Distributed Game Development

Harnessing Global Talent to Create Winning Games

Tim Fields

Focal Press

Distributed Game Development

Harnessing Global Talent to Create Winning Games

Tim Fields

ELSEVIER

AMSTERDAM • BOSTON • HEIDELBERG • LONDON
NEW YORK • OXFORD • PARIS • SAN DIEGO
SAN FRANCISCO • SINGAPORE • SYDNEY • TOKYO

Focal Press is an imprint of Elsevier

Focal Press is an imprint of Elsevier
30 Corporate Drive, Suite 400, Burlington, MA 01803, USA
The Boulevard, Langford Lane, Kidlington, Oxford, OX5 1GB, UK

Notices

Knowledge and best practice in this field are constantly changing. As new research and experience broaden our understanding, changes in research methods, professional practices, or medical treatment may become necessary.

Practitioners and researchers must always rely on their own experience and knowledge in evaluating and using any information, methods, compounds, or experiments described herein. In using such information or methods they should be mindful of their own safety and the safety of others, including parties for whom they have a professional responsibility.

To the fullest extent of the law, neither the Publisher nor the authors, contributors, or editors, assume any liability for any injury and/or damage to persons or property as a matter of product liability, negligence or otherwise, or from any use or operation of any methods, products, instructions, or ideas contained in the material herein.

Library of Congress Cataloging-in-Publication Data
Fields, Tim.
Distributed game development: harnessing global talent to create winning games/Tim Fields.
p. cm.
Includes bibliographical references and index.
ISBN 978-0-240-81271-7 (hardcover: alk. paper) 1. Computer games–Programming. 2. Computer software–Development. 3. Electronic data processing–Distributed processing. I. Title.
QA76.76.C672F55 2010
794.8'1526–dc22 2009049368

British Library Cataloguing-in-Publication Data
A catalogue record for this book is available from the British Library.

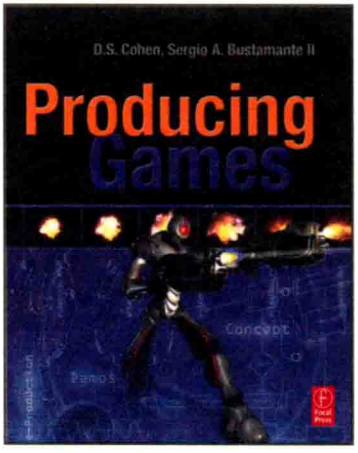
ISBN: 978-0-240-81271-7

For information on all Focal Press publications
visit our website at www.elsevierdirect.com

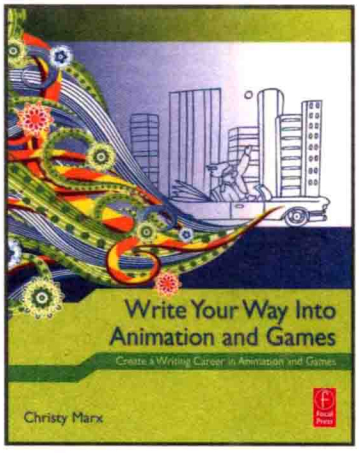
10 11 5 4 3 2 1

Printed in the United States of America

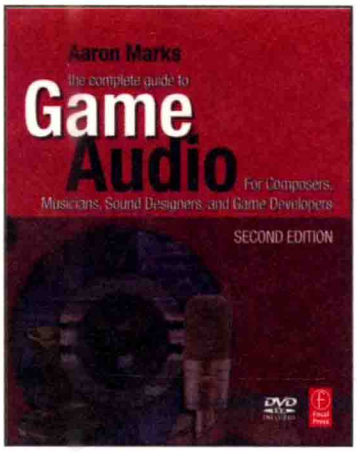
Recent Focal Press Titles available at bookstores and www.focalpress.com



