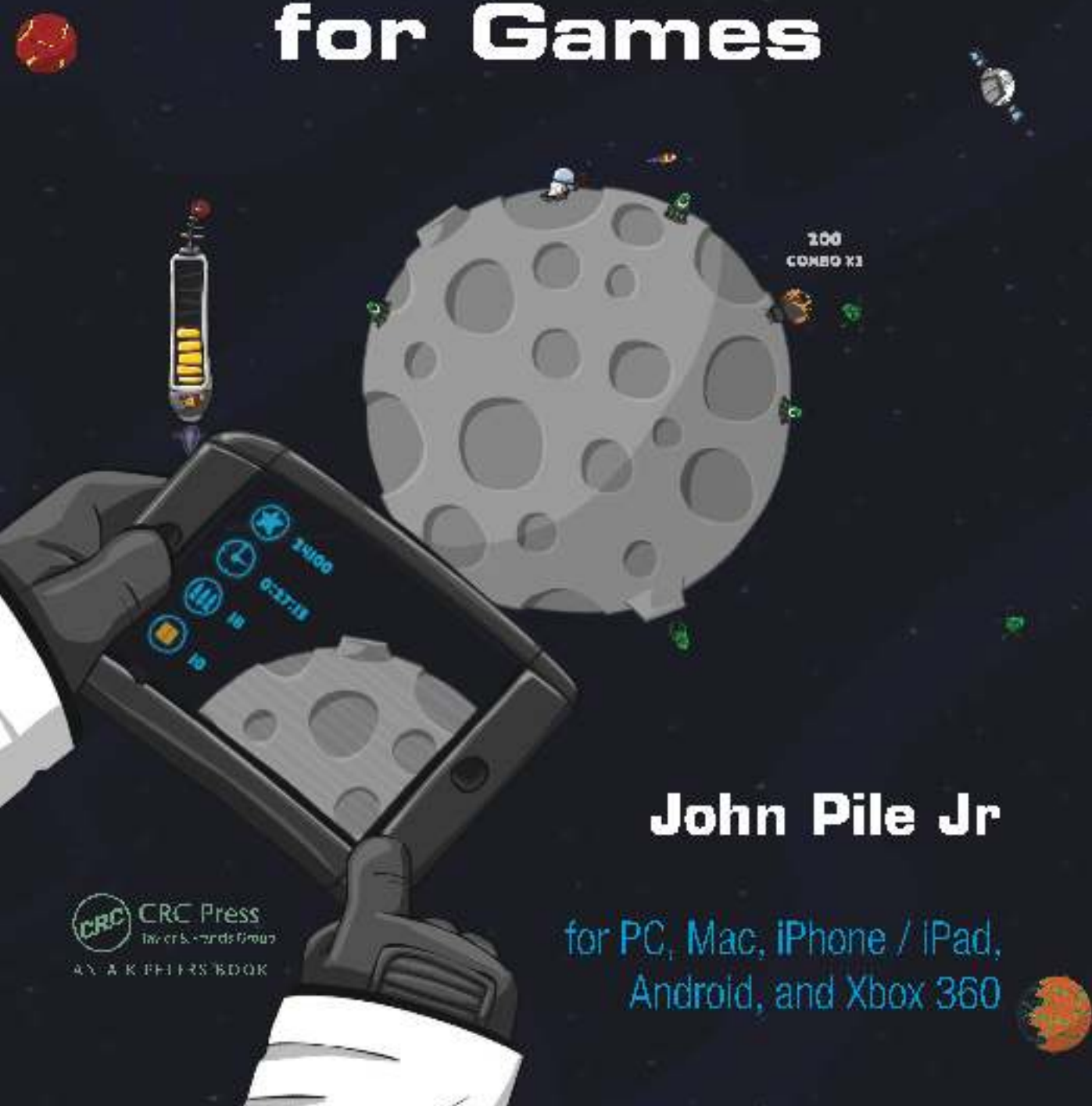


2D Graphics Programming for Games



John Pile Jr

for PC, Mac, iPhone / iPad,
Android, and Xbox 360

 **CRC Press**
Taylor & Francis Group
AN AK PETERS BOOK



2D Graphics Programming for Games



2D Graphics Programming for Games

John Pile Jr



CRC Press

Taylor & Francis Group

Boca Raton, London, New York

CRC Press is an imprint of the
Taylor & Francis Group, an informa business
AN & C PETERS BOOK

