

# An Interview

with Havok's  
Jeff Yates



Havok's Jeff Yates

Interview provided in  
association with  
[Gamasutra.com](http://Gamasutra.com)

For many gamers and developers, the name Havok is synonymous with physics. Havok's current physics SDK has shipped in some 200 games, including popular titles such as the Halo\* series, and Guitar Hero\* III.

For the past three years, the company has also been developing middleware and tools for game characters. Havok calls these products Havok Animation\* and Havok Behavior\*. To help deliver character assets into the cross-platform file formats used by these tools, Havok has also developed the Havok Content Tools\*, which are a set of plug-ins and exporters for Autodesk's Maya\* and 3ds Max\*, and XSI\*.

Using traditional 3D character animations, along with various procedural and blending techniques, these tools improve the fundamental interactions and low-level decisions that are crucial in helping bring in-game characters to life. Much of the intent behind these tools is to provide a back-end that is not only powerful but also comprehensible—even to members of development teams who may not have highly technical backgrounds.

Havok's Jeff Yates has 15 years of experience managing the development of content creation tools, such as Autodesk's 3ds Max\* and Maya\*. As vice president of product management at Havok, he deals directly with clients who have specific requirements and serves as the company's public face for high-level product strategy.

We caught up with Jeff to discuss Havok's expansion into areas beyond physics, Havok's underlying goals for middleware tools, game development's focus on characters, and what he sees for the future of animation and behavior technology.

## **Havok is mainly known for its physics, but your animation and behavior components have been in development for a few years now, right?**

*Jeff:* Yes. We had the coming-out launch of our animation product, Havok Animation, at Game Developers Conference (GDC) 2005. Since then we have continued to mature the product, and it is now in use in dozens of games. We've developed a complete content pipeline to help game developers get the animation assets in the game as effectively as possible.

For example, to deal with the natural memory limitations on many game systems, Havok Content Tools gather all the animation tracks in 3ds Max, Maya, and XSI, and compress them in a way that can be efficiently decompressed on a range of game consoles from the PS3 to the Wii. Compressing animation is a bit like compressing video: it relies on clever off-line analysis and packing that enables fast and relatively high-quality decompression in the game.

These compression settings are all controllable and previewable via the Havok Content Tools that install directly into popular 3D modeling tools. So you can get a sense of how much compression you are getting up front.

Havok Animation also offers various inverse kinematics (IK) solvers, with runtimes appropriate for game engines. We have Foot IK, which can adjust and climb over uneven terrain, as well as Hand IK, which can dynamically assess specific grab orientations for properly picking up objects and holding onto environmental elements.

Then there's what we call ragdoll mapping, which is basically runtime motion mapping between a simplified physical representation of the character, and its traditional "rig" or skeleton. For example, say you have a simple ragdoll specification with only a few spine bones (not a ton, because you don't need them for the physics), and also the higher-resolution bone system—between 50 and 100 bones—which is what animators usually work with. With real-time ragdoll mapping, we can let the physics drive the animation, the animation drive the physics, or some combination of the two. We do this to achieve full-body IK-like effects—with the added benefit that limbs do not pass through themselves or the body, nor through any other objects they interact with. It's the best of both worlds.

### **Are those animation features separable from the physics component?**

*Jeff:* Yes—Havok Animation is very modular and so things like the compression and decompression can be used by themselves as desired.

The IK is also independent. It's incorporated into the SDK structure in a way that is relevant for games, and it takes advantage of the underlying physics system to sense the environment and help make decisions. For example, consider what should happen to a character when one of its legs steps over a ledge. Should it strike an off balance pose, or transition into a falling state where it flails and grabs for targets, before hitting the ground below?

The Foot and Hand IK systems are equipped with all the bells and whistles to deal with these situations. And yet these systems can be used independent of physics if you like. But the physics gives the ability to make more informed decisions that make the animation look that much more convincing.

