

THE DARK SIDE OF MOTIONBUILDER

As with any tool, MotionBuilder has a few features and implementations that have left me with an unpleasant taste in my mouth. In some cases, aspects of several tools seem to require polishing, as if they were implemented as works-in-progress, and it seems Autodesk is waiting for the community to point out the problems, rather than implementing stable tools little by little.

To that extent, I tend to become frustrated with tiny problems as soon as I try to think outside the box. For example, the character rig is very easy to setup and use for standard characters, but it's cumbersome to use when replicating an existing rig or building on top of the basic MotionBuilder rig. Implementing a set of extra controls requires a good understanding of the software and can lead to frustration since it lacks some essential features of other favorite 3D software packages.

For instance, to set up a pair of wings on a character, you would need to build controllers from scratch without evolved rigging tools (in Maya, I would defer to the Set Driven Keys), and implement a character extension to plug into your current character rig.

In another example, if you want to create a simple snake character, you need to disregard the character rig completely and create a custom setup,

which could use last resort techniques or non-animator friendly controllers, thus slowing down your animation tasks.

From a production standpoint, I've always opted to use MotionBuilder only on standard characters, and I would strongly discourage using characters that don't fall into the symmetrical biped or quadruped mold.

While MotionBuilder 2009 is making progress in terms of opening its interface and features to programmers, the SDK and scripting language are, in my opinion, very immature compared to other packages. If we compare it to Maya, for which the software was built from the ground up using the MEL scripting language and C++ SDK, it seems like MotionBuilder does the exact opposite, trying to deconstruct the software to give access to some features (but not all) with a Python scripting language and C++ SDK.

For instance, there is no easy way of creating a custom hotkey. We had to use brute force keyboard hooks to access some menu items and tools in order to speed up our workflow.

Using a software with a dysfunctional Undo system can be quite frustrating too. For most tasks, Undo works as intended in MotionBuilder, but the button does not work for everything. When you come across certain actions that you can't undo, it forces you to revert to the last saved file, a painful moment in any game developer's workflow.

Along the same line, I have in the past been working on a file when suddenly MotionBuilder crashes—and with no last-minute save or memory dump—prevents me from retrieving a portion of my work. MotionBuilder does have a Save Reminder setting that warns you when you haven't saved for the past few minutes, but that's hardly a replacement for a proper crash recovery function.

AROUND THE BEND

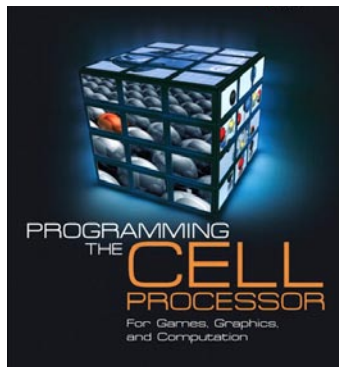
MotionBuilder is simply awesome for creating animation for video games. It's fast, easy to learn, and supports FBX file format across a wide variety of 3D packages. But you might want to take it for a test run before going full throttle into production (details about the 30 day free trial are available on Autodesk's website). Make sure you know what MotionBuilder can do for you.

From my experience, MotionBuilder was a time saver in most cases, but in certain situations, it was best to omit it from the pipeline.

MARC-ANDRÉ GUINDON has been in the visual effects and games industries for 10 years, specializing in pipelines using MotionBuilder and Maya. Ambitious to redefine the industry and explore new ways of working, he has founded NeoReel, Inc., a production and consultation studio located in Montreal. Email him at mguindon@gdmag.com.

Book Review by Bijan Forutanpour

PROGRAMMING THE CELL PROCESSOR FOR GAMES, GRAPHICS, AND COMPUTATION BY MATTHEW SCARPINO



WHAT DO SPIDER-MAN AND THE PLAYSTATION 3 HAVE IN COMMON? THEY SHARE A common theme: "With great power comes great responsibility." According to the *Guinness Book of World Records*, the honor of the world's most powerful distributed system goes to Folding@Home, a worldwide network of PCs and PlayStation 3 consoles. As of April 2008, the network consisted of roughly 2.4 million PCs, and half a million PlayStation 3 consoles. Yet the PCs contribute "only" .259 PetaFlops, compared to 1.235 PetaFlops from the PlayStation 3 consoles. The PlayStation 3 delivers several orders of magnitude more performance than a single PC. With all that processing power comes the responsibility and complexity of programming the PlayStation 3's Cell Broadband Engine, which is where *Programming the Cell Processor For Games, Graphics, and Computation* by Matthew Scarpino proves to be very valuable.

The Cell processor was developed by the STI Alliance, which is composed of Sony, Toshiba, and IBM. The processor is used in Sony's PlayStation 3, IBM's Roadrunner supercomputer and Blade Servers, and Toshiba's SPURS engine. Both IBM and

Sony have released Cell Software Development Kits. It was amusing to read the author's description of his reaction in trying to understand the material covered in IBM's Cell SDK, since I could relate: "The material flew over my head with a fierce whoosh."

