Crossroads
The ACM Student Magazine

Trends in Real-Time Rendering: An Interview with Bioware's Ben Earhart

by *James Stewart*

Fans of PC role-playing games need no introduction to Bioware—the Edmonton, Alberta based developer of Baldur's Gate*,* Neverwinter Nights*, and* Jade Empire*, among others. The company recently opened a studio in Austin, Texas to develop a massively multiplayer online role-playing game (MMORPG, or simply MMO) for an unannounced intellectual property. Ben Earhart, client technology lead on the new project, took a few hours out of his busy schedule to discuss with* Crossroads *the future of real-time rendering—3-D graphics that render fast enough to respond to user input, such as those required for video games.*

Before we get to the tech stuff, tell us how you got started in the industry. I started programming on my Apple II+ when I was about 12 years old. I taught myself BASIC and then 6502 assembly and spent a lot of time writing modem drivers. I remember feeling like I was on top of the world when I invented what I later found out was known as RLE compression.

I earned a bachelor's in EE from the University of Texas at Austin in 1994. I decided on EE for two reasons. First, I was given one of those circuitry kits from Radio Shack when I was a kid. I felt stupid when I couldn't understand the circuit diagrams, which really pissed me off. Second, CS seemed like it would be a waste of money and time because programming came so easily to me.

One day I was in an electronics store and they were running *Doom II* on some demo machines. I still clearly remember starting the game and entering the first room. As I

looked left, I saw a staircase and ran straight for it. It was like that classic scene of the man and woman running toward each other in the field of flowers. When I hit the stairs and glided right up them, I was a goner. I decided right then I wanted to make games as a career. I stuck it out and graduated though.

I worked several years on my own technology company, making the AMP engine and a couple of AMP-based games, the best known of which is *Gore: The Ultimate Soldier*. I've since left the indie life, but those years of hand-to-mouth existence and doing nothing but coding all the time left me with tons of valuable experience.

I worked for Sony Online Entertainment for about a year, first on a new project, and later doing optimizations to the *Star Wars Galaxies* engine.

Currently, I'm leading the client technology team on Bioware's first MMORPG, being done in their new Austin Studio. I'm not sure I'll ever be able to buy myself a stable of exotic Italian sports cars, but I'm thrilled to be working for a triple-A game company that is exceptionally good to its employees—not an easy find, I assure you.

Sounds like you've spent a lot of time on the rendering side of things. I've spent almost my entire career in games and am best described as an engine programmer. I've written several renderers, the first of which was a "quake-class" 6 D. O.F. [six degrees of freedom] polygon renderer just a few years before consumer 3-D accelerators were a reality.

Can you discuss some of the trends in 3-D graphics, especially as they relate to graphics hardware? A 3-D accelerator is a coprocessor, in that it allows you to throw more transistors at the problem. More importantly, though, it is a system highly optimized for a very narrow class of calculations. These optimizations are really the source of the pleasures and pains of 3-D graphics today. As any experienced programmer can tell you, a well-optimized piece of code is a joy to use and a nightmare to modify. Every time we sit down and try to push graphics to the next level, we are using a highly optimized system that is probably *not* designed to do what we want.

Accelerators started by providing a hardware implementation of a simple set of 3-D graphics equations. The industry quickly outgrew this, [so] vertex and pixel programs came along. In my opinion, this "overloading" approach to extending the graphics pipeline is long overdue for being subsumed by something completely different.

Accelerators also achieve big gains by pipelining, which is really another way of saying that they optimize towards series of calculations that do not require feedback (i.e., off-screen rendering). I believe that pipelining will always have merit, though current systems handle this far too crudely to take full advantage of what is really possible.

Also, accelerators take advantage of the parallel nature of graphics calculations. This will always be true. But as scene detail continues to climb, it will be less and less beneficial to assume that any two adjacent pixels will be undergoing exactly the same series of calculations.

In any case, I think some sort of general-purpose parallel calculation engine is what is really needed. I think we should quit trying to figure out clever ways to parallelize all of our primarily sequential algorithms on the main CPU and concentrate on an architecture that assists a small number of sequential processors with a large number of programmable, fast, simple, parallel processing units.

