley

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

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.

The authors 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.

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 international@pearsoned.com.

Visit us on the Web: informit.com/aw

Library of Congress Control Number: 2014936195

Copyright © 2015 Pearson Education, Inc.

All rights reserved. Printed in the United States of America. 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. To obtain permission to use material from this work, please submit a written request to Pearson Education, Inc., Permissions Department, One Lake Street, Upper Saddle River, New Jersey 07458, or you may fax your request to (201) 236-3290.

ISBN-13: 978-0-321-93316-4
ISBN-10: 0-321-93316-8

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

Second Printing: January 2015

Editor-in-Chief
Mark Taub

Senior Acquisitions Editor
Laura Lewin

Senior Development Editor
Chris Zahn

Managing Editor
Kristy Hart

Project Editor
Elaine Wiley

Copy Editor
Keith Cline

Indexer
Ken Johnson

Proofreader
Paula Lowell

Technical Editors
Marc Destefano
Charles Duba
Margaret Moser

Publishing Coordinator
Olivia Basegio

Cover Designer
Chuti Prasersith

Book Designer
Bumpy Design

Compositor
Nonie Ratcliff

This book is dedicated to:

My wife Melanie, the love of my life,
for her love, intellect, and support

My parents and sisters

And my many professors, colleagues, and students
who inspired me to write this book.

This page intentionally left blank

Contents at a Glance

Part I Game Design and Paper Prototyping 1

1 Thinking Like a Designer 3

2 Game Analysis Frameworks 19

3 The Layered Tetrad 31

4 The Inscribed Layer 39

5 The Dynamic Layer 61

6 The Cultural Layer 79

7 Acting Like a Designer 89

8 Design Goals 105

9 Paper Prototyping 125

10 Game Testing 141

11 Math and Game Balance 155

12 Puzzle Design 185

13 Guiding the Player 197

14 The Digital Game Industry 211

Part II Digital Prototyping 223

15 Thinking in Digital Systems 225

16 Introducing Our Development Environment: Unity 235

17 Introducing Our Language: C# .253

18 Hello World: Your First Program 263

19 Variables and Components 281

20 Boolean Operations and Conditionals 299

21 Loops 315

22 Lists and Arrays 327

23 Functions and Parameters 349

24 Debugging 363

25 Classes 379

26 Object-Oriented Thinking 391

27 The Agile Mentality 405

Part III Game Prototype Examples and Tutorials. 417
28 Prototype 1: Apple Picker 419
29 Prototype 2: Mission Demolition. 449
30 Prototype 3: Space SHMUP 487
31 Prototype 4: Prospector Solitaire. 561
32 Prototype 5: Bartok 621
33 Prototype 6: Word Game. 657
34 Prototype 7: QuickSnap 695
35 Prototype 8: Omega Mage 727

Part IV Appendices 791
A Standard Project Setup Procedure. 793
B Useful Concepts Reference 799
C Online References 851

Index. 857

Contents

Preface xxiv

Part I Game Design and Paper Prototyping 1

1 Thinking Like a Designer 3

You Are a Game Designer. 4

Bartok: A Game Exercise. 4

The Definition of *Game* 10

Summary 17

2 Game Analysis Frameworks 19

Common Frameworks for Ludology 20

MDA: Mechanics, Dynamics, and Aesthetics. 20

Formal, Dramatic, and Dynamic Elements 24

The Elemental Tetrad 27

Summary 29

3 The Layered Tetrad. 31

The Inscribed Layer 32

The Dynamic Layer. 33

The Cultural Layer 34

The Responsibility of the Designer. 36

Summary 37

4 The Inscribed Layer. 39

Inscribed Mechanics 40

Inscribed Aesthetics 46

Inscribed Narrative 49

Inscribed Technology 58

Summary 59

5 The Dynamic Layer 61
The Role of the Player 62
Emergence 63
Dynamic Mechanics 64
Dynamic Aesthetics 70
Dynamic Narrative 75
Dynamic Technology 77
Summary 77

6 The Cultural Layer 79
Beyond Play 80
Cultural Mechanics 81
Cultural Aesthetics 82
Cultural Narrative 83
Cultural Technology 84
Authorized Transmedia Are Not in the Cultural Layer 85
The Cultural Impact of a Game 86
Summary 87

7 Acting Like a Designer 89
Iterative Design 90
Innovation 97
Brainstorming and Ideation 98
Changing Your Mind 101
Scoping!. 103
Summary 104

8 Design Goals 105
Design Goals: An Incomplete List 106
Designer-Centric Goals 106
Player-Centric Goals 109
Summary 124

9 Paper Prototyping 125
The Benefits of Paper Prototypes. 126
Paper Prototyping Tools 127
An Example of a Paper Prototype. 129
Best Uses for Paper Prototyping 138
Poor Uses for Paper Prototyping 139
Summary 140

10 Game Testing 141
Why Playtest? 142
Being a Great Playtester Yourself 142
The Circles of Playtesters 143
Methods of Playtesting 146
Other Important Types of Testing 152
Summary 153

11 Math and Game Balance 155
The Meaning of Game Balance 156
Installing Apache OpenOffice Calc. 156
Examining Dice Probability with Calc 157
The Math of Probability 165
Randomizer Technologies in Paper Games. 170
Weighted Distributions 173
Permutations 175
Positive and Negative Feedback 176
Using Calc to Balance Weapons 177
Summary 183

12 Puzzle Design 185
Puzzles Are Almost Everywhere. 186
Scott Kim on Puzzle Design 186
Puzzle Examples in Action Games 193
Summary 195

13 Guiding the Player . . . 197
Direct Guidance . . . 198
Indirect Guidance . . . 200
Teaching New Skills and Concepts . . . 207
Summary . . . 210

14 The Digital Game Industry . . . 211
About the Game Industry . . . 212
Game Education . . . 215
Getting into the Industry . . . 217
Don't Wait to Start Making Games! . . . 221
Summary . . . 222

Part II Digital Prototyping . . . 223

15 Thinking in Digital Systems . . . 225
Systems Thinking in Board Games . . . 226
An Exercise in Simple Instructions . . . 226
Game Analysis: *Apple Picker* . . . 229
Summary . . . 234

16 Introducing Our Development Environment: Unity . . . 235
Downloading Unity . . . 236
Introducing Our Development Environment . . . 237
Running Unity for the First Time . . . 241
Setting Up the Unity Window Layout . . . 246
Learning Your Way Around Unity . . . 251
Summary . . . 251

17 Introducing Our Language: C# . . . 253
Understanding the Features of C# . . . 254
Reading and Understanding C# Syntax . . . 259
Summary . . . 262

18 Hello World: Your First Program . . . 263
Creating a New Project . . . 264
Making a New C# Script . . . 266
Making Things More Interesting . . . 271
Summary . . . 279

19 Variables and Components . . . 281
Introducing Variables . . . 282
Strongly Typed Variables in C# . . . 282
Important C# Variable Types . . . 283
The Scope of Variables . . . 286
Naming Conventions . . . 286
Important Unity Variable Types . . . 288
Unity GameObjects and Components . . . 294
Summary . . . 297

20 Boolean Operations and Conditionals . . . 299
Booleans . . . 300
Comparison Operators . . . 303
Conditional Statements . . . 307
Summary . . . 313

21 Loops . . . 315
Types of Loops . . . 316
Set Up a Project . . . 316
`while` Loops . . . 316
`do...while` Loops . . . 319
`for` Loops . . . 320
`foreach` Loops . . . 322
Jump Statements within Loops . . . 322
Summary . . . 325

22 Lists and Arrays . . . 327
C# Collections . . . 328
List . . . 328
Array . . . 333
Multidimensional Arrays . . . 337
Jagged Arrays . . . 340
Whether to Use Array or List . . . 344
Summary . . . 344
Summary Exercise . . . 344
Moving Forward . . . 347

23 Functions and Parameters . . . 349
Set Up the Function Examples Project . . . 350
Definition of a Function . . . 350
Function Parameters and Arguments . . . 353
Returning Values . . . 354
Proper Function Names . . . 356
When Should You Use Functions? . . . 356
Function Overloading . . . 358
Optional Parameters . . . 359
The `params` Keyword . . . 359
Recursive Functions . . . 361
Summary . . . 362

24 Debugging . . . 363
Getting Started with Debugging . . . 364
Stepping Through Code with the Debugger . . . 369
Summary . . . 377

25 Classes . . . 379
Understanding Classes . . . 380
Class Inheritance . . . 387
Summary . . . 390

26 Object-Oriented Thinking 391

The Object-Oriented Metaphor 392

An Object-Oriented Boids Implementation 394

Summary 403

27 The Agile Mentality 405

The Manifesto for Agile Software Development 406

Scrum Methodology 407

Creating Your Own Burndown Charts 416

Summary 416

Part III Game Prototype Examples and Tutorials 417

28 Prototype 1: Apple Picker 419

The Purpose of a Digital Prototype 420

Preparing 421

Coding the Apple Picker Prototype 428

GUI and Game Management 440

Summary 448

Next Steps 448

29 Prototype 2: Mission Demolition 449

Getting Started: Prototype 2 450

Game Prototype Concept 450

Art Assets 451

Coding the Prototype 456

Summary 485

Next Steps 485

30 Prototype 3: Space SHMUP 487

Getting Started: Prototype 3 488

Setting the Scene 490

Making the Hero Ship 491

Adding Some Enemies 504

Spawning Enemies at Random 509
Setting Tags, Layers, and Physics 510
Making the Enemies Damage the Player 513
Restarting the Game 518
Shooting (Finally) 519
Adding Power-Ups 531
Resolving Race Conditions in Code 538
Making Enemies Drop Power-Ups 541
Programming Other Enemies 543
Adding Particle Effects and Background 556
Summary 558
Next Steps 558

31 Prototype 4: Prospector Solitaire 561
Getting Started: Prototype 4 562
Build Settings 562
Importing Images as Sprites 564
Constructing Cards from Sprites 566
The Prospector Game 583
Implementing Prospector in Code 585
Adding Scoring to Prospector 604
Summary 619
Next Steps 620

32 Prototype 5: Bartok 621
Getting Started: Prototype 5 622
Build Settings 623
Coding Bartok 624
Summary 655
Next Steps 655

33 Prototype 6: Word Game 657
Getting Started: Word Game Prototype. 658
About the Word Game 658
Parsing the Word List 660
Setting Up the Game 665
Laying Out the Screen. 671
Adding Interactivity 680
Adding Scoring 684
Adding Animation 687
Adding Color. 690
Summary 692
Next Steps 692

34 Prototype 7: QuickSnap 695
Getting Started: QuickSnap Prototype 696
Building the Scene 697
Coding the Game. 706
Summary 725
Next Steps 726

35 Prototype 8: Omega Mage. 727
Getting Started: Omega Mage Prototype 728
Building the Scene 730
The Mage Character 735
Mouse Interaction 737
Movement 741
The Inventory and Selecting Elements 747
Casting the Fire Ground Spell 754
Changing Rooms 764
Spawning Enemies. 768
Abstracting the Enemy Interface 782

Making an EnemyFactory 785
Summary 789
Next Steps 789
Thanks! 789

Part IV Appendices 791

A Standard Project Setup Procedure 793

B Useful Concepts 799
C# and Unity Coding Concepts 801
Math Concepts 822
Interpolation 831
Roleplaying Games 846
User Interface Concepts 848

C Online Reference 851
Tutorials 852
Unity Resources 852
Programming 853
Searching Tips 854
Finding Assets 854
Educational Software Discounts 855

Index 857

FOREWORD

I have a theory about game designers and teachers. I think that, beneath the possible differences of our outer appearances, we're secretly the same; that many of the skills possessed by a good game designer are the same skills held by a great teacher. Have you ever had a teacher who held a class spellbound with puzzles and stories? Who showed you simple demonstrations of skills that were easy for you to understand and copy, but were difficult for you to master? Who gradually, cleverly, helped you put together pieces of information in your mind, maybe without your even realizing it, until one day your teacher was able to step aside and watch you do something amazing, something that you never would have thought was possible.

We video game designers spend a lot of our time finding ways to teach people the skills they need to play our games, while keeping them entertained at the same time. We sometimes don't want people to be aware that we're teaching them, though—the best tutorial levels that video games open with are usually the ones that simply seem like the beginning of a thrilling adventure. I was lucky to work at the award-winning game studio Naughty Dog for eight amazing years, where I was the Lead or Co-Lead Game Designer on all three PlayStation 3 games in the *Uncharted* series. Everyone at the studio was very happy with the sequence that opened our game *Uncharted 2: Among Thieves*. It effectively taught each player all the basic moves they would need to play the game, while keeping them on the edge of their seat because of the gripping predicament our hero Nathan Drake found himself in, dangling over the edge of a cliff in a ruined train carriage.

Video game designers do this kind of thing over and over again as they create digital adventures for us to play. Working on a sequence of player experiences like those found in the *Uncharted* games, I have to stay very focused on what the player has recently learned. I have to present my audience with interesting situations that use their new skills and that are easy enough that they won't get frustrated, but challenging enough that their interest will be held. To do this with complete strangers, through the channels of communication that a game provides—the graphics of the environments and the characters and objects within them, the sounds that the game makes, and the interactivity of the game's controls—is tremendously challenging. At the same time, it is one of the most rewarding things I know how to do.

Now that I've become a professor, teaching game design in a university setting, I've discovered firsthand just how many of the skills I developed as a game designer are useful in my teaching. I'm also discovering that teaching is just as rewarding as game design. So it came to me as no surprise when I discovered that Jeremy Gibson, the author of this book, is equally talented as a game designer and a teacher, as you're about to find out.

I first met Jeremy around ten years ago, at the annual Game Developers Conference in Northern California, and we immediately hit it off. He already had a successful career as a game developer, and his enthusiasm for game design struck a chord with me. As you'll see when you begin to read this book, he loves to talk about game design as a craft, a design practice and an emerging art. Jeremy and I stayed in touch over the years, as he went back to graduate school at Carnegie Mellon University's excellent Entertainment Technology Center to study under visionaries like Doctor Randy Pausch and Jesse Schell. Eventually, I came to know Jeremy as a professor and a colleague in the Interactive Media & Games Division of the School of Cinematic Arts at the University of Southern California—part of USC Games, the program in which I now teach.

In fact, I got to know Jeremy better than ever during his time at USC—and I did it by becoming his student. In order to acquire the skills that I needed to develop experimental research games as part of USC's Game Innovation Lab, I took one of Jeremy's classes, and his teaching transformed me from a Unity n00b with some basic programming experience into an experienced C# programmer with a strong set of skills in Unity, one of the world's most powerful, usable, adaptable game engines. Every single one of Jeremy's classes was not only packed with information about Unity and C#, but was also peppered with inspirational words of wisdom about game design and practical pieces of advice related to game development—everything from his thoughts about good "lerping," to great tips for time management and task prioritization, to the ways that game designers can use spreadsheets to make their games better. I graduated from Jeremy's class wishing that I could take it again, knowing that there was a huge amount more that I could learn from him.

So I was very happy when I heard that Jeremy was writing a book—and I became even happier when I read the volume that you now hold in your hands. The good news for both you and me is that Jeremy has loaded this book with everything that I wanted more of. I learned a lot in the game industry about best practices in game design, production, and development, and I'm happy to tell you that in this book, Jeremy does a wonderful job of summarizing those ways of making games that I've found work best. Within these pages, you'll find step-by-step tutorials and code examples that will make you a better game designer and developer in innumerable ways. While the exercises in this book might get complex—game design is among the most difficult things I know how to do—Jeremy won't ask you to do anything complicated without guiding you through it in clear, easy-to-follow language.

You'll also find history and theory in this book. Jeremy has been thinking deeply about game design for a long time and is very well-read on the subject. In the first part of this volume, you'll find an extraordinarily wide and deep survey of the state-of-the-art in game design theory, along with Jeremy's unique and strongly developed synthesis of the very best ideas he's encountered on his game design travels. Jeremy supports his discussion with interesting historical anecdotes and fascinating glimpses of the long traditions of play in human culture, all of which help to frame his conversation in valuable and progressive ways. He continually pushes you to question your assumptions about games, and to think beyond the console, the controller, the screen and the speakers, in ways that might just spur a whole new generation of game innovators.

Jeremy Gibson has moved on from USC, and now teaches at the University of Michigan Ann Arbor, and I'm very happy for the generations of U-M students that he'll lead to new understandings of the craft of game design in the coming years. This spring, when Jeremy walked into the restaurant at the annual GDC alumni dinner hosted by the USC Games program, the room full of our current and former students came alive with whoops and cheers and moments later broke out into applause. That tells you a lot about what Jeremy Gibson is like as a teacher. You're lucky that, thanks to this book, he can now be your teacher too.

The world of game design and development is changing at a rapid rate. You can be part of this wonderful world—a world unlike any other I know, and which I love with all my heart. You can use the skills you learn through reading this book to develop new prototypes for new kinds of games, and in doing so you might eventually create whole new genres of games, in expressive new styles, which appeal to new markets. Some of tomorrow's stars of game design are currently learning to design and program, in homes and schools all around the world. If you make good use of this book, by following the advice and doing the exercises you find in here, it might just help your chances of creating a modern game design classic.

Good luck, and have fun!

Richard Lemarchand
Associate Professor, USC Games

PREFACE

Welcome to *Introduction to Game Design, Prototyping, and Development*. This book is based on my work over many years as both a professional game designer and a professor of game design at several universities, including the Interactive Media and Games Division at the University of Southern California and the Department of Electrical Engineering and Computer Science at the University of Michigan Ann Arbor.

This preface introduces you to the purpose, scope, and approach of this book.

The Purpose of This Book

My goal in this book is quite simple: I want to give you all the tools and knowledge you need to get started down the path to being a successful game designer and prototyper. This book is the distillation of as much knowledge as I can cram into it to help you toward that goal. Unlike most books out there, this book combines the disciplines of game design and digital development (that is, computer programming) and wraps them both in the essential practice of iterative prototyping. The emergence of advanced, yet approachable, game development engines such as Unity has made it easier than ever before to create playable prototypes that express your game design concepts to others, and the ability to do so will make you a much more skilled (and employable) game designer.

The book is divided into four parts:

Part I: Game Design and Paper Prototyping

The first part of the book starts by exploring various theories of game design and the analytical frameworks for game design that have been proposed by several earlier books. This section then describes the *Layered Tetrad* as a way of combining and expanding on many of the best features of these earlier theories. The Layered Tetrad is explored in depth as it relates to various decisions that you must make as a designer of interactive experiences. This part also covers information about the interesting challenges of different game design disciplines; describes the process of paper prototyping, testing, and iteration; and gives you concrete information to help you become a better designer.

Part II: Digital Prototyping

The second part teaches you how to program in the C# language (pronounced "see-sharp"). This part draws upon my many years of experience as a professor teaching nontechnical students how to express their game design ideas through digital code. If you have no prior knowledge or experience with programming or development, this part is designed for you. However, even if you do have some programming experience, you might want to take a look at this part to learn a few new tricks or get a refresher on some approaches.

Part III: Game Prototype Examples and Tutorials

The third part of the book encompasses several different tutorials, each of which guides you through the development of a prototype for a specific style of game. The purpose of this part is twofold: It reveals some best practices for rapid game prototyping by showing you how I personally approach prototypes for various kinds of games, and it provides you with several basic foundations on which to build your own games in the future. Most other books on the market that attempt to teach Unity (our game development environment) do so by taking the reader through a single, monolithic tutorial that is hundreds of pages long. In contrast, this book takes you through several much smaller tutorials. The final products of these tutorials are necessarily less robust than those found in some other books, but it is my belief that the variety of projects in this book will better prepare you for creating your own projects in the future.

Part IV: Appendices

This book has several important appendices that merit mention here. Rather than repeat information throughout the book or require you to go hunting through various chapters for it, any piece of information that is referenced several times in the book or that I think you would be likely to want to reference later (after you've finished reading the book once) is placed in the appendices. Appendix A is just a quick step-by-step introduction to the initial creation process for a game project in Unity. The second and longest appendix is Appendix B, "Useful Concepts." Though it has a rather lackluster name, this is the portion of the book that I believe you will return to most often in the years following your initial read through the book. "Useful Concepts" is a collection of several go-to technologies and strategies that I use constantly in my personal game prototyping process, and I think you'll find a great deal of it to be very useful. The third and final appendix is a list of good online references where you can find answers to questions not covered in this book. It is often difficult to know the right places to look for help online; this appendix lists those that I personally turn to most often.

There Are Other Books Out There

As a designer or creator of any kind, I think that it's absolutely essential to acknowledge those on whose shoulders you stand. There have been many books written on games and game design, and the few that I list here are those that have had the most profound effect on either my process or my thinking about game design. You will see these books referenced many times throughout this text, and I encourage you to read as many of them as possible.

Game Design Workshop by Tracy Fullerton

Initially penned by Tracy Fullerton, Chris Swain, and Steven S. Hoffman, *Game Design Workshop* is now in its third edition. More than any other text, this is the book that I turn to for advice on game design. This book was initially based on the Game Design Workshop class that Tracy and Chris taught at the University of Southern California, a class that formed the foundation for the entire games program at USC (and a class that I myself taught there from 2009–2013). The USC Interactive Media and Games graduate program has been named the number one school for game design in North America by Princeton Review every year that they have been ranking game programs, and the Game Design Workshop book and class were the foundation for that success.

Unlike many other books that speak volumes of theory about games, Tracy's book maintains a laser focus on information that helps budding designers improve their craft. I taught from this book for many years (even before I started working at USC), and I believe that if you actually attempt all the exercises listed in the book, you can't help but have a pretty good paper game at the end.

> Fullerton, Tracy, *Game Design Workshop: A Playcentric Approach to Creating Innovative Games,* 3rd ed. (Boca Raton, FL: CRC Press, 2014)

The Art of Game Design by Jesse Schell

Jesse Schell was one of my professors at Carnegie Mellon University and is a fantastic game designer with a background in theme park design gained from years working for Walt Disney Imagineering. Jesse's book is a favorite of many working designers because it approaches game design as a discipline to be examined through 100 different lenses that are revealed throughout the book. Jesse's book is a very entertaining read and broaches several topics not covered in this book.

> Jesse Schell, *The Art of Game Design: A Book of Lenses* (Boca Raton, FL: CRC Press, 2008)

The Grasshopper by Bernard Suits

While not actually a book on game design at all, *The Grasshopper* is an excellent exploration of the definition of the word *game*. Presented in a style reminiscent of the Socratic method, the book presents its definition of game very early in the text as the Grasshopper (from Aesop's fable *The Ant and the Grasshopper*) gives his definition on his deathbed, and his disciples spend the remainder of the book attempting to critique and understand this definition. This book also explores the question of the place of games and play in society.

> Bernard Suits, *The Grasshopper: Games, Life and Utopia* (Peterborough, Ontario: Broadview Press, 2005)

Game Design Theory by Keith Burgun

In this book, Burgun explores what he believes are faults in the current state of game design and development and proposes a much narrower definition of *game* than does Suits. Burgun's goal in writing this text was to be provocative and to push the discussion of game design theory forward. While largely negative in tone, Burgun's text raises a number of interesting points, and reacting to it helped me to refine my personal understanding of game design.

> Keith Burgun, *Game Design Theory: A New Philosophy for Understanding Games* (Boca Raton, FL: A K Peters/CRC Press, 2013)

Imaginary Games by Chris Bateman

Bateman uses this book to argue that games are a legitimate medium for scholarly study. He pulls from several scholarly, practical, and philosophical sources; and his discussions of books like *Homo Ludens* by Johan Huizinga, *Man, Play, and Games* by Roger Caillois, and the paper "The Game Game" by Mary Midgley are both smart and accessible.

> Chris Bateman, *Imaginary Games* (Washington, USA: Zero Books, 2011)

Level Up! by Scott Rogers

Rogers distills his knowledge from many years in the trenches of game development into a book that is fun, approachable, and very practical. When he and I co-taught a level design class, this was the textbook that we used. Rogers is also a comic book artist, and his book is full of humorous and helpful illustrations that drive home the concepts of level, character, narrative, and many other aspects of design.

> Scott Rogers, *Level up!: The Guide to Great Video Game Design* (Chichester, UK: Wiley, 2010)

Our Digital Prototyping Environment: Unity and C#

All the digital game examples in this book are based on the Unity Game Engine and the C# programming language. I have taught students to develop digital games and interactive experiences for more than a decade, and in my experience, Unity is—by far—the best environment that I have found for learning to develop games. I have also found that C# is the best initial language for game prototypers to learn. Some other tools out there are easier to learn and require no real programming (Game Maker and Game Salad are two examples), but Unity allows you much more flexibility and performance in a package that is basically free (the free version of Unity includes nearly all the capabilities of the paid version, and it is the version used throughout this book). If you want to actually learn to program games, Unity is the engine you want to use.

Similarly, some programming languages are a little more approachable than C#. In the past, I have taught my students both ActionScript and JavaScript. However, C# is the one language I have used that continually impresses me with its flexibility and feature set. Learning C# means learning not only programming but also good programming practices. Languages such as JavaScript allow a lot of sloppy behaviors that I have found actually lead to slower development. C# keeps you honest (via things like strongly typed variables), and that honesty will not only make you a better programmer but will also result in your being able to code more quickly (for example, strong variable typing enables very robust code hinting and auto-completion, which makes coding faster and more accurate).