CRC Press
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742

© 2013 by Taylor & Francis Group, LLC
CRC Press is an imprint of Taylor & Francis Group, an Informa business

No claim to original U.S. Government works
Version Date: 20121220

International Standard Book Number-13: 978-1-4665-0190-4 (eBook - PDF)

This book contains information obtained from authentic and highly regarded sources. Reasonable efforts have been made to publish reliable data and information, but the author and publisher cannot assume responsibility for the validity of all materials or the consequences of their use. The authors and publishers have attempted to trace the copyright holders of all material reproduced in this publication and apologize to copyright holders if permission to publish in this form has not been obtained. If any copyright material has not been acknowledged please write and let us know so we may rectify in any future reprint.

Except as permitted under U.S. Copyright Law, no part of this book may be reprinted, reproduced, transmitted, or utilized in any form by any electronic, mechanical, or other means, now known or hereafter invented, including photocopying, microfilming, and recording, or in any information storage or retrieval system, without written permission from the publishers.

For permission to photocopy or use material electronically from this work, please access www.copyright.com (<http://www.copyright.com/>) or contact the Copyright Clearance Center, Inc. (CCC), 222 Rosewood Drive, Danvers, MA 01923, 978-750-8400. CCC is a not-for-profit organization that provides licenses and registration for a variety of users. For organizations that have been granted a photocopy license by the CCC, a separate system of payment has been arranged.

Trademark Notice: Product or corporate names may be trademarks or registered trademarks, and are used only for identification and explanation without intent to infringe.

Visit the Taylor & Francis Web site at
<http://www.taylorandfrancis.com>

and the CRC Press Web site at
<http://www.crcpress.com>

For Helen.

Contents

Preface	xi
Acknowledgments	xiii
About the Author	xv
I Getting Started in 2D	1
1 Introduction	3
1.1 About This Book	3
1.2 Why C# and XNA?	5
1.3 Game Development 101	8
1.4 Game Developer Platforms	9
1.5 Book Organization	12
2 Basics of Computer Graphics	15
2.1 Bits and Bytes	15
2.2 Display	24
2.3 Double Buffering	30
2.4 Graphic File Formats	31
Exercises	33
3 Sprites!	37
3.1 What Is a Sprite?	37
3.2 Layering with Depth	45
3.3 The Sprite Sheet and the GPU	47
3.4 Scaling Sprites	49
Exercises	52

II	Motion and Depth	55
4	Animation	57
4.1	Historical Animation	57
4.2	Cel Animation	59
4.3	A Few Principles of Animation	62
4.4	Animation Cycles	69
	Exercises	70
5	Camera and Tiling	73
5.1	A Simple Camera	73
5.2	Simple Camera Zoom	79
5.3	Tiling	80
5.4	Isometric Tiled Graphics	89
	Exercises: Challenges	91
6	The Illusion of Depth	93
6.1	A Historical Perspective on Perspective	93
6.2	Layering	95
6.3	The Six Principles of Depth	97
6.4	The Six Principles in Code	105
6.5	Traditional Perspective	116
6.6	Summary	120
	Exercises: Challenges	121
7	User Interface	123
7.1	UI Types	123
7.2	Fonts	124
7.3	Localization	126
7.4	Safe Frames	128
7.5	Menus	129
	Exercises: Challenges	130
III	Advanced Graphics	131
8	Particle Systems	133
8.1	What Is a Particle?	134
8.2	Creating Effects	141
8.3	Blending Types	146
8.4	Types of Effects	149
8.5	An Effect System	162
8.6	Optimization	164
	Exercises: Challenges	166

