FROM CONCEPT TO PLAYABLE GAME
WITH UNITY® AND C#

New Chapters, Coding Challenges and Expanded Tutorials!

Introduction to GAME DESIGN, PROTOTYPING, and DEVELOPMENT

Third Edition

Jeremy Gibson **BOND**

Foreword by Richard Lemarchand

FREE SAMPLE CHAPTER |



Praise for the Second Edition

"When teaching about game design and development, you often get asked the dreaded question: 'Where can I learn all this?' *Introduction to Game Design, Prototyping, and Development* has been my deliverance, as it provides a one-stop solution and answer. This book is quite unique in covering in-depth both game design and development: it embraces and exemplifies the idea that design, prototyping, development, and balancing combine in an iterative process. By sending the message that creating games is both complex and feasible, I believe this to be a great learning tool; and the new edition with even more detailed examples seems even better."

—**Pietro Polsinelli**, Applied Game Designer at Open Lab

"*Introduction to Game Design, Prototyping, and Development* has truly helped me in my game development journey and has opened my mind to many helpful techniques and practices. This book not only contains a full introduction to the C# language, but also includes information about playtesting, game frameworks, and the game industry itself. Jeremy is able to explain complex concepts in a way that is very informative and straightforward. I have also found the prototype tutorials to be useful and effective for developing good programming practices. I would highly recommend this book to anyone looking to learn game development from scratch, or simply brush up on their skills. I look forward to using it as a guide and reference for future projects."

—**Logan Sandberg**, Pinwheel Games & Animation

"Jeremy's approach to game design shows the importance of prototyping game rules and prepares the readers to be able to test their own ideas. Being able to create your own prototypes allows for rapid iteration and experimentation, and makes better Game Designers."

—**Juan Gril**, Executive Producer, Flowplay

"*Introduction to Game Design, Prototyping, and Development* 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. I regularly recommend this book to any aspiring game designers who are looking to learn new skills or strengthen their design chops. Jeremy uses his years of experience as a professor to teach you how to think with vital game design mindsets so that you can create a game with all the right

tools at hand. Regardless of how long you've been in the games industry, you're bound to find inspirational ideas that will help you improve your design process. I'm personally excited to dive into the updates in this latest edition and get a refresher course on some of the best practices for creating amazing games!"

—**Michelle Pun**, Game Producer at Osmo, former Lead Game Designer at Disney and Zynga

"I used Professor Bond's book to teach myself how to code in C# and familiarize myself with Unity. Since then I have used the book as the backbone for my high school Digital Game Design class. The programming lessons are top-notch, the prototypes clearly demonstrate the myriad facets of programming and how those are used to create recognizable game mechanics, and the prototypes are easily adapted for student personalization. I can't wait to get hold of the second edition and begin using it in my classroom."

—**Wesley Jeffries**, Game Design Teacher, Riverside Unified School District

"With the latest edition of *Introduction to Game Design, Prototyping, and Development*, Bond builds on the solid foundation of the first. The new edition adds new content throughout the book, with updated examples and topics across all the chapters. This is a thorough and thoughtful exploration of the process of making games."

—**Drew Davidson**, Director, Entertainment Technology Center at Carnegie Mellon University

"If you want to take your game development to the next level, this book is a must! Not only does it give you a lot of game examples from beginning to end, it also—and this is the most important part—makes you think like a game designer. What makes a game fun and engaging? What makes a player come back to your game over and over again? The answers are all here. This book gives you a lot more than a couple of online tutorials can give you. It gives you the whole picture!"

—**David Lindskog**, Founder, Monster Grog Games

Introduction to Game Design, Prototyping, and Development

This page intentionally left blank

Introduction to Game Design, Prototyping, and Development

From Concept to Playable Game with Unity and C#

Jeremy Gibson Bond

Addison-Wesley

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

Library of Congress Control Number: 2021947656

ISBN-13: 978-0-13-661994-9
ISBN-10: 0-13-661994-0

ScoutAutomatedPrintCode

Editor-in-Chief
Mark Taub

Acquisitions Editor
Malobika Chakraborty

Development Editor
Chris Zahn

Managing Editor
Sandra Schroeder

Senior Project Editors
Lori Lyons
Tonya Simpson

Copy Editor
Paula Lowell

Indexer
Ken Johnson

Proofreader
Donna E. Mulder

Technical Reviewer
Margaret Moser

Editorial Assistant
Cindy Teeters

Cover Designer
Chuti Prasertsith

Compositor
codeMantra

PEARSON'S COMMITMENT TO DIVERSITY, EQUITY, AND INCLUSION

Pearson is dedicated to creating bias-free content that reflects the diversity of all learners. We embrace the many dimensions of diversity, including but not limited to race, ethnicity, gender, socioeconomic status, ability, age, sexual orientation, and religious or political beliefs.

Education is a powerful force for equity and change in our world. It has the potential to deliver opportunities that improve lives and enable economic mobility. As we work with authors to create content for every product and service, we acknowledge our responsibility to demonstrate inclusivity and incorporate diverse scholarship so that everyone can achieve their potential through learning. As the world's leading learning company, we have a duty to help drive change and live up to our purpose to help more people create a better life for themselves and to create a better world.

Our ambition is to purposefully contribute to a world where

- Everyone has an equitable and lifelong opportunity to succeed through learning
- Our educational products and services are inclusive and represent the rich diversity of learners
- Our educational content accurately reflects the histories and experiences of the learners we serve
- Our educational content prompts deeper discussions with learners and motivates them to expand their own learning (and worldview)

While we work hard to present unbiased content, we want to hear from you about any concerns or needs with this Pearson product so that we can investigate and address them.

- Please contact us with concerns about any potential bias at https://www.pearson.com/report-bias.html.

This book is dedicated to:

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

My son Jordan and godson Phoenix,
whom I hope to make games with one day

My parents and sisters

My friend and mentor Mike Sellers,
a brilliant designer and professor

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

Contents at a Glance

Contents

FOREWORD

Jeremy Gibson Bond taught me how to code. When I joined the University of Southern California Games program in 2012, one of the first things I did was to sign up for Jeremy's class in Unity and C# programming. I had just left Naughty Dog, where I'd worked as a lead game designer on the *Uncharted* series. I'd done a lot of scripting—simplified programming—during my career, but I'd always had a chip on my shoulder about not being a "real" coder. Jeremy's class fixed that, in just fifteen weeks.

In the class, I made a version of the classic game *Asteroids* in Unity, which my teammate and I then modded into an original game, and even though it was probably the simplest game I'd made since I was a kid, it was one of the most satisfying development experiences of my life. Every single one of Jeremy's classes was not only jam-packed with information about Unity and C# but was also peppered with inspirational 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 was blown away by Jeremy's skill as a teacher, and by his ability to make the process of creating gameplay into its own kind of exciting fun. Of course, I was delighted when I learned that he was packing all of that inspiring knowledge into the book you're now starting to read.

I'd first met Jeremy at the Game Developers Conference in 2002, and we hit it off immediately. Jeremy already had a successful career as a game developer, and his enthusiasm for game design struck a chord with me. I was drawn to his sharp understanding of game development and design, his easy, friendly manner, and the engaging way he loves to talk about game design as a craft, a design practice, and an emerging art form. We stayed friends down the years; I was excited when Jeremy got his master's degree from the world-famous Entertainment Technology Center at Carnegie Mellon University and was happy to see him go from strength to strength in his career. And of course, I was delighted to briefly be colleagues with him at USC, before he moved to teach at Michigan State University.

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 now you're very fortunate, because you're holding in your hands what is essentially the textbook—and much, much more—of the class that I took with Jeremy. With an incredible wealth of knowledge

about game design, Unity, and C#, and highly detailed, step-by-step instructions, this book is a sure-fire method of realizing your game development dreams. Not only that, but it's the third edition, and over the years, Jeremy has been continuously refining and updating this superb volume, seeing it in action in his own classes and in those of others.