Who This Book Is For

There are many books about game design, and there are many books about programming. This book seeks to fill the gap between the two. As game development technologies like Unity become more ubiquitous, it is increasingly important that game designers have the ability to sketch their design ideas not only on paper but also through working digital prototypes. This book exists to help you learn to do just that:

- **If you're interested in game design but have never programmed,** this book is perfect for you. Part I introduces you to several practical theories of game design and presents you with the practices that can help you develop and refine your design ideas. Part II teaches you how to program from nothing to understanding object-oriented class hierarchies. Since I became a college professor, the majority of my classes have focused on teaching nonprogrammers how to program games. I have distilled all of my experience doing so into Part II of this book. Part III takes you through the process of developing eight different game prototypes across several different game genres. Each demonstrates fast methods to get from concept to working digital prototype. Lastly, the appendices will explain specific game development and programming concepts in-depth and guide you to resources to learn more once you've finished the book. This in-depth content was moved to Appendix B, "Useful Concepts," so that you could continue to use that section of the book as a reference in the years to come.

- **If you're a programmer who is interested in game design,** Parts I and III of this book will be of most interest to you. Part I introduces you to several practical theories for game design and presents you with the practices that can help you develop and refine your design ideas. You can skim Part II, which introduces C# and how it is used in Unity. If you are familiar with other programming languages, C# looks like C++ but has the advanced features of Java. Part III takes you through the process of developing eight different game prototypes across several different game genres. Game development in Unity is very different from what you may be used to from other game engines. Many elements of development are taken care of outside of the code. Each prototype will demonstrate the style of development that works best in Unity to get from concept to working digital prototype quickly. You will also want to look carefully at Appendix B, which is full of detailed information about various development concepts and is arranged as a reference that you can return to later.

Conventions

This book maintains several writing conventions to help make the text more easily understandable.

Any place that specific button names, menu commands, or other multi-word nouns are introduced in the text, they will be listed in *italics*. This includes terms like the *Main Camera* Game Object. An example menu command is *Edit* > *Project Settings* > *Physics*, which would instruct you to select the *Edit* menu from the menu bar, choose the *Project Settings* sub-menu, and then select *Physics*.

Book Elements

The book includes several different types of asides that feature useful or important information that does not fit in the flow of the regular body text.

note

Callouts in this format are for information that is useful but not critical. Information in notes will often be an interesting aside to the main text that provides a little bit more info about the topic.

tip

This element provides additional information that is related to the book content and can help you as you explore the concepts in the book.

warning

BE CAREFUL Warnings cover information about things that you need to be aware of to avoid mistakes or other pitfalls.

SIDEBAR

The sidebar is for discussions of longer topics that are important to the text but should be considered separately from it.

Code

Several conventions apply to the code samples in this book. When specific elements from the code listing are placed in regular paragraph text, they appear in a `monospaced` font. The variable `variableOnNewLine` from the following code listing is an example of this.

Code listings also utilize a monospaced font and appear as follows:

```
public class SampleClass {
    public GameObject        variableOnExistingLine;                      // 1
    public GameObject        variableOnNewLine;                           // 2
}
```

1 Code listings are often annotated; in this case, additional information about the line marked with `// 1` would appear in this first annotation.

2 Some code listings will be expansions on code that you've already written or that already exists in the C# script file for another reason. In this case, the old lines will be at `normal weight`, and the new lines will be at **`bold weight`**.

Note that occasionally lines of code in the chapters are too long to fit on the printed page. Where that occurs, a code-continuation arrow (➥) has been used to mark the continuation. For example:

```
21          jaggedList.Add ( new List<string>( new string[] {"complex",
                  ➥"initialization"} ) );
```

Most of the code listings in the first two parts of the book will include line numbers (as seen in the preceding listing). You do not need to type the line numbers when entering the code into MonoDevelop (it will automatically number all lines). In the final part of the book, there are no line numbers due to the size of the code listings.

Book Website

The website for this book includes all of the files referenced in the chapters, lecturer notes, and finished versions of each tutorial prototype. It is available at **http://book.prototools.net**.

ACKNOWLEDGMENTS

A tremendous number of people deserve to be thanked here. First and foremost, I want to thank my wife, Melanie, whose help and feedback on my chapters throughout the entire process improved the book tremendously. I also want to thank my family for their many years of support, with special thanks to my father for teaching me how to program as a child.

As a new author, there were several people at Pearson who provided support to me and shepherded me through this process. Chief among them were Chris Zahn, Laura Lewin, Olivia Basegio, Elaine Wiley, and Keith Cline who each demonstrated laudable patience in working with me. I also had the support of some fantastic technical reviewers: Marc Destefano, Charles Duba, and Margaret Moser. Their keen eyes and minds found many places in the original text that could be clarified or improved.

I would also like to thank all the educators who have taught me and worked as my colleagues. Special thanks go to Dr. Randy Pausch and Jesse Schell. Though I had worked as a professor and game designer before meeting them, they each had a profound effect on my understanding of design and education. I also owe tremendous thanks to Tracy Fullerton, Mark Bolas, and Scott Fisher, who were friends and mentors to me in the years I taught at the University of Southern California's Interactive Media and Games Division. There were also many other brilliant faculty and friends at USC who helped me to flesh out the ideas in this book, including Adam Liszkiewicz, William Huber, Richard Lemarchand, Scott Rogers, Vincent Diamante, Sam Roberts, and Logan Ver Hoef.

Many of my friends in the industry have also helped me by giving me suggestions for the book and feedback on the ideas presented therein. These included Michael Sellers, Nicholas Fortugno, Jenova Chen, Zac Pavlov, Joseph Stevens, and many others.

Thanks as well to all the fantastic students whom I have taught over the past decade. It is you who inspired me to want to write this book and who convinced me that there was something important and different about the way that I was teaching game development. Every day that I teach, I find myself inspired and invigorated by your creativity, intelligence, and passion.

Finally, I would like to thank you. Thank you for purchasing this book and for your interest in developing games. I hope that this book helps you get started, and I would love to see what you make with the knowledge you gain here.

ABOUT THE AUTHOR

Jeremy Gibson is a lecturer teaching computer game design for the Electrical Engineering and Computer Science department at the University of Michigan Ann Arbor and is the founder of ExNinja Interactive, LLC. From 2009 to 2013, he was an Assistant Professor teaching game design and protyping for the Interactive Media and Games Division of the University of Southern California's School of Cinematic Arts, which was the number one game design school in North America throughout his tenure there. Jeremy serves the IndieCade independent game festival as the Chair for Education and Advancement, where he is responsible for the IndieXchange and GameU conference tracks, and he has spoken at the Game Developers Conference every year since 2009.

Jeremy earned a Master of Entertainment Technology degree from Carnegie Mellon University's Entertainment Technology Center in 2007 and a Bachelor of Science degree in Radio, Television, and Film from the University of Texas at Austin in 1999. Jeremy has worked as a programmer and prototyper for companies such as Human Code and frog design, has taught classes for Great Northern Way Campus (in Vancouver, BC), Texas State University, the Art Institute of Pittsburgh, Austin Community College, and the University of Texas at Austin, and has worked for Walt Disney Imagineering, Maxis, and Electronic Arts/Pogo.com, among others. While in graduate school, his team created the game *Skyrates*, which won the Silver Gleemax Award at the 2008 Independent Games Festival. Jeremy also apparently has the distinction of being the first person to ever teach game design in Costa Rica.

This page intentionally left blank

CHAPTER 16

INTRODUCING OUR DEVELOPMENT ENVIRONMENT: UNITY

This is the start of your programming adventure.

In this chapter, you download Unity, the game development environment that you will use throughout the rest of this book. We talk about why Unity is a fantastic game development tool for any budding game designer or developer and why we've chosen C# as the language for you to learn.

You also take a look at the sample project that ships with Unity, learn about the various window panes in the Unity interface, and move these panes into a logical arrangement that will match the examples you see in the rest of the book.

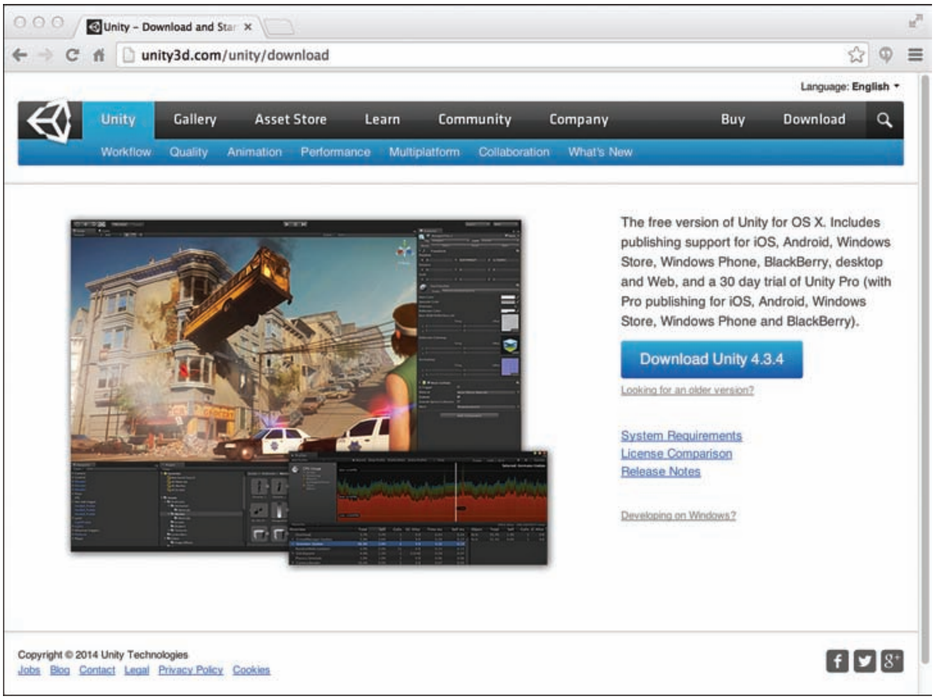
Downloading Unity

First things first, let's start downloading Unity. The Unity installer is over 1 GB in size, so depending on your Internet speed, this could take anywhere from a few minutes to a couple of hours. After you've gotten this process started, we can move on to talking about Unity.

As of this writing, the latest major version of Unity is Unity 4. Because Unity is under constant development, the current minor version should be something like 4.x.y, with the *x* and *y* being sub-version numbers. Regardless of version, Unity is always available for free from Unity's official website:

http://unity3d.com/unity/download

This should take you to a page that provides the latest download link for your system (see Figure 16.1). Unity is available for both PC and OS X, and it is nearly identical on both platforms.



The free version of Unity for OS X. Includes publishing support for iOS, Android, Windows Store, Windows Phone, BlackBerry, desktop and Web, and a 30 day trial of Unity Pro (with Pro publishing for iOS, Android, Windows Store, Windows Phone and BlackBerry).

Copyright © 2014 Unity Technologies

Figure 16.1 The web page to download Unity

tip

Unity is free, but you will still need to acquire a license, and this requires that you have an available Internet connection the first time that you run the application.

Introducing Our Development Environment

Before you can begin prototyping in earnest, you first need to become familiar with Unity, our chosen development environment. Unity itself can really be thought of as a synthesis program; while you will be bringing all the elements of your game prototypes together in Unity, the actual production of the assets will largely be done in other programs. You will program in MonoDevelop; model and texture in a 3D modeling program like Maya, Autodesk 3ds Max, or Blender; edit images in a photo editor such as Photoshop or GIMP; and edit sound in an audio program such as Pro Tools or Audacity. Because a large section of this book is about programming and learning to program in C# (pronounced "see-sharp"), you'll be spending most of the time with tutorials using MonoDevelop, but it's still critically important to understand how to use Unity and how to effectively set up your Unity environment.

Why Choose Unity?

There are many game development engines out there, but we've chosen to focus on Unity for several reasons:

- **Unity is free:** With the free version of Unity, you can create and sell games that run on OS X, PC, the Web, Linux, iOS, Android, BlackBerry, Windows Phone, Windows Store, and more. While the Pro version of Unity includes a few additional useful features, for a game designer just learning to prototype, the free version is really all that you need. The Pro version normally costs $1,500 (or $75/month), but if you're a student, a one-year license for Unity Pro is about ten times less!

tip

STUDENT PRICING If you are a student, you can purchase a 1-year educational license for Unity Pro at a tremendous discount (about $150 instead of $1,500). This license does prevent you from being able to sell your game directly to players, but it lets you use the full power of Unity Pro to develop your game and make excellent portfolio pieces. After you're done developing, if you know you've got a hit on your hands, you can purchase the commercial version of Pro before attempting to sell your game. Unity has also recently added Pro student licenses that do allow you to sell your games, but those have a higher cost.

To find the latest student pricing for Unity, I recommend searching the Web for "unity educational student pricing." That will make sure that you're looking at the latest.

- **Write once, deploy anywhere:** The free version of Unity can build applications for OS X, PC, the Web, Linux, iOS, Android, BlackBerry, Windows Phone, Windows Store, and more, all from the same code and files. This kind of flexibility is at the core of Unity; in fact, it's what the product and company are named for. There are also paid extensions to Unity Pro that professionals can use to create games for the PlayStation 3, Xbox 360, and several other game consoles.

- **Great support:** In addition to excellent documentation, Unity has an incredibly active and supportive development community. Hundreds of thousands of developers are using Unity, and many of them contribute to the discussions on Unity forums across the web.
- **It's awesome!:** My students and I have joked that Unity has a "make awesome" button. Although this is not strictly true, there are several phenomenal features built in to Unity that will make your games both play and look better by simply checking an option box. Unity engineers have already handled a lot of the difficult game programming tasks for you. Collision detection, physics simulation, pathfinding, particle systems, draw call batching, shaders, the game loop, and many other tough coding issues are all included. All you need to do is make a game that takes advantage of them!

Why Choose C#?

Within Unity, you have the choice to use any of three programming languages: UnityScript, C#, or Boo. Very, very few people actually use Boo, so you're really left with two choices.

UnityScript, A Version of JavaScript

JavaScript is often seen as a language for beginners; it's easy to learn, the syntax is forgiving and flexible, and it's also used for scripting web pages. JavaScript was initially developed in the mid-1990s by Netscape as a "lite" version of the Java programming language. It was used as a scripting language for web pages, though early on that often meant that various JavaScript functions worked fine in one web browser but didn't work at all in another. The syntax of Java Script was the basis for HTML5 and is very similar to Adobe Flash's ActionScript 3. Despite all of this, it is actually JavaScript's flexibility and forgiving nature that make it an inferior language for this book. As one example, JavaScript uses weak typing, which means that if we were to create a variable (or container) named `bob`, we could put anything we wanted into that variable: a number, a word, an entire novel, or even the main character of our game. Because the variable `bob` doesn't have a variable type, Unity never really knows what kind of thing `bob` is, and that could change at any time. These flexibilities in JavaScript make scripting more tedious and prevent programmers from taking advantage of some of the most powerful and interesting features of modern languages.

C#

C# was developed in 2000 as Microsoft's response to Java. They took a lot of the modern coding features of Java and put them into a syntax that was much more familiar to and comfortable for traditional C++ developers. This means that C# has all the capabilities of a modern language. For you experienced programmers, these features include function virtualization and delegates, dynamic binding, operator overloading, lambda expressions, and the powerful Language INtegrated Query (LINQ) query library among many others. For those of you new to programming, all you really need to know is that working in C# from the beginning will make you a better programmer and prototyper in the long run. In my prototyping class at the

University of Southern California, I taught using both UnityScript and C# in different semesters, and I found that students who were taught C# consistently produced better game prototypes, exhibited stronger coding practices, and felt more confident about their programming abilities than their peers who had been taught UnityScript in prior semesters of the class.

RUNTIME SPEED OF EACH LANGUAGE

If you've had some experience programming, you might assume that C# code in Unity would execute faster than code written in JavaScript or Boo. This assumption would come from the understanding that C# code is usually compiled while JavaScript and Boo are interpreted (meaning that compiled code is turned into a computer's machine language by a compiler as part of the coding process, while interpreted code is translated on-the-fly as the player is playing the game, making interpreted code generally slower). However, in Unity, every time you save a file of C#, UnityScript, or Boo code, Unity imports it, converts any of the three languages to the same Common Intermediate Language (CIL), and then compiles that CIL into machine language. So, regardless of the language you use, your Unity game prototypes will execute at the same speed.

On the Daunting Nature of Learning a Language

There's no way around it, learning a new language is tough. I'm sure that's one of the reasons that you bought this book rather than just trying to tackle things on your own. Just like Spanish, Korean, Mandarin, French, or any other human language, there are going to be things in C# that don't make any sense at first, and there are places that I'm going to tell you to write something that you don't immediately understand. There will also probably be a point where you are just starting to understand some things about the language but feel utterly confused by the language as a whole (which is the exact same feeling you'd have if you took one semester of Spanish class and then tried to watch soap operas on Telemundo). This feeling comes for almost all of my students about halfway through the semester, and by the end of the semester, every one of them feels much more confident and comfortable with both C# and game prototyping.

Rest assured, this book is here for you, and if you read it in its entirety, you will emerge with not only a working understanding of C# but also several simple game prototypes that you can use as foundations on which to build your own projects. The approach that I take in this book comes from many semesters of experience teaching "nonprogrammers" how to find the hidden coder within themselves and, more broadly, how to convert their game ideas into working prototypes. As you'll see throughout this book, that approach is composed of three steps:

1. **Concept introduction:** Before asking you to code anything for each project, I'll tell you what we're doing and why. This general concept of what you're working toward in each tutorial will give you a framework on which to hang the various coding elements that are introduced in the chapter.

2. **Guided tutorial:** You'll be guided step by step through a tutorial that will demonstrate these concepts in the form of a playable game. Unlike some other approaches, we will be compiling and testing the game throughout the process so that you can identify and repair bugs (problems in the code) as you go, rather than trying to fix all of them at the end. Additionally, I'll even guide you to create some bugs so that you can see the errors they cause and become familiar with them; this will make it easier when you encounter your own bugs later.
3. **Lather, rinse, repeat:** In many tutorials, you'll be asked to repeat something. For instance, in a top-down shooter game like *Galaga*, the tutorial would guide you through the process of making one single enemy type, and then it would ask you to create three others on your own. Don't skip this part! This repetition will really drive the concept home, and it will help your understanding solidify later.

pro tip

90% OF BUGS ARE JUST TYPOS I've spent so much time helping students fix bugs that now I can very quickly spot a typo in code. The most common include the following:

- **Misspellings:** If you type even one letter wrong, the computer won't have any idea what you're talking about.
- **Capitalization:** To your C# compiler, `A` and `a` are two completely different letters, so `variable`, `Variable`, and `variAble` are all completely different words.
- **Missing semicolons:** Just like almost every sentence in English should end in a period, nearly every statement in C# should end in a semicolon (;). If you leave the semicolon out, it will often cause an error on the next line. FYI: A semicolon is used because the period was needed for decimal numbers and what's called *dot syntax* in variable names and subnames (e.g., varName.subVarName.subSubVarName).

Earlier, I mentioned that most of my students feel confused and daunted by C# at about the midway point of the semester, and it's at exactly that time that I assign them the Classic Games Project. They are asked to faithfully recreate the mechanics and game feel of a classic game over the course of four weeks. Some great examples have included *Super Mario Bros.*, *Metroid*, *Castlevania*, *Pokemon*, and even the original *Legend of Zelda*. By being forced to work things out on their own, to schedule their own time, and to dig deeply into the inner workings of these seemingly simple games, the students come to realize that they understand much more C# than they thought, and that is the time that everything really falls into place. The key component here is that the thought process changes from "I'm following this tutorial" to "I want to do this...now how do I make it happen?" At the end of this book, you will be prepared to tackle your own game projects (or your own Classic Game Project, if you want). The tutorials in this book can be a fantastic starting point on which to build your own games.

Running Unity for the First Time

Hopefully reading all of that will have given Unity enough time to download in the background. Congratulations! You're about to embark on a challenging but rewarding journey.

Installing Unity

Depending on your personal system settings, the Unity installer should have placed itself in a *Downloads* folder somewhere on your hard drive. I'm sure you've done this kind of thing several times before, so find the file, run the installer with all default options, and let's get to work. This is a big install, so it could take a while. In the final bit of the installation, it may look like it has frozen; but don't worry, just give it some time to complete.

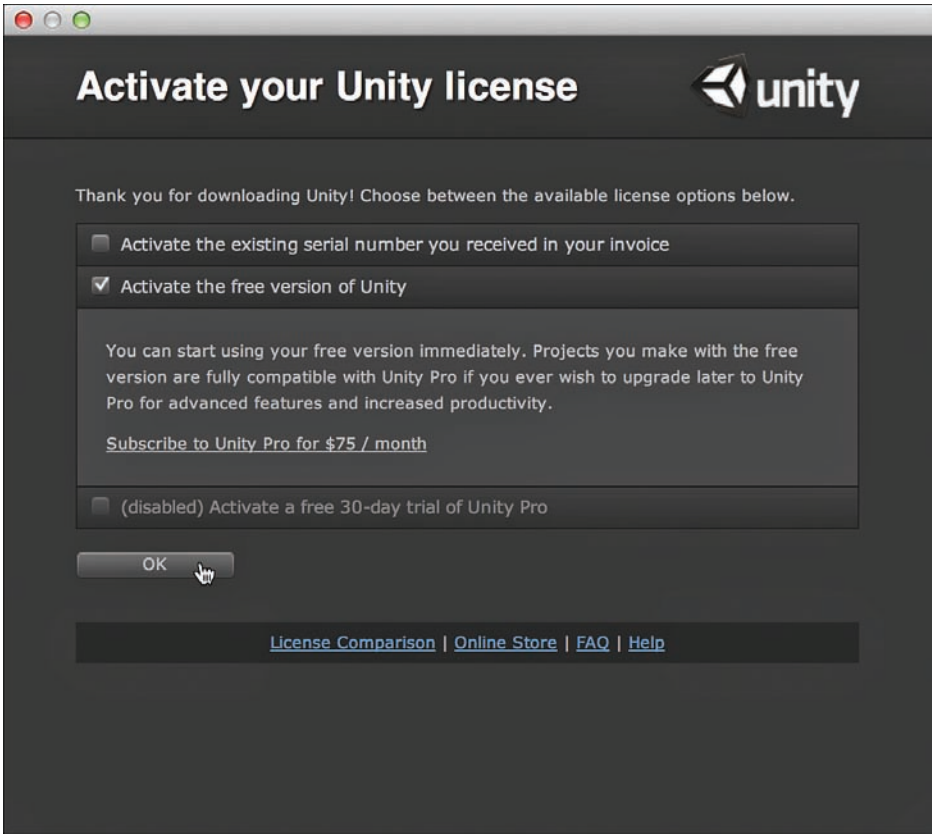
Your First Launch: Licensing

The first time you run Unity, it will open a built-in web page that will ask you to create a license and register (see Figure 16.2), but it's really quite painless, and it shouldn't take much time at all. You will need to choose between the free license and a 30-day trial of Unity Pro. At this time, I recommend activating the free version of Unity, especially if you plan to work through this book slowly. The Pro version will be nice to have for the prototype you'll make in Chapter 34, "QuickSnap," so I recommend waiting until then to start the 30-day trial of Unity Pro. However, choosing the 30-day Unity Pro trial now would allow you to see the beautiful reflections and depth-of-field shaders in Figure 16.4.

You can choose to activate the 30-day trial any time, although you can only activate it once, and once the trial is over, you will be reverted to the free version. If you choose the free version now, you can always go back and upgrade to the Pro trial by selecting *Unity > Manage License* from the menu bar on OS X (on PC, choose *Help > Manage License*).

Once you click *OK*, you are prompted to create a Unity account. They'll send you an email to confirm this (so you need to give them a valid email address). Then, you may be asked to take part in a survey, which you can choose to skip if you want (through a link at the bottom of the survey).

After this, Unity will automatically open the *AngryBots* demo project. This is a large project, so it may take several seconds to load. It may appear that Unity has frozen or is unresponsive, but if you wait a bit, everything will show up.



Activate your Unity license

Thank you for downloading Unity! Choose between the available license options below.

- [ ] Activate the existing serial number you received in your invoice
- [x] Activate the free version of Unity

You can start using your free version immediately. Projects you make with the free version are fully compatible with Unity Pro if you ever wish to upgrade later to Unity Pro for advanced features and increased productivity.

Subscribe to Unity Pro for $75 / month

- [ ] (disabled) Activate a free 30-day trial of Unity Pro

Figure 16.2 Unity licensing window

Example Project: *AngryBots*

When you first launch Unity, it will open a demo project and will show you a *Welcome to Unity* window that pops up over the main Unity window. For now, close the *Welcome to Unity* window, but feel free to explore the introductory videos and other links there later if you want more of an introduction to Unity than is provided in this chapter.