Let's discuss some more specific trends. The Xbox 360 and PS3 both offer some support for real-time procedural generation of content, and numerous tools are already available for PC developers. Where is this technology heading? Of course anything that provides more freedom for the developer is a good thing. Geometry shading, from what I've seen of it so far, appears to be a programmable geometry compression scheme that is ideal for creating high-frequency detail. Things like this will allow developers to effectively increase memory capacity and bus bandwidth in the same way compression increases a modem's effective baud rate.

Right now, I'm dubious it will be a real revolution in graphics, but I do think it's a great step. I'd also like to point out that a big increase in geometric detail is going to make running with FSAA [full scene anti-aliasing] an absolute necessity due to aliasing issues.

How has your content pipeline changed to take advantage of new graphics-hardware features and the content demands of next-gen titles? Until recently, I was in the business of doing shooters, and the rising cost of all that very unique art was driving more and more focus on the art pipeline. I would say that being able to eliminate, minimize, or even accelerate preprocessing stages is made possible, in large

part, by the increase in raw processing power of 3-D cards. For example, switching to a fully dynamic lighting system not only gives you a lot of design freedom but also gets you out of waiting on lengthy radiosity calculations.

When I started working on MMOs, I found out that the RPG/MMO camp had been driving rapid development techniques and asset reuse much harder, simply for game design reasons. This was often done to the detriment of visual quality, which is heresy in the shooter world. But really, how many painstakingly built rooms did I blow through in *Doom III* without so much as a second look? I think the need to make competitive game designs and the corresponding recognition that we're in the entertainment business is what really drives the whole content creation issue.

Some developers have been quoted as saying that Level-Of-Detail is a thing of the past on next-gen systems—that it costs more to evaluate than it earns in performance. What's your stance on this? I doubt that, especially since discrete LOD is so incredibly cheap to evaluate. If you're doing primarily indoor scenes, you could probably skip it.

You can carry the "Don't worry about being vertex-bound" flag all you want, but as you go into the distance, there will eventually be more vertices than on-screen pixels. And, to make matters worse, your scene complexity is order cubed with distance. It's safe to say that neglecting LOD will build a wall in the distance that you just won't be able to cross.

From what's been published on the hardware, it looks like the PS3 takes advantage of CPU parallelism, while the 360 leans more toward GPU parallelism. Which way do you think the industry will head in the future? My impression has always been that the Japanese console designers have an almost pathological tendency to make systems tough for the developers. I suppose it's all in the name of making the base unit as cheap as possible, which may well be a good strategy. However, my suspicion is that the more mainstream architecture of the Xbox will eventually win out, if only by virtue of standardization.

Also, I think parallelism of the main CPU has its limits. Only certain types of calculations can be highly (and easily) parallelized. These parallel calculations need to be executed on a subsystem that is designed for it. Giving the developer several slow, general-purpose CPUs instead of one fast one is not a solution.

It looks like general-purpose processors have hit the performance wall, hence the move to multicores. Intel touts optimizing compilers that will automatically parallelize code. Is this hype? Will hand-tuned parallel code be significantly faster? I think it will be a good incremental improvement, but it's really nothing new. For several years now, CPU pipelines have been "parallelizing" code by doing out-of-order execution of instructions (or portions of instructions) whenever a lack of dependencies allows it.

CPU parallelization is done through a very small window of instructions, and I suspect that processor manufacturers will try to move some of that to the compiler, which has complete visibility into code and can extract much longer "fibers" of independent instruction sequences. As a side benefit, it could simplify the CPU pipeline, allowing for secondary performance gains at the silicon level.

I see concurrency as an optimization, and, like all optimizations, the biggest benefits come from the high-level changes. I definitely agree that the programmer's approach to design and implementation will always have the biggest impact.

Okay, last question: What's the single most powerful force driving progress in real-time rendering? The game developer. All advancement is driven by the consumer, so the frontline engineers working on retail titles represent the authority on what is needed. Programmable computing made it all possible. Putting flexibility into the hands of the developer should be the golden rule for any hardware or middleware producer.

Crossroads wishes to thank Ensemble Studio's Colt McAnlis for technical consultation on this interview.