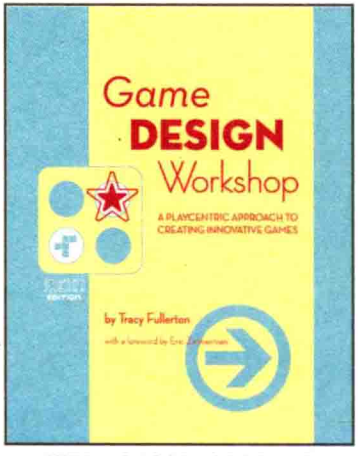
ISBN: 9780240810706



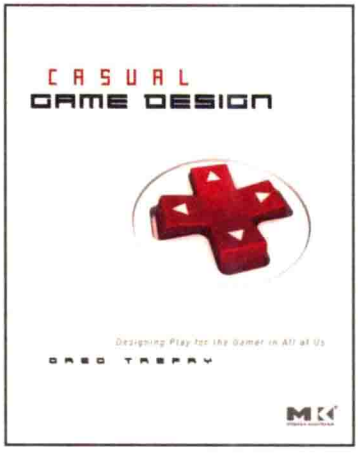
ISBN: 9780240813431



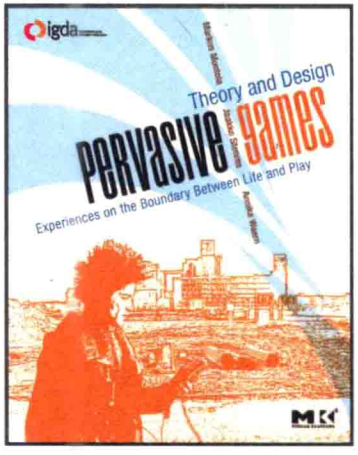
ISBN: 9780240810744



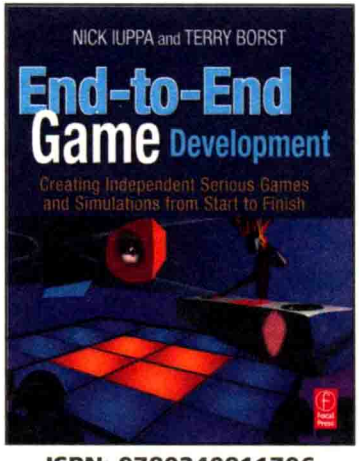
ISBN: 9780240809748



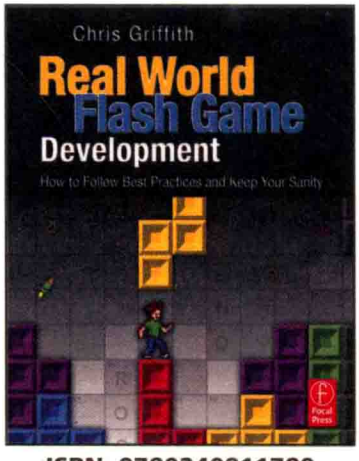
ISBN: 9780123749536



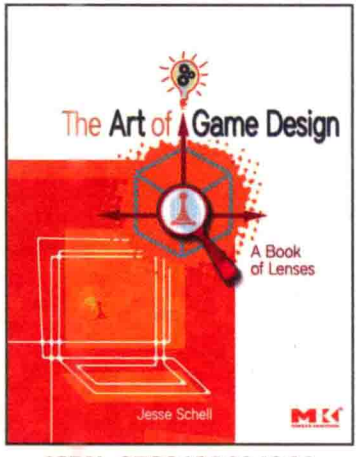
ISBN: 9780123748539



ISBN: 9780240811796



ISBN: 9780240811789



ISBN: 9780123694966

I dedicate this work to all of the wise mentors who have taught me so much over the years: V. Fields, V. Jewett, B. Fregger, E. Boling, D. Stafford, L. Acton, J. Ybarra, DB23, E. Roberts, Alanha, P. Watt, S. Barcia, L. Lapierre, R. Wallace. And always, RS.

About the Author

Tim Fields is a 16-year game industry veteran producer, project manager, design lead, and business developer. Tim has helped small studios and top publishers such as EA and Microsoft run teams that create great games. He has worked on shooters, sports games, racing titles, and RPGs using talent and teams from North America, Asia, Europe, and the United Kingdom. Tim has been involved in one way or another with franchises like Need for Speed, Halo, SSX, Brute Force, and Call of Duty. He loves visiting about game development and design and can be reached at tfields@distributedgamedevelopment.com.

Contents

CHAPTER One

Preface and Overview

1.1 Introduction

Some time over the last 15 years, the geeks won. Bill Gates became the richest man in the world, at least for a while. The personal computer moved to the center of private life for hundreds of millions worldwide. The planet got wired, got flat, and got an e-mail address. Most fun of all, games moved from being a marginalized form of entertainment at the fringe of social acceptance to being mainstream. And beyond mainstream, video games became cool, no longer just the province of the stereotypical outcast adolescent male. Consequently, video games became delightfully profitable.