Unless you tell it not to (by holding the Option key at launch), Unity will open an existing project every time you launch it. The default project for this is *AngryBots* (see Figure 16.3), a game created internally by the Unity team to show off the capabilities of the engine. If for some reason the default scene doesn't open automatically, you will need to double-click the *AngryBots* Scene Asset to open it; it should be the first one listed in the Project window pane in the bottom half of the screen. You'll see Project and several other window panes on screen that I'll explain later, but for now, just click the large *Play* button at the top of the Unity window (the triangle pointing to the right in the top, center of the Unity window) and enjoy playing this game for a while. You can read about the controls for this game in the nearby tip.

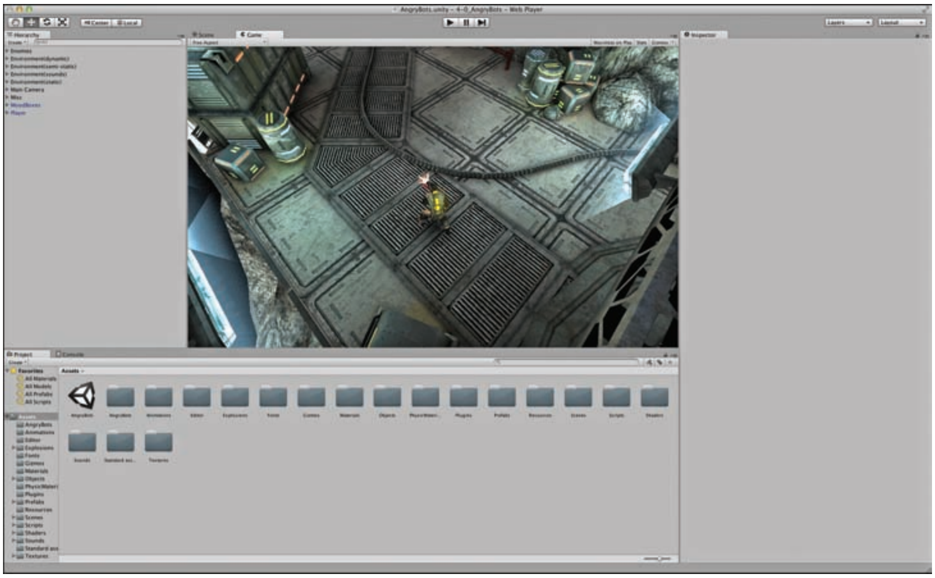


Figure 16.3 The Unity window when it opens for the first time

tip

ANGRYBOTS CONTROLS

- Movement is controlled by the W, A, S, and D or arrow keys.
- The gun will always aim at your mouse pointer.
- Hold down the left mouse button to fire.

You must stand very close to any circular door for a couple of seconds for it to open.

There are several computers that you need to stand in front of in order to unlock them (turn the color of electric wires coming out of them from red to green).

Here are some things to notice while you're playing:

- **Shaders:** AngryBots is rife with *shaders* (see Figure 16.4), code written specifically for the graphics card with the sole purpose of making the game look amazing. Special ones to check out include the following:
  A. The depth-of-field image effect that makes some parts of the scene in-focus while others are out-of-focus (see letter *A* in Figure 16.4). This will only appear in Unity Pro.

B. The reflections on the floors (especially of the laser sight) (see letter *B* in Figure 16.4). This will only appear in Unity Pro.

C. The animated water droplets on the floor when outside (see letter *C* in Figure 16.4). This appears regardless of whether you are using Unity Pro or free.

As explained earlier, if you chose to activate the free license rather than the 30-day Unity Pro trial, you will not see the most advanced shaders. This is one of the few differences between the free and Pro versions of Unity.

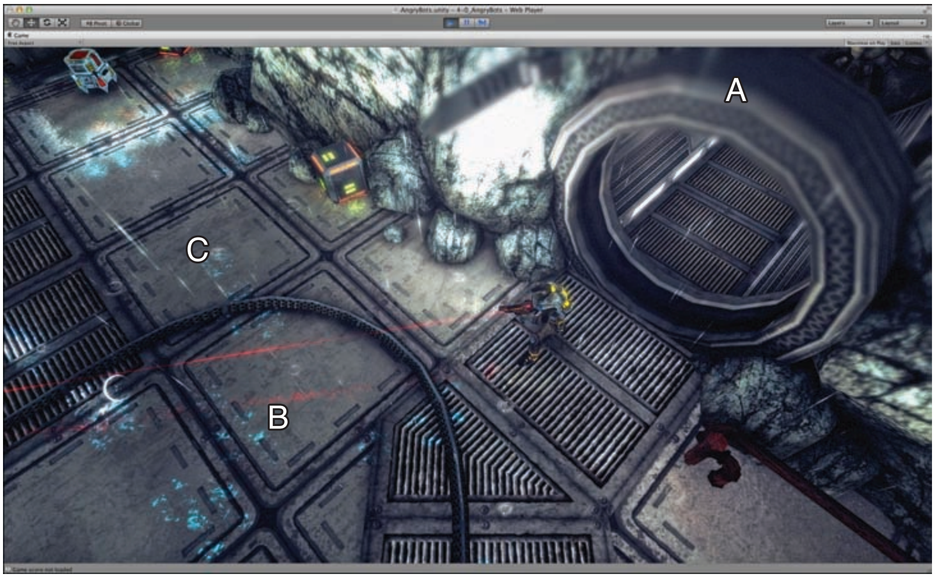


Figure 16.4 Screen showing the effects of various shaders

- **Character rigging and animation:** Unity makes use of animation blending to enable the player character to walk in one direction while looking and shooting in another.
- **AI pathing:** Enemies will move around objects in a room to find and attack the player.

Feel free to explore the whole space and see what elements of AngryBots you might want to use in your own project. Go ahead, I'll wait.

...

...

So, what did you think? Did you blow up the base, or did you escape the exploding station? Did you find the white museum? The controls of this game are a little unusual, but regardless, it's a good showcase for how beautiful Unity can look.

Now, let's do something really cool.

Compile and Deploy AngryBots for the Web

Once you've clicked the blue Stop button at the top of the Unity window (the square next to the Play button), choose *File > Build Settings* from the menu bar (meaning that you should choose the item *Build Settings* from the File menu, as shown in Figure 16.5).

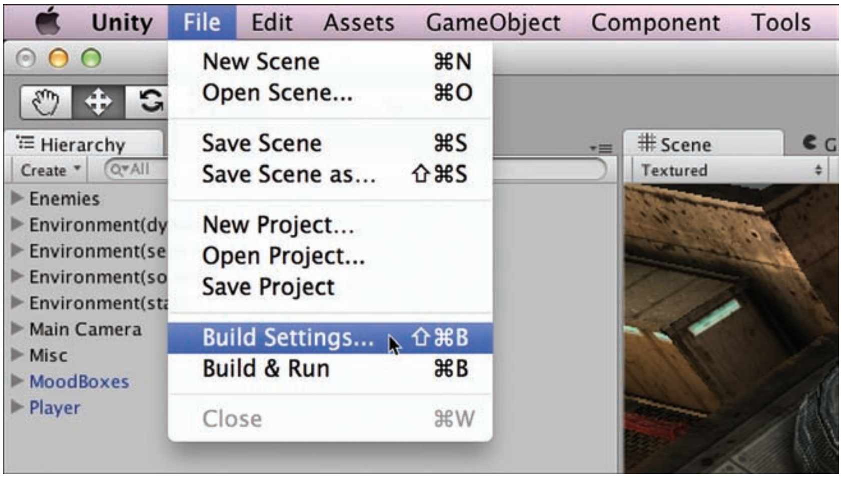
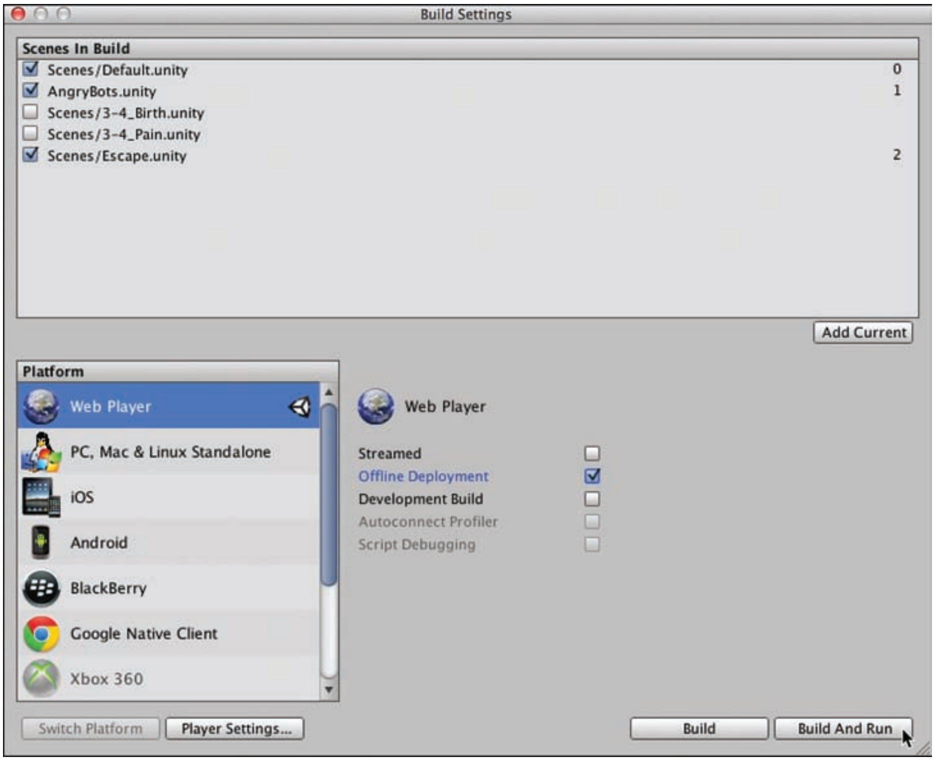


Figure 16.5 Build Settings menu selection

You should see the *Build Settings* window shown in Figure 16.6.

From here, be sure to click *Web Player* on the left and then check *Offline Deployment* in the *Web Player* options area. Click *Build and Run*, and Unity will ask you where to save the files. Type **AngryBots Web Build** for the filename and click *Save*.

Unity will process this for a while and build a web version of the game for you. Once it's built, your web browser will automatically be opened and sent to the page you just made as shown in Figure 16.7. Depending on your browser, you may be prompted to give the Unity plug-in permission to run.



Build Settings

Scenes In Build

✓	Scenes/Default.unity	0
✓	AngryBots.unity	1
☐	Scenes/3-4_Birth.unity	
☐	Scenes/3-4_Pain.unity	
✓	Scenes/Escape.unity	2

Platform: Web Player (selected)

Web Player

Streamed	☐
Offline Deployment	✓
Development Build	☐
Autoconnect Profiler	☐
Script Debugging	☐

Figure 16.6 Unity build settings for the web player

And there you go. You've compiled *AngryBots* for the web. Unity makes things like this very easy so that you can focus on the interesting work: game design and development.

Setting Up the Unity Window Layout

The last thing we need to do before we start actually making things in Unity is to get our environment laid out properly. Unity is very flexible, and one of those flexibilities is that it allows you to arrange its window panes however you like. You can see several window layouts by choosing various options from the *Layout* pop-up menu in the top-right corner of the Unity window (see Figure 16.8).

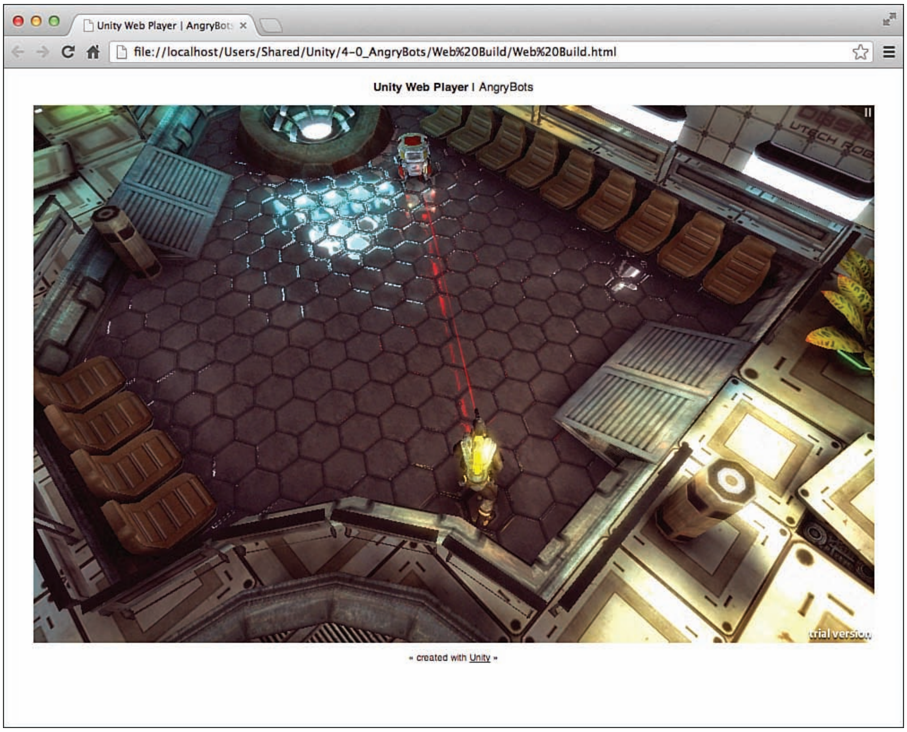

Unity Web Player | AngryBots
file://localhost/Users/Shared/Unity/4-0_AngryBots/Web%20Build/Web%20Build.html
Unity Web Player | AngryBots
trial version
« created with Unity »

Figure 16.7 *AngryBots* running in a browser window

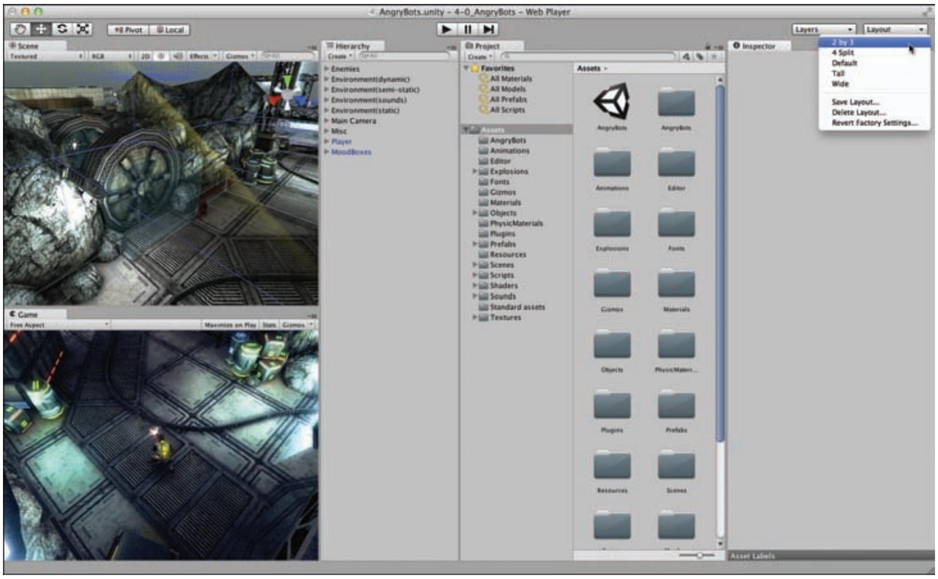


Figure 16.8 Position of the *Layout* pop-up menu and selection of the *2 by 3* layout

Choose *2 by 3* from this pop-up menu. This will be the starting point for making our layout.

Before doing anything else, let's make the Project pane look a little cleaner. Click on the options pop-up for the Project pane (shown in the black circle in Figure 16.9) and choose *One Column Layout*.

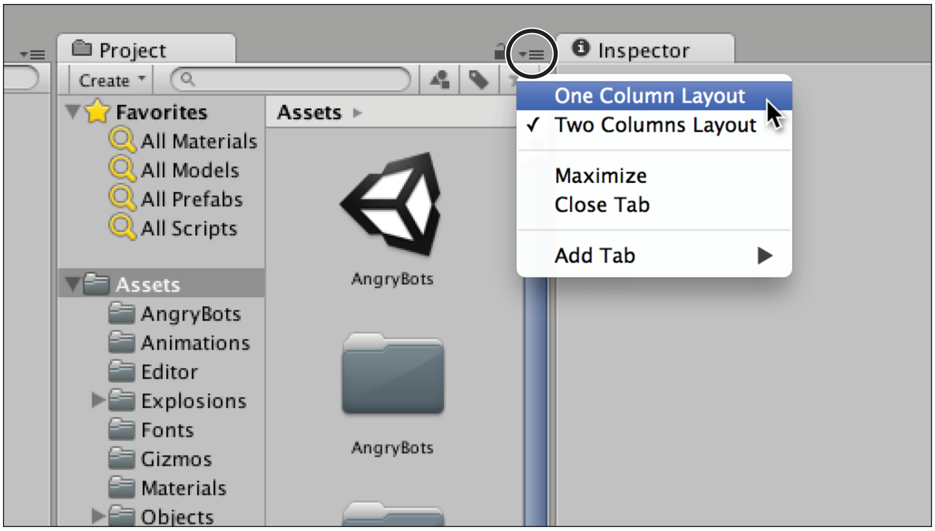


Figure 16.9 Choosing the One Column Layout for the Project pane

Unity enables you to both move window panes around and adjust the borders between them. As shown in Figure 16.10, you can move a pane by dragging its tab (the arrow cursor) or adjust a border between panes by dragging the border between them (the left-right resize cursor).

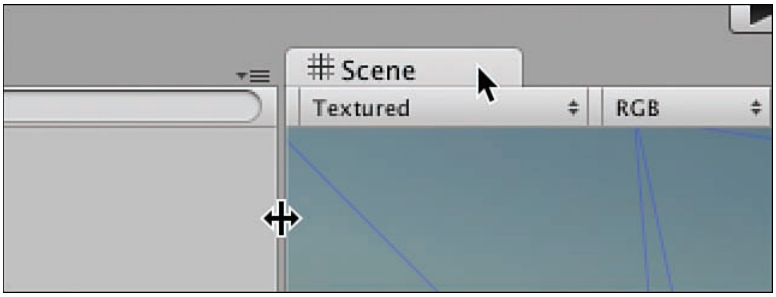


Figure 16.10 Two types of cursors for moving and resizing Unity's window panes

When you drag a pane by its tab, a small ghosted version will appear (see Figure 16.11). Some locations will cause the pane to snap into place. When this happens, the ghosted version of the tab will appear in the new location.

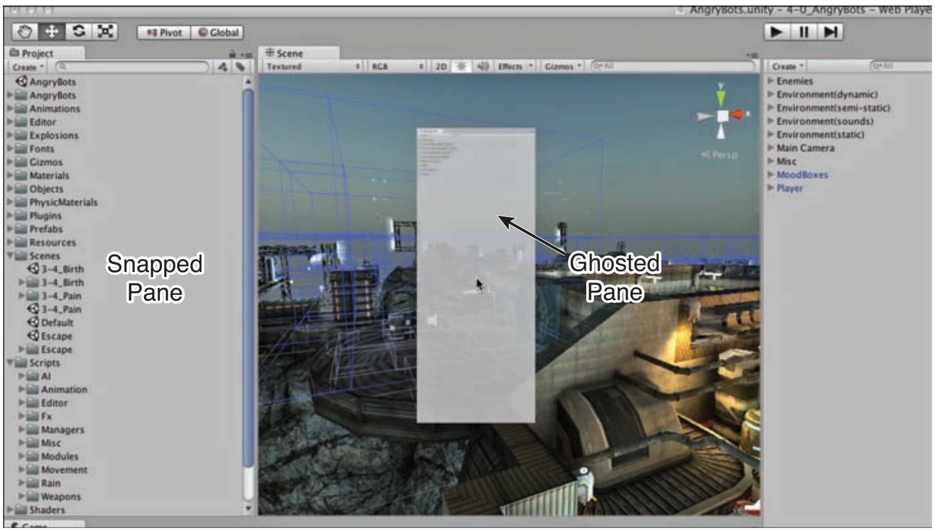


Figure 16.11 Ghosted and snapped panes when moving them around the Unity window

Play around with moving the window panes until your window looks like Figure 16.12.

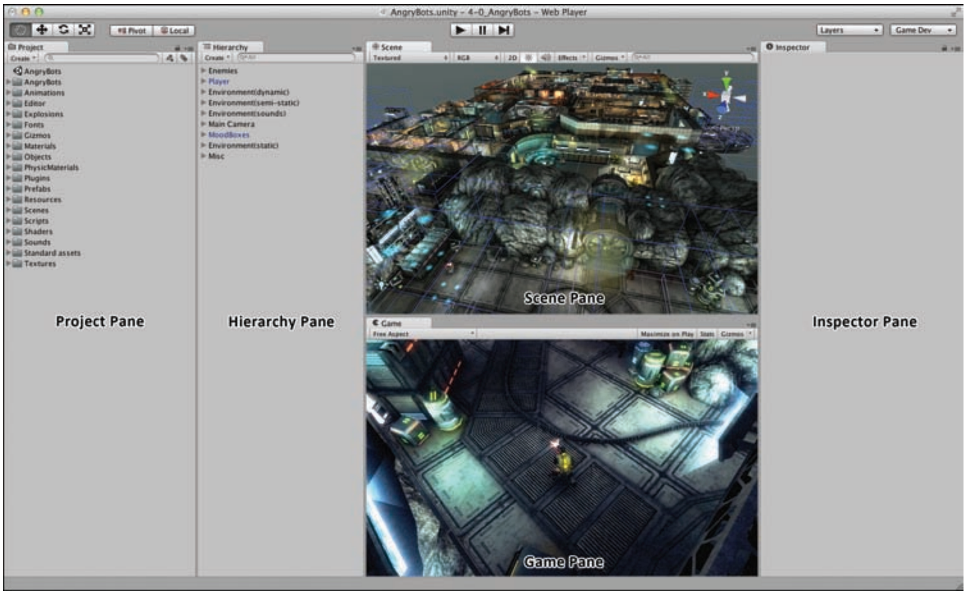


Figure 16.12 Proper layout for the Unity window...but it's still missing something

Now the last thing we need to add is the Console pane. From the menu bar, choose *Window > Console*. Then drag the Console pane below the Hierarchy pane. You'll also need to move the Project pane after you've done this to create the final layout shown in Figure 16.13.

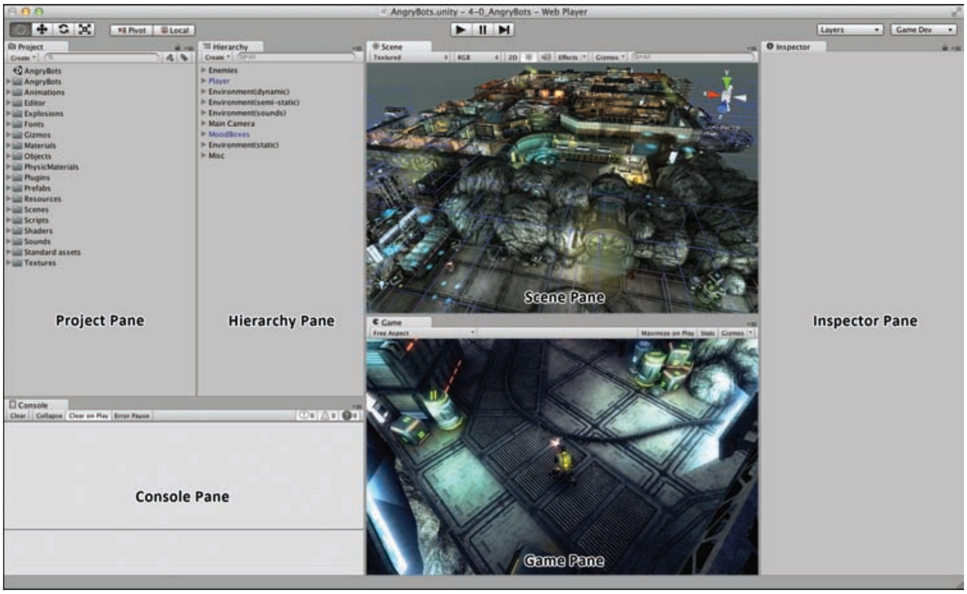


Figure 16.13 Final layout of the Unity window, including the Console pane

Now you just need to save this layout in the Layout pop-up menu so that you don't have to go through all that again. Click the Layout pop-up menu and choose *Save Layout*, as shown in Figure 16.14.

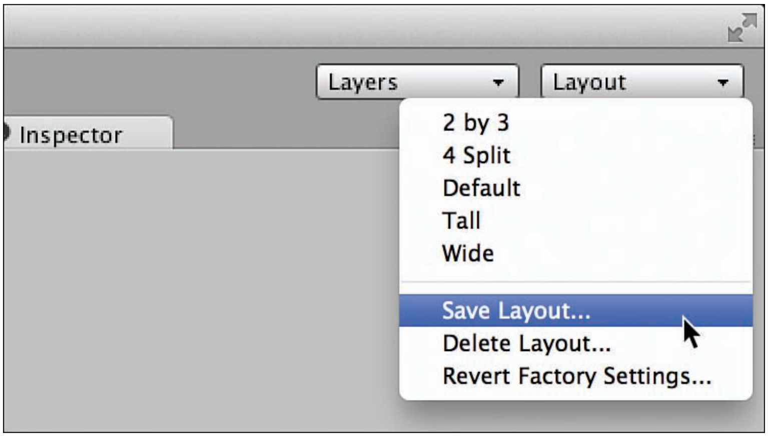


Figure 16.14 Saving the layout

Save this layout with the name *Game Dev*, with a leading space before the G in Game (i.e., " Game Design"). By putting a space at the beginning of the name, you make sure that this layout is sorted to the top of the menu. Now, any time you need to return to this layout, you can simply choose it from this pop-up menu.