9 GPU Programming	169
9.1 Pixel Modification	169
9.2 Full-Screen Pixel Modifications	174
9.3 What Is a Shader?	178
9.4 Shader Languages	178
9.5 Pixel Shader Examples	182
Exercises: Challenges	186
10 Polish, Polish, Polish!	187
10.1 Transitions	188
10.2 Sinusoidal Movement	193
10.3 Splines	195
10.4 Working with Your Artist	197
10.5 Conclusion	197
Exercises: Challenges	198
IV Appendices	199
A Math Review: Geometry	201
A.1 Cartesian Mathematics	201
A.2 Line	201
A.3 Circle	201
A.4 Pythagorean Theorem	202
A.5 Distance	202
A.6 Distance Squared	202
B Math Review: Vectors	203
B.1 Vectors and Notation	203
B.2 Vector Comparison	204
B.3 Length, Addition, and Subtraction	206
B.4 Unit Vectors and Normalizing a Vector	207
B.5 Vector Properties	207
B.6 Standard Unit Vectors and Polar Representation	208
C Math Review: Trigonometry	211
C.1 Triangle Trigonometry	211
C.2 Unit-Circle Trigonometry	212
C.3 Trigonometry as a Collection of Periodic Functions	213
C.4 The Tangent Function	214
C.5 Translations and Transforms of Trigonometric Functions	215
C.6 Circles and Ellipses	216

Bibliography	217
Glossary	219
Index	223

Preface

There are already some great books on programming 2D games, so why write one that focuses only on 2D graphics?

The answer is that whereas other books might succeed at covering a breadth of topics, they don't necessarily go into the depth required to make professional-looking games. Some great texts cover other advanced game development topics, such as game physics, game AI, real-time 3D graphics, and game architectures, but the information on 2D graphics has been difficult to find in a single text. Until now, that is.

Further, the books that do discuss the creation of 2D games focus on only one platform (OpenGL, DirectX, Flash, XNA). In reality, as you will see in this book, the core concepts of graphics programming are the same, regardless of platform.

Throughout this book you will learn the concepts and techniques used in making great 2D graphics. Much of what is included in this book might be considered general knowledge by many game developers, but those same developers would be at a loss to tell you where they actually picked up the information. The truth is that it has been gained by years of experience developing games.

When I was hired to teach a course on 2D graphics, I spent a great deal of time looking for a textbook that covered the topics I believe are most important for new game developers to learn. I could not find one, and the result is the content within this book.

My goal is that by the time you finish reading and working through the exercises in this text, you will be able to look at a game such as *Castle Crashers* [Zynga Dallas 11] and think, "Sure, I could do that."

In addition, I suspect you'll have a newfound respect for the roles of game artists and designers.

Acknowledgments

Among teaching, coding, and fulfilling a variety of other obligations, I have managed to finish writing a book during what is also my first two years of marriage. I therefore want to thank my beautiful wife, Helen, who has happily dealt with the glow of my computer screens until the wee hours of the morning and a year of too much work and not enough play.

I would also like thank my parents for their continual support and patience over the years. Even I am aware that having a middle-aged son who still plays computer games and watches cartoons is a little odd. Through the years, they have led by example, instilling a combined work and play ethic epitomized by my dad's motto: "Do what you love and the rest will follow." That has been my guiding principle and helps to explain why I look forward to each day of work.

At the end of this book are two appendices reviewing the basic math principles needed for this text, which are provided courtesy of Dr. Scott Stevens, Mathematics Coordinator at Champlain College. My thanks go out to him for putting these together. For further exploration of these topics, Scott developed an advanced course of the math needed for 3D game development. The textbook for that course, *Matrices, Vectors, and 3D Math*, is available online [Stevens 12].

My students also deserve a great deal of thanks. They keep me inspired and on my toes. Throughout this book you will find that many of the visual examples are screenshots of games created by my students. In addition, one of the great rewards of teaching at a time when all the latest software development information can be found online is that for those who want to learn, the classroom has now become an amazing two-way information exchange. When I give students a bit of background and point them in the right direction, they come back with all kinds of new and interesting stuff that I never could have found on my own.