### **So how do you decide what development issues to move into outside of your core area of physics?**

*Jeff:* That is a great question. You know, a lot of it is deep soul-searching. [laughs]

At Havok, we try to extend from our areas of strength, from the core competencies that we already have. We would not want to jump into an adjunct area where we are totally green. We grew into animation through our demos, because people would say, "Well, you guys say you can blend animations and physics, but show us."

In our demo creations, we began to notice that what we were writing was beginning to look more and more like a product. So we decided to build and support a full animation system and tool chain that people could license separately or with Havok Physics—that's Havok Animation.

### **What about behavior? That seems like a different branch than physics and animation are.**

*Jeff:* Havok Behavior came about in much the same way as the Havok Animation product. When we were building our GDC 2005 demo to show off Animation and Physics, we wanted to show a guy walking around on uneven terrain—getting knocked about by objects, falling down, and then getting up again convincingly, using a blend of animation assets. We had the animation and physics bits, but we had to build a lot more code to deal with blending setup, and also the articulation of "states" to change the character's mode from running to falling, etc.

As it goes with games, the demo required a lot of creative tweaking to get the right game-play. Unfortunately, that tweaking had to be done by the programming team, and the creative iteration loop was slower than having the designers work directly with a tool. In the end we got the demo done "the old fashioned way," but we saw first-hand how the creative process could be improved by moving more of the tweaking and tuning process for the data-components "up stream," to free up the programming team.

So we built Havok Behavior—an SDK for the programmers, wrapped by a tool for animators and designers.

This was obviously a huge step, but we believe that characters are central to games and that it was definitely time to see some innovation in that area, in addition to physics. If we had been looking at the problem purely from the standpoint of knocking objects around in a physical world, I think we would have missed the boat.

### **With the behavior components, do you deal with AI at all?**

*Jeff:* That's an interesting question, because it gets down to the definition of "AI". With Havok Behavior, I would call it "personal AI." This is quite different from what people usually refer to as AI, which is often about macro path-finding for non-player characters. Path-finding is an important and difficult problem in its own right, but it doesn't take you that last mile when it comes to the finer-grained reactions of a character to its immediate situation.

For example, how does a character react to standing next to a door that's hot or to a loud sound coming from behind? His head is snapping back, his eyes are opening wide; maybe he's scratching his head because he's bored from waiting. Those scenarios can be authored and simulated using Havok Behavior without a lot of extra special code. In fact, using the Havok Behavior Tool, you can author these types of scenarios in ways that let you reuse one behavior between characters that have different bone structures and animation sets. Yet the character can still respond to requests from the game's AI system, which has a more global view of what's going on in the game.

In that sense, we definitely provide a kind of AI. And it helps eliminate the more basic tasks that the game's traditional AI system would otherwise have to deal with. So instead of having to think about how the character approaches the stairway and telling the character how to climb, the AI system can inform the character at a higher level and simply let the character's behavior figure out how to deal with the details. This gives AI developers more time to focus on innovative, macro issues like path-finding, group dynamics, and overall game AI.

Since we've developed Havok Behavior as an extensible plug-in system, there are a variety of ways the game's AI can be integrated with Behavior so they can work together. We're actually working with customers who are integrating AI systems with Behavior now, depending on the game play model. It's really fascinating to see how people are approaching that.

**Often, those low-level reactions with geometry are not accounted for, but gamers tend to accept those flaws because they're so accustomed to them.**

*Jeff:* True—picking up and grabbing objects is one such area. This is often covered up in games in clever ways that hide the actual problem. It's definitely hard, because anyone who dares to cross that line and put something forward as a technology that is better—but still not perfect—is still in a no-man's land. But we are making some big steps in that direction, and the physics enable that.

Using the Havok Content Tools back in the modelers, artists can now tag objects with handles and markers that play a critical role in the behavior of a character. We can use the physics optimally at run-time to find these handles, within a specific range, and then do something with them.

If a character is reaching for a weapon, we don't want to have to hand-code it in and say, "If gun X is nearby, do Y, else if club W is nearby do Z." Instead, we want to enable graphical authoring of general behaviors that say something like this: "If there is a handle of type "grab me" within .25 meters of my right hand, turn on my IK and ragdoll mapping systems to reach and lean and attach to that handle in its orientation, and then form a physical constraint between that object and my hand."