Learning Your Way Around Unity

Before we can really get into coding things, you need to get to know the various window panes that you've just arranged. Refer back to Figure 16.13 as we discuss each pane:

- **Scene pane:** The Scene pane allows you to navigate around your scene in 3D and to select, move, rotate, and scale objects.
- **Game pane:** The Game pane is where you will preview your actual gameplay; it's the window in which you played *AngryBots* before compiling the web build. This pane also shows you the view from the *Main Camera* in your scene.
- **Hierarchy pane:** The Hierarchy pane shows you every GameObject that is included in your current scene. For now, you can think of each scene as a level of your game. Everything that exists in your scene, from the camera to your player-character, is a GameObject.
- **Project pane:** The Project pane contains all of the assets that are included in your project. An asset is any kind of file that is part of your project, including images, 3D models, C# code, text files, sounds, fonts and so on. The Project pane is a reflection of the contents of the Assets folder within your Unity project folder on your computer hard drive. These assets are not necessarily in your current scene.
- **Inspector pane:** Any time you click on an asset in the Project pane or a GameObject in the Scene or Hierarchy panes, you will be able to see and edit information about it in the Inspector pane.
- **Console pane:** The Console pane will allow you to see messages from Unity about errors or bugs in your code as well as messages from yourself that will help you understand the inner workings of your own code.[1] We will use the Console pane extensively in Chapter 18, "Hello World: Your First Program," and Chapter 19, "Variables and Components."

Summary

That's it for setup. Now, let's move on to actually developing! As you've seen in this chapter, Unity can create some pretty stunning visuals and compelling gameplay. Though the process of making beautiful 3D models and shaders is outside the scope of this book, it's important for you to know the extent of Unity's graphical capabilities. In the next chapter, you'll learn more about C#, the language you'll be using for game development.

[1] Unity's `print()` and `Debug.Log()` functions allow you to print messages to the Console pane.

This page intentionally left blank

This page intentionally left blank

INDEX

Numbers

3D animation/model resources, 854-855
3D printing, touch as an Inscribed Layer aesthetic, 47-48

A

A Pattern Language, 45
AAA (top) games, costs in developing, 213
Achiever player type (diamonds), 67
acquaintances as playtesters, 145
action, five-act dramatic narrative structures
  falling action (Act IV), 51
  rising action (Act II), 51
action games
  Omega Mage
    changing rooms, 764-768
    creating an inventory, 747-754
    creating the game environment, 730-735
    customizing setup, 789
    damaging enemies, 772-777
    damaging players, 777-782
    enemy factories, 785-789
    enemy interfaces, 782-785
    EnemyBug GameObjects, 770-780
    EnemySpiker GameObjects, 780-782
    example of play, 728-729
    fire ground spell, 754-762
    fire spell, 761-762
    fire-and-forget spells, 762-764
    ground spell, 756-761
    importing Unity asset packages, 729
    Mage GameObject (player character), 735-737
    mouse interaction, 737-747
    project setup, 729
    selecting elements from inventory, 749-754
    spawning enemies, 768-782
  puzzles in, 188
    boss fights, 195
    chain reaction puzzles, 194
    physics puzzles, 194
    sliding block/position puzzles, 193
    stealth puzzles, 194
    traversal puzzles, 194
action lists (GameObjects), *Apple Picker* game analysis, 231-232
actions
  discernable actions (meaningful play), 64
  integrated actions (meaningful play), 64-65
  tracking and reacting to (empathetic characters versus avatars), 57
Activision, *Kaboom!* game analysis (systems thinking), 229-234
Adkinson, Peter
  innovation and the design process, 97-98
ADL (Automated Data Logging) and playtesting, 151
Adobe software, educational software discounts, 855
Aeon of Strife, game mods and cultural mechanics, 81-82
aesthetics
  Cultural Layer (Layered Tetrad), 82
    cosplay, 82
    defining, 35
    fan art, 82
    gameplay as art, 83
  Dynamic Layer (Layered Tetrad), 70
    defining, 34
    environmental aesthetics, 70, 73-74
    procedural aesthetics, 70-73
  Elemental Tetrad framework, 27-28

Inscribed Layer (Layered Tetrad), 46
conveying information, 48-49
conveying mood, 48
defining, 33
five aesthetic senses, 47-48
goals of aesthetic design, 48-49
hearing, 47
immediacy of sound (Inscribed Layer aesthetics), 47
smell, 48
touch, 47-48
vision, 47
MDA framework, 21-24
Snakes and Ladders, 21-24
Agile software development methodologies, 405-407, 416
Agon (competitive play), *Le Jeux et Le Hommes* and the four different kinds of play, 110
Aguilar, Chris, *Vectorized Playing Cards 1.3*
Bartok, 622
Prospector Solitaire, 562
Alea (chance-based play), *Le Jeux et Le Hommes* and the four different kinds of play, 110
Alexander, Christopher, and the purpose of space (inscribed game mechanics), 45
alpha phase (project development process), 103
ambiguous decisions, importance of, 121
analysis phase (iterative design), 90-91
audience, determining, 91
fastest path to testing, determining, 92
prior art, researching, 92
repetition, importance of, 96-97
resources, determining, 91-92
analyzing game play (playtesting) in *Bartok*
questions, asking, 7
rounds, comparing, 9
AND operator (&&) in C# coding, 300
Anderson, Nels, and intent versus execution, 187
AngryBots, Unity project example, 242-246
Animal Crossing, fulfilling play as a player-centric goal of game design, 110
animation
Bartok digital prototype, 623
model resources and, 854-855
procedural animation, 72
Word Game, 687-690
antagonism (Act II), three-act dramatic narrative structure, 52
Apache OpenOffice Calc. *See* Calc
Apple Picker, 419
Apple GameObject, 424-425
AppleTree GameObject, 421-423
art assets, 421-425
Basket GameObject, 425
C# coding, 425-426
adding high scores, 445-448
adding points, 441-442
basic movement, 431-432
catching apples, 439-440
changing direction, 432-434
destroying baskets, 444-445
dropped apple notifications, 443-444
dropping apples, 434-435
Game Over screen, 448
GUI and game management, 440-448
increasing difficulty, 448
instantiating baskets, 437-438
moving baskets via mouse, 438-439
score counter, 440-441
setting physics layers, 435-436
Start screen, 448
stopping apples from falling too far, 437
camera setup, 425-426
game analysis (systems thinking), 229
GameObject action lists, 231-232
GameObject flowcharts, 232-234
GameObjects, 230-231
gameplay, 230
preparing for, 421
project setup, 420
time-based games, 431-432
ARG (Alternate Reality Games)
Assassin, 17
boundary mechanics, 44
Majestic, 17

arrays (C# coding), 328, 333-344
  basic array creation, 333-334
  converting
    arrays to lists, 336-337
    lists to arrays, 331
  empty elements
    foreach loops and, 335
    empty elements within arrays, 334-335
  jagged arrays, 340-342
  lists versus, 344
  methods, 336
  multidimensional arrays, 337-340
  properties, 336
  static methods, 336
  zero indexed arrays, 331
arrows, indirectly guiding players by, 202-203
art
  art assets, 854
  fan art (cultural aesthetics), 82
  gameplay as art (cultural aesthetics), 83
  prior art, researching (analysis phase of iterative design), 92
  procedural visual art, 71-72
Art of Game Design: A Book of Lenses, The, 4, 11, 20, 27
  guiding players indirectly, 200-206
  interest as a player-centric goal of design, 119
  listening during design phase (iterative design), importance of, 93
  Ten Rules of Probability Every Game Designer Should Know, 165-169
  testing phase (iterative design), 96
Art of Puzzle Design, The
  eight steps of puzzle design, 191-192
  genres of puzzles, 187-188
  goals of effective design, 192-193
  reasons for playing puzzles, 189
  required modes of thought in solving, 189
aspect ratios
  QuickSnap, 697
  Space SHMUP, 490
Assassin, 17
Assassin's Creed
  player guidance, 202, 205
  limited possibilities and interactive or linear narrative, 53-54
  resolution (screen), 73
Assassin's Creed IV: Black Flag
  experiences and space (Inscribed Layer mechanics), 46
  player guidance, 199
  premises narratives, examples of, 49
asset packages (Unity)
  Prospector Solitaire, 562
  QuickSnap, 696-697
  Space SHMUP, 488-490
Assets folder (Unity), 265-266
assigning tasks in burndown charts, 414-415
AT (Automated Testing), 153
Atkinson, Kevin, on word lists and *Word Game,* 660
Atlas Games, *Lunch Money* and house rules (dynamic mechanics), 67
attention and involvement as player-centric goals of design, 118-120
audience, determining (analysis phase of iterative design), 91
audio assets, Unity Asset Store, 854
audio design, indirectly guiding players by, 204-205
auditory play environments (environmental aesthetics), 73-74
  noisy environments, 74
  volume control, 74
AutoDesk software, educational software discounts, 855
Automated Testing (AT), 153
Automated Data Logging (ADL) and play testing, 151
autotelic empowerment as a player-centric goal of design, 116-117
avatars
  empathetic characters versus (interactive versus linear narrative), 55-57
    multiple dialogue choices, 56-57
    role fulfillment, 56
    silent protagonists, 56

tracking and reacting to player actions, 57
guiding players indirectly by, 205
average damage (weapons and game balance)
calculating, 179
charting, 179-180
axis mapping
InputManager (Unity), 491-494
Microsoft controllers, 848-849

B

balance (See game balance)
Bartle, Richard, and types of players, 67-68
Bartok, 4-5
analyzing game play, 7
deblocking, 6-7
digital prototype, building, 621-622, 655
backgrounds, 622-623
build settings, 623-624
card animation, 623
creating cards, 624-629
fanning the hand, 638-640
game logic, 653-655
layouts, 629-638
LINQ (Language Integrated Query) and C# coding, 640-641
managing initial deal, 642-643
managing turns, 646-653
moving cards, 641-642
rules of the game, 622
sorting order (2D-depth), 643-646
digital version, obtaining, 4-5
dynamic procedures, 64
objective, 5
playtesting, 6-10
riffle shuffling, 7
rounds, comparing, 9
rules of, 5, 8
Bateman, Chris
games, defining, 15
Ilinx (vertigionous play), 110
Imaginary Games, 15, 110, 115-116
structured conflict as a player-centric goal of design, 115-116
BDV (Burndown Velocity) and burndown charts, 414-416
Beale, Alan, on word lists and *Word Game,* 660
Beck, Kent, *Manifesto for Agile Software Development*, 406-407
behavioral change, designing games for, 109
Bejeweled 2, 186
beta phase (project development process), 103
beta tests (playtesting)
closed playtesting, 150
limited playtesting, 150
open playtesting, 150-151
Bethesda Softworks
actions (players), tracking and reacting to (empathetic characters versus avatars), 57
Creation Kit, game mods and cultural mechanics, 82
Fallout 3, side quests, 54
Skyrim
conflicting objectives (Inscribed layer mechanics), 42
final outcomes (dynamic mechanics), 69
game mods and cultural mechanics, 82
importance of objectives (Inscribed layer mechanics), 42
narrative game mods, 83
side quests, 54
Bézier curves, 841-843
recursive function to solve, 844-845
recursive functions and, 815
Bioware
Mass Effect
multiple dialogue choices (empathetic characters versus avatars), 57
player interaction patterns, 43
Star Wars: Knights of the Old Republic, limited possibilities and interactive or linear narrative, 53-54
bitwise Boolean operators (C# coding), 302, 801-802
Blade Runner, multiple dialogue choices (empathetic characters versus avatars), 57
Blender software, educational software discounts, 855

Blizzard
- *Starcraft*, game mods and cultural mechanics, 81-82
- *Warcraft III*, game mods and cultural mechanics, 81-82

board games, systems thinking, 226

Bogost, Ian, and the magic circle concept and lusory attitude, 112

Boids project (OOP in C# coding), 394
- building simple models, 394-396
- project setup, 394
- scripts, 397-403

Book of Games: Strategies, Tactics & History, The, 22

bool variables (C# coding), 283

Boolean operations (C# coding), 299-300, 313
- bitwise Boolean operators, 302, 801-802
- combining, 302-303
- logical equivalence of, 303
- AND operator (&&), 300
- NOT operator (!), 300
- OR operator (||), 300
- shorting operators versus non-shorting operators, 301-302

boss fights as puzzles, 195

Botermans, Jack
- *Book of Games: Strategies, Tactics & History, The*, 22
- *Snakes and Ladders*, 22

boundaries
- Alternative Reality Games, 44
- Formal, Dynamic, and Dramatic framework, 25
- Inscribed Layer (Layered Tetrad), 40, 44

Box Collider component (GameObjects), 272, 295

brainstorming (ideation), 98
- collection phase, 99
- collision phase, 100-101
- discussions, 101
- expansion phase, 98-99
- rating phase, 101

Brice, Mattie, and *Mainichi*
- experiential understanding as a player-centric goal of design, 122-123
- personal expression/communication as a goal of game design, 108

bridge puzzles, 188

brightness. *See also* color
- environmental aesthetics (Dynamic Layer), 73
- indirectly guiding players by (contrast), 204

Brigs, Jeff, and music in *C.P.U. Bach*, 71

bullets, determining percent chance of to hit (weapons and game balance), 178

Bungie and *Halo*
- inscribed dramatics, example of, 58
- limited possibilities and interactive or linear narrative, 53-54
- prior art, researching (analysis phase of iterative design), 92
- *Red vs. Blue*, machinima example, 83-84

Burgun, Keith
- ambiguous decisions, importance of, 121
- fun in game design, the importance of, 109-110
- *Game Design Theory: A New Philosophy for Understanding Games*, 11, 83, 118
- gameplay as art (cultural aesthetics), 83
- games, defining, 11, 14-15
- interesting decisions as a player-centric goal of design, 120-121
- performative empowerment as a player-centric goal of design, 118

burndown charts, 409
- assigning tasks, 414-415
- BDV (Burndown Velocity), 414-416
- creating, 416
- estimated hours versus real hours, 413
- example of, 410-412
- prioritizing tasks, 414-415

button mapping, Microsoft controllers, 848-849

C

C#, 253, 262
- arrays, 328, 333, 344
  - *basic array creation, 333-334*
  - *converting arrays to lists, 336-337*
  - *converting lists to arrays, 331*

empty elements and foreach loops, 335
empty elements within arrays, 334-335
jagged arrays, 340-342
lists versus arrays, 344
methods, 336
multidimensional arrays, 337-340
properties, 336
static methods, 336
zero indexed arrays, 331
Bézier curves and recursive functions, 815
Boolean operations, 299-300, 313
bitwise Boolean operators, 302, 801-802
combining, 302-303
logical equivalence of, 303
AND operator (&&), 300
NOT operator (!), 300
OR operator (||), 300
shorting operators versus non-shorting operators, 301-302
C# 4.0 Pocket Reference, 854
camelCase naming conventions, 286
classes, 379-380, 390
elements of, 380-381
Enemy Class sample project, 381-384, 387-388
inheritance, 387-389
instances, 289-288
instances as GameObject components, 385-387
naming conventions, 287
Object-Oriented Programming (OOP), 258-259
properties, 384-385
subclasses, 388-389
superclasses, 388-389
collections, 327-328
arrays, 328, 333-337
jagged, 340-342
multidimensional, 337-340
lists, 328-333
jagged, 342-344
comparison statements, 299, 303, 313
Greater Than comparison operator (>), 306
Greater Than or Equal To comparison operator (>=), 306-307
Is Equal To comparison operator (==), 304
Less Than comparison operator (<), 306
Less Than or Equal To comparison operator (<=), 306-307
Not Equal To comparison operator (!=), 306
compiled language, C# as, 254-256
conditionals, 299, 307
if statements, 307-310
switch statements, 310-313
coroutines, 660, 802-803
CSharp Yellow Book, 854
debugging, 363-364
attaching debugger to Unity, 372-374
attaching/removing scripts, 366-367
compile-time bugs, 364-366
QuickSnap, 713-714
runtime errors, 367-369
stepping through code, 369-371, 373-377
variables, 375-376
Enemy Class sample project
class inheritance, 387-388
project setup, 381-384
enum, 500, 803-804
equality testing, 304
features of (overview), 254
function delegates, 525-531, 805-807
functions, 257-258, 349-350, 362
defining, 350-353
naming, 287, 356
optional parameters, 359
overloading, 358
parameters and arguments, 353-354
params keyword, 359-361
reasons for using, 356-357
recursive functions, 361-362
returning values, 354-356
static functions, 288
void, returning, 355-356
GameObjects, 293-294
Box Collider component, 295
Capsule Collider component, 295
Collider components, 295-296

Mesh Collider component, 295
Mesh Filter component, 295
Renderer component, 295
RigidBody component, 296
scripts as GameObject components, 296-297
Sphere Collider component, 295
Transform component, 294-295
interfaces, 782-785, 807-810
LINQ (Language Integrated Query) and, 640-641
lists, 328-331, 344
arrays versus lists, 344
converting arrays to lists, 336-337
converting lists to arrays, 331
jagged lists, 342-344
methods, 331
properties, 331
zero indexed lists, 331
loops, 315, 325
do...while loops, 316, 319-320
for loops, 316, 320-322, 342
foreach loops, 316, 322, 335
infinite loops, 317-318
jump statements in loops, 322-324
types of, 316
while loops, 316-319
managed code, C# as, 256
modulus operator (%), 325
MonoDevelop editor, creating C# scripts, 266-271
naming conventions, 286-287, 810
online resources, 853-854
OOP (Object-Oriented Programming), 258-259, 391, 403
Boids project, 394-403
defining through metaphors, 392-393
operator precedence, 810
order of operations, 810
race conditions, 811-814
reasons for choosing, 238-239
reserve functions, 814-815
runtime speed, 239
scripts
attaching to scene Main Camera, 797-798
creating, 266-271, 797
elements of, 380-381
GameObject components, scripts as, 296-297
search tips, 854
software design patterns, 815
Factory design pattern, 816
Singleton design pattern, 815-816
Strategy design pattern, 816-817
strongly typed language, C# as, 256-257
subclasses, 589-592
superclasses, 591
syntax of, 259-260
systems thinking and, 227
time-based games, 431-432
tips for learning, 239-241
Unity, online resources, 853-854
variables, 256-257, 282
bool variables, 283
char variables, 285
class variables, 286
Color variables, 290-291
debugging C# coding, 375-376
declaring, 282-283
defining, 282-283
equality testing, 304
float variables, 284
instance variables, 289
int variables, 284
literal values, 282-283
Mathf variables, 292
naming conventions, 286-287
private variables, 287
Quaternion variables, 291-292
scope of, 286, 817-820
Screen variables, 292
string variables, 285
SystemInfo variables, 293
types of, 283-286
Unity variables, 287-294
Vector3 variables, 288-290
XML and, 817-821

Caillois, Roger, on *Le Jeux et Le Hommes* and the four different kinds of play, 110
Calc, Open Office
  charts, creating, 164
  COUNTIF formula, weighted probabilities, 174-175
  dice probability, determining, 157-158
    adjusting column widths, 160
    charting results, 164
    counting die roll sums, 163-164
    creating Die A row, 160-161
    creating Die B row, 161-162
    creating probability distribution charts, 164
    creating rows, 159
    entering data in cells, 158-159
    labeling rows, 162-163
    spreadsheets, creating, 158
    summing dice role results, 163
  FLOOR formula, weighted probabilities, 174-175
  formula editing, exiting, 164
  Function Wizard
    creating rows, 160-161
  INDEX formula, weighted probabilities, 174-175
  installing, 156-157
  MOD formula, 160-161
  RAND formula, weighted probabilities, 174-175
  REPT formula, showing overall damage (weapons and game balance), 181
  ROUND formula, showing overall damage, (weapons and game balance), 181
  SUM formula, showing overall damage (weapons and game balance), 181
  weapons, balancing, 177-178
    average damage, 179-180
    Chapter 9 prototype example, 182-183
    duplicating weapon data, 180-181
    overall damage, showing, 181
    percent chance for each bullet, determining, 178
    rebalancing weapons, 177-178
  weighted probabilities, 174-175
Call of Duty, limited possibilities and interactive or linear narrative, 53-54
Call of Duty: Modern Warfare, researching prior art (analysis phase of iterative design), 92
calls to action, directly guiding players by, 199
camelCase naming conventions (C# coding), 286
camera setups
  Apple Picker, 425-426
  follow cameras, 462-466
  indirectly guiding players by, 203
  Mission Demolition, 451-453, 462-466
  orthographic cameras, 425-426
  perspective cameras, 425-426
  Space SHMUP, 490
campaigns (RPG), tips for running, 846-847
Capsule Collider component (GameObjects), 295
card games
  Bartok, 621-622, 655
    backgrounds, 622-623
    build settings, 623-624
    card animation, 623
    creating cards, 624-629
    fanning the hand, 638-640
    game logic, 653-655
    layouts, 629-638
    LINQ (Language Integrated Query) and C# coding, 640-641
    managing the initial deal, 642-643
    managing turns, 646-653
    moving cards, 641-642
    rules of the game, 622
    sorting order (2D-depth), 643-646
  custom card decks, 172
  Poker
    game balance and, 176-177
    Red Dead Redemption rule mechanics, 44
  Prospector Solitaire, 561
    art assets, 614-615
    backgrounds, 614-615

beginning/end of round announcements, 615-616
chain scoring, 605-607
clickable card functionality, 597-600
creating cards from sprites, 566-581
displaying scores to players, 607-614
draw and discard actions, 597-604
earning points, 604-605
example of play, 584-585
game logic, 597-604
giving player feedback on scores, 616-619
gold cards, 620
importing images as sprites, 564-566
importing Unity asset packages, 562
matching drawn cards, 600-604
mine tableau layout, 585-592
mobile device build settings, 562-563, 620
positioning cards on tableau, 592-594
project setup, 562
rules of the game, 583-584
scoring, 604-619
setting up sorting layers on tableau, 594-597
shuffling cards, 581-583
randomization, 171-173
riffle shuffling, 7
shuffling, 7, 172-173
Vectorized Playing Cards 1.3 by Chris Aguilar
Bartok, 622
Prospector Solitaire, 562
careers in the digital game industry, 217
following up, 218-219
interviewing, 219-220
networking, 217-218
salaries, 220
Cash, Bryan
Skyrates, 111
sporadic-play games and lusory attitude, 111-112
cells (Calc spreadsheets)
adjusting column widths, 160
entering data, 158-159
MOD formula, 160-161
SUM formula, 163
chain reaction puzzles, 194
chance-based play (Alea), *Le Jeux et Le Hommes* and the four different kinds of play, 110
changing
direction, *Apple Picker*, 432-434
your mind (design process), 101
char variables (C# coding), 285
characters
empathetic characters versus avatars (interactive versus linear narrative), 55-57
FDD framework, 26
Inscribed Layer (Layered Tetrad), 49
NPC, development of (interactive narratives), 55
charts (Calc)
average damage (weapons and game balance), 179-180
creating, 164
Cheap Ass Games, touch as an Inscribed Layer aesthetic, 47
Cheater player type, 68
Chen, Jenova
flow as a player-centric goal of design, 113
Journey, 144-145
tissue playtesters, 144-145
chess puzzles, 188
Chief Plenty-Coups and coup-counting as a game, structured conflict as a player-centric goal of design, 115-116
Chowanec, John, on fortune as a goal of game design, 107
Chrono Trigger, limited possibilities and interactive or linear narrative, 53-54
Chutes and Ladders, 26
Cialdini, Robert B., and *Influence: The Psychology of Persuasion*, 4
citizens (player roles), 44
Civilization, tables, 40-41
class variables (C# coding)
classes (C# coding), 379-380, 390. *See also* OOP (Object-Oriented Programming) in C#
elements of, 380-381
Enemy Class sample project
inheritance, 387-388
project setup, 381-384
inheritance, 387-388