Without sounding too much like an award speech, I want to give credit to the team I worked with at Proper Games: Mike, Danny, Andy, Fritz, Janek, Chris Bradwell, Chris Brown, Paddy, John, and, of course, Smithy. Additionally, much of the artwork in this book was provided by my Proper Games colleague and good friend Geoff Gunning. His unique artistic style

and attention to detail are an inspiration. Geoff is truly a hidden talent and all-around good guy. I'm lucky to have had the privilege to work with him on almost every one of my major game projects.

Finally, I would like to thank two good friends who are gone too soon. Mike and Jenny, you are missed.

About the Author

John Pile Jr is a game developer and educator. He has taught courses in graphics, game physics, and game networking for the Game Studio at Champlain College since 2010. He holds a BS in mathematics from Fairmont State University and an MS in software engineering for computer game technology from the University of Abertay in Dundee, Scotland.

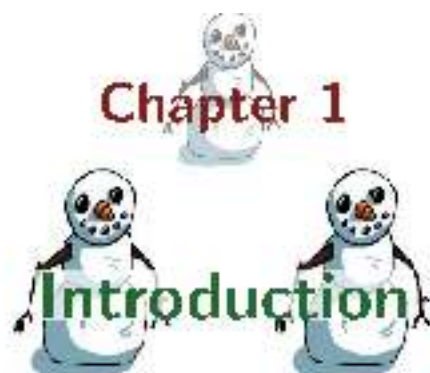
John also has an extensive career as a software engineer both in and out of the game industry, with credited titles for Xbox 360, PlayStation 3, PC, iOS, and Android. His most recently released title was *aliEnd* for Android.

While not teaching, writing books, or developing games, John spends his summers with his wife exploring his home state of Alaska, her home country of Scotland, and wherever else the wind might take them.

Part I



Getting Started in 2D



1.1 About This Book

This book is about programming, but at times also presents aspects of 2D graphics that might otherwise be considered more appropriate for a discussion on art or design. These are useful topics because they allow you, as a graphics programmer, to communicate effectively with both your art and design counterparts. They also give you the perspective to offer meaningful dialogue and suggestions on how a particular art or design challenge can be solved with a programmatic solution.

My emphasis in this book, as it is in my classroom, is threshold theory, minimal code, and experimentation. By starting with a basic concept that demonstrates both the understanding of what we are trying to accomplish as well as why we are taking a particular approach, we set the proper context for the code we write. Minimal code keeps us clear as readers to see a particular line of code in action or as it relates to the code around it. These code snippets are provided without the usual volume of good coding comments. However, this is done for the purpose of assessing the code chunk in isolation in isolation. A variety of best practices available on good coding practices for any language of choice, as well as an object-oriented programming and design paradigm. Apply these principles to the code you write.

The final and most important part of my emphasis is experimentation. This learning experience that most learning occurs when exploring through a problem, experimenting with solutions, and generally tinkering with code. The challenges listed in the book are for you to try. In addition to these challenges, other suggestions throughout the text present possible projects and added functionality. Take these suggestions to heart. The reader who experiments is the reader who learns.

1.1.1 Required Knowledge

This book assumes you already have a basic understanding of programming. The code samples listed in the text are written in C# but can easily be applied to most programming languages. When I teach this course at the college level, the students have only one year of C++ programming as their background. Assuming you already know Java, C++, or Objective-C, you should find the transition to C# fairly effortless.

The companion website, <http://www.2dGraphicsProgramming.com>, offers code samples in other programming languages. However, the focus of this book is on coding graphics, not the specifics of the language. To use a fairly bad analogy: when driving from Seattle to Florida, you need to understand the basic rules of the road and how to navigate, no matter what your vehicle. As long as you know how to drive at least one vehicle, the differences between driving a tractor or a sports car are irrelevant. Both vehicles need fuel and have an accelerator, brake, and transmission. As long as you can drive one, you will have the other figured out by the time you get there.