As you'll see, the book opens with a section on Jeremy's wide-reaching, wise, and grounded philosophy of game design, a section that is worth the price of admission on its own. Jeremy is extraordinarily well-read about game design, and this book is going to give you an overview of all the best game design theory to know about. After laying out the most useful definitions of "game," Jeremy will present you with his idea of the "Layered Tetrad," a valuable synthesis of the finest game analysis frameworks. He'll go on to give you a clear breakdown of how to design a game, including paper prototyping, playtesting, game balancing, guiding the player, and designing puzzles. He'll talk you through the best practices of Agile development, including the "burndown chart" scheduling tool, which Jeremy taught me. This tool is now a core part of my own classes and has helped countless game developers to both keep their projects on track and avoid running out of time. Part II of the book will ease you painlessly into the world of programming in C# and working in Unity. The careful way that Jeremy introduces and explains often abstract and difficult-to-grasp concepts is brilliant and works like magic to turn non-programmers into wised-up coders. Once you've worked through this section, you'll be ready to dive into the excellent tutorials in Part III.

This third edition of the book is the best yet, packed with up-to-date and essential information. It includes a new chapter on Data-Oriented Design—which thinks about code from the point of view of how data is managed by the computer—and Unity's new Data-Oriented Tech Stack, which can help you speed up the performance of your games enormously. The C# terms and samples in the book are now highlighted and color-coded in a very similar way to that of Microsoft Visual Studio, the C# development environment installed with Unity. The book's tutorials, an important part of the special magic of Jeremy's teaching, are more refined and detailed than ever before. These tutorials will supercharge your game coding practice, just like they did mine, as Jeremy guides you through the creation of small games that build your knowledge in a systematic way. In addition, Jeremy is now providing you with a set of "Coding Challenges" that can be found on the book's website: partially complete games that guide you in the creation of the code to make them work. These will help you transition from the tutorials in the book to writing your own games from scratch. As if all that wasn't enough, the book now has excellent new 3D art by Peter Burroughs and an appendix on how to teach using the book, which will be invaluable to game professors around the world. And don't miss the other appendices in the last part of the book, a grab-bag of knowledge and wisdom that are the diamonds and rubies at the very bottom of this mine.

Jeremy is an immensely talented and knowledgeable game developer and game educator. He's put in many multiples of the ten thousand hours said to be needed to become an expert, and done it several times over, in the disciplines of game design, C#, Unity, and game education. Not only that, but his integrity, his kindness, and his sense of fun shine through in these pages. This is the book that I recommend to my students when I want to help them transform themselves from a game engine dabbler into a Unity adept, and I am delighted to recommend it to you.

Good luck, and have fun!

Richard Lemarchand
Associate Professor, USC Games

PREFACE

Welcome to the third edition of *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 Media and Information Department at Michigan State University and the Interactive Media and Games Division at the University of Southern California.

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

The Purpose of This Book

My goal in this book is 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 both the disciplines of game design and digital game programming and development and wraps them both in the essential practice of iterative prototyping. The growth 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. Whether you wish to be a game programmer, game designer, or a bit of both, this book has much to offer you.

What's New in the Third Edition?

Since 2017, when the second edition of this book was published, Unity has grown and changed considerably. To give you the best possible book for learning Unity, I had to change this book as well. Some of the major changes include:

- **About 400 additional pages:** With the inclusion of the online appendices, the additional content in this book is longer than the entirety of some books on learning to program. Unity has grown significantly over the years, and you need to know more to be able to use it well. I have added tons of content to the game prototypes so that you can experience more of the expanded features of Unity. Among the new chapters I have added is a new chapter explaining Data-Oriented Design, a new approach to Unity programming that can drastically increase performance and efficiency but

requires a completely different mindset from the Object-Oriented Programming that has been taught at universities for the past 30 years.

- **Improved, more polished tutorials:** The first edition contained eight rather small tutorials that provided a good introduction to Unity at the time. Now that Unity has more capabilities, I have worked these into the tutorials as well. Of the five tutorials in Part III of this book, three are now spread across two chapters (a space shooter, a card game, and an action/adventure game). For each of these, the first chapter sets up the underlying technology and gets you to a rough prototype, while the second chapter expands the prototype into a *first playable*, a more polished version of the game that is ready to be shown to others for feedback.
- **Better code throughout:** As I have improved as a programmer, so has the code in this book. Each of the Part III prototypes are designed to be a framework upon which you could build your own games, so code throughout the book has been revised to be more understandable and extensible. Additionally, immense care has been taken to implement consistent syntax coloring throughout the book, making the code clearer and easier to read. In the many places where you are modifying existing scripts, the areas that you must modify have also been made clearer.
- **Coding Challenges:** One wholly new aspect of the book is the online Coding Challenges, which are designed to aid your transition from following the book tutorials to creating your own games from scratch. Each challenge is a complete Unity game project with much of the key code missing. In place of this code are comments that explain what the code should do and how it should work. Replacing the missing code draws upon your experience from this book and helps you better internalize what you learn here. I have used these successfully in my classes for a few years to great effect!
- **Unity 2020.3 LTS:** Unity's new commitment to Long Term Support (LTS) releases means that they will make only bug-fix and security fixes to LTS releases and will avoid any changes that could break code or tutorials like those in this book. By committing the book to version 2020.3 LTS, I avoid many of the issues that could come up if you tried to use a more recent version of Unity with these tutorials. 2020.3 LTS was released in mid-2021 and will be updated monthly until mid-2023, but it will be a viable, solid release for years after that.
- **Better online tools:** Many of the online tools that I offer you through this book are the same tools that I developed for my own game development projects and the classes I teach. Hundreds of students have used these tools for dozens of projects, and I have improved them every semester. This now even includes an online code-checker that can help you find issues in your code at any point in one of the tutorials.

These are just a few of the many improvements I have worked into the book since the previous edition. While the game design chapters in Part I and the C# programming

chapters in Part II have several revisions throughout, the game prototypes in Part III and beyond contain the most drastic changes. I have put well over 1,000 hours into improving this book to make it the best possible way for you to learn Unity. It contains as much content as I could possibly fit into it (more pages than they would allow me to print!), and I know it will be a great resource for you.

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: Game Design and Paper Prototyping** 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.
  - **Part II: Programming C# in Unity** teaches you how to program from nothing to understanding object-oriented class hierarchies in C# (pronounced *See-Sharp*). 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: Game Prototype Tutorials** takes you through the process of developing several different game prototypes across several different game genres. Each demonstrates fast methods to get from concept to working digital prototype.
  - **Part IV: Next Steps** covers what you can do once you've finished this book. It introduces the Coding Challenges that have been extremely successful in helping my students transition from following tutorials to creating their own games and gives you many ideas for what you can do next in your journey.
  - Lastly, the online **Appendices** in **Part V** explain specific game development and programming concepts in-depth and guide you to other online resources that may be useful.
- **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: Game Design and Paper Prototyping** 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: Programming C# in Unity**, which introduces C# (pronounced *See-Sharp*) 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: Game Prototype Tutorials** takes you through the process of developing several 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, as many elements of development are managed 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.
- **Part IV: Next Steps** covers what you can do once you've finished this book. It introduces the Coding Challenges that have been extremely successful in helping my students transition from following tutorials to creating their own games and gives you many ideas for what you can do next in your journey.
- You will also want to look carefully at **Part V: Appendices**, which is full of detailed information about various Unity development concepts and is arranged as a reference that you can return to later.

- **If you're teaching game design or programming**, you're not alone. Many universities worldwide use this book as their game design and programming textbook. I have added a new **Appendix D** that outlines how I recommend teaching from this book.

The Structure of This Book

The book is divided into five 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; gives you concrete information to help you become a better designer; and presents you with effective project and time management strategies to help keep your projects on track. The final chapter examines the game industry and gives you several tips for how to approach finding a job.

Part II: Programming C# in Unity

The second part teaches you C#—our programming language—from the basics through class inheritance and object-oriented programming. 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.

The final chapters of this part explore Object-Oriented Programming and Data-Oriented Design, two very different approaches to designing advanced code. Data-Oriented Design is the core of Unity's new Data-Oriented Tech Stack (DOTS), which can drastically improve the performance and efficiency of your code.

Part III: Game Prototype 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 a basic foundation on which to build your own games in the future. Many 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.