The book is a brave attempt at reducing the whoosh effect associated with Cell development. Three main topics are covered in order to give an understanding of the big picture. The first is the hands-on development tool set, the second is the hardware architecture, and finally the available software libraries. It is important to note that this book is not for the novice programmer and requires a very solid background in computer science. Knowledge of some form of Assembly is very helpful, as is some understanding of hardware architectures, operating systems, mathematics, and computer graphics.

Programming the Cell Processor For Games, Graphics, and Computation is very well written and organized, and does a great job describing the hardware architecture and how things work. That is not to say it is easy to read, because the subject matter is pretty complex. The majority of the book is wisely spent discussing in detail the internals of the PowerPC Processor Unit (PPU), the Synergistic Processor Unit (SPU), and SIMD (Single Instruction, Multiple Data) programming on the PPU and SPU. Communication and synchronization with the SPUs are discussed in the chapters covering Direct Memory Access, events, signals, and mailboxes. The chapter on advanced topics discusses SPU overlays, software caching, and SPU isolation for security purposes. Finally, the material on SPUs concludes with an exhaustive coverage of SPU assembly. At some point during reading the many pages involving hardware architecture, as things became blurry, the question inevitably crossed my mind: Why should I care? The author must have sensed the same, and explained: Because the main goal in having all this hardware is to use it, and to do so simultaneously whenever possible. Knowing how the hardware pipeline works helps prevent "pipeline bubbles," or wasted stages and wasted time.

There are a few very important distinctions that must be made regarding the context of this book. The

focus is on the Cell Broadband Engine, and not the PlayStation 3 specifically. The SDK covered is the IBM Cell SDK, and not the Sony PlayStation 3 SDK. Given that the book's title includes the words "For Games and Graphics" assumptions may arise that this is a book specifically written for the PS3 game developer. This is not the case. The book does not describe tools that are standard in the game development community. Important elements not discussed include the PlayStation 3 developer hardware, Microsoft Visual Studio as the Integrated Developer Environment (IDE), combined with the suite of tools by SN Systems such as the ProDG compiler, debugger, tuner, and target manager tools. Additionally, the libraries provided by the IBM SDK are different from those provided by the Sony SDK in many significant areas.

In fact, the book focuses on Linux as the development operating system of choice, which is uncommon in the game development industry, at least for PlayStation 3 development. But this does explain the choice of software tools covered, as the IBM Cell SDK is only available for Linux. The IDE described is Eclipse, working in tandem with the GNU GCC compiler. The author describes how to install Linux on a standard PlayStation 3 and install the needed software tools. Instead of the SN Systems Target Manager, it is suggested to use PuTTY or WinSCP. Obviously, there is more than one way to skin a cat, and it is interesting to see how it is done.

After a little while I was struck by one amusing fact. All the tools described by the author are open source or are freely available at no cost. The IBM Cell SDK, the GNU compiler, GDB debugger, Eclipse, and finally Linux, all come at a total cost of zero. All that is needed is an off-the-shelf PlayStation 3. The last few chapters in the book address game development lightly, and suggest the use of the open source graphics engine Ogre3D and Collada for game data representation.

Realizing the actual goal and focus of the book, it comes as no surprise that the PlayStation 3's NVidia RSX graphics chip is not discussed. Instead there is a chapter on accessing the Linux frame buffer as a device directly, which I found to be interesting. Usually one is bound to using OpenGL or

DirectX. The next chapter is a brief introduction to OpenGL, and how to download and compile the open source implementations called Mesa or Gallium.

In summary, *Programming the Cell Processor For Games, Graphics, and Computation* is an excellent book that tackles a difficult subject admirably. The only point of caution is that not all chapters may relate to what the reader is looking for, depending on preset expectations. In either case, the chapters describing programming the Cell Broadband Engine are very effective and well worth the read.

BIJAN FORUTANPOUR is a senior graphics programmer at Sony Online Entertainment in San Diego with 16 years experience in the visual effects and games industries. He is also the author of *Enzo 3D paint for Photoshop* (www.enzo3d.com). Email him at bforutanpour@gdmag.com.

BOOK REVIEW

★★★★

TITLE

Programming the Cell Processor For Games, Graphics, and Computation

AUTHOR

Matthew Scarpino

PRICE

\$69.99 (hardcover)

PUBLISHER INFORMATION

Prentice Hall
1330 Avenue of the Americas
New York, NY 10019
www.informit.com

Published: October 2008
744 pp.
ISBN-10: 0136008860

www.rtpatch.com

RTPatch and Patched Cell are registered trademarks of Prentice Hall, Inc.