instances, 289-288, 385-387
naming conventions, 287
properties, 384-385
subclasses, 388-389
superclasses, 388-389
clear decisions as part of player-centric design, 122
climaxes
Red Dead Redemption, 53
five-act dramatic narrative structures, 51
three-act dramatic narrative structure, 53
closed playtesting (beta tests), 150
Clover Studios and *Okami*
empathetic characters versus avatars, 56
touch as an Inscribed Layer aesthetic, 48
code libraries and systems thinking, 228
collaborators (player roles), 43
collection phase (brainstorming/ideation), 99
collections (C# coding), 327-328
arrays, 328, 333, 344
basic array creation, 333-334
converting arrays to lists, 336-337
converting lists to arrays, 331
empty elements and foreach loops, 335
empty elements within arrays, 334-335
jagged arrays, 340-342
lists versus arrays, 344
methods, 336
multidimensional arrays, 337-340
properties, 336
static methods, 336
zero indexed arrays, 331
lists, 328-331, 344
arrays versus lists, 344
converting arrays to lists, 336-337
converting lists to arrays, 331
jagged lists, 342-344
methods, 331
properties, 331
zero indexed lists, 331
colleges/universities, Games Education Programs, 215-217
Collider components (GameObjects), 272, 295-296
Collins, Andy on XP (Experience Points) and cumulative outcomes (dynamic mechanics), 69
collision phase (brainstorming/ideation), 100-101
color. *See also* brightness
Color variables (C# coding), 290-291
colorblindness, player considerations (environmental aesthetics), 74
indirectly guiding players by, 204
Word Game, 690-692
columns (Calc spreadsheets)
adjusting widths, 160
labeling rows, 162-163
combat and gameplay development (paper prototyping)
control points, capturing, 137
counterattacks, 135
cover, 136
health, 136
interception fire, 137
weapons/firing, 134-135
communication (personal) as a goal of game design, 108
community as a goal of game design, 107-108, 120
comparison statements (C# coding), 299, 303, 313
Greater Than comparison operator (>), 306
Greater Than or Equal To comparison operator (>=), 306-307
Is Equal To comparison operator (==), 304
Less Than comparison operator (<), 306
Less Than or Equal To comparison operator (<=), 306-307
Not Equal To comparison operator (!=), 306
competition
Agon (competitive play), *LeJeux et Le Hommes* and the four different kinds of play, 110

multilateral competition (player interaction patterns), 43
player roles, 43
team competition (player interaction patterns), 43
unilateral competition (player interaction patterns), 43
compiling
AngryBots, Unity project example, 245-246
compile-time bugs, debugging in C# coding, 364-366
complex problems, breaking down (systems thinking), 229
computer languages and systems thinking, 227
Concept of Flow, *The,* 113-114, 116-117
concepts
developing concepts (paper prototyping), 129-130
new concepts, teaching by player guidance, 207
integration, 209
sequencing, 207-209
conditionals (C# coding), 299, 307
if statements, 307-310
switch statements, 310-313
conferences
careers in the digital game industry, 217-218
items to take when attending, 218
conflict (structured) as a player-centric goal of design, 115-116
conflicting objectives (Inscribed layer mechanics), 42
Conrad, Joseph, *The Heart of Darkness* as an example of inscribed dramatics, 57-58
Console pane (Unity), 251
constraints, indirectly guiding players by, 200
construction puzzles, 188
contrast (brightness), indirectly guiding players by, 204
control points and gameplay development (paper prototyping), capturing, 137
controllers (Microsoft)
axis mapping, 848-849
button mapping, 848-849
cooperative play (player interaction patterns), 43
Core War, example of player's role in game design, 62
coroutines (C# coding), 663, 802-803
cosine/sine, 822-825
cosplay (cultural aesthetics), 82
costs of AAA (top) game development, 213
Counter Strike, game mods and cultural mechanics, 81-82
counterattacks, gameplay development (paper prototyping), 135
COUNTIF formula (Calc), weighted probabilities, 174-175
coup-counting, structured conflict as a player-centric goal of design, 115-116
cover in gameplay development (paper prototyping), determining, 136
C.P.U. Bach, music in, 71
Crazy Cakes, ADL (Automated Data Logging) and playtesting, 151
Creation Kit (Bethesda Softworks), game mods and cultural mechanics, 82
crossword puzzles, 188
CSharp Yellow Book, 854
Csíkszentmihályi, Mihaly
autotelic empowerment as a player-centric goal of design, 116-117
Concept of Flow, 113-114, 116-117
flow as a player-centric goal of design, 113-114
Flow: The Psychology of Optimal Experience, 114, 116-117
Cultural Layer (Layered Tetrad), 34-35, 37-38, 79, 87
aesthetics, 82
cosplay, 82
defining, 35
fan art, 82
gameplay as art, 83
cultural impact of games, 86-87

defining, 80-81
designer responsibilities, 37
law and cultural impact of games, 86-87
mechanics, 81
custom game levels, 82
defining, 35
game mods, 81-82
narrative, 83
defining, 36
fan fiction as cultural narrative, 83
game mods as cultural narrative, 83
machinima, 83-84
technology, 84
defining, 36
external tools (player-built) and cultural technology, 84-85
game technology used in other fields, 84
transmedia, 85-86
cumulative outcomes (dynamic mechanics), 69
custom card decks, 172
custom game levels (cultural mechanics), 82

D

damage, weapons and game balance
average damage
calculating, 179
charting, 179-180
overall damage, showing, 181
deblocking in *Bartok*, 6-7
debugging C# coding, 363-364
attaching debugger to Unity, 372-374
attaching/removing scripts, 366-367
compile-time bugs, 364-366
QuickSnap, 713-714
runtime errors, 367-369
stepping through code, 369-371, 373-377
variables, 375-376
decimals and probability, 165-166
decision-making as a player-centric goal of design, 120-122
ambiguous decisions, importance of, 121
clear decisions, 122
double-edged decisions, 121
meaningful play (dynamic procedures)
discernable actions, 121
integrated actions, 121
novel decisions, 121
decks of cards
Bartok, 621-622, 655
backgrounds, 622-623
build settings, 623-624
card animation, 623
creating cards, 624-629
fanning the hand, 638-640
game logic, 653-655
layouts, 629-638
LINQ (Language Integrated Query) and C# coding, 640-641
managing the initial deal, 642-643
managing turns, 646-653
moving cards, 641-642
rules of the game, 622
sorting order (2D-depth), 643-646
custom card decks, 172
art assets, 614-615
backgrounds, 614-615
beginning/end of round announcements, 615-616
chain scoring, 605-607
clickable card functionality, 597-600
creating cards from sprites, 566-581
displaying scores to players, 607-614
draw and discard actions, 597-604
earning points, 604-605
example of play, 584-585
game logic, 597-604
giving player feedback on scores, 616-619
gold cards, 620
importing images as sprites, 564-566
importing Unity asset packages, 562
matching drawn cards, 600-604
mine tableau layout, 585-592
mobile device build settings, 562-563, 620

positioning cards on tableau, 592-594
project setup, 562
rules of the game, 583-584
scoring, 604-619
setting up sorting layers on tableau, 594-597
shuffling cards, 581-583
Prospector Solitaire, 561
randomization, 171-173
riffle shuffling, 7
shuffling, 7, 172-173
Vectorized Playing Cards 1.3
Bartok, 622
Prospector Solitaire, 562
Defense of the Ancients, game mods and cultural mechanics, 81-82
degrees (colleges/universities), Games Education Programs, 215-217
delegates (function) in C# coding, 805-807
Demo Projects section (Unity website), 852
denouement (Act V), five-act dramatic narrative structures, 51
deploying *AngryBots*, Unity project example, 245-246
design goals, 105-106, 124
designer-centric goals, 106
community, 107-108
fame, 107
fortune, 107
greater good, 108-109
personal development/experience, 109
personal expression/communication, 108
player-centric goals, 106
attention and involvement, 118-120
empowerment, 116-118
experiential understanding, 122-123
fiero, the concept of, 110
flow, 113-115
fun, 109-110
interesting decisions, 120-122
lusory attitude, 110-112
structured conflict, 115-116
Design Patterns
software design patterns, 815
Factory design pattern, 816
Singleton design pattern, 815-816
Strategy design pattern, 816-817
spawning enemies/enemy factories, 769-770
design phase (iterative design), 90
listening, importance of, 92-94
repetition, importance of, 96-97
designers, 89
Agile software development methodologies, 405-407, 416
brainstorming (ideation), 98
collection phase, 99
collision phase, 100-101
discussions, 101
expansion phase, 98-99
rating phase, 101
burndown charts, 409
assigning tasks, 414-415
BDV (Burndown Velocity), 414-416
creating, 416
estimated hours versus real hours, 413
example of, 410-412
prioritizing tasks, 414-415
changing your mind (design process), 101
designer-centric goals of design, 106
community, 107-108
fame, 107
fortune, 107
greater good, 108-109
personal development/experience, 109
personal expression/communication, 108
ideation (brainstorming), 98
collection phase, 99
collision phase, 100-101
discussions, 101
expansion phase, 98-99
rating phase, 101
innovation, 97-98
iterative design, 90
analysis phase, 90-92
design phase, 90
implementation phase, 90-91
testing phase, 91

overscoping (design process), 103-104
player-centric goals of design, 106
attention and involvement, 118-120
empowerment, 116-118
experiential understanding, 122-123
fiero, the concept of, 110
flow, 113-115
fun, 109-110
interesting decisions, 120-122
lusory attitude, 110-112
structured conflict, 115-116
project development process, 102
alpha phase, 103
beta phase, 103
gold phase, 103
post-release phase, 103
preproduction phase, 102
production phase, 102-103
responsibilities within Layered Tetrad frameworks
Cultural Layer, 37
Dynamic Layer, 36-37
Inscribed Layer, 36
scoping and the design process, 103-104
Scrum software development methodologies, 407, 416
meetings, 409
product backlogs/feature lists, 408
releases/sprints, 408
teams, 408
development environments. *See* Unity
development process (projects), 102
Agile software development methodologies, 405-407, 416
alpha phase, 103
beta phase, 103
burndown charts, 409
assigning tasks, 414-415
BDV (Burndown Velocity), 414-416
creating, 416
estimated hours versus real hours, 413
example of, 410-412
prioritizing tasks, 414-415
gold phase, 103
post-release phase, 103
preproduction phase, 102
production phase, 102-103
Scrum software development methodologies, 407, 416
meetings, 409
product backlogs/feature lists, 408
releases/sprints, 408
teams, 408
Diaconis, Persi, 7
dialogue
immediacy of sound (Inscribed Layer aesthetics), 47
multiple dialogue choices (empathetic characters versus avatars), 56-57
Diamante, Vincente, and music in *Flower*, 71
dice probability, determining with Calc, 157-158
cells (Calc spreadsheets), entering data in, 158-159
charting results, 164
counting die roll sums, 163-164
probability distribution charts, creating via Calc, 164
randomization, 170
rows (Calc spreadsheets)
adjusting column widths, 160
creating Die A row, 160-161
creating Die B row, 161-162
creating rows, 159
labeling rows, 162-163
spreadsheets, creating, 158
summing dice role results, 163
Unity example, 825-829
difficulty, increasing, *Apple Picker*, 448
digital game industry, 211-212, 222
changes to, 213
costs of AAA (top) game development, 213
freemium games, 214
indie games, 214
working conditions, 213
Entertainment Software Association Essential Facts list, 212
Games Education Programs, 215-217
getting into (careers in), 217
following up, 218-219

interviewing, 219-220
networking, 217-218
salaries, 220
projects
joining, 221
starting, 221-222
digital prototypes, purpose of, 420
digital systems thinking, 225, 234
Apple Picker game analysis, 229
GameObject action lists, 231-232
GameObject flowcharts, 232-234
GameObjects, 230-231
gameplay, 230
board games, 226
breaking down complex problems into simpler ones, 229
code libraries, 228
computer languages, 227
development environment, 228
simple instructions exercise, 226-227
direction in *Apple Picker*, changing, 432-434
directionality (contrast), indirectly guiding players by, 204
directly guiding players
Assassin's Creed IV: Black Flag, 199
Grand Theft Auto V, 199
Kya: Dark Legacy, 198
Legend of Zelda: Ocarina of Time, 199
methods of, 199
calls to action, 199
instructions, 199
maps/guidance systems, 199
pop-ups, 199
quality of guidance, determining, 198
Skyrim, 198
Valkyrie Chronicles, 198
discernable actions (meaningful play), 64, 121
discounts (educational software), 855
Disneyland, indirect guidance example, 202
distributions (weighted) and game balance, 173-175
Doctor Who, foreshadowing and interactive narrative, 55
dot product, 829-830
double-edged decisions as part of player-centric design, 121
do.while loops (C# coding), 316, 319-320
downloading, Unity, 236
DPS (Damage Per Second) calculators, player-built external tools as example of cultural technology, 84
Dramatic elements (FDD framework), 24-26
characters, 26
premises, 25-26
stories, 26
dramatic narrative structures
dynamic dramatics, 75
emergent narrative, 76-77
interactive narrative, 75-76
five-act dramatic narrative structure
climax (Act III), five-act dramatic narrative structures, 51
denouement (Act V), five-act dramatic narrative structures, 51
exposition (Act I), five-act dramatic narrative structures, 50-51
falling action (Act IV), five-act dramatic narrative structures, 51
rising action (Act II), five-act dramatic narrative structures, 51
inscribed dramatics, purposes of, 57-58
three-act dramatic narrative structure
antagonism (Act II), 52
climaxes, 53
exposition (Act I), 52
Field, Syd, 51-52
first plot point, 52
Foundations of Screenwriting, The, 51-52
hooks, 52
inciting incidents, 52
resolution (Act III), 52
second plot point, 52
Star Wars: A New Hope as an example of, 51-52
Dungeons & Dragons, 846
dynamic dramatics, 75-77
Emergent Narrative (FDD framework), 27

interactive versus linear narrative, 55
progression tables, 46
tips for running good campaigns, 846-847
XP (Experience Points) and cumulative outcomes (dynamic mechanics), 69

duplicating weapon data (game balance), 180-181

Dynamic elements (FDD framework), 24, 26-27
Emergence, 26-27
Emergent Narrative, 27
playtesting, 27

Dynamic Layer (Layered Tetrad), 33-34, 37, 61, 77
aesthetics, 70
defining, 34
environmental aesthetics, 70, 73-74
procedural aesthetics, 70-73
designer responsibilities, 36-37
dramatics, 75
emergent narrative, 76-77
interactive narrative, 75-76
emergence
Bartok, 63
mechanics and unexpected emergence, 63-64
mechanics
defining, 34, 64
house rules, 66-67
meaningful play, 64-65
outcomes, 69
player intent, 67-68
players, types of, 67-68
procedures, 64
strategy, 65-66
narrative, defining, 34
player's role in game design, 62-63
technology, 34, 77

E

editing formulas (Calc spreadsheets), 164

education
Games Education Programs, 215-217
educational software discounts, 855

Electronic Arts
Crazy Cakes, ADL (Automated Data Logging) and playtesting, 151
Majestic, boundary mechanics, 44

Elemental Tetrad framework (ludology), 20, 27-29
aesthetics, defining, 27-28
mechanics, defining, 27
story's role in, 28
technology's role in, 28

Elite Beat Agents, music in, 71

emergence
Dynamic Layer
Bartok, 63
mechanics and unexpected emergence, 63-64
FDD framework, 26-27

Emergent Narrative (FDD framework), 27, 76-77

emotion, evoking
dynamic dramatics, *Planetfall*, 76
inscribed dramatics, 57

empathetic characters versus avatars (interactive versus linear narrative)
actions (players), tracking and reacting to, 57
multiple dialogue choices, 56-57
role fulfillment, 56
silent protagonists, 56

employment in the digital game industry, 217
following up, 218-219
interviewing, 219-220
networking, 217-218
salaries, 220

empowerment as a player-centric goal of design, 116-118

Enemy Class sample project, project setup, 381

engaging play as a player-centric goal of game design, 110

enjoyable play as a player-centric goal of game design, 110

enum (C# coding), 501, 803-804

enumerating and probability, 166

environmental aesthetics (Dynamic Layer), 70, 73
  auditory play environments, 73-74
    noisy environments, 74
    volume control, 74
  player considerations, 74
    colorblindness, 74
    epilepsy, 74
    headaches, 74
    migraines, 74
  visual play environments, 73
    brightness, 73
    resolution (screen), 73
environments (procedural), 73
epilepsy, player considerations (environmental aesthetics), 74
equipment (required), paper prototyping and gameplay development, 132
Ernst, James, on touch as an Inscribed Layer aesthetic, 47
estimated hours versus real hours (burndown charts), 413
Eve Online, player-built external tools as example of cultural technology, 84-85
Evil Hat Productions and *FATE system*, 846
  interactive versus linear narrative, 55
  tips for running good campaigns, 846-847
expansion phase (brainstorming/ideation), 98-99
experience
  personal experience/development, designing games for, 109
  shared experiences, developing relationships through (dynamic dramatics), 76
  space and experience (Inscribed Layer mechanics), 46
  XP (Experience Points) and cumulative outcomes (dynamic mechanics), 69
experiential understanding as a player-centric goal of design, 122-123
Explorer player type (spades), 67
exposition (Act I)
  five-act dramatic narrative structures, 50-51
  three-act dramatic narrative structure, 52
expressing yourself through game design (designer-centric goals), 108
external tools (player-built) and cultural technology, 84-85
extrapolation (linear), 835-837

F

Fable, limited possibilities and interactive or linear narrative, 53-54
Facade, autotelic empowerment as a player-centric goal of design, 117
Factory software design pattern, 816
falling action (Act IV), five-act dramatic narrative structures, 51
Fallout 3, side quests, 54
fame as a goal of game design, 107
fan art (cultural aesthetics), 82
fan fiction as cultural narrative, 83
Farmville
  lusory attitude and, 111-112
  spoilage mechanic, 112
Farscape, foreshadowing and interactive narrative, 55
fastest path to testing, determining (analysis phase of iterative design), 92
FATE system, 846
  interactive versus linear narrative, 55
  tips for running good campaigns, 846-847
FDD framework (ludology), 20, 24, 29
  defining, 24
  Dramatic elements, 24-26
    characters, 26
    premises, 25
    purposes of, 26
    stories, 26
  Dynamic elements, 24, 26-27
    Emergence, 26-27
    Emergent Narrative, 27
    playtesting, 27
  Formal elements, 24-25
    boundaries, 25

objectives, 25
outcomes, 25
player interaction patterns, 24
procedures, 25
resources, 25
rules, 25
feature lists/product backlogs (Scrum software development methodologies), 408
feedback (positive/negative), game balance, 176-177
feel, designing for, 9-10
Fellowship of the Ring, emergent narrative example, 77
Field, Syd
Foundations of Screenwriting, The, 51-52
three-act dramatic structure, 51-52
fiero, the concept of (player-centric game design), 110
Final Fantasy III (U.S. release), NPC development, 55
Final Fantasy VI (Japanese release), NPC development, 55
Final Fantasy VII
empathetic characters versus avatars, 56
final outcomes (dynamic mechanics), example of, 69
limited possibilities and interactive or linear narrative, 53-54
Final Fantasy X, limited possibilities and interactive or linear narrative, 53-54
final outcomes (dynamic mechanics), 69
firing weapons, gameplay development (paper prototyping), 134-135, 137-138
first plot point (three-act dramatic narrative structure), 52
five-act dramatic narrative structure
climax (Act III), 51
denouement (Act V), 51
exposition (Act I), 50-51
falling action (Act IV), 51
rising action (Act II), 51
float variables (C# coding), 284
Flocks, Herds, and Schools: A Distributed Behavior Model, OOP (Object-Oriented Programming), 393
FLOOR formula (Calc), weighted probabilities, 174-175
flow
Flow: The Psychology of Optimal Experience, 114, 116-117
flow as a player-centric goal of design, 113-115
space and flow (Inscribed Layer vmechanics), 45
flowcharts (GameObjects)
Apple Picker game analysis, 232-234
frames as, 232
Flower, music in, 71
Flurry Analytics, freemium games, 214
focus testing, 152
follow cameras, *Mission Demolition*, 462-466
font-related resources, 855
for loops (C# coding), 316, 320-322, 342
force-quitting applications, 317-318
OS X, 371
Windows, 371-372
foreach loops (C# coding), 316, 322, 335
foreshadowing and interactive narrative, 55
Formal, Dramatic, and Dynamic (FDD) Framework, 24-27
Formal elements (FDD framework), 24-25
boundaries, 25
objectives, 25
outcomes, 25
player interaction patterns, 24
procedures, 25
resources, 25
rules, 25
formal group playtesting method, 146-147
formal individual playtesting method, 147-148
labs, 148-149
running playtests, 149
formulas (Calc spreadsheets)
COUNTIF formula, weighted probabilities, 174-175
entering into cells, 158-159
exiting formula editing, 164
FLOOR formula, weighted probabilities, 174-175

INDEX formula, weighted probabilities, 174-175
MOD formula, 160-161
RAND formula, weighted probabilities, 174-175
REPT formula, overall damage, showing (weapons and game balance), 181
ROUND formula, overall damage, showing (weapons and game balance), 181
SUM formula, 163
overall damage, showing (weapons and game balance), 181
fortune as a goal of game design, 107
Foundations of Screenwriting, The, and the three-act dramatic structure, 51-52
FPS (First-Person Shooter) games
QuickSnap, 695
adding sound/visual effects to shots, 723-725
aspect ratios, 697
build settings, 696
camera setup, 702-705
comparing shots, 717-721
customizing setup, 725-726
debugging, 713-714
deleting shots, 714-715
displaying player progress, 721-723
first-person controllers, 697-698
GUI elements, 703-705
importing Unity asset packages, 696-697
layers, 703-705
lightmapping, 698-701
maximizing target window, 716-717
project setup, 696
quality settings, 701-702
recording player progress, 721-723
replacing shots, 715-716
setting up, 706-707
storing shots, 707-712
fractions and probability, 165-166
frames
defining, 232
flowcharts (GameObjects), 232
free will versus plot (interactive versus linear narratives), 53-55
freemium games, 214
free-to-play games, 214
Frequency, music in, 70-71
Freytag, Gustav
five-act dramatic structure, 50-51
Technique of Dramas, The, 50-51
friends as playtesters, 144
fulfilling play as a player-centric goal of game design, 110
Fullerton, Tracy
dynamic mechanics, 64
FDD framework, 20, 24, 29
boundaries, 25
characters, 26
Dramatic elements, 24-26
Dynamic elements, 24, 26-27
Emergence, 26-27
Emergent Narrative, 27
Formal elements, 24-25
objectives, 25
outcomes, 25
player interaction patterns, 24
playtesting, 27
premises, 25-26
procedures, 25
resources, 25
rules, 25
stories, 26
Game Design Workshop, 10, 13, 20, 24, 64
player interaction patterns, 43
games, defining, 10, 13
inscribed mechanics, 40
player interaction patterns, 43
fun in game design, the importance of (player-centric game design goals), 109-110
function delegates (C# coding), 525-531, 805-807
Function Wizard (Calc)
MOD formula, 160-161
rows, creating, 160-161
functions (C# coding), 257-258, 349-350, 362
defining, 350-353

naming, 286, 356
overloading, 358
parameters
arguments and parameters, 353-354
optional parameters, 359
params keyword, 359-361
reasons for using, 356-357
recursive functions, 361-362
returning
values, 354-356
void,, 355-356
static functions, 288

G

game balance, 155, 183
defining, 156
Calc, installing, 156-157
dice probability, determining with Calc, 157-158
adjusting column widths, 160
charting results, 164
counting die roll sums, 163-164
creating Die A row, 160-161
creating Die B row, 161-162
creating probability distribution charts, 164
creating rows, 159
entering data in cells, 158-159
labeling rows, 162-163
spreadsheets, creating, 158
summing dice role results, 163
Unity example, 825-829
permutations, 175-176
positive/negative feedback, 176-177
randomization
custom card decks, 172
dice, 170
playing cards, 171-173
shuffling decks, 172-173
spinners, 170
Ten Rules of Probability Every Game Designer Should Know, 165-169
weapons, balancing, 177-178
average damage, 179-180
Chapter 9 prototype example, 182-183
determining percent chance for each bullet, 178
duplicating weapon data, 180-181
overall damage, showing, 181
rebalancing weapons, 177-178
weighted distributions, 173-175
weighted probabilities, 174-175
Game Design Theory: A New Philosophy for Understanding Games, 11
gameplay as art (cultural aesthetics), 83
performative empowerment as a player-centric goal of design, 118
Game Design Workshop, 10, 13, 18, 20, 24, 43
Game Feel: A Game Designer's Guide to Virtual Sensation, digital prototypes, 420
Game Game, The, 15-16
game industry (digital), 211-212, 222
changes to, 213
costs of AAA (top) game development, 213
freemium games, 214
indie games, 214
working conditions, 213
Entertainment Software Association Essential Facts list, 212
Games Education Programs, 215-217
getting into (careers in), 217
following up, 218-219
interviewing, 219-220
networking, 217-218
salaries, 220
projects
joining, 221
starting, 221-222
game mods
cultural mechanics, 81-82
narrative game mods, 83
Game Over screen, *Apple Picker*, 448
Game pane (Unity), 251
game play analysis (playtesting) and *Bartok*
questions, asking, 7
rounds, comparing, 9
GameCareerGuide.com, game developer salary surveys, 220
GameObjects, 281, 293-294
Apple Picker, 230-231