The three final projects of this part each span two chapters. The first chapter gets you to the playable prototype stage of the project, where the basic technology is in place and the core mechanics of the game work. The second chapter of each takes the game to what is known in the industry as a *first playable*, the state of the game where you would actually show it to other people and get their feedback. Each of these projects has grown as Unity has grown. The versions of these projects in the first edition of the book were basic and rough, while the versions in this third edition have grown more refined and take advantage of more interesting and useful aspects of both Unity and C# programming.

Part IV: Next Steps

This entirely new section comprises two chapters that will help you take the next steps in your game programming journey after you have finished the book. After completing

prior editions of the book, readers and students often had difficulty transitioning from following the detailed book tutorials to creating their own projects from scratch. To rectify this in my classes, I introduced Coding Challenges, game prototypes that are nearly complete except for the code. In place of the code, there are detailed comments outlining what the code needs to do there, and you can follow those comments to create the needed code and make the prototypes work. Chapter 37, "Coding Challenges," introduces you to these challenges and guides you to finding them online. I plan to add a new challenge at least once per semester following publication of the book. The final chapter of Part IV, "Beyond This Book," gives you several ideas for the next projects you can tackle and where to find resources to do so.

Part V: Online 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 want to look back on (after you've finished reading the book once) is placed in the appendices, which are online-only, both to reduce the immense size of this book and make searching them easier. To find them, head to this book's website: http://book.prototools.net or informit.com/title/9780136619949.

- **Appendix A: Standard Project Setup Procedure** is a step-by-step introduction to the initial creation process for a game project in Unity. There is a lot to know, and this appendix will make sure your projects start on the right foot.
- The 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. To be honest, I brush up on topics in this appendix pretty often myself!
- **Appendix C: Online Reference** is a list of very useful 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.
- **Appendix D: Tips for Teaching from This Book** covers my best practices for instructors using this book in a classroom. I have taught from this book every semester since the first edition was published, and I have iterated many times to find the right way to present the information. This includes sample schedules for both Game Design and Game Programming classes.

Book Website

The website for this book includes all of the files referenced in the chapters, lecturer notes, starter packages, and errata for anything that we somehow failed to correct in the many passes through editing and the tutorial projects. Find it at

http://book.prototools.net

or

informit.com/title/9780136619949

Why You Should Learn Unity and C#

All the digital game examples in this book are based on the C# programming language and the Unity Game Engine. I have taught students to develop digital games and interactive experiences for two decades now, 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.

The Unity 2020.3 LTS Development Environment

Some other tools out there are easier to learn and require no real programming (Game Maker is a great example), 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). Unreal is another game engine used by some studios, but in Unreal, there is very little middle ground between the simplified graphical programming of the Blueprint system and the very complex C++ code on which the engine is built. If you want to actually learn to program games and have success doing it, Unity is the engine you want to use.

Unity has both *Tech Stream* releases that include all the newest features (in a sometimes buggy state) and *Long Term Support (LTS)* releases that are stable and supported for many years. Unity 2020.3 LTS, which we use in this book, was initially released in 2021 and represents an extremely stable and feature-rich release of Unity. It will be updated monthly until 2023 and will be stable and usable for several years beyond that. I do not recommend attempting to follow the book tutorials with a future version of Unity, but transitioning from 2020.3 LTS to later versions of Unity after completing the book will be easy for you.

The C# Programming Language

In the past, I have taught my students many languages, including C++, JavaScript, and ActionScript. However, C# is the one language that 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 (e.g., strong variable typing enables very robust code hinting and auto-completion, which makes coding faster and more accurate).

Conventions in This Book

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 appear in the text, they will be listed in *italics*. This includes terms like the *Main Camera* GameObject. 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* submenu, and then select *Physics*. I also tend to *italicize* important terms when first introducing them and use **bold** and *italics* for emphasis throughout the book. When specific terms from C# code are used in text, they are in bold code font for emphasis and clarity. Examples include `float`, `List<>`, and text like `"Hello World"`, and `MonoBehaviour` (which uses the British spelling because Unity originated in Europe).

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 code listing below is an example of this.

Code Listings also utilize a monospaced font and appear as follows. Code Listings are all numbered (here 0.1), and the name of the code file you're editing is also included (e.g., SampleClass.cs).

Code Listing 0.1 SampleClass.cs

```
1 public class SampleClass {
2     public GameObject       variableOnExistingLine;                    // a
3     public GameObject       variableOnNewLine;                         // b
4 }
```

a. Code Listings are often annotated; in this case, additional information about the line marked with `// a` would appear in this first annotation. Annotations are always bold to call attention to them.

b. 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`**.

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 Visual Studio (it will automatically number all lines). In Part III of the book, there are no line numbers due to the size and complexity of the code listings increasing the chance that your line numbers would differ from mine. However, later

code listings precede each line with a pipe character "|" to clarify the indentation level of each line of code, and new lines are preceded by a bold right angle bracket ">" as shown in Code Listing 0.2. You also *should not type* these | or > characters.

Code Listing 0.2 SampleClassFromLaterInTheBook.cs

```
| public class SampleClassFromLaterInTheBook {
|     public GameObject       variableOnExistingLine;
>     public GameObject       variableOnNewLine;
| }
```

tip

THE CODE YOU WRITE IN YOUR PROJECTS WON'T LOOK LIKE MINE
This is something that a reader asked me to add to the beginning of this book. I spend many hours and many passes working to make my code as clear and understandable as possible. When you start writing your own C# code for your own games, it is not going to be as clean, and that is absolutely okay. Game prototyping is not about beautiful, clean code; it is about getting a game working as quickly as possible. Once the game is working, if you want to continue and expand the project, you can *always* go back and refactor the code into something cleaner. And, if you want to write tutorials to teach other people, you can refactor it a third or even fourth time, like I have.

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. Many books have been 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 several of 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. 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 taught at USC from 2009–2013). The USC Interactive Media and Games graduate program has been named the number one private university for game design in North America by Princeton Review nearly every year that it has 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.

> Tracy Fullerton, Christopher Swain, and Steven Hoffman, *Game Design Workshop: A Playcentric Approach to Creating Innovative Games,* 2nd ed. (Boca Raton, FL: Elsevier Morgan Kaufmann, 2008)

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)

Advanced Game Design: A Systems Approach by Michael Sellers

Mike Sellers once told me that "Systems thinking is the literacy of the 21st century," and I think that he is absolutely right. A large portion of the world population began the 20th century unable to read, but it is now a nearly ubiquitous skill. Similarly, he believes that to survive in the future, people must learn to understand the interrelated systems that impact their lives, and one of the best ways to understand systems is to design games. Mike is one of the most brilliant game and artificial intelligence developers that I have ever met, and he distilled a tremendous amount of his knowledge, understanding, and processes of design into this excellent book.

> Michael Sellers, *Advanced Game Design: A Systems Approach* (Indianapolis: Pearson Education, Inc., 2018)

A Playful Production Process: For Game Designers (and Everyone) by Richard Lemarchand

Richard Lemarchand, the author of the forewords to all three editions of this book, has thought deeply for nearly 30 years about how the production process of a game (or any project) can be designed to make the experience of working on the game playful and joyful for everyone involved. Those years of experience as both a co-lead designer on the *Uncharted* series and a professor at the University of Southern California have all led to this book. Discussions with Richard and my reading of this book changed how I approach some of my group-based game development classes and led to many of the third edition changes that I made to Chapter 7, "Acting Like a Designer."

> Richard Lemarchand, *A Playful Production Process: For Game Designers (and Everyone)* (Cambridge, MA: MIT Press, 2021)

Games, Design and Play by Colleen Macklin and John Sharp

Unlike some other game design texts that talk about the process of game design in theory, *Games, Design and Play* digs deeply into the details of design; into the nuts and bolts of what actually goes into making good design decisions as well as the impact of those decisions. Colleen and John do not cover game development at all—in fact, they recommend that you read this book to learn game programming—and instead focus exclusively on game design. This book illustrates its points with real examples from independent game developers, meaning that their examples are much more similar in scope to those you might encounter as you're getting into game development.