Hand in hand with this rise to popularity, our entertainment software has become more complicated. Gaming hardware, from PCs to high-end consoles, has become more powerful, and the types of content have become much more involved. Gone are the days of single textured polygons or even basic hardware shaders. Simple LAN-style multiplay is long gone too, and an escalating war of feature brinksmanship is leading to ever more sophisticated ways to play. Input mechanisms have become more varied and sophisticated, with motion-sensing devices like the Wiimote and those fantastic little plastic guitars opening up new types of gameplay to all new audiences. The proliferation of mobile phones capable of running complex software has created new markets for casual gamers who don't even own dedicated high-end hardware. At the same time, our users expect ever more accessible interface models, better matchmaking that is more transparent as well as more accurate, and so on. Finally, the profusion of different platforms, from PC to handhelds, means that it is no longer enough to build a great game on one gaming system. To reach massive commercial success, it often needs to be built for six or seven. As if all of that weren't enough, the marketplace has become so crowded (because of the delightful profits I mentioned previously) that you need to have brilliantly marketed products. Moreover, all versions of those products need to simultaneously hit store shelves on the same day so you can get the most out of

doi: 10.1016/B978-0-240-81271-7.00001-1

those brilliant marketing dollars. Beyond that, you'll need to have downloadable content, expansion packs, and sequel or franchise plans in place so you can ensure that your hit game isn't a flash in the pan but instead starts a franchise dynasty that will have your investors rolling in the Benjamins until the next ice age.

1.2 How Is a Team Leader to Juggle All of This?

Luckily, a few things have evolved to make this daunting task a little easier. First, the software tools we use to create games have gotten better – a lot better, in fact. From modern versions of 3D packages such as Maya to middleware such as Havok or Gamebryo and the software development kits that we use to interact with console platforms, our tools are just plain better. Our defect tracking software has improved, as has the server hardware and the version and source control software. Moreover, there are many more professional game developers now, and our methods of communication have become much more varied and powerful. Gone are the days of firing up a dial-up modem and logging into a BBS to ask technical questions. There are thousands of websites devoted to helping engineers ask questions and wiki-type collaborative projects that serve up vastly better documentation than was common a decade ago.

Finally, our organizational processes have become more advanced. First, we've embraced a level of specialization in many roles (physics engineer, rigger, CG Sup, lighter, etc.) that would have been unheard of when all game developers were expected to be generalists. Second, we've refined some of our management roles and added a layer of facilitators such as development directors and associate producers. Teams can grumble about the introduction of these kinds of middle management roles, but it seems clear that when used properly, they help reduce friction, increase communication, and make possible the feats of coordination required to deliver top-selling titles in such a complex marketplace.

Perhaps the most sweeping change to the way we organize ourselves to facilitate the creation of entertainment software is a process that has only recently come online. It is widely remarked that the world has become "flat." Advances in communication, the opening of global markets, and the widespread adoption of English as the lingua franca assist an educated class that can collaborate across borders regardless of distance or time zones.

To be fair, "collaborate" is my word. And It's chosen to frame our subject in the appropriate light from the outset. Unfortunately, too much of the discussion about our new flat world has centered on fear mongering about lost jobs and discussions of the perils of outsourcing. Although much of this seems to be little more than political pandering in the developed world, it does speak to a greater truth: The ways in which we can turn the world to our advantage are often ill understood. An educated, global workforce with the tools of communication that allow them to collaborate can be used to benefit most industries. The creation of goods and services, like gaming software, is not a zero sum game. It is possible to make better

products, more efficiently, which reach broader markets, and delight a wider range of consumers, if the power of distributed collaboration can be effectively harnessed. We can increase the number of jobs, make better games, and all make more money than we have in years past if we can embrace the options we have available.

In particular, we're going to spend the next few hundred pages visiting about how to harness the global talent pool to create winning games – games that exceed sales expectations, games that thrill our customers, and games that help build sustainable franchises and make a lot of money for our investors and, it is hoped, for you.

This is a book about the organization of teams – about how to make use of a wide, flat world of resources and eager developers in order to accomplish the daunting tasks described previously.

Approximately 15 years ago, I was handed a book on software project management by one of my bosses, software guru David Stafford. The book, the venerable *Debugging the Development Process* by Steve Maguire, is one of the bibles of software development, published at a time when Microsoft Press was working hard to create a world in which a lot more people would understood how to develop professional software for Windows. In the book, Maguire used the metaphor of the software project as a large truck. This big rig is moving, ideally moving fast, and has to be somewhere that (hopefully) everyone has agreed upon in advance. As a leader of this project, it is your job to ensure that the truck does not encounter any roadblocks, does not run out of gas, is not broadsided by another giant truck that wants to monopolize the same section of road, and so on. To extend the metaphor, Maguire likens the software project leader as a member of an advance crew who goes ahead of the truck, looking for likely obstacles that will slow or stop its progress and radioing back course corrections to the folks with their foot on the pedal. I have always liked this metaphor and thought it nicely described the way one should approach software project management.

Only now, things are different. With projects of the complexity we've discussed previously, and a world in which you can and should be distributing the workload among several different teams, your job is more akin to that of central dispatch, or an air traffic controller. To deliver complex entertainment software on time, across a variety of platforms, in a host of different languages, to a bunch of varied markets, all on time and on budget, you need a fleet of different trucks, each manned by capable drivers.