Apple GameObject, 424-425
AppleTree GameObject, 421-423
Basket GameObject, 425
GameObject action lists, 231-232
GameObject flowcharts, 232-234
Box Collider component, 272, 295
Capsule Collider component, 295
class instances as GameObject components (C# coding), 385-387
Collider components, 272, 295-296
creating, 271-272
editing, 272, 277-278
Mesh Collider component, 295
Mesh Filter component, 272, 295
Mesh Renderer component, 272
Mission Demolition
adding levels, 479-485
C# coding, 456-485
camera setup, 451-453
castles, 471-473
clouds, 467-471
directional light GameObject, 451
follow cameras, 462-466
game logic, 479-485
ground GameObject, 451
hitting the goal, 478-479
instantiating projectiles, 458-462
projectile component, 455-456
projectile trails, 456-462
returning shots, 473-474
slingshot GameObject, 453-455
slingshots, 456-462
vection/sense of speed, 466-471
Omega Mage
EnemyBug GameObjects, 770-780
EnemySpiker GameObjects, 780-782
Mage GameObject (player character), 735-737
playing, 272-273
prefabs, 273-277
Renderer component, 295
RigidBody component, 272, 296
scripts as GameObject components, 296-297
Space SHMUP
adding elements, 559-560
adding enemies, 543-556
art assets, enemies, 504-506
aspect ratios, 490
boosting weapons, 531-538
camera bounds, 498-500
camera setup, 490
damaging players, 513-518
directional light GameObject, 490
enemies dropping power-ups, 541-543
enemy attack/damage, 513-518
enemy GameObjects, 504-506
enemy scripts, 506-509, 543-556
function delegate for firing, 525-531
hero shield, 493-495
hero ship, 491-493
hero ship bounds, 495-498
layers, 510-513
power-ups, 531-538, 541-543
restarting games, 518-519
shooting, 519-531
spawning enemies, 509-510
tags, 511-513
testing overlapping bounds, 500-504
weapon definitions, 521-525
Sphere Collider component, 295
Transform component, 272, 294-295
gameplay as art (cultural aesthetics), 83
gameplay development (paper prototyping), 132
control points, capturing, 137
counterattacks, 135
cover, 136
health
health packs, 136
starting health, 136
interception fire, 137
movement systems, 134, 138
objectives, 132
required equipment, 132
setup, 133
visibility, 133-134, 136
weapons/firing, 134-135, 137-138
games, defining
Bateman, Chris, 15
Burgun, Keith, 11, 14-15

evolution of definitions, 16-17
Fullerton, Tracy, 10, 13
importance of, 15-16
importance of definitions, 17-18
IndieCade, 17
Meier, Sid, 10, 13
Midgley, Mary, 15-16
nature of definitions, 16-17
Pearce, Celia, 17
Roberts, Sam, 17
Schell, Jesse, 11, 13-14
Suits, Bernard, 10-15
Wittgenstein, Ludwig, 15
Games Education Programs, 215-217
gametesting. *See* playtesting
Gamma, Erich
Design Patterns, 769-770, 815-817
software design, 815
Factory design pattern, 816
Singleton design pattern, 815-816
Strategy design pattern, 816-817
spawning enemies/enemy factories, 769-770
Garfield, Richard
innovation and the design process, 97-98
RoboRally, 97-98
Georgia Institute of Technology, interactive narrative example, 75
gizmos (Unity), 277
GM (Game Master) strategies, paper prototyping, 137
goals, indirectly guiding players by, 200
goals of design, 105-106, 124
designer-centric goals, 106
community, 107-108
fame, 107
fortune, 107
greater good, 108-109
personal development/experience, 109
personal expression/communication, 108
player-centric goals, 106
attention and involvement, 118-120
empowerment, 116-118
experiential understanding, 122-123
fiero, the concept of, 110
flow, 113-115
fun, 109-110
interesting decisions, 120-122
lusory attitude, 110-112
structured conflict, 115-116
God of War, flow as a player-centric goal of design, 115
gold phase (project development process), 103
Goldstone, Will, and Unity-related websites, 853
Grand Theft Auto, resource mechanics, 45
Grand Theft Auto V, player guidance, 199
graphics (frames)
defining, 232
flowcharts (GameObjects), 232
Grasshopper, The, 10-11
lusory attitude as a player-centric goal of design, 110-111
Suits, Bernard, 15
greater good, designing games for the, 108-109
Greater Than comparison operator (>) in C# coding, 306
Greater Than or Equal To comparison operator (>=) in C# coding, 306-307
Green, Ryan, and *That Dragon, Cancer* as a personal expression/communication as a goal of game design, 108
Gregory and *Tic-Tac-Toe*, Kia, 65
Groundhog Day, experiential understanding as a player-centric goal of design, 122-123
group playtesting methods, 146-147
GUI (Graphical User Interfaces)
Apple Picker, 440-448
paper prototyping, 139
guides, fan-made game guides (cultural technology), 84-85
guiding players, 210
direct guidance, 199
Assassin's Creed IV: Black Flag, 199
calls to action, 199
Grand Theft Auto V, 199
instructions, 199
Kya: Dark Legacy, 198
Legend of Zelda: Ocarina of Time, 199
maps/guidance systems, 199
quality of guidance, determining, 198
Skyrim, 198

Valkyrie Chronicles, 198
indirect guidance, 200
arrows, 202-203
Assassin's Creed, 205
audio design, 204-205
brightness, 204
camera, 203
color, 204
constraints, 200
contrast, 204
directionality, 204
Disneyland, 202
goals, 200
Guitar Hero, 200-201
Journey, 201-202, 206
Kya: Dark Legacy, 205
landmarks, 201-202
light, 201-202
Minecraft, 200
NPC, 205-206
physical interface, 200-201
player avatars, 205
Rock Band, 200-201
similarities, 201
texture, 204
trails, 201
Uncharted 3, 203
visual design, 201-204
integration, 209
quality of guidance, determining, 197-198
teaching new skills/concepts, 207-209
Guitar Hero, player guidance, 200-201

H

HAL Laboratories, *Kirby* and integrated actions (meaningful play), 64-65
Half-Life
game mods and cultural mechanics, 81-82
narrative premises, examples of, 49
Halo
inscribed dramatics, example of, 58
limited possibilities and interactive or linear narrative, 53-54
prior art, researching (analysis phase of iterative design), 92
Red vs. Blue, machinima example, 83-84
Hamlet on the Holodeck, interactive narrative example, 75
headaches, player considerations (environmental aesthetics), 74
health (avatar/character) in gameplay development (paper prototyping)
health packs, 136
starting health, 136
hearing (Inscribed Layer aesthetics), 47
Heart of Darkness, The, as example of inscribed dramatics, 57-58
Heinsoo, Rob, on XP (Experience Points) and cumulative outcomes (dynamic mechanics), 69
Hello World program, 263, 279
arrays in, 344-347
C# scripts, creating, 266-271
debugging C# coding, 364
GameObjects
creating, 271-272
editing, 272
playing, 272-273
gizmos, 277
lists in, 344-347
naming projects, 266
prefabs, 273-277
scene views, changing, 277-278
standard project setup procedures, 264
Helm, Richard
Design Patterns, 769-770, 815-817
software design, 815
Factory design pattern, 816
Singleton design pattern, 815-816
Strategy design pattern, 816-817
spawning enemies/enemy factories, 769-770
Hierarchy pane (Unity), 251
Hitchhiker's Guide to the Galaxy, The, and autotelic empowerment as a player-centric goal of design, 117
Homo Ludens, magic circle concept and lusory attitude, 112

hooks (three-act dramatic narrative structure), 52
hours (estimated versus real) and burndown charts, 413
house rules (dynamic mechanics), 66-67
Hoye, Mike
  Legend of Zelda: The Wind Waker, The, 83
  narrative game mods, 83
HRS (Horizontal Re-Sequencing), procedural music, 70
Huizinga, Johan
  Homo Ludens, 112
  magic circle concept and lusory attitude, 112
Hunicke and MDA framework, Robin, 20, 29
  aesthetics-based design, 21-24
  defining, 20-21
  designer views of games, 21
  player views of games, 21

I

ideation (brainstorming), 98
  collection phase, 99
  collision phase, 100-101
  discussions, 101
  expansion phase, 98-99
  rating phase, 101
if statements (C# coding), 307-310
Ilinx (vertigionous play)
  Imaginary Games, 110
  Jeux et Le Hommes, Le, and the four different kinds of play, 110
Imaginary Games, 15
  Ilinx (vertigionous play), 110
  structured conflict as a player-centric goal of design, 115-116
immediacy of objectives (Inscribed layer mechanics), 41
immediacy of sound (Inscribed Layer aesthetics), 47
immediate outcomes (dynamic mechanics), 69
implementation phase (iterative design), 90-91, 94-97
importance of objectives (Inscribed layer mechanics), 42
iMUSE (Interactive MUsic Streaming Engine), 70
inciting incidents (three-act dramatic narrative structure), 52
increasing difficulty, *Apple Picker*, 448
Incredible Machine, The, 188
incremental innovation, 97
INDEX formula (Calc), weighted probabilities, 174-175
indie games and the digital game industry, 214
IndieCade Game Festival
  games, defining, 17
  indie games and the digital game industry, 214
  scoping and the design process, 104
indirectly guiding players, 200
  arrows, 202-203
  Assassin's Creed, 205
  audio design, 204-205
  brightness, 204
  camera, 203
  color, 204
  constraints, 200
  contrast, 204
  directionality, 204
  Disneyland, 202
  goals, 200
  Guitar Hero, 200-201
  Journey, 201-202, 206
  Kya: Dark Legacy, 205
  landmarks, 201-202
  light, 201-202
  Minecraft, 200
  NPC, 205-206
  physical interface, 200-201
  player avatars, 205
  Rock Band, 200-201
  similarities, 201
  texture, 204
  trails, 201
  Uncharted 3, 203

visual design, 201-204
individual playtesting methods
formal method, 147-148
labs, 148-149
running playtests, 149
informal method, 146
infinite loops (C# coding), 317-318
Influence: The Psychology of Persuasion, 4
Infocom
Hitchhiker's Guide to the Galaxy, The, and autotelic empowerment as a player-centric goal of design, 117
Planetfall, developing relationships through shared experience, 76
informal individual playtesting method, 146
information, conveying (Inscribed Layer aesthetics), 48-49
inheritance in classes (C# coding), 387
innovation and the design process, 97-98
Inscribed Layer (Layered Tetrad), 32, 37, 39, 59
aesthetics, 46
defining, 33
five aesthetic senses, 47-48
goals of aesthetic design, 48-49
hearing, 47
immediacy of sound (Inscribed Layer aesthetics), 47
information, conveying, 48-49
mood, conveying, 48
smell, 48
touch, 47-48
vision, 47
designer responsibilities, 36
mechanics, 40
boundaries, 40, 44
defining, 32
objectives, 40-42
player relationships, 40, 42-44
resources, 40, 45
rules, 40, 44
spaces, 40, 45-46
tables, 40, 46
narrative, 49
characters, 49
components of, 49-50
defining, 33
five-act dramatic structure, 50-51
inscribed dramatics, purposes of, 57-58
interactive versus linear narrative, 53-57
linear versus interactive narrative, 53-57
plots, 50
premises, 49
settings, 49
three-act dramatic structure, 51-52
technology, 58
defining, 33
paper game technologies, 58-59
Inspector pane (Unity), 251
editing GameObjects, 272
GameObjects, components of, 294
installing
Calc, 156-157
Unity, 241
instance variables (C# coding), 289
instructions
directly guiding players by, 199
simple instructions exercise (systems thinking), 226-227
int variables (C# coding), 284
integration
integrated actions (meaningful play), 64-65, 121
teaching players by, 209
interactive narrative
characters, 49
components of, 49-50
dynamic dramatics, 75
developing relationships through shared experience, 76
interactive fiction and player experience, 75-76
five-act dramatic structure, 50-51
linear narrative versus, 53
empathetic characters versus avatars, 55-57
foreshadowing, 55
limited possibilities, 53-54
NPC development, 55
pen-and-paper RPG, 55
plot versus free will, 53-55
side quests, 54

plots, 50
premises, 49
settings, 49
three-act dramatic structure, 51-52
interception fire, gameplay development (paper prototyping), 137
interest as a player-centric goal of design, 118-120
interesting decisions as a player-centric goal of design, 120-122
interesting polling (playtesting), 152
interfaces (C# coding), 769-770, 782-785, 807-810
Internet's role in playtesting, 145-146, 150
closed playtesting, 150
limited playtesting, 150
open playtesting, 150-151
interpolation, 831
interpolating values other than position, 834-835
linear interpolation, 831
easing, 837-841
time-based linear interpolation, 831-832
Zeno's Dichotomy Paradox and linear interpolation, 833-834
intersectional innovation, 97, 99
interviewing, finding a career in the digital game industry, 219-220
investigators (playtesting), defining, 142
involvement and attention as player-centric goals of design, 118-120
Is Equal To comparison operator (==), 304
iterative design, 90
analysis phase, 90-91
determining audience, 91
determining fastest path to testing, 92
determining resources, 91-92
importance of repetition, 96-97
researching prior art, 92
design phase, 90
importance of listening, 92-94
importance of repetition, 96-97
implementation phase, 90-91, 94-97
repetition, importance of, 96-97
testing phase, 91, 95-97

J

jagged arrays (C# coding), 340-342
jagged lists (C# coding), 342-344
JavaScript, 238-239
Jenkins on transmedia, Henry, 86
Jeux et Le Hommes, Le, and the four different kinds of play, 110
jobs in the digital game industry, 217
following up, 218-219
interviewing, 219-220
networking, 217-218
salaries, 220
Johansson, Frans
innovation and the design process, 97
Medici Effect, The, 97
Johnson, Ralph
Design Patterns, 769-770, 815-817
software design, 815
Factory design pattern, 816
Singleton design pattern, 815-816
Strategy design pattern, 816-817
spawning enemies/enemy factories, 769-770
joining game projects in development, 221
Journey, 41
player guidance, 201-202, 206
tissue playtesters, 144-145
jump statements in loops (C# coding), 322-324
justification (inscribed dynamics), 57-58

K

Kaboom!, game analysis (systems thinking), 229
GameObject action lists, 231-232
GameObject flowcharts, 232-234
GameObjects, 230-231
gameplay, 230
Kaplan, Larry, on *Kaboom!* game analysis (systems thinking), 229-234

Killer player type (clubs), 67
Kim, Scott, and puzzle design
- action puzzles, 188
- *Art of Puzzle Design, The*, 186
  - *eight steps of puzzle design, 191-192*
  - *genres of puzzles, 187-188*
  - *goals of effective design, 192-193*
  - *reasons for playing puzzles, 189*
  - *required modes of thought in solving, 189*
- bridge puzzles, 188
- chess puzzles, 188
- construction puzzles, 188
- crossword puzzles, 188
- eight steps of puzzle design, 191-192
- genres of puzzles, 187-188
- goals of effective design
- puzzles, defining, 186-187
- reasons for playing puzzles, 189
- required modes of thought in solving, 189-190
- story puzzles, 188
- strategy puzzles, 188
- *Sudoku*, 188

Kirby, integrated actions (meaningful play), 64-65
Klei Entertainment, *Mark of the Ninja* and intent versus execution, 187
Kya: Dark Legacy and player guidance, 198, 205

L

lab setups for formal individual playtesting, 148-149
labeling rows (Calc spreadsheets), 162-163
landmarks
- indirectly guiding players by, 201-202
- space and landmarks (Inscribed Layer mechanics), 45-46

languages (computer) and systems thinking, 227
Larson, Josh, and *That Dragon, Cancer* as a personal expression/communication as a goal of game design, 108
Last of Us, The, and screenresolution, 73
law and cultural impact of games, 86-87
layer masks and bitwise Boolean operators (C# coding), 801-802
Layered Tetrad
- Cultural Layer, 34-35, 37-38, 79, 87
  - *aesthetics, 35, 82-83*
  - *cultural impact of games, 86-87*
  - *defining, 80-81*
  - *designer responsibilities, 37*
  - *law and cultural impact of games, 86-87*
  - *mechanics, 35, 81-82*
  - *narrative, 36, 83-84*
  - *technology, 36, 84-85*
  - *transmedia, 85-86*
- designer responsibilities
  - *Cultural Layer, 37*
  - *Dynamic Layer, 36-37*
  - *Inscribed Layer, 36*
- Dynamic Layer, 33-34, 37, 61, 77
  - *aesthetics, 34*
  - *designer responsibilities, 36-37*
  - *dramatics, 75-77*
  - *emergence, 63-64*
  - *mechanics, 34, 64-69*
  - *narrative, 34*
  - *player's role in game design, 62-63*
  - *technology, 34, 77*
- Inscribed Layer, 32, 37, 39, 46-49, 59
  - *aesthetics, 33*
  - *boundaries, 40, 44*
  - *designer responsibilities, 36*
  - *mechanics, 32, 40-46*
  - *narrative, 33*
  - *objectives, 40-42*
  - *player relationships, 40, 42-44*
  - *resources, 40, 45*
  - *rules, 40, 44*
  - *spaces, 40, 45-46*
  - *tables, 40, 46*
  - *technology, 33*
- introduction to, 31-32

Lazzaro, Nicole, and the concept of fiero (player-centric game design), 110
Learn section (Unity website), 852
LeBlanc, Mark, and MDA framework, 20, 29
- aesthetics-based design, 21-24

defining, 20-21
designer views of games, 21
player views of games, 21
Legend of Zelda: Ocarina of Time, The, and player guidance, 199
Legend of Zelda, The
resource mechanics, 45
silent protagonists (empathetic characters versus avatars), 56
Legend of Zelda: The Wind Waker, The
inscribed dramatics, example of, 57-58
narrative game mods, 83
Legend of Zelda: Twilight Princess, The, andtouch as an Inscribed Layer aesthetic, 48
LeMarchand, Richard
attention and involvement as player-centric goals of design, 118-120
engaging play as a player-centric goal of game design, 110
Less Than comparison operator (<) in C# coding, 306
Less Than or Equal To comparison operator (<=) in C# coding, 306-307
Level Up! The Guide to Great Video Game Design, 64, 202
levels (custom) and cultural mechanics, 82
licensing Unity, 241-242
light, indirectly guiding players by, 201-202
limited playtesting (beta tests), 150
limited possibilities and interactive narrative, 53-54
Linderman, Frank Bird,
Chief Plenty-Coups and coup-counting as a game, 115-116
structured conflict as a player-centric goal of design, 115-116
linear extrapolation, 835-837
linear interpolation, 831
easing, 837-841
time-based linear interpolation, 831-832
Zeno's Dichotomy Paradox and linear interpolation, 833-834
linear narrative
characters, 49
components of, 49-50
five-act dramatic structure, 50-51
interactive narrative versus linear narrative, 53
empathetic characters versus avatars, 55-57
foreshadowing, 55
limited possibilities, 53-54
NPC development, 55
pen-and-paper RPG, 55
plot versus free will, 53-55
side quests, 54
plots, 50
premises, 49
settings, 49
three-act dramatic structure, 51-52
LineRenderer (Unity), ground spells in *Omega Mage*, 756-761
LINQ (Language Integrated Query) and C# coding, 640-641
Lionhead Studios, *Fable* and the limited possibilities of interactive or linear narrative, 53-54
listening, importance of during design phase (iterative design), 92-94
lists (C# coding), 328-331, 344
arrays versus lists, 344
converting arrays to lists, 336-337
converting lists to arrays, 331
jagged lists, 342-344
methods, 331
properties, 331
zero indexed lists, 331
long-term objectives, space and (Inscribed Layer mechanics), 46
loops (C# coding), 315, 325
do.while loops, 316, 319-320
for loops, 316, 320-322, 342
foreach loops, 316, 322
empty array elements and foreach loops, 335
infinite loops, dangers of, 317-318
jump statements in loops, 322-324
modulus operator (%), 324
project setup, 300-301

types of, 316
while loops, 316-319
Lucas, George
scoping and the design process, 104
Star Wars: A New Hope as an example of three-act dramatic narrative structure, 51-52
LucasArts
iMUSE (Interactive MUsic Streaming Engine), 70
X-Wing
information, conveying (Inscribed Layer aesthetics), 48-49
music in, 70
ludology, 19. *See also* Layered Tetrad
Elemental Tetrad framework, 20, 27-29
aesthetics, 27-28
mechanics, 27
story's role in, 28
technology's role in, 28
FDD framework, 20, 24, 29
boundaries, 25
characters, 26
Dramatic elements, 24, 25-26
Dynamic elements, 24, 26-27
Emergence, 26-27
Emergent Narrative, 27
Formal elements, 24-25
objectives, 25
outcomes, 25
player interaction patterns, 24
playtesting, 27
premises, 25-26
procedures, 25
resources, 25
rules, 25
stories, 26
MDA framework, 20, 29
aesthetics-based design, 21-24
defining, 20-21
designer views of games, 21
player views of games, 21
Ludwig, Manfred
optimal strategy, determining (dynamic mechanics), 65-66
Up The River, 65-66
Lunch Money, house rules (dynamic mechanics), 67
lusory attitude, 13-14, 110-112

M

machinima, 83-84
magic circle concept and lusory attitude, 112
Magic:The Gathering, illegal cards and emergence (Dynamic Layer), 64
Mainichi
experiential understanding as a player-centric goal of design, 122-123
personal expression/communication as a goal of game design, 108
Majestic, 17, 44
Mangle of Play, The, and the cultural aspects of game development, 80-81
Manifesto for Agile Software Development, 406-407
maps/guidance systems, directly guiding players by, 199
Marathon, researching prior art (analysis phase of iterative design), 92
Mario Kart and game balance, 177
Mark of the Ninja, intent versus execution, 187
Mass Effect
multiple dialogue choices (empathetic characters versus avatars), 57
player interaction patterns, 43
resolution (screen), 73
Mateas, Michael
autotelic empowerment as a player-centric goal of design, 117
Façade, 117
math and game balance, 155, 183
balance, defining, 156

Calc, installing, 156-157
dice probability, determining with Calc, 157-158
adjusting column widths, 160
charting results, 164
counting die roll sums, 163-164
creating Die A row, 160-161
creating Die B row, 161-162
creating probability distribution charts, 164
creating rows, 159
entering data in cells, 158-159
labeling rows, 162-163
spreadsheets, creating, 158
summing dice role results, 163
Unity example, 825-829
permutations, 175-176
positive/negative feedback, 176-177
randomization
custom card decks, 172
dice, 170
playing cards, 171-173
shuffling decks, 172-173
spinners, 170
Ten Rules of Probability Every Game Designer Should Know, 165-169
weapons, balancing, 177-178
average damage, 179-180
Chapter 9 prototype example, 182-183
duplicating weapon data, 180-181
overall damage, showing, 181
percent chance for each bullet, determining, 178
rebalancing weapons, 177-178
weighted distributions, 173-175
weighted probabilities, 174-175
math concepts
Bézier curves, 841-845
dice probability, Unity example, 825-829
dot product, 829-830
extrapolation (linear), 835-837
interpolation
easing linear interpolation, 837-841
interpolating values other than position, 834-835
linear interpolation, 831
time-based linear interpolation, 831-832
Zeno's Dichotomy Paradox and linear interpolation, 833-834
recursive Bézier curves, 844-845
sine/cosine, 822-825
mathf variables (C# coding), 292
MDA framework (ludology), 20, 29
aesthetics-based design, 21-24
defining, 20-21
designer views of games, 21
player views of games, 21
meaningful play
dynamic mechanics, 64-65
dynamic procedures
discernable actions, 121
integrated actions, 121
interesting decisions as a player-centric goal of design, 120-121
mechanics
attention and involvement as a player-centric goal of design, 120
Cultural Layer (Layered Tetrad), 81
custom game levels, 82
defining, 35
game mods, 81-82
Dynamic Layer (Layered Tetrad)
defining, 34, 64
defining strategies, 65
designing for strategies, 66
determining optimal strategy, 65-66
determining player intent, 67-68
emergence (unexpected), 63-64
house rules, 66-67
meaningful play, 64-65
outcomes, 69
player types, 67-68
procedures, 64
Elemental Tetrad framework, 27
Inscribed Layer (Layered Tetrad), 40
boundaries, 40, 44
defining, 32
objectives, 40-42
player relationships, 40, 42-44
resources, 40, 45

rules, 40, 44
spaces, 40, 45-46
tables, 40, 46
reinforcing via inscribed dramatics, 58
spoilage mechanic (*Farmville*), 112
Medici Effect, The, innovation and the design process, 97
medium-term objectives and space (Inscribed Layer mechanics), 46
meetings and Scrum software development methodologies, 409
Meier, Sid
Civilization, tables, 40-41
C.P.U. Bach, music in, 71
games, defining, 13
interesting decisions as a player-centric goal of design, 121
melee, pronunciation of, 42
Meretzky, Steven
developing relationships through shared experience, 76
Planetfall, 76
Mesh Collider component (GameObjects), 295
Mesh Filter component (GameObjects), 272, 295
Mesh Renderer component (GameObjects), 272
Microsoft controllers
axis mapping, 848-849
button mapping, 848-849
Midgley, Mary
Game Game, The, 15-16
games, defining, 15-16
migraines, player considerations (environmental aesthetics), 74
Miles, Rob, and *CSharp Yellow Book*, 854
Mimicry, *Le Jeux et Le Hommes* and the four different kinds of play, 110
Minecraft
autotelic empowerment as a player-centric goal of design, 117
player guidance, 200
player-built external tools as example of cultural technology, 84
procedural environments, 73
Minority Media, *Papa y Yo* as a personal expression/communication as a goal of game design, 108
Mission Demolition, 449, 485
art assets, 451-456
C# coding, 456
adding levels, 479-485
castles, 471-473
clouds, 467-471
follow cameras, 462-466
game logic, 479-485
hitting the goal, 478-479
instantiating projectiles, 458-462
projectile trails, 474-478
returning shots, 473-474
slingshots, 456-462
vection/sense of speed, 466-471
camera setup, 451-453
concept of, 450-451
directional light GameObject, 451
ground GameObject, 451
project setup, 450
projectile component, 455-456
sequence of events during play, 450-451
slingshot GameObject, 453-455
mission types (paper prototyping), determining, 131-133
MMORPG (Massively Multiplayer Online Role-Playing Game), player interaction patterns, 43
MOD formula (Calc), 160-161
model/animation resources, 854-855
Modern Warfare as example of inscribed dramatics, 58
modifying games (cultural mechanics), 81-83
modulus operator (%) in C# coding, 324
Mojang and *Minecraft*
autotelic empowerment as a player-centric goal of design, 117
player guidance, 200
player-built external tools as example of cultural technology, 84
procedural environments, 73

money as a goal of game design, 107
MonoDevelop editor
creating C# scripts, 266-271
debugging C# coding
attaching debugger to Unity, 372-374
stepping through code, 369-371, 373-377
variables, 375-376
Monopoly
conflicting objectives (Inscribed layer mechanics), 42
immediate outcomes (dynamic mechanics), example of, 69
resource mechanics, 45
roles of players (player relationships), defining, 43
mood, conveying (Inscribed Layer aesthetics), 48
motivation (inscribed dynamics), 57-58
mouse interaction, programming in *Omega Mage*, 737-741
movement, 741-747
moving when dragging, 746-747
tap indicators, 743-745
movement systems, developing, 127-128, 138
MUD (Multi-User Dungeon) games, 67-68
multidimensional arrays (C# coding), 337-340
multilateral competition (player interaction patterns), 43
multiplayer games
puzzles and, 186
roles of players (player relationships), defining, 44
multiple individual players versus game (player interaction patterns), 43
Murray, Janet
developing relationships through shared experience, 76
Hamlet on the Holodeck, 75
interactive narrative, 75
Planetfall, 76
music, 70
C.P.U. Bach, music in, 71
Elite Beat Agents, 71
Flower, music in, 71
Frequency, 70-71
HRS (Horizontal Re-Sequencing), 70
immediacy of sound (Inscribed Layer aesthetics), 47
iMUSE (Interactive MUsic Streaming Engine), 70
Osu Tatake Ouendan!, 71
PaRappa the Rapper, 70-71
PCO (Procedural Composition), procedural music, 71
VRO (Vertical Re-Orchestration), procedural music, 70-71
Myst, 188