> Colleen Macklin and John Sharp, *Games, Design and Play: A Detailed Approach to Iterative Game Design* (Boston, MA: Addison-Wesley, 2016)

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. Scott is also a comic book artist, and his book is full of humorous and helpful illustrations that drive home the design concepts.

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

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)

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 Bernard 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 helped me 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)

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 of all three editions of this book improved them tremendously. She is not only my inspiration but has also been an excellent copy editor. 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 with every edition, there were several people at Pearson who provided support to me and shepherded me through this process. Chief among them was Chris Zahn, who has been with me since the first edition. Laura Lewin initially approached me about writing a book and served as the acquisitions editor for the first two editions. Also at Pearson, Malobika Chakraborty, Lori Lyons, and Tonya Simpson each demonstrated incredible patience in working with me as I worked to complete this book with a new child and throughout the COVID pandemic. Margaret Moser continued her fantastic work as a technical reviewer on this edition of the book and not only caught my mistakes but also added her brilliant insight throughout the book. Thanks also to the excellent proofreader, Donna E. Mulder, as well as Aswini Kumar and the team at Codemantra for their work in the production phase.

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 Games and Interactive Media Division. There were also many other brilliant faculty and friends at USC who helped me flesh out the ideas in this book, including Adam Sulzdorf-Liszkiewicz, William Huber, Richard Lemarchand, Scott Rogers, Vincent Diamante, Sam Roberts, and Logan Ver Hoef. My current colleagues at Michigan State University have also contributed ideas and feedback on the third edition of the book, especially Andrew Dennis, Elizabeth LaPensée, Adam Sulzdorf-Liszkiewicz, and Ryan Thompson.

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 that 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 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 Bond is a Professor of Practice teaching game design and development at Michigan State University, which in 2022 was ranked the #1 public university for undergraduate game development by Princeton Review three of the last four years. Since 2013, he has served the IndieCade independent game festival and conference as the Chair of Education and Advancement, where he co-chairs the IndieXchange summit each year and has also chaired the GameU summit. In 2013, Jeremy founded the company ExNinja Interactive, through which he develops his independent game projects. Jeremy has spoken several times at the Game Developers Conference. He also created the official Unity Certified Programmer Exam Review specialization on Coursera, which thousands of developers (including several Unity employees) used to prepare for the UCP exam from 2018–2022.

Prior to joining the Games faculty at Michigan State, Jeremy taught for three years as a lecturer in the Electrical Engineering and Computer Science department at the University of Michigan Ann Arbor where he taught game design and software development. From 2009–2013, Jeremy was an assistant professor teaching game design for the Games and Interactive Media Division of the University of Southern California's School of Cinematic Arts, which was named the #1 game design school in North America throughout his tenure there.

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.

Figure Credits

Cover image by Rost9/Shutterstock

Figure 1-2: Jason Rohrer

Figure 2-1: Robin Hunicke

Figure 2-3, Figure 3-1, Figure 3-2: Taylor & Francis Group

Figure 4-1: Thatgamecompany, Inc

Figure 5-2: Richard A. Bartle

Chapter 5, Zork screengrab: Infocom

Figure 7-1: Elsevier

Figure 8-5: Mattie Brice

Figure 9-3, Figure 9-5: Nintendo

Figure 10-2: You Run Ltd

Figure 19-3, Figure 25-7, Figure 25-10, Figure 25-11, Figure 25-12 (bottom), Figure 26-1, Figure 26-3, Figure 35-5, Figure B-7: Microsoft Corporation

Figure 12-1, Figure 12-5: Sony Interactive Entertainment

Figure 12-2, Figure 12-3: Naughty Dog

Figure 12-4, Figure 12-6: Eden Games

Figure 13-1, Figure 13-2: Scott Kim

Figure 16-1, Figure 16-2, Figure 17-1–Figure 17-10, Figure 19-1, Figure 19-4–Figure 19-14, Figure 20-1, Figure 23-1, Figure 23-2, Figure 25-1–Figure 25-6, Figure 25-8, Figure 25-9, Figure 26-2, Figure 27-1–Figure 27-8, Figure 28-3, Figure 28-5–Figure 28-7, Figure 28-9–Figure 28-11, Figure 28-12B, Figure 28-13, Figure 28-14, Figure 29-1–Figure 29-9, Figure 29-11–Figure 29-15, Figure 30-1–Figure 30-7, Figure 30-10–Figure 30-17, Figure 31-1–Figure 31-10, Figure 32-1, Figure 32-2, Figure 32-3c, Figure 32-4–Figure 32-12, Figure 33-1, Figure 33-4–Figure 33-11, Figure 33-13–Figure 33-15, Figure 34-1–Figure 34-9, Figure 35-3, Figure 35-4, Figure 35-8, Figure 35-10–Figure 35-12, Figure 36-2–Figure 36-7, Figure 37-1, Figure 37-2, Figure A-1–Figure A-7, Figure B-1, Figure B-2, Figure B-4, Figure B-5, Figure B-9, Figure C-1: Unity Technologies

Figure 19-2, Figure 25-12 (top): Apple Inc

Figure 11-1–Figure 11-8, Figure 11-13, Figure 11-15–Figure 11-18, Figure 14-1–Figure 14-7, Figure 28-12A, Figure B-6: Google LLC

Figure 30-9: Skyrates

Figure 33-1A, Figure 33-2, Figure 33-3, Figure 33-7A, Figure 33-9A, Figure 33-11A, Figure 33-13A, Figure 33-15A: Chris Aguilar

Figure 35-1, Figure 35-2, Figure 35-6, Figure 35-9, Figure 35-13, Figure 35-14, Figure 36-1: SKIPSTONE PICTURES

PART I

GAME DESIGN AND PAPER PROTOTYPING

This page intentionally left blank

CHAPTER 1

THINKING LIKE A DESIGNER

Our journey starts here. This chapter presents the basic theories of design upon which the rest of the book is built. In this chapter, you also encounter your first game design exercise and learn more about the underlying philosophy of this book.

You Are a Game Designer

As of this moment, you are a game designer, and I want you to say it out loud:[1]

> "I am a game designer."

It's okay. You can say it out loud, even if other people can hear you. In fact, according to psychologist Robert Cialdini's book, *Influence: The Psychology of Persuasion*,[2] if other people hear you commit to something, you're more likely to follow through. So, go ahead and tell your friends, tell your family, shout it from the mountain tops, post it to social media:

> "I am a game designer!"

But, what does it mean to be a game designer? This book will help you answer that question *and* will give you the tools to start making your own games. Let's start with a design exercise.

Bartok: A Game Design Exercise

I first saw this exercise used by game designer Malcolm Ryan as part of a Game Design Workshop session at the Foundations of Digital Gaming conference. The goal of this exercise is to demonstrate how even a simple change to the rules of a game can have a massive effect on the experience of playing the game.

Bartok is a simple game played with a single deck of standard cards that is very similar to the commercial game *Uno*. In the best-case scenario, you would play this game with three friends who are also interested in game design; however, I've also made a digital version of the game that you can play solo. Either the paper or digital version will work fine for our purposes.[3]

---

1. I thank my former professor Jesse Schell for asking me to make this statement publicly in a class full of people. He also includes this request in his excellent book, *The Art of Game Design: A Book of Lenses* (Boca Raton, FL: CRC Press, 2008).
2. Robert B. Cialdini, *Influence: The Psychology of Persuasion* (New York: Morrow, 1993).
3. The card images in this book and in the digital card games presented in the book are based on Vectorized Playing Cards 1.3, Copyright 2011, Chris Aguilar, https://sourceforge.net/projects/vector-cards/ . Licensed under LGPL 3 (http:// www.gnu.org/copyleft/lesser.html).

PLAYING THE DIGITAL VERSION OF BARTOK

To play the digital version of *Bartok*, simply visit the website for this book:

http://book.prototools.net

You will find the game in the section of the website for Chapter 1.

You can, of course, also just grab a standard deck of playing cards and a few friends and play the game in person, which will allow you to talk with your friends about the feel of the game and the changes you want to make to it.

Objective

Be the first player to get rid of all the cards in your hand.

