Game Engine Architecture

Jason Gregory

Foreword by **Jeff Lander and Matt Whiting**

Game Engine Architecture

Jason Gregory

A K Peters, Ltd.
Natick, Massachusetts

Editorial, Sales, and Customer Service Office

A K Peters, Ltd.
5 Commonwealth Road, Suite 2C
Natick, Massachusetts
www.akpeters.com

Library of Congress Cataloging-in-Publication Data

Gregory, Jason, 1970-
Game engine architecture / Jason Gregory.
p. cm.
Includes bibliographical references and index.
ISBN 978-1-56881-413-1 (alk. paper)
1. Computer games--Programming. 2. Computer architecture. I. Title.
QA76.76.C672G77 2009
794.8'1526--dc22

2009013092

Printed in the United States of America
14 13 12 11 10

10 9 8 7 6 5 4 3

Game Engine Architecture

Dedicated to
Trina, Evan and Quinn Gregory,

in memory of our heros,
Joyce Osterhus and Kenneth Gregory.

Foreword

The very first video game was built entirely out of hardware, but rapid advancements in microprocessors have changed all that. These days, video games are played on versatile PCs and specialized video game consoles that use software to make it possible to offer a tremendous variety of gaming experiences. It's been 50 years since those first primitive games, but the industry is still considered by many to be immature. It may be young, but when you take a closer look, you will find that things have been developing rapidly. Video games are now a multibillion-dollar industry covering a wide range of demographics.

Video games come in all shapes and sizes, falling into categories or "genres" covering everything from solitaire to massively multiplayer online role-playing games, and these games are played on virtually anything with a microchip in it. These days, you can get games for your PC, your cell phone, as well as a number of different specialized gaming consoles—both handheld and those that connect to your home TV. These specialized home consoles tend to represent the cutting edge of gaming technology, and the pattern of these platforms being released in cycles has come to be called console "generations." The powerhouses of this latest generation are Microsoft's Xbox 360 and Sony's PLAYSTATION 3, but the ever-present PC should never be overlooked, and the extremely popular Nintendo Wii represents something new this time around.

The recent explosion of downloadable and casual games has added even more complexity to the diverse world of commercial video games. Even so, big games are still big business. The incredible computing power available on today's complicated platforms has made room for increased complexity in the software. Naturally, all this advanced software has to be created by someone, and that has driven up the size of development teams—not to mention development costs. As the industry matures, we're always looking for better, more efficient ways to build our products, and development teams have begun compensating for the increased complexity by taking advantage of things like reusable software and middleware.

With so many different styles of game on such a wide array of platforms, there cannot be any single ideal software solution. However, certain patterns have developed, and there is a vast menu of potential solutions out there. The problem today is choosing the right solution to fit the needs of the particular project. Going deeper, a development team must consider all the different aspects of a project and how they fit together. It is rare to find any one software package that perfectly suits every aspect of a new game design.

Those of us who are now veterans of the industry found ourselves pioneering unknown territory. Few programmers of our generation have Computer Science degrees (Matt's is in Aeronautical Engineering, and Jason's is in Systems Design Engineering), but these days many colleges are starting to programs and degrees in video games. The students and developers of today need a good place to turn to for solid game-development information. For pure high-end graphics, there are a lot of sources of very good information from research to practical jewels of knowledge. However, these sources are often not directly applicable to production game environments or suffer from not having actual production-quality implementations. For the rest of game development, there are so-called beginner books that so gloss over the details and act as if they invented everything without giving references that they are just not useful or often even accurate. Then there are high-end specialty books for various niches like physics, collision, AI, etc. But these can be needlessly obtuse or too high level to be understood by all, or the piecemeal approach just doesn't all fit together. Many are even so directly tied to a particular piece of technology as to become rapidly dated as the hardware and software change.

Then there is the Internet, which is an excellent supplementary tool for knowledge gathering. However, broken links, widely inaccurate data, and variable-to-poor quality often make it not useful at all unless you know exactly what you are after.