N

Nakamura, Jeanne
autotelic empowerment as a player-centric goal of design, 116-117
Concept of Flow, 113-114, 116-117
flow as a player-centric goal of design, 113-114
naming
C# naming conventions, 286-287
projects in Unity, 266
narrative
attention and involvement as player-centric goals of design, 120
Cultural Layer (Layered Tetrad), 83
defining, 36
fan fiction as cultural narrative, 83
game mods as cultural narrative, 83
machinima, 83-84
Dynamic Layer (Layered Tetrad), defining, 34
emergent narrative, 76-77
Inscribed Layer (Layered Tetrad), 49
characters, 49
components of, 49-50
defining, 33
five-act dramatic structure, 50-51
inscribed dramatics, purposes of, 57-58
interactive versus linear narrative, 53-57
linear versus interactive narrative, 53-57
plots, 50
premises, 49

settings, 49
three-act dramatic structure, 51-52
interactive narrative (dynamic dramatics), 75
developing relationships through shared experience, 76
interactive fiction and player experience, 75-76
Naughty Dog
Uncharted
inscribed dramatics, example of, 58
limited possibilities and interactive or linear narrative, 53-54
role fulfillment (empathetic characters versus avatars), 56
Uncharted 2: Drake's Deception, machinima example, 84
Uncharted 3
particle systems (procedural visual art), 71-72
player guidance, 203
navigating Unity, 251
negative/positive feedback, game balance, 176-177
networking, finding a career in the digital game industry, 217-218
Neverwinter Nights, narrative game mods, 83
Nintendo, *Mario Kart* and game balance, 177
noisy environments (environmental aesthetics), 74
Not Equal To comparison operator (!=) in C# coding, 306
NOT operator (!) in C# coding, 300
novel decisions as part of player-centric design, 121
NPC (Non-Player Characters)
development of (interactive narratives), 55
indirectly guiding players by, 205-206

O

objectives
FDD framework, 25
gameplay development (paper prototyping), 132
Inscribed Layer (Layered Tetrad), 40-41
conflicting objectives, 42
immediacy of objectives, 41
importance of objectives, 42
long-term objectives and space, 46
medium-term objectives and space, 46
short-term objectives and space, 46
objects, C# as OOC (Object-Oriented Coding), 258-259
Okami
empathetic characters versus avatars, 56
touch as an Inscribed Layer aesthetic, 48
Omega Mage
changing rooms, 764-768
creating the game environment, 730-735
customizing setup, 789
damaging players, 777-782
enemies, 768-770
damaging, 772-777
enemy factories, 785-789
EnemyBug GameObjects, 770-780
EnemySpiker GameObjects, 780-782
interfaces, 782-785
example of play, 728-729
fire ground spell, 754-762
fire spell, 761-762
fire-and-forget spells, 762-764
ground spell, 756-761
importing Unity asset packages, 729
inventories
camera setup, 748-749
creating, 747-748
selecting elements, 749-754
Mage GameObject (player character), 735-737
mouse interaction, 737-741
moving when dragging, 746-747
tap indicators, 743-745
movement, 741-747
project setup, 729
online playtesting, 150
closed playtesting, 150
limited playtesting, 150
open playtesting, 150-151

online resources
- animation/model resources, 854-855
- art assets, 854
- audio assets, 854
- C# resources, 853-854
- educational software discounts, 855
- font-related resources, 855
- game developer salary surveys, 220
- model/animation resources, 854-855
- Unity
  - *tutorials, 852*
  - *Unity-related websites, 852-853*

OOC (Object-Oriented Coding), C# as, 258-259
OOP (Object-Oriented Programming) in C#. *See also* classes (C# coding), 391, 403
- *Boids* project, 394
  - *building simple models, 394-396*
  - *project setup, 394*
  - *scripts, 397-403*
- defining through metaphors, 392-393

open playtesting (beta tests), 150-151
OpenOffice Calc. *See* Calc
OR operator (||) in C# coding, 300
Origin Systems, *Ultima IV*, actions (players), tracking and reacting to (empathetic characters versus avatars), 57
orthographic cameras, 425-426
OS X
- force-quitting applications, 317-318, 371
- right-clicking and, 265, 849-850
- Unity, new project setup procedures, 794-796

Osu Tatake Ouendan!
- fulfilling play as a player-centric goal of game design, 110
- music in, 71

outcomes
- Dynamic Layer (Layered Tetrad), 69
- FDD framework, 25

overall damage, showing (weapons and game balance), 181
overscoping (design process), 103-104

P

Pajitnov, Alexi
- *Art of Puzzle Design, The*, 186
  - *eight steps of puzzle design, 191-192*
  - *genres of puzzles, 187-188*
  - *goals of effective design, 192-193*
  - *reasons for playing puzzles, 189*
  - *required modes of thought in solving, 189*
- *Tetris*, 186

panes (Unity), 251
- Console pane, 251
- Game pane, 251
- Hierarchy pane, 251
- Inspector pane, 251
- layouts, configuring, 246-251
- Project pane, 251
- Scene pane, 251

Papa Sangre, sound as an Inscribed Layer aesthetic, 47
Papa y Yo, personal expression/communication as a goal of game design, 108
paper games
- *Dungeons & Dragons*, 846-847
- *FATE system*, 846-847
- progression, 59
- randomization, 59
  - *custom card decks, 172*
  - *dice, 170*
  - *playing cards, 171-173*
  - *shuffling decks, 172-173*
  - *spinners, 170*
- state tracking, 59
- tips for running good campaigns, 846-847

paper prototyping, 125, 140
- benefits of, 126
- best uses for, 138-139
- example of, 129
  - *determining mission types, 131-133*
  - *gameplay development, 132-137*
  - *GM strategies, 137*
  - *initial concept development, 129-130*
  - *playtesting, 138*
  - *prototype construction, 131-133*
- movement systems, 127-128, 134, 138
- poor uses for, 139

tools for, 127-129
PaRappa the Rapper, music in, 70-71
particle systems (procedural visual art), 71-72
Passage, 11-12
Pauling, Linus, brainstorming (ideation), 98
PCO (Procedural Composition), procedural music, 71
Pearce, Celia, and game definitions, 17
pen-and-paper RPG
 Dungeons & Dragons, 846-847
 FATE system, 846-847
 progression, 59
 randomization, 59
  custom card decks, 172
  dice, 170
  playing cards, 171-173
  shuffling decks, 172-173
  spinners, 170
 state tracking, 59
 tips for running good campaigns, 846-847
performative empowerment as a player-centric goal of design, 118
permutations, 175-176
 permutations without repeating elements, 176
 repeating elements in, 176
personal development/experience, designing games for, 109
personal expression/communication as a goal of game design, 108
perspective cameras, 425-426
Philosophical Investigations, 15
physical interface, indirectly guiding players by, 200-201
physics games
 Mission Demolition, 449, 485
  adding levels, 479-485
  art assets, 451-456
  C# coding, 456-485
  camera setup, 451-453
  castles, 471-473
  clouds, 467-471
  concept of, 450-451
  directional light GameObject, 451
  follow cameras, 462-466
  game logic, 479-485
  ground GameObject, 451
  hitting the goal, 478-479
  instantiating projectiles, 458-462
  project setup, 450
  projectile component, 455-456
  projectile trails, 474-478
  returning shots, 473-474
  sequence of events during play, 450-451
  slingshot GameObject, 453-455
  slingshots, 456-462
  vection/sense of speed, 466-471
 puzzles in, 194
Planetfall
 developing relationships through shared experience, 76
 interactive narrative example, 75
player interaction patterns, 24
 Inscribed Layer (Layered Tetrad), 43
  cooperative play, 43
  multilateral competition, 43
  multiple individual players versus game, 43
  player versus player, 43
  single player versus game, 43
  team competition, 43
  unilateral competition, 43
 Mass Effect, 43
players
 avatars, indirectly guiding players by, 205
 Bartle's four players who suit MUDs, 67-68
 community as a goal of game design, 107-108, 120
 environmental aesthetics (Dynamic Layer), 74
  colorblindness, 74
  epilepsy, 74
  headaches, 74
  migraines, 74
  volume control, 74
 external tools (player-built) and cultural technology, 84-85
 gameplay development (paper prototyping), 132-137
 guiding, 197-198, 210
  direct guidance, 198-199

indirect guidance, 200-206
teaching new skills/concepts, 207-209
intent, determining (dynamic mechanics), 67-68
interaction patterns (FDD framework), 24, 43
movement systems and gameplay development (paper prototyping), 134, 138
player versus player (player interaction patterns), 43
player-centric goals of design, 106
attention and involvement, 118-120
empowerment, 116-118
experiential understanding, 122-123
fiero, the concept of, 110
flow, 113-115
fun, 109-110
interesting decisions, 120-122
lusory attitude, 110-112
structured conflict, 115-116
relationships at the Inscribed Layer (Layered Tetrad), 40, 42
roles of, 43-44, 62-63
teaching new skills/concepts
integration, 209
sequencing, 207-209
types of, Bartle's four players who suit MUDs, 67-68
playing cards/card decks
Bartok, 621-622, 655
backgrounds, 622-623
build settings, 623-624
card animation, 623
creating cards, 624-629
fanning the hand, 638-640
game logic, 653-655
layouts, 629-638
LINQ (Language Integrated Query) and C# coding, 640-641
managing the initial deal, 642-643
managing turns, 646-653
moving cards, 641-642
rules of the game, 622
sorting order (2D-depth), 643-646
custom card decks, 172
Poker
game balance and, 176-177
Red Dead Redemption rule mechanics, 44
art assets, 614-615
backgrounds, 614-615
beginning/end of round announcements, 615-616
chain scoring, 605-607
clickable card functionality, 597-600
creating cards from sprites, 566-581
displaying scores to players, 607-614
draw and discard actions, 597-604
earning points, 604-605
example of play, 584-585
game logic, 597-604
giving player feedback on scores, 616-619
gold cards, 620
importing images as sprites, 564-566
importing Unity asset packages, 562
matching drawn cards, 600-604
mine tableau layout, 585-592
mobile device build settings, 562-563, 620
positioning cards on tableau, 592-594
project setup, 562
rules of the game, 583-584
scoring, 604-619
setting up sorting layers on tableau, 594-597
shuffling cards, 581-583
Prospector Solitaire, 561
randomization, 171-173
riffle shuffling, 7
shuffling, 7, 172-173
Vectorized Playing Cards 1.3
Bartok, 622
Prospector Solitaire, 562
playtesting, 141, 153
ADL (Automated Data Logging), 151
analyzing game play, *Bartok*, 7
AT (Automated Testing), 153
Bartok, 6-10
beta tests
closed playtesting, 150
limited playtesting, 150
open playtesting, 150-151

data tables, Inscribed Layer mechanics (Layered Tetrad), 46
Dynamic elements (FDD framework), 27
feel, designing for, 9-10
flukes, 8
focus testing, 152
importance of, 142
interesting polling, 152
investigators (playtesting), defining, 142
methods of testing, 146
formal group testing, 146-147
formal individual testing, 147-149
informal individual testing, 146
online playtesting, 150-151
scripts, 147
paper prototyping, 138
playtesters
acquaintances as playtesters, 145
circle of playtesters, 143-146
defining, 142
friends as, 144
Internet as, 145-146
tissue playtesters, 144-145
ways to be a great playtester, 142-143
you as, 143-144
QA (Quality Assurance) testing, 152-153
questions, asking, 7
rounds, comparing, 9
rules, modifying, 8
usability testing, 152
plots
first plot point (three-act dramatic narrative structure), 52
free will versus plot (interactive versus linear narratives), 53-55
Inscribed Layer (Layered Tetrad), 50
second plot point (three-act dramatic narrative structure), 52
Pogo.com, *Crazy Cakes* playtesting and ADL (Automated Data Logging), 151
point-and-click adventure games, loss of popularity, 117
points, adding in *Apple Picker*, 441-442
Pokemon
epilepsy, 74
transmedia example, 85
Poker
game balance and, 176-177
Red Dead Redemption rule mechanics, 44
polling (interest) and playtesting, 152
pop-ups, directly guiding players by, 199
position puzzles/sliding blocks in action games, 193
positive/negative feedback, game balance, 176-177
possible outcomes of probability, 166
post-release phase (project development process), 103
power-ups, building in *Space SHMUP*, 531-538, 541-543
practical probability versus theoretical probability, 169
prefabs (Unity), 273-277
premises
FDD framework, 25
Inscribed Layer (Layered Tetrad)
defining, 49
examples of, 49
preproduction phase (project development process), 102
Prince of Persia, limited possibilities and interactive or linear narrative, 53-54
Prince of Persia: The Sands of Time, limited possibilities and interactive or linear narrative, 53-54
prior art, researching (analysis phase of iterative design), 92
prioritizing tasks in burndown charts, 414-415
probability
adding probabilities, 166
decimals, 165-166
dice probability, determining with Calc, 157-158
adjusting column widths, 160
charting results, 164
counting die roll sums, 163-164
creating Die A row, 160-161
creating Die B row, 161-162
creating probability distribution charts, 164
creating rows, 159

creating spreadsheets, 158
entering data in cells, 158-159
labeling rows, 162-163
summing dice role results, 163
Unity example, 825-829
enumeration, 166
fractions, 165-166
multiplying probabilities, 166-167
one minus the probability, 167-168
possible outcomes, 166
practical probability versus theoretical probability, 169
range of probabilities, 166
sought outcomes, 166
Ten Rules of Probability Every Game Designer Should Know, 165-169
theoretical probability versus practical probability, 169
weighted probabilities, 174-175
probability tables, Inscribed Layer mechanics (Layered Tetrad), 46
procedural aesthetics (Dynamic Layer), 70
procedural animation, 72
procedural environments, 73
procedural music, 70
HRS (Horizontal Re-Sequencing), 70
PCO (Procedural Composition), procedural music, 71
VRO (Vertical Re-Orchestration), procedural music, 70-71
procedural visual art, 71-72
procedural music, 70
procedures
Dynamic Layer (Layered Tetrad), 64
FDD framework, 25
product backlogs/feature lists (Scrum software development methodologies), 408
production phase (project development process), 102-103
Professor Layton, *Myst*, 188
Profiler (Unity), 663-665
programming. *See also* systems thinking
C#, 253, 262
compiled language, C# as, 254-256
creating scripts, 266-271
features of (overview), 254
functions, 257-258
managed code, C# as, 256
OOC (Object-Oriented Coding), C# as, 258-259
reasons for choosing, 238-239
runtime speed, 239
strongly typed language, C# as, 256-257
syntax of, 259-260
tips for learning, 239-241
variables, 256-257
Hello World program, 263, 279
creating C# scripts, 266-271
creating GameObjects, 271-272
creating projects in Unity, 264-266
editing GameObjects, 272, 277-278
gizmos, 277
naming projects, 266
playing, 272-273
prefabs, 273-277
standard project setup procedures, 264
JavaScript, 238, 239
naming projects, 266
progression
inscribed dramatics, 58
paper game technologies, 59
progression tables, Inscribed Layer mechanics (Layered Tetrad), 46
project development process, 102
alpha phase, 103
beta phase, 103
gold phase, 103
post-release phase, 103
preproduction phase, 102
production phase, 102-103
Project pane (Unity), 251, 264-266
Project Wizard (Unity), new project setup procedures, 794-796
projects
Agile software development methodologies, 405-407, 416
Apple Picker. See Apple Picker
arrays (C# coding), 328, 333-337

Hello World program, 344-347
jagged arrays, 340-342
multidimensional arrays, 337-340
Bartok. See Bartok
Boids project (OOP in C# coding), 394
building simple models, 394-396
project setup, 394
scripts, 397-403
burndown charts, 409
assigning tasks, 414-415
BDV (Burndown Velocity), 414-416
creating, 416
estimated hours versus real hours, 413
example of, 410-412
prioritizing tasks, 414-415
C#, *Boids* project (OOP in C# coding), 394-403
classes (C# coding)
Enemy Class sample project, 381-384, 387-388
inheritance, 387-389
instances as GameObject components, 385-387
properties, 384-385
subclasses, 388-389
superclasses, 388-389
collections (C# coding), 328
arrays, 333-337
Hello World program, 344-347
jagged arrays, 340-342
jagged lists, 342-344
lists, 328-333
multidimensional arrays, 337-340
debugging C# coding, 363-364
attaching debugger to Unity, 372-374
attaching/removing scripts, 366-367
compile-time bugs, 364-366
runtime errors, 367-369
stepping through code, 369-371, 373-377
variables, 375-376
development process
Agile software development methodologies, 405-407, 416
burndown charts, 409-416
Manifesto for Agile Software Development, 406-407
Scrum software development methodologies, 407-409, 416
Enemy Class sample project
class inheritance, 387-388
project setup, 381-384
functions (C# coding), 349-350
arguments and parameters, 353-354
defining, 350-353
naming, 356
optional parameters, 359
overloading, 358
params keyword, 359-361
reasons for using, 356-357
recursive functions, 361-362
returning values, 354-356
void, returning, 355-356
Hello World program. *See* Hello World program
joining game projects in development, 221
lists (C# coding), 328-333
Hello World program, 344-347
jagged lists, 342-344
loops (C# coding)
do.while loops, 319-320
foreach loops, 322
jump statements in loops, 322-324
for loops, 320-322
modulus operator (%), 324
project setup, 300-301
while loops, 316-319
Mission Demolition. See Mission Demolition
naming, 266
Omega Mage. See Omega Mage
OOP (Object-Oriented Programming), *Boids* project (OOP in C# coding), 394-403
Prospector Solitaire. See Prospector Solitaire
QuickSnap. See QuickSnap
Scrum software development methodologies, 407, 416
meetings, 409
product backlogs/feature lists, 408
releases/sprints, 408

teams, 408
setup procedures, 264, 793, 796
attaching C# scripts to scene Main Camera, 797-798
creating C# scripts, 797
new projects, 794-796
saving scenes, 796-797
Space SHMUP. See Space SHMUP
starting, 221-222
Word Game. See Word Game
Projects folder (Unity), viewing contents of, 265
Prospector Solitaire, 561
art assets, 614-615
backgrounds, 614-615
beginning/end of round announcements, 615-616
clickable card functionality, 597-600
draw and discard actions, 597-604
example of play, 584-585
game logic, 597-604
gold cards, 620
importing Unity asset packages, 562
matching drawn cards, 600-604
mine tableau layout, 585-592
positioning cards on tableau, 592-594
setting up sorting layers, 594-597
mobile device build settings, 562-563, 620
project setup, 562
rules of the game, 583-584
scoring
chain scoring, 605-607
displaying scores to players, 607-614
earning points, 604-605
giving player feedback on scores, 616-619
shuffling cards, 581-583
sprites
creating cards from sprites, 566-581
importing images as sprites, 564-566
protagonists (player roles), 43, 56
prototypes (digital)
Apple Picker. See Apple Picker
Bartok. See Bartok
Game Feel: A Game Designer's Guide to Virtual Sensation, 420
Mission Demolition. See Mission Demolition
Omega Mage. See Omega Mage
Prospector Solitaire. See Prospector Solitaire
purpose of, 420
QuickSnap. See Quick Snap
Space SHMUP. See Space SHMUP
Word Game. See Word Game
prototypes (paper), 125, 140
benefits of, 126
best uses for, 138-139
example of, 129
gameplay development, 132-137
GM strategies, 137
initial concept development, 129-130
mission types, determining, 131-133
playtesting, 138
prototype construction, 131-133
movement systems, 127-128, 134, 138
poor uses for, 139
tools for, 127-129
puzzles and puzzle design, 185, 195
action games, sliding blocks/position puzzles, 193
action puzzles, 188
Art of Puzzle Design, The, 186
eight steps of puzzle design, 191-192
genres of puzzles, 187-188
goals of effective design, 192-193
reasons for playing puzzles, 189
required modes of thought in solving, 189
boss fights, 195
bridge puzzles, 188
chain reaction puzzles, 194
chess puzzles, 188
construction puzzles, 188
crossword puzzles, 188
defining, 186-187
eight steps of puzzle design, 191-192
genres of puzzles, 187-188
goals of effective design, 191-192
Kim, Scott, 186-193
multiplayer games, 186
physics puzzles, 194
QuickSnap, 695

adding sound/visual effects to shots, 723-725
aspect ratios, 697
build settings, 696
camera setup, 702-705
comparing shots, 717-721
customizing setup, 725-726
debugging, 713-714
deleting shots, 714-715
displaying player progress, 721-723
first-person controllers, 697-698
GUI elements, 703-705
importing Unity asset packages, 696-697
layers, 703-705
lightmapping, 698-701
maximizing target window, 716-717
project setup, 696
quality settings, 701-702
recording player progress, 721-723
replacing shots, 715-716
setting up, 706-707
storing shots, 707-712
reasons for playing puzzles, 189
required modes of thought in solving, 189-190
stealth puzzles, 194
story puzzles, 188
strategy puzzles, 188
Sudoku, 188
traversal puzzles, 194

Q

QA (Quality Assurance) testing, 152-153
Quake, machinima example, 84
Quake 2, game mods and cultural mechanics, 81
quaternion variables (C# coding), 291-292
quest outcomes (dynamic mechanics), 69
questions, asking (game play analysis), *Bartok*, 7
QuickSnap, 695
aspect ratios, 697
build settings, 696
camera setup, 702-705
customizing setup, 725-726
debugging, 713-714
first-person controllers, 697-698
GUI elements, 703-705
importing Unity asset packages, 696-697
layers, 703-705
lightmapping, 698-701
maximizing target window, 716-717
project setup, 696
quality settings, 701-702
recording/displaying player progress, 721-723
setting up, 706-707
shots
adding sound/visual effects, 723-725
comparing, 717-721
deleting, 714-715
replacing, 715-716

R

race conditions (C# coding), 812-814
RAND formula (Calc), weighted probabilities, 174-175
randomization
dice, 170
paper game technologies, 59
playing cards, 171-172
custom card decks, 172
shuffling decks, 172-173
spinners, 170
rate of fire (weapons), gameplay development (paper prototyping), 134-135, 137-138
rating phase (brainstorming/ideation), 101
Ravensburger and *Up The River*
inscribed dramatics, example of, 58
optimal strategy, determining (dynamic mechanics), 65-66
real hours versus estimated hours (burndown charts), 413
rebalancing weapons, 177-178
recursive Bézier curves, 844-845
recursive functions (C# coding), 361-362
Red Dead Redemption
climax, example of (dramatic narrative structure), 53
Poker in, 44
rule mechanics, 44