The text also assumes that the reader has a basic background in mathematics, including geometry and trigonometry. If it has been a while since your basic math days, the math primers in the appendices should help.

Be forewarned: the sample code included in the beginning of the text includes every line of code, but later you will be required to fill in the blanks yourself. Code snippets for a new concept are included, but after a while it is not necessary to repeat the same pieces of code for each sample.

1.1.2 Why 2D Games?

The last five years or so have demonstrated that it is still possible to create fun, addictive, and immersive game experiences in two dimensions. Run-away hits such as *Angry Birds* [Rovio Entertainment 09], *Peggle* [PopCap Games 07], and *Fruit Ninja* [Halfbrick Studios 10] are all examples of highly successful 2D games, and you probably can think of many more.

On a scale of realistic to symbolic, 2D games tend to fall to the symbolic side, although this is not always the case. These games speak to us on a more abstract level, and we are actually quite comfortable with that: we often communicate in 2D in the form of letters, numbers, symbols, and charts [Rasmussen 05].

In addition, some developers simply consider 2D a better platform for achieving certain artistic goals. Game artist Geoff Gunning put it this way: “I’ve never been a fan of 3D game art . . . I can appreciate how impressive the talent is, but it never looks as characterful as 2D art.”

Another important point is that 2D games usually require significantly less in art assets than their 3D counterparts. This can be a big deal for a small development team for whom resources are limited. But even in a 3D game, it is likely that some work is done in 2D. The user interface, heads-up display, and/or menuing system are likely rendered in 2D. In fact, unless a game is developed for a 3D television, games are still 2D media. The final output for most games is still a 2D screen.

Finally, from the perspective of someone who also loves to teach 3D graphics programming, I believe that focusing on 2D graphics is a great introduction to the broader graphics topics. In later chapters you will be able to create particle systems and write your own graphics shaders without the added confusion of 3D matrix math, lighting algorithms, and importing 3D models. I believe it is a valuable step in the learning process of those who want to become 3D graphics programmers to first understand 2D graphics thoroughly.

Beyond these justifications, 2D graphics are simply fun. They provide instant gratification and allow developers to quickly prototype ideas and mechanics.

1.2 Why C# and XNA?

The code samples included in this book are in C# with XNA. Every language has its advantages and disadvantages, but for the goals of this book, I strongly believe C#/XNA is the best choice for a number of reasons.

First, like Java, C# is a managed coding language. This means you won't get distracted by pointers and memory management. But, this comes at a cost. C# is not as fast as C++, however most platforms (even mobile devices) are able to handle this added overhead without that being much of an issue.

Second, using C#/XNA will allow your game to run natively on PCs (Windows), game consoles (Xbox 360), and even some mobile devices (Windows 7 Phone) without any significant modification. Then, with the help of an environment such as Mono, your C# game can easily be ported to Android, iOS, Mac PCs, Linux, and Sony platforms.

Let's pause for a moment here for emphasis because this second point should not be passed lightly. C#/XNA allows you to develop richly graphical games for almost any platform. Very few game development environments are able to make this same claim—and those that do come with their own set of challenges.

Third, XNA was created specifically for game development. It provides common structures and functions useful to game creation that are outside

the scope of this text. At the same time, the tools provided by XNA are not so abstract that they become irrelevant to other platforms. For example, Unity3D has a great particle system, but using that particle system won't necessarily give you the experience to create your own.

Finally, XNA allows us to have direct access to the graphics card through the implementation of shader programming. This tool is powerful for creating advanced graphics effects, and the knowledge is easily transferable to both DirectX and OpenGL.

At the risk of repeating myself, the concepts discussed in this book are not specific to any one programming language or graphics library. This book is about understanding and exploring 2D graphics programming concepts; the language is just a means to an end.

1.2.1 Why not C++?

Before we get too far, I would like to address an often-cited reason for avoiding XNA. This is an idea that I see printed over and over, that *real game programming* is done in C++. Unfortunately, I have to admit that even a few years ago, I too was guilty of uttering that tired refrain.