Getting Started

Here are the basic rules for *Bartok*:

1. Start with a regular deck of playing cards. Remove the Jokers, leaving you with 52 cards (13 of each suit ranked Ace–King).
2. Shuffle the deck and deal seven cards to each player.
3. Place the rest of the cards face-down in a *draw pile*.
4. Pick the top card from the draw pile and place it on the table face-up to start the *discard pile*.
5. Starting with the player to the left of the dealer and proceeding clockwise, each player must play a card onto the discard pile if possible, and if they cannot play a card, the player must draw a single card from the draw pile (see Figure 1.1).
6. A player may play a card onto the discard pile if the card is either:
   a. The same *suit* as the top card of the discard pile. (For example, if the top card of the discard pile is a 2 of Clubs (2C), any other Club may be played onto the discard pile.)
   b. The same *rank* as the top card of the discard pile. (For example, if the top card of the discard pile is a 2C, any other 2 may be played onto the discard pile.)
7. The first player to successfully get rid of all their cards wins.

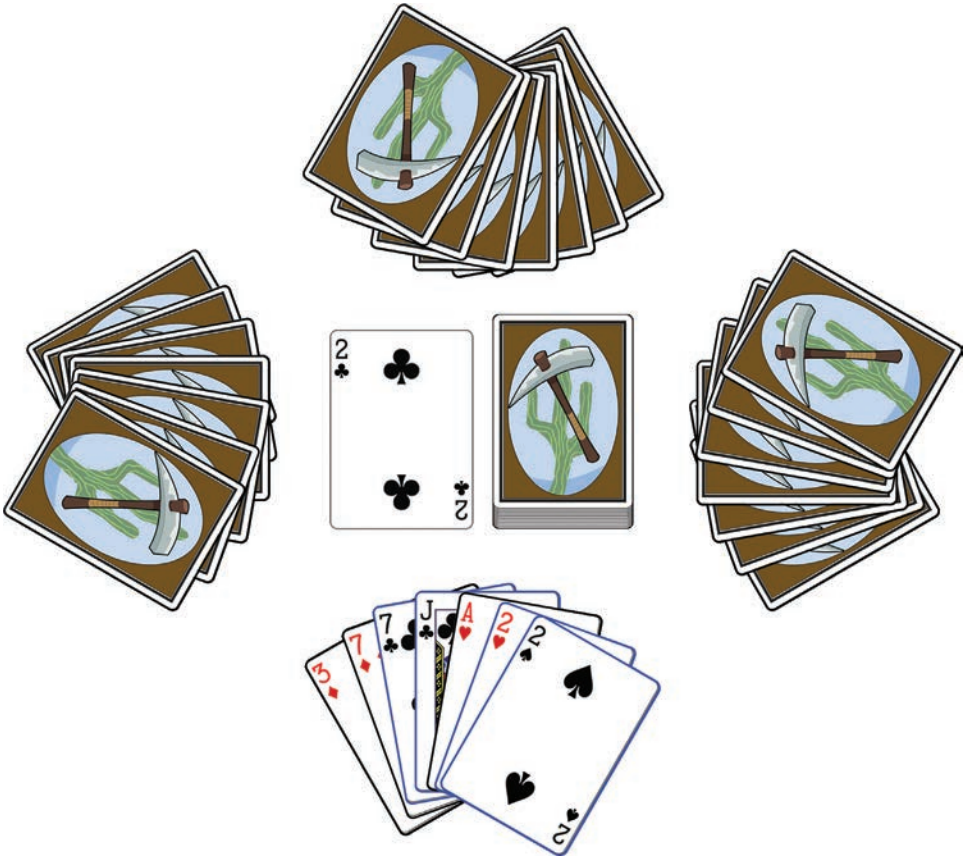

Figure 1.1 The initial layout of *Bartok*. In the situation shown, the player can choose to play any one of the cards highlighted with blue borders (7C, JC, 2H, 2S).

Playtesting

Try playing the game a couple times to get a feel for it. Be sure to shuffle the cards thoroughly between each playthrough. Games will often result in a somewhat sorted discard pile, and without a good shuffle, subsequent games may have results weighted by the nonrandom post-game card distribution.

tip

DEBLOCKING *Deblocking* is the term for strategies used to break up blocks of cards (i.e., groups of similar cards). In *Bartok*, each successful game ends with all the cards sorted into blocks of the same suit and blocks of the same rank. If you don't deblock those groups, the subsequent game will end much faster because players are more likely to be dealt cards that match each other.

According to mathematician and magician Persi Diaconis, seven good *riffle*[4] *shuffles* should be sufficient for nearly all games;[5] if you run into issues, though, some of these deblocking strategies can help.

Here are some standard strategies for deblocking a deck of cards if standard shuffling doesn't work:

- Deal the cards into several different piles. Then shuffle these piles together.
- Deal the cards out face-down into a large, spread-out pool. Then use both hands to move the cards around almost like mixing water. This is how dominoes are usually shuffled, and it can help break up your card blocks. Then gather all the cards into a single stack.
- Play *52 Pickup*: Throw all the cards on the floor and pick them up.

Analysis: Asking the Right Questions

After each playtest, it's important to ask the right questions. Of course, each game will require slightly different questions, though many of them will be based on these general guidelines:

- **Is the game of the appropriate difficulty for the intended audience?** Is it too difficult, too easy, or just right?

4. A riffle shuffle is one where half of the deck starts in each hand and you bend the cards up with the thumb and hold the cards down with the index finger of each hand, causing the cards from the left and right to alternate falling into a center pile. See more at https://en.wikipedia.org/wiki/Shuffling#Riffle.
5. Persi Diaconis, "Mathematical Developments from the Analysis of Riffle Shuffling," *Groups, Combinatorics and Geometry,* edited by Ivanov, Liebeck, and Saxl. *World Scientific* (2003): 73–97. Also available online at http://statweb.stanford.edu/~cgates/PERSI/papers/Riffle.pdf.

- **Is the outcome of the game based more on strategy or chance?** Does randomness play too strong a role in the game, or, alternatively, is the game too deterministic so that after one player takes the lead, the other players don't have any chance to catch up?
- **Does the game have meaningful, interesting decisions?** When it's your turn, do you have several choices, and is the decision between those choices an interesting one?
- **Is the game interesting when it's not your turn?** Do you have any effect on the other players' turns, or do their turns have any immediate effect on you?

We could ask many other questions, but these are some of the most common.

Take a moment to think about your answers to these questions relative to the games of *Bartok* you just played and write them down. If you're playing the paper version of this game with other human players, asking them to write down their own answers to the questions individually and then discussing them after they're written is worthwhile, because it keeps each player's responses from being influenced by the other players.

Modifying the Rules

As you'll see throughout this book, from a process standpoint, game design is pretty straightforward. The process is almost always:

1. Incrementally modify the rules, changing very few things between each playtest.
2. Playtest the game with the new rules.
3. Analyze how the feel of the game is altered by the new rules.
4. Design new rules that you think might move the feel of the game in the direction you want.
5. Repeat this process until you're happy with the game.

Iterative design is the term for this repetitive process of deciding on a small change to the game design, implementing that change, playtesting the game, analyzing how the change affected the gameplay, and then starting the process over again by deciding on another small change. Chapter 7, "Acting Like a Designer," covers iterative design in detail.

For the *Bartok* example, why don't you start by picking one of the following three rule changes and playtesting it:

- **Rule 1:** If a player plays a 2, the person to her left must draw two cards instead of playing.
- **Rule 2:** If any player has a card that matches the rank and color (red or black) of the top card, they may announce "Match card!" and play it out of turn. Play then

continues with the player to the left of the one who just played the out-of-turn card. This can lead to players having their turns skipped.

For example: In a four-player game, the first player plays a 3C (three of Clubs). The third player has the 3S (which matches both the rank and color of the 3C), so they call "Match card!" and play the 3S on top of the 3C out-of-turn, skipping the second player's turn. Play then continues with the fourth player.

- **Rule 3:** A player must announce "Last card" when they have only one card left. If someone else calls it first, the player must draw two cards (bringing their total number of cards to three).

Choose only one of the rule changes from the previous listing and play the game a couple times with the new rule. Then have each player write their answers to the four playtest questions. You should also try playing with another one of the rules (although I would recommend still only using one of them at a time when trying a new rule for the first time).