Red vs. Blue, machinima example, 83-84
registering Unity, 241-242
relationships, developing through shared experiences (dynamic dramatics), 76
releases/sprints (Scrum software development methodologies), 408
Renderer component (GameObjects), 295
REPT formula (Calc), showing overall damage (weapons and game balance), 181
required equipment (paper prototyping and gameplay development), 132
researching prior art (analysis phase of iterative design), 92
reserve functions (C# coding), 814-815
resolution (Act III), three-act dramatic narrative structure, 52
resolution (screen), environmental aesthetics (Dynamic Layer), 73
resources
  animation/model resources, 854-855
  art assets, 854
  audio assets, 854
  C# resources
    C# 4.0 Pocket Reference, 3rd Edition, 854
    CSharp Yellow Book, 854
    online resources, 853-854
    search tips, 854
  determining (analysis phase of iterative design), 91-92
  educational software discounts, 855
  FDD framework, 25
  font-related resources, 855
  Inscribed Layer (Layered Tetrad), 40, 45
  model/animation resources, 854-855
  online resources
    game developer salary surveys, 220
    Unity tutorials, 852
    Unity-related websites, 852-853
restarting games, *Space SHMUP*, 518-519
reward (inscribed dramatics), 58
Reynolds, Craig W.
  Flocks, Herds, and Schools: A Distributed Behavior Model, 393
  OOP (Object-Oriented Programming), 393
riffle shuffling, 7
right-clicking and OS X, 265, 849-850
RigidBody component (GameObjects), 272, 296
rising action (Act II), five-act dramatic narrative structures, 51
Roberts, Sam, and game definitions, 17
RoboCup tournaments, example of player's role in game design, 62
RoboRally, innovation and the design process, 97-98
Rock Band, player guidance, 200-201
Rockstar Studios
  Grand Theft Auto, resource mechanics, 45
  Grand Theft Auto V, player guidance, 199
  Red Dead Redemption
    climax, example of (dramatic narrative structure), 53
    Poker and rule mechanics, 44
Rocky Horror Picture Show, The, and audience participation, 62
Rogers, Scott
  Disneyland as example of indirect guidance, 202
  emergence (Dynamic Layer), mechanics and unexpected emergence, 63-64
  Level Up! The Guide to Great Video Game Design, 64, 202
Rogue, final outcomes example (dynamic mechanics), 69
Rohrer, Jason
  attention and involvement as player-centric goals of design, 120
  Passage, 11-12
role fulfillment (empathetic characters versus avatars), 56
roles of players (player relationships), defining, 43
  citizens, 44
  collaborators, 43
  competitors, 43
  multiplayer games, 44
  protagonists, 43
Romeo and Juliet
  empathetic characters versus avatars, 55-56
  five-act dramatic narrative structures, example of, 50-51

Rooster Teeth Productions, *Red vs. Blue* machinima example, 83-84
ROUND formula (Calc), showing overall damage (weapons and game balance), 181
rounds, comparing in *Bartok* (game play analysis), 9
rows (Calc spreadsheets)
  creating, 159
    Die A row, 160-161
    Die B row, 161-162
  labeling, 162-163
RPG (Role-Playing Games)
  Dungeons & Dragons, 846-847
  FATE system, 846-847
  tips for running good campaigns, 846-847
rules
  FDD framework, 25
  feel, designing for, 9-10
  house rules (dynamic mechanics), 66-67
  Inscribed Layer (Layered Tetrad), 40, 44
  modifying, *Bartok*, 8
Rules of Play: Game Design Fundamentals, 64, 120-121
runtime errors, debugging (C# coding), 367-369
Ryan, Malcolm, and *Bartok,* 4

S

salaries in the digital game industry, 220
Salen, Katie
  meaningful play
    defining, 64
    interesting decisions as a player-centric goal of design (dynamic procedures), 120-121
  Rules of Play: Game Design Fundamentals, 64, 120-121
scenes (Unity)
  saving, 796-797
  Scene pane, 251
Schell, Jesse
  Art of Game Design: A Book of Lenses, The, 4, 11, 20, 27
    indirectly guiding players, 200-206
    inscribed mechanics, 40-41
    interest as a player-centric goal of design, 119
    listening, importance of during design phase (iterative design), 93
    Ten Rules of Probability Every Game Designer Should Know, 165-169
    testing phase (iterative design), 96
  Elemental Tetrad framework, 20, 27-29
    aesthetics, 27-28
    mechanics, 27
    story's role in, 28
    technology's role in, 28
  games, defining, 11, 13-14
  indirectly guiding players, 200-206
  interest as a player-centric goal of design, 119
  listening, importance of during design phase (iterative design), 93
  probability, 165-169
  Skyrates, 111-112
  Ten Rules of Probability Every Game Designer Should Know, 165-169
  testing phase (iterative design), 96
schools, Games Education Programs, 215-217
scoping (design process), 103-104
score counter, *Apple Picker*, 440-441
screen variables (C# coding), 292
Scripting Reference (Unity), 430-431
scripts
  C# scripts, creating, 266-271
  formal group playtesting method, 147
  GameObject components, scripts as, 296-297
Scrum software development methodologies, 407, 416
  meetings, 409
  product backlogs/feature lists, 408
  releases/sprints, 408
  teams, 408
searches (online), C# resources, 854
second plot point (three-act dramatic narrative structure), 52
sequencing, teaching players by, 207-209

serious games, 109
setting up projects in Unity, 793, 796
  C# scripts
    attaching to scene Main Camera, 797-798
    creating, 797
  new projects, 794-796
  saving scenes, 796-797
settings, Inscribed Layer (Layered Tetrad), 49
Settlers of Catan
  resource mechanics, 45
  strategy, designing for (dynamic mechanics), 66
setup (paper prototyping and gameplay development), 133
Shakespeare, William, and *Romeo and Juliet*
  empathetic characters versus avatars, 55-56
  five-act dramatic narrative structures, example of, 50-51
shared experiences, developing relationships through (dynamic dramatics), 76
SHMUP (shoot-em-up) games. *See Space SHMUP*
shorting operators versus non-shorting operators (C# coding), 301-302
short-term objectives and space (Inscribed Layer mechanics), 46
shuffling card decks, 7, 172-173
side quests and interactive narrative, 54
Sierra OnLine, *Space Quest II* and autotelic empowerment as a player-centric goal of design, 117
silent protagonists (empathetic characters versus avatars), 56
similarities, indirectly guiding players by, 201
sine/cosine, 822-825
single player versus game (player interaction patterns), 43
Singleton software design pattern, 815-816
skills, teaching by player guidance, 207
  integration, 209
  sequencing, 207-209
Skyrates
  closed playtesting, 150
  open playtesting, 150-151
  sporadic-play games, 111
Skyrim
  conflicting objectives (Inscribed layer mechanics), 42
  final outcomes (dynamic mechanics), example of, 69
  game mods and cultural mechanics, 82
  importance of objectives (Inscribed layer mechanics), 42
  narrative game mods, 83
  player guidance, 198
  side quests, 54
sliding blocks/position puzzles in action games, 193
smell (Inscribed Layer aesthetics), 48
Snakes and Ladders
  aesthetics-based design example, 21-24
  American name change, 26
  strategic game play, modifying for, 23-24
    Factory design pattern, 816
    Singleton design pattern, 815-816
    Strategy design pattern, 816-817
social change, designing games for, 109
social network games and lusory attitude, 111-112
Socializer player type (hearts), 67
software design patterns, 815
software development
  Agile methodologies, 405-406, 416
    Manifesto for Agile Software Development, 406-407
  burndown charts, 409
    assigning tasks, 414-415
    BDV (Burndown Velocity), 414-416
    creating, 416
    estimated hours versus real hours, 413
    example of, 410-412
    prioritizing tasks, 414-415
  Scrum methodologies, 407, 416
    meetings, 409
    product backlogs/feature lists, 408
    releases/sprints, 408
    teams, 408

software, educational software discounts, 855
Somethin' Else, *Papa Sangre*, sound as an Inscribed Layer aesthetic, 47
sought outcomes of probability, 166
sound
  auditory play environments (environmental aesthetics), 73-74
    noisy environments, 74
    volume control, 74
  immediacy of (Inscribed Layer aesthetics), 47
  sound effects, 47
Space Quest II, autotelic empowerment as a player-centric goal of design, 117
Space SHMUP, 487
  adding elements, 559-560
  adding enemies, 543-556
  art assets, enemies, 504-506
  aspect ratios, 490
  backgrounds, 556-558
  boosting weapons, 531-538
  bounds
    camera bounds, 498-500
    hero ship bounds, 495-498
    testing overlapping bounds, 500-504
  camera setup, 490
  damaging players, 513-518
  directional light GameObject, 490
  enemies dropping power-ups, 541-543
  enemy attack/damage, 513-518
  enemy GameObjects, 504-506
  enemy scripts, 506-509, 543-556
  fine-tuning game play, 559
  function delegate for firing, 525-531
  hero shield, 493-495
  hero ship, 491-493
  importing Unity asset packages, 488-490
  layers, 510-513
  particle effects, 556-558
  physics, 511-513
  power-ups, 531-538, 541-543
  project setup, 488-490
  race conditions, 538-541
  restarting games, 518-519
  shooting, 519-531
  spawning enemies, 509-510
  starfields, 556-558
  tags, 511-513
  weapon definitions, 521-525
spaces, Inscribed Layer (Layered Tetrad), 40, 45
  experiences, 46
  flow, 45
  landmark, 45-46
  long-term objectives, 46
  medium-term objectives, 46
  purposes of space, 45
  short-term objectives, 46
Spec Ops: The Line
  inscribed dramatics, example of, 57-58
  limited possibilities and interactive or linear narrative, 53-54
speed (sense of)/vection, *Mission Demolition*, 466-471
Spider-Man 2, quest outcomes (dynamic mechanics), example of, 69
spinners (randomization), 170
spoilage mechanic (*Farmville*), 112
Spoilsport player type, 68
sporadic-play games and lusory attitude, 111-112
Spore, procedural animation, 72
spreadsheet programs. *See* Calc
spreadsheets (Calc)
  cells, entering data, 158-159
  columns
    adjusting column widths, 160
    labeling rows, 162-163
  creating, 158
  formulas, exiting formula editing, 164
  rows
    creating, 159
    creating Die A row, 160-161
    creating Die B row, 161-162
    labeling rows, 162-163
sprints/releases (Scrum software development methodologies), 408
sprites and *Prospector Solitaire*
  creating cards from sprites, 566-581
  importing images as sprites, 564-566
Stack Overflow website, C# online resources, 853

Star Control, conflicting objectives (Inscribed layer mechanics), 42
Star Wars: A New Hope
  narrative
    premises, examples of, 49
    settings, example of, 49
  scoping and the design process, 104
  three-act dramatic narrative structure, example of, 51-52
Star Wars: Knights of the Old Republic, limited possibilities and interactive or linear narrative, 53-54
Starcraft, game mods and cultural mechanics, 81-82
Start screen, *Apple Picker*, 448
state tracking, paper game technologies, 59
static functions (C# coding), 288
stealth puzzles, 194
Steinkuehler and the cultural aspects of game development, Constance, 80-81
Stern and autotelic empowerment as a player-centric goal of design in *Facade*, Andrew, 117
stories
  Elemental Tetrad framework, story's role in, 28
  FDD framework, 26
  story puzzles, 188
strategy
  defining, 65
  designing for, 66
  modifying *Snakes and Ladders* for strategic game play, 23-24
  optimal strategy, determining, 65-66
  strategy puzzles, 188
  Strategy software design pattern, 816-817
string variables (C# coding), 285
structured conflict as a player-centric goal of design, 115-116
subclasses (C# coding), 388-389, 589-592
Sudoku, 188
Suits, Bernard
  games, defining, 10-15
  Grasshopper, The, 10-11, 15, 110-111
  lusory attitude, 13-14, 110-111
SUM formula (Calc), 163, 181
Super Mario Brothers, integrated actions (meaningful play), 64-65
Super Mario Galaxy, particle systems (procedural visual art), 71-72
superclasses (C# coding), 388-389, 591
Swain, Chris
  FDD framework, 20, 24, 29
    boundaries, 25
    characters, 26
    Dramatic elements, 24-26
    Dynamic elements, 24, 26-27
    Emergence, 26-27
    Emergent Narrative, 27
    Formal elements, 24-25
    objectives, 25
    outcomes, 25
    player interaction patterns, 24
    playtesting, 27
    premises, 25-26
    procedures, 25
    resources, 25
    rules, 25
    stories, 26
  game design, 18
  Game Design Workshop, 10, 18, 20, 24
  iterative design, 90
Swink, Steve
  digital prototypes, 420
  Game Feel: A Game Designer's Guide to Virtual Sensation, 420
switch statements (C# coding), 310-313
SystemInfo variables (C# coding), 293
systems thinking, 225, 234
  Apple Picker game analysis, 229
    GameObject action lists, 231-232
    GameObject flowcharts, 232-234
    GameObjects, 230-231
    gameplay, 230
  board games, 226
  breaking down complex problems into simpler ones, 229
  code libraries, 228
  computer languages, 227
  development environment, 228

simple instructions exercise, 226-227

T

tables
  Civilization, 40-41
  Inscribed Layer (Layered Tetrad), 40, 46
    playtest data, 46
    probability tables, 46
    progression tables, 46
Tales of the Arabian Nights, probability tables, 46
teaching new skills/concepts, sequencing, 207-209
team competition (player interaction patterns), 43
Team Fortress 2
  Dynamic Layer (Layered Tetrad) example, 36-37
  freemium games, 214
Technique of Dramas, The, five-act dramatic structure, 50-51
technology
  Cultural Layer (Layered Tetrad), 84
    defining, 36
    external tools (player-built) and cultural technology, 84-85
    game technology used in other fields, 84
  Dynamic Layer (Layered Tetrad), defining, 34
  dynamic technology, 77
  Elemental Tetrad framework, 28
  Inscribed Layer (Layered Tetrad), 58
    defining, 33
    paper game technologies, 58-59
Tekken, fulfilling play as a player-centric goal of game design, 110
Ten Rules of Probability Every Game Designer Should Know, 165-169
testing phase (iterative design), 91, 95-96. *See also* playtesting
  fastest path to testing, determining (analysis phase of iterative design), 92
  repetition, importance of, 96-97
Tetris, 186
text
  font-related resources, 855
  text-based adventure games, loss of popularity, 117
texture (contrast), indirectly guiding players by, 204
That Dragon, Cancer, personal expression/communication as a goal of game design, 108
thatgamecompany
  Flower, music in, 71
  Journey, 41, 201-202
theoretical probability versus practical probability, 169
three-act dramatic narrative structure
  antagonism (Act II), 52
  climaxes, 53
  exposition (Act I), 52
  Field, Syd, 51-52
  first plot point, 52
  Foundations of Screenwriting, The, 51-52
  hooks, 52
  inciting incidents, 52
  resolution (Act III), 52
  second plot point, 52
  Star Wars: A New Hope as an example of, 51-52
Tic-Tac-Toe, optimal strategy, determining (dynamic mechanics), 65
time-based games, 431-432
time-based linear interpolation, 831-832
tissue playtesters, 144-145
Titanfall, researching prior art (analysis phase of iterative design), 92
Tolkien, J.R.R., *Fellowship of the Ring* as an emergent narrative example, 77
Tomb Raider, role fulfillment (empathetic characters versus avatars), 56
Tony Hawk's Pro Skater, performative empowerment as a player-centric goal of design, 118
top (AAA) games, costs in developing, 213
touch (Inscribed Layer aesthetics), 47-48
tracking (state), paper game technologies, 59
trails, indirectly guiding players by, 201
Transform component (GameObjects), 272, 294-295

transmedia and Cultural Layer (Layered Tetrad), 85-86
traversal puzzles, 194
Tueber, Klaus
  designing for strategy (dynamic mechanics), 66
  Settlers of Catan, 66
tutorials (Unity), 852
  Demo Projects section (Unity website), 852
  Learn section (Unity website), 852

U

Ultima IV, tracking and reacting to (empathetic characters versus avatars), 57
Uncharted
  inscribed dramatics, example of, 58
  limited possibilities and interactive or linear narrative, 53-54
  role fulfillment (empathetic characters versus avatars), 56
Uncharted 2: Drake's Deception, machinima example, 84
Uncharted 3
  particle systems (procedural visual art), 71-72
  player guidance, 203
unilateral competition (player interaction patterns), 43
Unity, 235, 237, 251
  AngryBots project example, 242-246
  Apple Picker. See Apple Picker
  art assets, Unity Asset Store, 854
  aspect ratios
    QuickSnap, 69
    Space SHMUP, 490
  asset packages
    Omega Mage, 729
    Prospector Solitaire, 562
    QuickSnap, 696-697
    Space SHMUP, 488-490
  Assets folder, 265-266
  audio assets, Unity Asset Store, 854
  axis mapping, InputManager (Unity), 491-494
  Bézier curves, 842-845
  C# coding
    attaching scripts to scene Main Camera, 797-798
    coroutines, 802-803
    creating scripts, 266-271, 797
    interfaces, 807-810
    online resources, 853-854
    race conditions, 812-814
  creating projects in, 264-266
  debugging C# coding
    attaching debugger to Unity, 372-374
    stepping through code, 369-371, 373-377
    variables, 375-376
  downloading, 236
  educational software discounts, 855
  equality testing, 304
  first-person controllers, *QuickSnap*, 697-698
  force-quitting, 371-372
  functions (C# coding), defining, 350-353
  GameObjects, 281, 293-294
    Box Collider component, 295
    Box Colllider component, 272
    Capsule Collider component, 295
    Collider components, 295-296
    Colllider components, 272
    creating, 271-272
    editing, 272, 277-278
    gizmos, 277
    Mesh Collider component, 295
    Mesh Filter component, 272
    Mesh Renderer component, 272
    Mesh Filter component, 295
    playing, 272-273
    prefabs, 273-277
    Renderer component, 295
    RigidBody component, 272, 296

scripts as GameObject components, 296-297
Sphere Collider component, 295
Transform component, 272, 294-295
InputManager, axis mapping, 491-494
Inspector pane
editing GameObjects, 272
GameObject components, 294
installing, 241
interpolation
easing linear interpolation, 837-841
interpolating values other than position, 834-835
time-based linear interpolation, 831-832
Zeno's Dichotomy Paradox and linear interpolation, 833-834
licensing, 241-242
lightmapping, *QuickSnap*, 698-701
linear extrapolation, 836-837
LineRenderer, ground spells in *Omega Mage*, 756-761
naming projects, 266
navigating, 251
panes, 251
configuring layout, 246-251
Console pane, 251
Game pane, 251
Hierarchy pane, 251
Inspector pane, 251
Project pane, 251
Scene pane, 251
particle systems (procedural visual art), 71-72
Profiler, 663-665
Project pane, creating new projects, 264-266
project setup procedures, 793, 796
attaching C# scripts to scene Main Camera, 797-798
creating C# scripts, 797
new projects, 794-796
saving scenes, 796-797
Project Wizard, new project setup procedures, 794-796
Projects folder, viewing contents of, 265
Prospector Solitaire, importing Unity asset packages, 562
QuickSnap, importing Unity asset packages, 696-697
reasons for choosing, 237-238
recursive Bézier curves, 844-845
registering, 241-242
Scripting Reference, 430-431
sine/cosine, 822-825
Space SHMUP
aspect ratios, 490
importing Unity asset packages, 488-490
systems thinking and, 228
tutorials, 852
Demo Projects section (Unity website), 852
Learn section (Unity website), 852
variables, 287-289
color variables, 290-291
instance variables, 289
mathf variables, 292
quaternion variables, 291-292
screen variables, 292
SystemInfo variables, 293
Vector3 variables, 288-290
web resources
Goldstone's websites, Will, 853
Unity Asset Store, 854
Unity Gems website, 852
website
Demo Projects section (Unity website), 852
Learn section (Unity website), 852
tutorials, 852
universities/colleges, Games Education Programs, 215-217
Up The River
inscribed dramatics, example of, 58
optimal strategy, determining (dynamic mechanics), 65-66
usability testing, 152
user interfaces
Microsoft controllers
axis mapping, 848-849
button mapping, 848-849
OS X and right-clicking, 849-850

V

Valkyria Chronicles, 129, 198
Valve
  Counter Strike, game mods and cultural mechanics, 81-82
  Half-Life, game mods and cultural mechanics, 81-82
  Team Fortress 2, Dynamic Layer (Layered Tetrad) example, 36-37
variables (C# coding), 256-257, 282
  bool variables, 283
  char variables, 285
  class variables, 286
  debugging C# coding, 375-376
  declaring, 282-283
  defining, 282-283
  equality testing, 304
  float variables, 284
  int variables, 284
  literal values, 283
  naming conventions, 286-287
  private variables, 287
  scope of, 286, 817-820
  string variables, 285
  types of, 283-286
  Unity variables, 287-289
    color variables, 290-291
    instance variables, 289
    mathf variables, 292
    quaternion variables, 291-292
    screen variables, 292
    SystemInfo variables, 293
    Vector3 variables, 288-290
vection/sense of speed, *Mission Demolition*, 466-471
Vector3 variables (C# coding), 288-290
Vectorized Playing Cards 1.3
  Bartok, 622
  Prospector Solitaire, 562
vertiginous play (Ilinx)
  Imaginary Games, 110
  Jeux et Le Hommes, Le, and the four different kinds of play, 110
visibility (paper prototyping and gameplay development), 133-134, 136
vision (Inscribed Layer aesthetics), 47
Vissides, John
  Design Patterns, 769, 815
  software design, 815
    Factory design pattern, 816
    Singleton design pattern, 815-816
    Strategy design pattern, 816-817
  spawning enemies/enemy factories, 769-770
visual art (procedural), 71-72
visual design, indirectly guiding players by, 201-204
visual play environments (environmental aesthetics), 73
  brightness, 73
  resolution (screen), 73
volume control, auditory play environments (environmental aesthetics), 74
VRO (Vertical Re-Orchestration), procedural music, 70-71

W

Warcraft III, game mods and cultural mechanics, 81-82
wealth as a goal of game design, 107
weapons and game balance, 177-178
  average damage
    calculating, 179
    charting, 179-180
  Chapter 9 prototype example, 182-183
  duplicating weapon data, 180-181
  overall damage, showing, 181
  percent chance for each bullet, determining, 178
  rebalancing weapons, 181-182
weapons/firing, gameplay development (paper prototyping), 134-135, 137-138
web resources, game developer salary surveys, 220
websites
  animation/model resources, 854-855
  art assets, 854
  audio assets, 854
  educational software discounts, 855
  font-related resources, 855

Goldstone's websites, Will, 853
model/animation resources, 854-855
Stack Overflow website, C# online resources, 853
Unity Gems website, 852
Unity website
Learn section (Unity website), 852
tutorials, 852
weighted distributions and game balance, 173-175
weighted probabilities in Calc, 174-175
Westwood Studios, *Blade Runner* and multiple dialogue choices (empathetic characters versus avatars), 57
while loops (C# coding), 316-319
Williams, John, *X-Wing*, music in, 70
Windows
force-quitting applications, 317-318, 371-372
Unity, new project setup procedures, 794-796
windows (Unity). *See* panes (Unity)
Wittgenstein, Ludwig
games, defining, 15
Philosophical Investigations, 15
Wizards of the Coast
Dungeons & Dragons, 846
dynamic dramatics, 75-77
emergent narrative example, 76-77
interactive versus linear narrative, 55
progression tables, 46
tips for running good campaigns, 846-847
XP (Experience Points) and cumulative outcomes (dynamic mechanics), 69
RoboRally, innovation and the design process, 97-98
Word Game, 657, 692-693
adding color, 690-692
adding interactivity, 680-684
animation, 687-690
design goals, 658-659
layouts, 671-680
parsing the word list, 660-663
project setup, 658
scoring, 684-687
setting up, 664-671
working in the digital game industry, 217
following up, 218-219
interviewing, 219-220
networking, 217-218
salaries, 220
working conditions, 213
World of Warcraft
player interaction patterns, 43
player-built external tools as example of cultural technology, 84
Wright, Will
procedural animation, 72
Spore, 72
Wyatt, James, on XP (Experience Points) and cumulative outcomes (dynamic mechanics), 69

X-Y-Z

XML (Extensible Markup Language)
C# coding and, 817-821
Omega Mage, creating the game environment, 730-735
XP (Experience Points) and cumulative outcomes (dynamic mechanics), 69
X-Wing
information, conveying (Inscribed Layer aesthetics), 48-49
music in, 70
Yager Development, *Spec Ops: The Line* and limited possibilities and interactive or linear narrative, 53-54
Yee, Nick, on types of players, 68
Zeno's Dichotomy Paradox and linear interpolation, 833-834
Zimmerman, Eric
meaningful play, 64, 120-121
Rules of Play: Game Design Fundamentals, 64, 120-121
Zork, interactive narrative example, 75-76

Zubek, Robert, and MDA framework, 20, 29
  aesthetics-based design, 21-24
  defining, 20-21
  designer views of games, 21
  player views of games, 21
Zynga, freemium games, 214