So physics is not just about bouncing off of things. It is also a key to enabling an efficient sensing of the run-time game's physical environment to enable lots of behavioral interrogation. It's fast and actually quite easy to set up in the Behavior Tool.

**How far along do you think we are in terms of reaching the point where those object-interaction decisions are achieved largely procedurally, as opposed to having to craft a lot of individual "reaching for the doorknob" animations, so to speak?**

*Jeff:* In the days of motion capture versus key-frame animation, people disagreed over which one was the "right" approach. I think everybody learned from that exercise that there was no single right answer, and it was best to have a system that could blend several approaches as desired.

To that end, we've engineered Havok Behavior with a variety of motion types in mind, combining real-time procedural motion, IK, and physics along with sample-based animation. The next step is for people to try them out and convince themselves that they can apply a range of techniques without losing artistic control and potentially reduce the amount of discrete animations that need to be created.

It's great to be able to progressively flesh out in-game character behavior in the early stages with a little animation and a lot of procedural fill-in, and thereafter drop in more animations as time and budget permit. It's analogous to sampling in the digital keyboard industry.

Of course, there is no replacement for fantastic, characterized hand-animated key-frame animation or motion capture. But if you have a system that supports IK and all of these things are dynamically blended where you can have pieces of both, you can go far with a smaller set of animations. I think we are there now.

Middleware companies like Havok who are providing these solutions are, in typical fashion, slightly ahead of the adoption curve. Game developers, mostly out of comfort, are still developing games by solving their problems using old approaches. Their content pipeline is set up to address things in a very literal way; and they don't have a lot of time and resources to innovate. Even getting out of the console cycle is painful, because everybody has been in cost-control mode.

In the same vein, when shaders originally came out, people were still using fixed-function pipelines and graphics pipelines. It took two years before it really found its way into production.

I think we are there now in terms of middleware technology and tools, and game teams are just starting to realize they can do these things—they just need a bit of time to try them out.

**Seeing a demo of these tools, it was surprising how many animation steps can be cut through the use of some blending and mirroring, and creative use of a few animations to imply a greater variety. It doesn't seem intuitive at first.**

*Jeff:* Yeah. I am one of those people who probably would have wanted to be classically trained as an animator, but as things played out, I ended up following the technical path. But I never stopped tinkering with animations on the side.

When I came to Havok, I had quite a bit of experience under my belt on every aspect of 3-D digital content creation, from modeling to rendering to animation. And it still took me a year to really get my head into the paradigm shift of where game characters were going. Once I did, we said, "Well, let's build a tool that takes this different way of authoring into consideration."

Now with a few character assets and Havok Behavior, I can get my character to navigate around and pick stuff up and throw things and do things functionally pretty quickly. Using Havok Behavior is a different way of thinking from the way that people generally approach these characters inside of Autodesk 3ds Max, Autodesk Maya, or Softimage. Because you get to move beyond literal key frames and polygons, and start thinking about the character's behavior. And that informs what you may want more or less of from the animation asset perspective. It is definitely a creative process of discovery.

Trying to educate the next generation of digital character creators on these differences is probably going to take some time. Many will still want to build just the off-line digital character content. But as tools like Havok Behavior become more accessible, game character behavior authoring will hopefully become a career path in its own right.

To that end, we've recently made the Havok Content Tools and the Havok Behavior Tool publicly available from our download site at [www.havok.com/tryhavok](http://www.havok.com/tryhavok). The SDK for physics and animation is there as well. But for folks who are interested in exploring this new zone of creativity for game character behavior, this kind of tool access is really unprecedented. With these tools today, character animators and designers can build astoundingly sophisticated content and literally "play-it" in the tool with a game controller and get a very real sense of how their game character will act in the game. If nothing else, that's just a whole lot of fun.