If you're playing the digital version of the game, you can use the check boxes on the menu screen to choose various game options.

> warning
>
> **WATCH OUT FOR PLAYTESTING FLUKES** A weird shuffle or other external factor can sometimes cause a single play through the game to feel really different from the others. This is known as a *fluke*, and you want to be careful not to make game design decisions based on flukes. If something you do seems to affect the game feel in a very unexpected way, be sure to play through the game multiple times with those same rules to make sure you're not experiencing a fluke.

Analysis: Comparing the Rounds

Now that you've played through the game with some different rule options, analyze the results from the different rounds. Look back over your notes and see how each different rule set felt to play. As you experienced, even a simple rule change can greatly change the feel of the game. Here are some common reactions to the previously listed rules:

- **The original rules**

  Many players find the original version of the game to be pretty boring. There are no interesting choices to make, and as the players remove cards from their hands, the number of possible choices dwindles as well, often leaving the player with only one valid choice for most of the later turns of the game. The game is largely based

on chance, and players have no real reason to pay attention to other players' turns because they don't really have any way of affecting each other.

- **Rule 1:** *If a player plays a 2, the person to her left must draw two cards instead of playing.*

  This rule allows players to directly affect each other, which generally increases interest in the game. However, whether a player has 2s is based entirely on luck, and each player only really has the ability to affect the player on their left, which often seems unfair. However, this does make other players' turns a bit more interesting because other players (or at least the player to your right) have the ability to affect you.

- **Rule 2:** *If any player has a card that matches the number and color (red or black) of the top card, they may announce "Match card!" and play it out of turn. Play then continues with the player to the left of the one who just played the out-of-turn card.*

  This rule often has the greatest effect on player attention. Because any player has the opportunity to interrupt another player's turn, all players tend to pay a lot more attention to each other's turns. Games played with this rule often feel more dramatic and exciting than those played with the other rules.

- **Rule 3:** *A player must announce "Last card!" when they have only one card left. If someone else calls it first, the player must draw two cards.*

  This rule only comes into play near the end of the game, so it doesn't have any effect on the majority of gameplay; however, it does change how players behave at the end. This can lead to some interesting tension as players try to jump in and say, "last card" before the player who is down to only one card. This is a common rule in both domino and card games where the players are trying to empty everything from their hands because it gives other players a chance to catch up to the lead player if the leader forgets about the rule.

Designing for the Game Feel That You Want

Now that you've seen the effects of a few different rules on *Bartok*, it's time to do your job as a designer and make the game better. First, decide on the feel that you want the game to have: Do you want it to be exciting and cutthroat, do you want it to be leisurely and slow, or do you want it to be based more on strategy than chance?

After you have a general idea of how you want the game to feel, think about the rules that you tested and try to come up with additional rules that can push the feel of the game in the direction that you want. Here are some tips to keep in mind as you design new rules for the game:

- Change only one thing in between each playtest. If you change (or even tweak) a number of rules between each play through the game, it can be difficult to determine which rule is affecting the game in what way. Keep your changes incremental, and you'll be better able to understand the effect that each is having.

- The bigger change you make, the more playtests will be required to understand how it changes the game feel. If you only make a subtle change to the game, one or two plays can tell you a lot about how that change affects the feel. However, if it's a major rule change, you will need to test it more times to avoid being tricked by a fluke game. Additionally, if the small rule change only happens in rare circumstances, you also may need multiple plays through the game to experience that circumstance.
- Change a number, and you change the experience. Even a seemingly small change can have a huge effect on gameplay. For instance, think about how much faster this game would end if there were two discard piles to choose from or if the players started with five cards instead of seven.

Of course, adding new rules is a lot easier to do when playing the card game in person with friends than when working with a digital prototype. That's one of the reasons that paper prototypes can be so important, even when you're designing digital games. The first part of this book discusses both paper and digital design, but most of the design exercises are done with paper games because they can be so much faster to develop and test than digital games.

The Definition of *Game*

Before moving too much further into design and iteration, we should probably clarify what we're talking about when we use terms such as *game* and *game design.* Many very smart people have tried to accurately define the word *game*. Here are a few of them in chronological order:

- In his 1978 book *The Grasshopper,* Bernard Suits (who was a professor of philosophy at the University of Waterloo) declares that "a game is the voluntary attempt to overcome unnecessary obstacles."[6]
- Game design legend Sid Meier says that "a game is a series of interesting choices."[7]
- In *Game Design Workshop,* Tracy Fullerton defines a game as "a closed, formal system that engages players in a structured conflict and resolves its uncertainty in an unequal outcome."[8]
- In *The Art of Game Design,* Jesse Schell playfully examines several definitions for *game* and eventually decides on "a game is a problem-solving activity, approached with a playful attitude."[9]

6. Bernard Suits, *The Grasshopper: Games, Life, and Utopia* (Toronto: Toronto University Press, 1978), 41.
7. Andrew Rollings and Dave Morris. *Game Architecture and Design* (Scottsdale: Coriolis, 2000), 38.
8. Tracy Fullerton, Christopher Swain, and Steven Hoffman. *Game Design Workshop: A Playcentric Approach to Creating Innovative Games*, 2nd ed. (Boca Raton, FL: Elsevier Morgan Kaufmann, 2008), 43.
9. Jesse Schell, *Art of Game Design: A Book of Lenses* (Boca Raton, FL: CRC Press, 2008), 37.

- In the book *Game Design Theory,* Keith Burgun presents a much more limited definition of game: "a system of rules in which agents compete by making ambiguous, endogenously meaningful decisions."[10, 11]

As you can see, all of these are compelling and correct in their own way. Perhaps even more important than the individual definition is the insight that it gives us into each author's intent when crafting that definition.

Bernard Suits' Definition

In addition to the short definition "a game is the voluntary attempt to overcome unnecessary obstacles," Suits also offers a longer, more robust version:

> *To play a game is to attempt to achieve a specific state of affairs, using only means permitted by rules, where the rules prohibit use of more efficient in favor of less efficient means, and where the rules are accepted just because they make possible such activity.*[12]

Throughout his book, Suits proposes and refutes various attacks on this definition; and having read the book, I am certainly willing to say that he has found the definition of "game" that most accurately matches the way that the word is used in day-to-day life.

However, it's also important to realize that this definition was crafted in 1978, and even though digital games and roleplaying games existed at this time, Suits was either unaware of them or intentionally ignored them. In fact, in Chapter 9 of *The Grasshopper,* Suits laments that there is no kind of game with rules for dramatic play through which players could burn off dramatic energy (much like children can burn off excess athletic energy via play of any number of different sports), exactly the kind of play that was enabled by games like *Dungeons & Dragons*.[13]

Although this is a small point, it gets at exactly what is missing from this definition: Whereas Suits' definition of game is an accurate definition of the word, it offers nothing to designers seeking to craft good games for others.

---

10. Keith Burgun. *Game Design Theory: A New Philosophy for Understanding Games* (Boca Raton, FL: A K Peters/CRC Press, 2013), 10, 19.
11. *Endogenous* means inherent to or arising from the internal systems of a thing, so "endogenously meaningful decisions" are those decisions that actually affect the game state and change the outcome. Choosing the color of your avatar's clothing in *Farmville* is not endogenously meaningful, whereas choosing the color of your clothing in *Metal Gear Solid 4* is, because the color of your clothing affects whether your avatar is visible to enemies.
12. Bernard Suits, *The Grasshopper: Games, Life, and Utopia* (Toronto: Toronto University Press, 1978), 41.
13. Suits, *The Grasshopper,* 96.

For an example of what I mean, take a moment to play Jason Rohrer's fantastic game *Passage*[14] (see Figure 1.2). The game only takes 5 minutes to play, and it does a fantastic job of demonstrating the power that even short games can have. Try playing through it a couple times. If you can't find a playable version for your computer, try watching some videos online, though playing it yourself is certainly better.