This book will teach you how to direct them all effectively.

1.3 Who Is This Book For?

There are still some games that do not require any sort of distributed development. Let's say that you're a member of a small team building a free web game. Let's imagine that you're all in the same physical space, a small office maybe, and you aren't planning any particular marketing or QA process for your title, outside of

what can be provided by friends or a few local testers. You don't have a publisher – you're self-publishing. You're on the smaller side of indie, and you're comfortable staying that way. If this describes your team, then you likely don't need this book (though I'd hope that you still might find it illuminating, if only to see how big and complex the machine can get). Otherwise, if you are involved professionally in making games that you hope will reach a large audience, you'll be interested in what we'll be talking about here.

Specifically, however, this is a book for project leads or those who hope someday to become project leads. It's for the harried executive producer at one of the top publishers – Activision, Microsoft, Electronic Arts, Tencent, Ubisoft, THQ, or similar – who has just finished another game and can't help but think that there must be a way to bring a little sanity to the process. It's for the development director helping keep a team alive at a small development studio doing work-for-hire for its publisher. It's for the art director of an art outsourcing house. It's for the motion graphics expert at a video production company's gaming division. It's for the lead designer trying to ensure that her vision, her baby for all intents and purposes, gets properly translated across to even the handheld version. It's for the marketing product manager who is trying to get a grip on how to help the development team create a more predictable process and a better Metacritic-rated game.

This is also a book for teachers and students. If you are a student of the games business or in an RTF program who wants to understand more about how modern games are built – and how to find your niche in this fascinating, profitable, dizzying industry – I believe you'll find a lot here. It's also a book for teachers: My goal is for this book to serve as a backbone textbook for courses on production.

Finally, this is a book for investors. If you are one of the millions of people who owns stock in a company that derives profits (or seeks to) from creating games, then this book will teach you a lot about how games are made and how to evaluate what's really happening behind the glossy press releases.

Although this book deals with team organization and the best practices for running software projects at a fairly advanced level, I endeavor to explain industry jargon as clearly as possible for those who are not already familiar with it. If you've never worked on software before, just hang in there – there's a lot to pick up, and you'll likely be amazed at how complex it all gets. But then, the inside of a sausage factory never has been a pretty place. By the time you've finished the tour, however, it won't seem so foreign. Now here's your hardhat. Let's proceed.

1.4 Preamble on Distributed Development

1.4.1 *Why Would I Use Distributed Development?*

There are dozens of reasons why different teams find themselves using a distributed model. It can be a load-balancing technique for larger companies that need to find work for their teams temporarily between production phases. Or maybe there

are possible synergies to exploit between different teams using similar technology. Alternately, distributed development can be a great way to allow smaller companies to avoid the burden of carrying too many full-time staffers.

Any time you've got a project that is meant to hit store shelves simultaneously across several different platforms (Xbox 360, iPhone, PS3, PC, PSP, Nintendo DS, and so on), you're likely to need some level of distributed development. If your project is tied into a movie license such that you're coordinating with a Hollywood studio, then you're likely distributed. Also, if you're working with a publisher of almost any size, then you're probably distributed to some degree (even if it's just your localization, QA, marketing, or sales departments that are located elsewhere).

Ultimately, however, there seem to be two main reasons to use distributed development:

- To get the best people and teams on the job
- To save money

Since the latter can be a nice side effect of clever resource organization, I strongly suggest focusing always on the former. Racing toward the bottom almost never gets you a great product, and poor products seldom create the kinds of long-term sustainable franchises that generate real revenues. Focus on using the techniques in this book to get the best minds on the planet thinking about how to make your game great. If you do this, the money will follow.

1.4.2 *So You Don't Like Outsourcing and Think It's a Bad Idea*

While having lunch with a young designer recently, I mentioned the ways in which a current project of mine was working and discussed my plan to write this document. His response surprised me: "I don't like outsourcing, or working with remote teams. I'm not sure teams should do it."

What surprised me wasn't the attitude; many people are afraid of job loss, of losing control of a project, or just afflicted with plain old xenophobia at the thought of having to collaborate with people in a distant land. What surprised me more was the idea that anyone believed that using some type of distributed team was optional.

Make no mistake. We are now firmly in the era of distributed, collaborate development and production. Personal preferences do not much enter into the question anymore. For projects of even a modest size, the question for you isn't "Will I have to collaborate with some external teams in order to succeed?" The question for you is "How do I ensure that this collaboration results in a better end product and more effectively meets my parameters for success?" There's no "if," only "how."

According to a 2009 article from *Gamasutra*, the online arm of *Game Developer* magazine, 86% of teams polled reported using some form of externalized development, and 20% of those projects polled spent more than $2 million on an externalized