Enter Jason Gregory, himself an industry veteran with experience at Naughty Dog—one of the most highly regarded video game studios in the

world. While teaching a course in game programming at USC, Jason found himself facing a shortage of textbooks covering the fundamentals of video-game architecture. Luckily for the rest of us, he has taken it upon himself to fill that gap.

What Jason has done is pull together production-quality knowledge actually used in shipped game projects and bring together the entire game-development picture. His experience has allowed him to bring together not only the ideas and techniques but also actual code samples and implementation examples to show you how the pieces come together to actually make a game. The references and citations make it a great jumping-off point to dig deeper into any particular aspect of the process. The concepts and techniques are the actual ones we use to create games, and while the examples are often grounded in a technology, they extend way beyond any particular engine or API.

This is the kind of book we wanted when we were getting started, and we think it will prove very instructive to people just starting out as well as those with experience who would like some exposure to the larger context.

Jeff Lander
Matthew Whiting

Preface

Welcome to *Game Engine Architecture*. This book aims to present a complete discussion of the major components that make up a typical commercial game engine. Game programming is an immense topic, so we have a lot of ground to cover. Nevertheless, I trust you'll find that the depth of our discussions is sufficient to give you a solid understanding of both the theory and the common practices employed within each of the engineering disciplines we'll cover. That said, this book is really just the beginning of a fascinating and potentially life-long journey. A wealth of information is available on all aspects of game technology, and this text serves both as a foundation-laying device and as a jumping-off point for further learning.

Our focus in this book will be on game engine technologies and architecture. This means we'll cover both the theory underlying the various subsystems that comprise a commercial game engine and also the data structures, algorithms, and software interfaces that are typically used to implement them. The line between the game engine and the game is rather blurry. We'll focus primarily on the engine itself, including a host of low-level foundation systems, the rendering engine, the collision system, the physics simulation, character animation, and an in-depth discussion of what I call the *gameplay foundation layer*. This layer includes the game's object model, world editor, event system, and scripting system. We'll also touch on some aspects of game-

play programming, including player mechanics, cameras, and AI. However, by necessity, the scope of these discussions will be limited mainly to the ways in which gameplay systems interface with the engine.

This book is intended to be used as a course text for a two- or three-course college-level series in intermediate game programming. Of course, it can also be used by amateur software engineers, hobbyists, self-taught game programmers, and existing members of the game industry alike. Junior engineers can use this text to solidify their understanding of game mathematics, engine architecture, and game technology. And some senior engineers who have devoted their careers to one particular specialty may benefit from the bigger picture presented in these pages, as well.

To get the most out of this book, you should have a working knowledge of basic object-oriented programming concepts and at least some experience programming in C++. Although a host of new and exciting languages are beginning to take hold within the game industry, industrial-strength 3D game engines are still written primarily in C or C++, and any serious game programmer needs to know C++. We'll review the basic tenets of object-oriented programming in Chapter 3, and you will no doubt pick up a few new C++ tricks as you read this book, but a solid foundation in the C++ language is best obtained from [39], [31], and [32]. If your C++ is a bit rusty, I recommend you refer to these or similar books to refresh your knowledge as you read this text. If you have no prior C++ experience, you may want to consider reading at least the first few chapters of [39], or working through a few C++ tutorials online, before diving into this book.

The best way to learn computer programming of any kind is to actually write some code. As you read through this book, I strongly encourage you to select a few topic areas that are of particular interest to you and come up with some projects for yourself in those areas. For example, if you find character animation interesting, you could start by installing Ogre3D and exploring its skinned animation demo. Then you could try to implement some of the animation blending techniques described in this book, using Ogre. Next you might decide to implement a simple joypad-controlled animated character that can run around on a flat plane. Once you have something relatively simple working, expand upon it! Then move on to another area of game technology. Rinse and repeat. It doesn't particularly matter what the projects are, as long as you're *practicing* the art of game programming, not just reading about it.

Game technology is a living, breathing thing that can never be entirely captured within the pages of a book. As such, additional resources, errata, updates, sample code, and project ideas will be posted from time to time on this book's website at http://gameenginebook.com.