Figure 1.2 *Passage* by Jason Rohrer (released December 13, 2007)

Suits' definition will tell you that, yes, this is a game. In fact, it is specifically an "open game," which he defines as: a game that has as its sole goal the continuance of the game.[15] In *Passage*, the goal is to continue to play for as long as possible...or is it? *Passage* has several potential goals, and it's up to the player to choose which of these they want to achieve. These goals could include the following:

- Moving as far to the right as possible before dying (exploration)
- Earning as many points as possible by finding treasure chests (achievement)
- Finding a wife (socialization)

The point of *Passage* as an artistic statement is that each of these can be a goal in life, and to some extent, these goals are all mutually exclusive. If you find a wife early in the game, getting treasure chests becomes more difficult because the two of you are unable to enter areas that could be entered singly. If you choose to seek treasure, you will spend your time exploring the vertical space of the world and won't be able to see the different scenery to the right. If you choose to move as far to the right as possible, you won't rack up nearly as much treasure.

In this incredibly simple game, Rohrer exposes a few of the fundamental decisions that every one of us must make in life and demonstrates how even early decisions can have a major effect on the rest of our lives. The important thing here is that he is giving players choice and demonstrating to them that their choices matter.

14. *Passage* is downloadable from Rohrer's website at http://hcsoftware.sourceforge.net/passage/, or you can find an online version at http://passage.toolness.org/.

15. Suits contrasts these with closed games, which have a specific goal (e.g., crossing a finish line in a race or ridding yourself of all your cards in *Bartok*). Suits' example of an open game is the game of make-believe that children play.

This is an example of the first of a number of designer's goals that I will introduce in this book: *experiential understanding.* Whereas a linear story like a book can encourage empathy with a character by exposing the reader to the character's life and the decisions that they have made, games can allow players to understand not only the outcome of decisions but also to be complicit in that outcome by giving the player the power and the responsibility of decision and then showing them the outcome wrought by their decisions. Chapter 8, "Design Goals," explores these in much greater depth.

Sid Meier's Definition

By stating that "a game is a series of interesting choices," Meier is saying very little about the definition of the word *game* (there are many, many things that could be categorized as a series of interesting choices and yet are not games) and quite a bit about what he personally believes makes for a good game. As the designer of games such as *Pirates*, *Civilization*, *Alpha Centauri*, and many more, Sid Meier is one of the most successful game designers alive, and he has consistently produced games that present players with interesting choices. This, of course, raises the question of what makes a choice or decision *interesting.* An interesting decision is generally one where:

- The player has multiple valid options from which to choose.
- Each option has both positive and negative potential consequences.
- The outcome of each option is predictable but not guaranteed.

This brings up the second of our designer's goals: to create *interesting decisions.* If a player is presented with a number of choices, but one choice is obviously superior to the others, the experience of deciding which to choose doesn't actually exist. If a game is designed well, players will often have multiple choices from which to choose, and the decision will often be a tricky one.

Tracy Fullerton's Definition

As she states in her book, Tracy Fullerton is much more concerned with giving designers tools to make better games than she is with the philosophical definition of *game.* Accordingly, her definition of a game as "a closed, formal system that engages players in a structured conflict and resolves its uncertainty in an unequal outcome" is not only a good definition of *game* but also a list of elements that designers can modify in their games:

- **Formal elements:** The elements that differentiate a game from other types of media: rules, procedures, players, resources, objectives, boundaries, conflict, and outcome.
- **(Dynamic) systems:** Methods of interaction that evolve as the game is played.

- **Conflict structure:** The ways in which players interact with each other.
- **Uncertainty:** The interaction between randomness, determinism, and player strategy.
- **Unequal outcome:** How does the game end? Do players win, lose, or something else?

Another critical element in Fullerton's book is her continual insistence on *actually making games.* The only way to become a better game designer is to make games. Some of the games you'll design will probably be pretty awful—some of mine certainly have been—but even designing a terrible game is a learning process, and every game you create will improve your design skills and help you better understand how to make great games.

Jesse Schell's Definition

Schell defines a game as "a problem-solving activity, approached with a playful attitude." This is similar in many ways to Suits' definition, including its consideration of the player's perspective. According to both, it is the playful attitude of the player that makes something a game.

Suits argues in his book that two people could both be involved in the same activity, and to one, it would be a game, whereas to the other, it would not be. His example is a foot race where one runner is just running because she wants to take part in the race, but the other runner knows that at the finish line there is a bomb they must defuse before it explodes. According to Suits, although the two runners would both be running in the same foot race, the one who is simply racing would follow the rules of the race because of what Suits calls her *lusory attitude*. On the other hand, the bomb-defusing runner would break the rules of the game the first chance they got because they have a serious attitude (as is required to defuse a bomb) and are not engaged in the game. *Ludus* is the Latin word for play, so Suits proposes the term *lusory attitude* to describe the attitude of one who willingly takes part in playing a game.

It is because of their lusory attitude that players will happily follow the rules of a game even though there may be an easier way to achieve the stated goal of the game (what Suits would call the *pre-lusory goal*). For example, the pre-lusory goal of golf is to get the golf ball into the cup, but there are many easier ways to do so than to stand hundreds of yards away and hit the ball with a bent stick. When people have a lusory attitude, they set challenges for themselves just for the joy of overcoming them.

So, another design goal is to *encourage a lusory attitude.* You should design your game to encourage players to enjoy the limitations placed on them by the rules. Think about why each rule is there and how it changes the player experience. If a game is balanced well and has the proper rules, players will enjoy the limitations of the rules rather than feel exasperated by them.

Keith Burgun's Definition

Burgun's definition of a game as "a system of rules in which agents compete by making ambiguous, endogenously meaningful decisions" is his attempt to push the discourse on games forward from a rut that he feels it has fallen into by narrowing the meaning of game down to something that can be better examined and understood. The core of this definition is that the player is making choices and that those choices are both ambiguous (the player doesn't know exactly what the outcome of the choice will be) and endogenously meaningful (the choice is meaningful because it has a noticeable effect upon the game system).

Burgun's definition is intentionally limited and purposefully excludes several of the things that many people think of as games (including foot races and other competitions based on physical skill) as well as reflective games like *The Graveyard*, by Tale of Tales, in which the player experiences wandering through a graveyard as an old woman. Both of these are excluded because the decisions in them lack ambiguity and endogenous meaning.

Burgun chooses such a limited definition because he wants to get down to the essence of games and what makes them unique. In doing so, he makes several good points, including his statement that whether an experience is fun has little to do with the question of whether it is a game. Even a terribly boring game is still a game; it's just a bad game.

In my discussions with other designers, I have found that a lot of contention can exist around this question of what types of things should fall under the term *game.* Games are a medium that has experienced a tremendous amount of growth, expansion, and maturation over the last few decades, and the explosion of independent game development this decade has only hastened the pace. Today, more people than ever before—with disparate voices and varied backgrounds—are contributing work to the field of games, and as a result, the definition of the medium is expanding, which is understandably bothersome to some people because this expanding definition can be seen as blurring the lines of what is considered a game. Burgun's response to this is his concern that it is difficult to rigorously advance a medium if we lack a good definition of what the medium is. I'll come back to this topic in a little while.

Why Care About the Definition of Game?

In his 1953 book *Philosophical Investigations,* Ludwig Wittgenstein proposed that the term *game*, as it is used colloquially, had come at that time to refer to several very different things that shared some traits (which he likened to a family resemblance) but couldn't be encapsulated in a single definition. In 1978, Bernard Suits attacked this idea by using his book, *The Grasshopper,* to argue very stringently for the specific definition of game that you read earlier in this chapter. However, as Chris Bateman points out in his book *Imaginary Games,* though Wittgenstein used the word *game* as his example,

he was really trying to make a larger point: the point that words are created to define things rather than things being created to meet the definition of words.