The truth is that even though AAA game development almost always requires the programming performance available only through C++, we are quickly finding that a thriving new game market is being driven by non-AAA games. Combined with the power of modern multicore processors, most of these non-AAA games are being developed on a variety of non-C++ platforms.

That's not to say that you should avoid C++. It really is a powerful programming language that should be the foundation of any programming or computer science degree. However, we just don't need it for this text, and it could potentially provide an unnecessary barrier to many of the graphical concepts we cover here.

I have no doubt that we will continue to hear the "real game development" cliché in the future, but it comes from the same naysayers who claimed there was no future in online, social, mobile, or *indie* (independent video) games. It's just so 2006.

1.2.2 The Future of XNA

Another, more fundamental, concern with C# and XNA is that Microsoft appears to be on a path to sunset the XNA framework. Early in the discussion of Windows 8, there were rumors that the new operating system would not support XNA. Now that the operating system (OS) has been released, it is clear that there has been a specific choice not to provide direct support for XNA. Although games can be written to run on Windows 8–

based PCs, they cannot directly be deployed to Windows 8–based mobile devices and tablets.

While there is currently no team at Microsoft developing further versions of the XNA framework, their policy is to continue supporting software for ten years beyond the last point release. XNA 4.0 was released at the end of 2011 and I have been assured by my friends at Microsoft that XNA will be supported until at least 2021. Just know that we may need to do a little extra work to prepare our XNA game for Windows 8 devices and associate marketplace.

The good news is that there is a path to publishing XNA games on Windows 8 mobile devices via Mono and MonoGame (the same technology that allows us to use the XNA framework on Android devices, which conveniently also happen to run the ARM architecture).

The future of XNA might remain uncertain, but for now I am quite content that, as a game developer, the framework meets my cross-platform 2D game development needs. And if something better does come along, I'll be the first to give it a try.

1.2.3 Required Software

Microsoft provides the developer tools for free. To run the code samples in this book, you will need (at a minimum)

- Visual C# Express 2010,
- XNA Game Studio 4.0.

These development tools are available for download. It may be easiest to get them directly from <http://create.msdn.com>; I have also provided a link on the book's companion website <http://www.2dGraphicsProgramming.com> in case that ever changes.

In addition to the required software, I suggest you become familiar with graphics tools, including Adobe Photoshop or the open source alternative Gimp. Knowing how to work with these tools, even if only to do minor edits such as resizing, will help you in the long run. It is well worth knowing the tools of the artist.

1.2.4 An Artistic Programmer

The common perception is that there is a dichotomy between the creative artist and the logical programmer—right-brained versus left-brained, cold and calculating versus warm, fuzzy, and touchy-feely. And even though I might argue that these stereotypes are unfair in any setting, the best attributes of both are required in game development when programming graphics for games.

- [download online Angle of Yaw here](#)
- [download Goodbye, Columbus And Five Short Stories](#)
- [The First Idea: How Symbols, Language, and Intelligence Evolved from Our Primate Ancestors to Modern Humans book](#)
- **[download online How to Negotiate Like a Child: Unleash the Little Monster Within to Get Everything You Want](#)**
- [download John Dies at the End](#)
- [download Indigo Springs \(Astrid Lethewood, Book 1\) online](#)

- <http://wind-in-herleshausen.de/?freebooks/Greywalker--Greywalker--Book-1-.pdf>
- <http://www.netc-bd.com/ebooks/The-Singularity-Is-Near--When-Humans-Transcend-Biology.pdf>
- <http://dadhoc.com/lib/Food-and-Culture--6th-Edition-.pdf>
- <http://bestarthritiscare.com/library/How-to-Negotiate-Like-a-Child--Unleash-the-Little-Monster-Within-to-Get-Everything-You-Want.pdf>
- <http://transtrade.cz/?ebooks/John-Dies-at-the-End.pdf>
- <http://rodrigocaporal.com/library/Twisting-Steele--Sarah-Steele--Book-2-.pdf>