Acknowledgments

No book is created in a vacuum, and this one is certainly no exception. This book would not have been possible without the help of my family, friends, and colleagues in the game industry, and I'd like to extend warm thanks to everyone who helped me to bring this project to fruition.

Of course, the ones most impacted by a project like this one are invariably the author's family. So I'd like to start by offering a special thank-you to my wife Trina, who has been a pillar of strength during this difficult time, taking care of our two boys Evan (age 5) and Quinn (age 3) day after day (and night after night!) while I holed myself up to get yet another chapter under my belt, forgoing her own plans to accommodate my schedule, doing my chores as well as her own (more often than I'd like to admit), and always giving me kind words of encouragement when I needed them the most. I'd also like to thank my eldest son Evan for being patient as he endured the absence of his favorite video game playing partner, and his younger brother Quinn for always welcoming me home after a long day's work with huge hugs and endless smiles.

I would also like to extend special thanks to my editors, Matt Whiting and Jeff Lander. Their insightful, targeted, and timely feedback was always right on the money, and their vast experience in the game industry has helped to give me confidence that the information presented in these pages is as accurate and up-to-date as humanly possible. Matt and Jeff were both a pleasure to work with, and I am honored to have had the opportunity to collaborate with such consummate professionals on this project. I'd like to thank Jeff in particular for putting me in touch with Alice Peters and helping me to get this project off the ground in the first place.

A number of my colleagues at Naughty Dog also contributed to this book, either by providing feedback or by helping me with the structure and topic content of one of the chapters. I'd like to thank Marshall Robin and Carlos Gonzalez-Ochoa for their guidance and tutelage as I wrote the rendering chapter, and Pål-Kristian Engstad for his excellent and insightful feedback on the text and content of that chapter. I'd also like to thank Christian Gyrling for his feedback on various sections of the book, including the chapter on animation (which is one of his many specialties). My thanks also go to the entire Naughty Dog engineering team for creating all of the incredible game engine systems that I highlight in this book. Special thanks go to Keith Schaeffer of Electronic Arts for providing me with much of the raw content regarding the impact of physics on a game, found in Section 12.1. I'd also like to thank Paul Keet of Electronic Arts and Steve Ranck, the

lead engineer on the *Hydro Thunder* project at Midway San Diego, for their mentorship and guidance over the years. While they did not contribute to the book directly, their influences are echoed on virtually every page in one way or another.

This book arose out of the notes I developed for a course called *ITP-485: Programming Game Engines*, which I have been teaching under the auspices of the Information Technology Program at the University of Southern California for approximately three years now. I would like to thank Dr. Anthony Borquez, the director of the ITP department at the time, for hiring me to develop the ITP-485 course curriculum in the first place. I'd also like to extend warm thanks to Ashish Soni, the current ITP director, for his continued support and encouragement as ITP-485 continues to evolve.

My extended family and friends also deserve thanks, in part for their unwavering encouragement, and in part for entertaining my wife and our two boys on so many occasions while I was working. I'd like to thank my sister- and brother-in-law, Tracy Lee and Doug Provins, my cousin-in-law Matt Glenn, and all of our incredible friends, including: Kim and Drew Clark, Sherilyn and Jim Kritzer, Anne and Michael Scherer, and Kim and Mike Warner. My father Kenneth Gregory wrote a book on investing in the stock market when I was a teenager, and in doing so he inspired me to write a book. For this and so much more, I am eternally grateful to him. I'd also like to thank my mother Erica Gregory, in part for her insistence that I embark on this project, and in part for spending countless hours with me when I was a child, beating the art of writing into my cranium—I owe my writing skills (not to mention my work ethic... and my rather twisted sense of humor...) entirely to her!

Last but certainly not least, I'd like to thank Alice Peters and Kevin Jackson-Mead, as well as the entire A K Peters staff, for their Herculean efforts in publishing this book. Alice and Kevin have both been a pleasure to work with, and I truly appreciate both their willingness to bend over backwards to get this book out the door under very tight time constraints, and their infinite patience with me as a new author.

Jason Gregory
April 2009

Contents