In 1974 (between the publications of *Philosophical Investigations* and *The Grasshopper*), the philosopher Mary Midgley published a paper titled, "The Game Game," in which she explored and refuted the "family resemblance" claim by Wittgenstein not by arguing for a specific definition of game herself but instead by exploring why the word *game* existed. In her paper, she agrees with Wittgenstein that the word *game* came into being long after games existed, but she makes the statement that words like *game* are not defined by the *things* that they encompass but instead by the *needs* that they meet. As she states:

> *Something can be accepted as a chair provided it is properly made for sitting on, whether it consists of a plastic balloon, a large blob of foam, or a basket slung from the ceiling. Provided you understand the need you can see whether it has the right characteristics, and aptness for that need is what chairs have in common.*[16]

In her paper, Midgley seeks to understand some of the needs that games fulfill. She completely rejects the idea that games are closed systems by both citing many examples of game outcomes that have effects beyond the game and pointing out that games cannot be closed because humans have a reason for entering into them. To her, that reason is paramount. The following are just a few reasons for playing games:

- **Humans desire structured conflict:** As Midgley writes, "The Chess Player's desire is not for general abstract intellectual activity, curbed and frustrated by a particular set of rules. It is a desire for a particular kind of intellectual activity, whose channel is the rules of chess." As Suits pointed out in his definition, the rules that limit behavior are there precisely because the challenge of those limitations is appealing to players.
- **Humans desire the experience of being someone else:** We are all acutely aware that we have but one life to live (or at least one at a time), and play can allow us to experience another life. Just as a game of *Call of Duty* allows a player to pretend to experience the life of a soldier, so too does *The Graveyard* allow the player to pretend to experience the life of an old woman, and playing the role of Hamlet allows an actor to pretend to experience the life of a troubled Danish prince.
- **Humans desire excitement:** Much popular media is devoted to this desire for excitement, be it action films, courtroom dramas, or romance novels. The thing that makes games different in this regard is that the player is actively taking part in the excitement rather than vicariously absorbing it, which is the only option for the majority of linear media. As a player, you aren't watching someone else be chased by zombies, you're being chased yourself.

16. Mary Midgley. "The Game Game," *Philosophy 49,* no. 189 (1974): 231–53.

Midgley found it critical to consider the needs that are fulfilled by games in order to understand both their importance in society and the positive and negative effects that games can have on the people who play them. Both Suits and Midgley spoke about the potentially addictive qualities of games in the 1970s, long before video games became ubiquitous and public concern emerged about players becoming addicted. As game designers, it is useful for us to understand these needs and respect their power.

The Nebulous Nature of Definitions

As Midgley pointed out, it is useful to think of the word *game* as being defined by the need that it fills. However, she also stated that a chess player doesn't want to play just any kind of game; they specifically want to play chess. Not only is it difficult to come up with an all-encompassing definition for game, it's also true that the same word will mean different things to different people at different times. When I say that I'm going to play a game, I usually mean a console or video game; when my wife says the same thing, though, she usually means *Scrabble* or another word game. When my parents say they want to play a game, it means something like Alan R. Moon's *Ticket to Ride* (a board game that is interesting but doesn't require players to be overly competitive with each other), and my in-laws usually mean a game of cards or dominoes when they use the word. Even within our family, the word has great breadth.

The meaning of the word *game* is also constantly evolving. When the first computer games were created, no one could have possibly imagined the multi-billion-dollar industry that we now have or the rise of the fantastic indie renaissance that we've seen over the past decade. All that they knew was that these things people were doing on computers were kind of like tabletop war board games (I'm thinking of *Space War* here), and these new games were called "computer games" to differentiate them from the pre-existing meanings of *game*.

The evolution of digital games was a gradual process with each new genre building in some way on the ones that had come before, and along the way, the term *game* expanded further and further to encompass all of them.

Now, as the art form matures, many designers are entering the field from various other disciplines and bringing with them their own concepts about what can be created with the technologies and design methodologies that have been developed to make digital games. (You may even be one of them.) As these new artists and designers enter the space, some of them are making things that are very different from what we think of as a stereotypical game. That's okay; in fact, I think it's fantastic! And, this isn't just my opinion. IndieCade, the international festival of independent games, seeks every year to find games that push the envelope of what is meant by *game.* According to Festival

Chair Celia Pearce and Festival Director Sam Roberts, if an independent developer wants to call the interactive piece that they have created a game, IndieCade will accept it as one.[17]

Summary

After all these interwoven and sometimes contradictory definitions, you may be wondering why this chapter has spent so much time exploring the definition of the word *game.* I have to admit that in my day-to-day work as an educator and game designer, I don't spend a lot of time wrestling with the definitions of words. As Shakespeare points out, were a rose to be named something else, it would still smell as sweet, still have thorns, and still be a thing of fragile beauty. However, I believe that an understanding of these definitions can be critical to you as a designer in the following three ways:

- Definitions help you understand what people expect from your games. This proves especially true if you're working in a specific genre or for a specific audience. Understanding how your audience defines the term will help you to craft better games for them.
- Definitions can lead you to understand not only the core of the defined concept but also the periphery (i.e., games that fit the definition perfectly (the core) and games that just barely fit the definition (the periphery)). As you read through this chapter, you encountered several different definitions by different people, and each had both a core and a periphery. The places where these peripheries don't mesh can be hints at some of the interesting areas to explore with a new game. For example, the area of disagreement between Fullerton and Midgley about whether a game is a closed system highlights the previously untracked ground that in the 2000s grew into alternate reality games (ARGs), a genre centered on perforating the closed magic circle of play.[18]
- Definitions can help you speak eloquently with others in the field. This chapter has more references and footnotes than any other in the book because I want you to be able to explore the philosophical understanding of games in ways that are beyond the scope of this one book (especially since this book is really focused on the practicalities of actually making digital games). Following these footnotes and reading the source material can help improve the critical thinking that you do about games.

17. This was stated during the Festival Submission Workshop given by Celia Pearce and Sam Roberts at IndieCade East 2014 and is paraphrased on the IndieCade submissions website at https://www.indiecade.com/submissions-help-section/eligibility/#mean-by-game (accessed June 14, 2021).

18. The first large-scale ARG was *Majestic* (Electronic Arts, 2001), a game that would phone players in the middle of the night and send them faxes and emails. Smaller-scale ARGs include the game *Assassin*, which is played on many college campuses, where players can "assassinate" each other (usually with Nerf or water guns, or by snapping a photo) any time that they are outside of classes. One of the fun aspects of these games is that they are always happening and can interfere with normal life.

The Core Lessons of This Book

This book will actually teach you how to design a lot more than just games. In fact, it will teach you how to craft any kind of *interactive experience*. As I define it:

> *An interactive experience is any experience created by a designer; inscribed into rules, media, or technology; and decoded by people through play.*

That makes *interactive experience* a pretty expansive term. In fact, any time that you attempt to craft an experience for people—whether you're designing a game, planning a surprise birthday party, or even planning a wedding—you're using the same tools that you will learn as a game designer. The processes that you will learn in this book are more than just the proper way to approach game design. They are a meaningful way to approach any design problem, and the iterative process of design that is introduced in Chapter 7, "Acting Like a Designer," is *the* essential method for improving the quality of any design.

No one bursts forth from the womb as a brilliant game designer. My friend Chris Swain[19] is fond of saying that "Game design is 1% inspiration and 99% iteration," a play on the famous quote by Thomas Edison. He is absolutely correct, and one of the great things about game design (unlike the previously mentioned examples of the surprise party and the wedding) is that you get the chance to iterate on your designs, to playtest the game, make subtle tweaks, and play it again. With each prototype you make—and with each iteration of your prototypes—your skills as a designer will improve. Similarly, once you reach the parts of this book that teach digital development, be sure to keep experimenting and iterating. The code samples and tutorials are designed to show you how to make a playable game prototype, but every tutorial in this book will end where your work as a designer should begin. Each one of these prototypes could be built into a larger, more robust, better balanced game, and I encourage you to do so.

Moving Forward

Now that you've experienced a bit of game design and explored various definitions of *game,* it's time to move on to a more in-depth exploration of a few different analytical frameworks that game designers use to understand games and game design. The next chapter explores various frameworks that have been used over the past several years, and the chapter that follows synthesizes those into the framework used throughout the remainder of this book.

19. Chris Swain co-wrote the first edition of *Game Design Workshop* with Tracy Fullerton and taught the class of the same name at the University of Southern California for many years, which I took over from him in 2009. He is now an entrepreneur and independent game designer.

INDEX

Symbols

Numbers

A

B

C

D

E

F

G

H

I

J

K

L

M

N

P

Q

R

S

T

U

V

W

X

Y

Z