**Where do you see animation blending, procedural animation, and behavior technology going from your end?**

*Jeff:* Chris Hecker, whom I am sure you know, gave a fantastic talk last year at the Montreal International Game Summit. In talking about music and synthesizers, he mentioned how in the old days, synthesizers were purely synthetic, and then they became sampled. And rather than just one or the other, there was a blending of the synthesis approach and the sampling approach, which allowed musicians to achieve stellar results.

I believe the future is going to be driven by great samples—which in the case of animation could be fundamentally great leaping animations or reactive animations.

Think of it as a node tree or a processing graph—in the runtime—in which those types of animations actually get mixed in with constraints in the world. Factors such as where the ground is and how fast the character is moving become very malleable results that can change every time you play the game. That is definitely where things are headed.

**That could be comforting to traditional animators who can count on not having their skills replaced entirely by procedural algorithms.**

*Jeff:* Yes, that lesson has been taught and learned many times over in the film industry. Many efforts were made to "help" artists by giving them a procedural system that made it totally unnecessary for them to do what they loved to do. These systems have failed time and time again.

It's true that the potential wins with a procedural approach are greater for games than in other mediums like film, but I still think that it's basically about how the character is expressed. In that respect, the procedural way of expressing a character's personality is pretty limited when you think about the massive number of parameters that are there. The samples that animators make are really going to be here for a very long time.

**Do you work to integrate with various software suites, and do you work with developers to integrate into their own custom solutions?**

*Jeff:* Many people have asked if they can use our behavior products with their own animation system. Yes, we'll work with them to try to do that.

Inevitably many people will find that our product works well out of the box, and they'll go for our system. But we try very hard to be open. We answer questions and provide all the source code.

### **Visual Adrenaline: Do you also deal with console manufacturers to optimize for their platforms?**

*Jeff:* Absolutely. It's critical, especially on newer multi-core architectures. On every platform—whether a game console or a PC—we've always had a very strong history and a core competency in optimizing.

We have strong ties with all the platform manufacturers. The guys who sit in Havok's core architecture groups are brilliant at what they do, and they know how to optimize at the absolute lowest machine level.

### **Would Havok ever consider expanding its tool sets into a more self-contained engine package with a renderer, or are you staying focused on targeted tools?**

*Jeff:* I'm an advocate of making sure that you have components in your adjacent areas correct before expanding. But we would like to go further out. Since 2005, we've gone from basically one area to also having Havok Animation and Havok Behavior. And we just added two new products last year, Havok Destruction\* and Havok Cloth\*. I think we have our hands full [laughs].

We'll remain focused on that for awhile to make sure that we get it right. In our upcoming 6.5 release, we will be showing, in the behavior tool itself, the advantages of this kind of integration. The cloth effects created in our Cloth product can be pre-visualized entirely inside of the Behavior Tool. If you are blending a couple of locomotion states to a stop state in the state machine, the Cloth is going to behave differently every time depending on the game-play.

We are looking forward to bringing out more of that integration around the character. Cloth really makes much more sense for the character. The fact that you can preview clothing on characters right there in the tool will be a lot of fun.

### **Where do you see all these developments going, broadly?**

*Jeff:* I am interested to see what is going to happen as we look around the corner to the next generational cycle.

The industry seems to have settled in appropriately and comfortably to a level of computation costs on games, with last-generation tools and middleware technology. We are venturing out in front of people right now, and working on interesting run-time tools and solutions that are a little more computationally expensive, but let us do some really cool stuff. In line with that, we already see PC multi-core consumer sales figures climbing, and with the current generation of game consoles already offering multi-core capabilities, it seems reasonable to expect the next generation of consoles will only offer more. All of this indicates a good potential for driving more sophisticated simulation in the game around characters.

I am looking forward to a year from now, when people start thinking about targeting the next generation of PC and console architectures—how far will they be willing to reach out? I think that's when we are really going to see these pipelines turn over and push the technology again.

**EDITORS NOTE:** This article was independently written and commissioned by Gamasutra's editors, since it was deemed of value to the community. Its publishing has been made possible by Intel, as a platform and vendor-agnostic part of Intel's Visual Computing microsite at: [www.gamasutra.com/visualcomputing](http://www.gamasutra.com/visualcomputing).