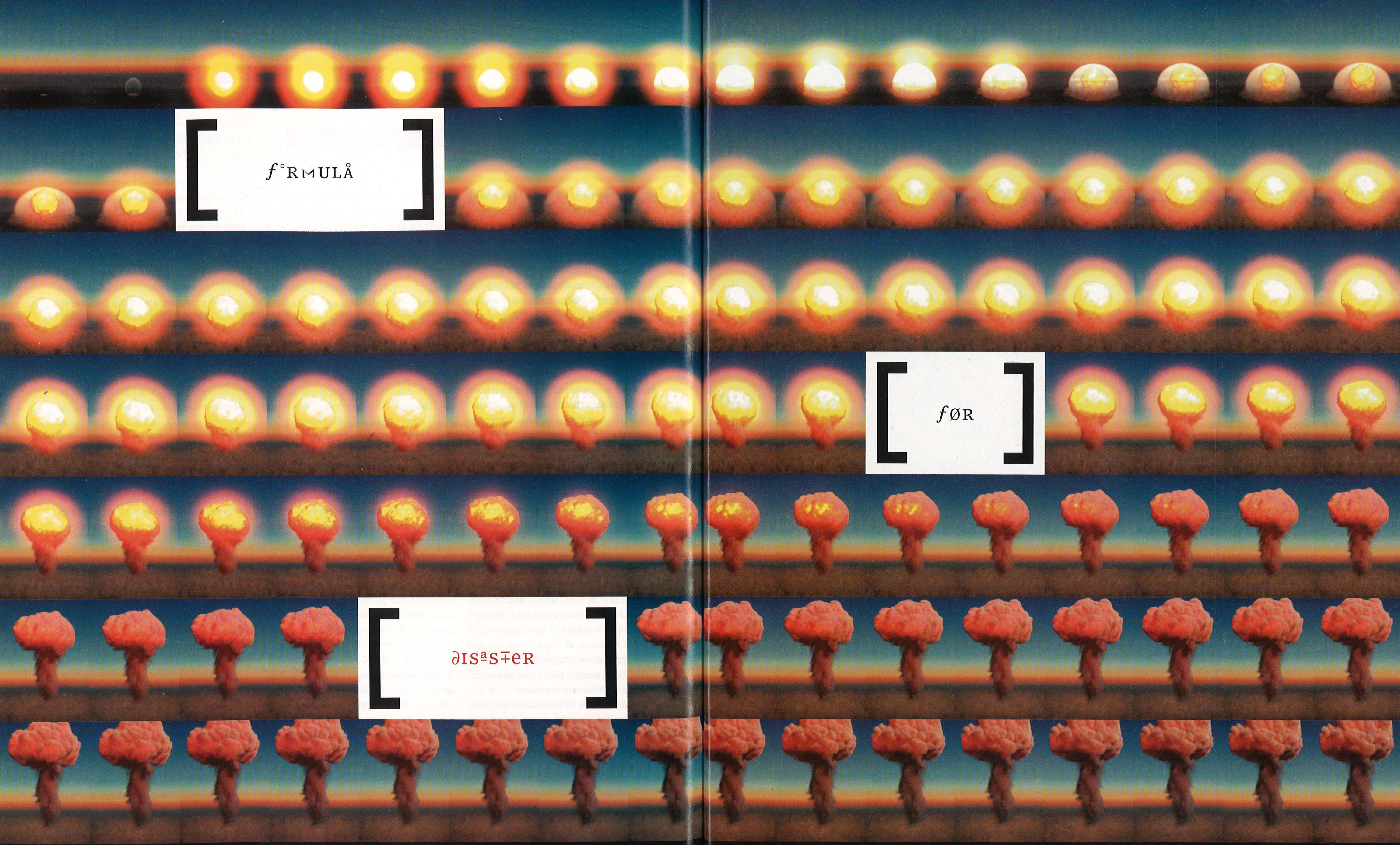
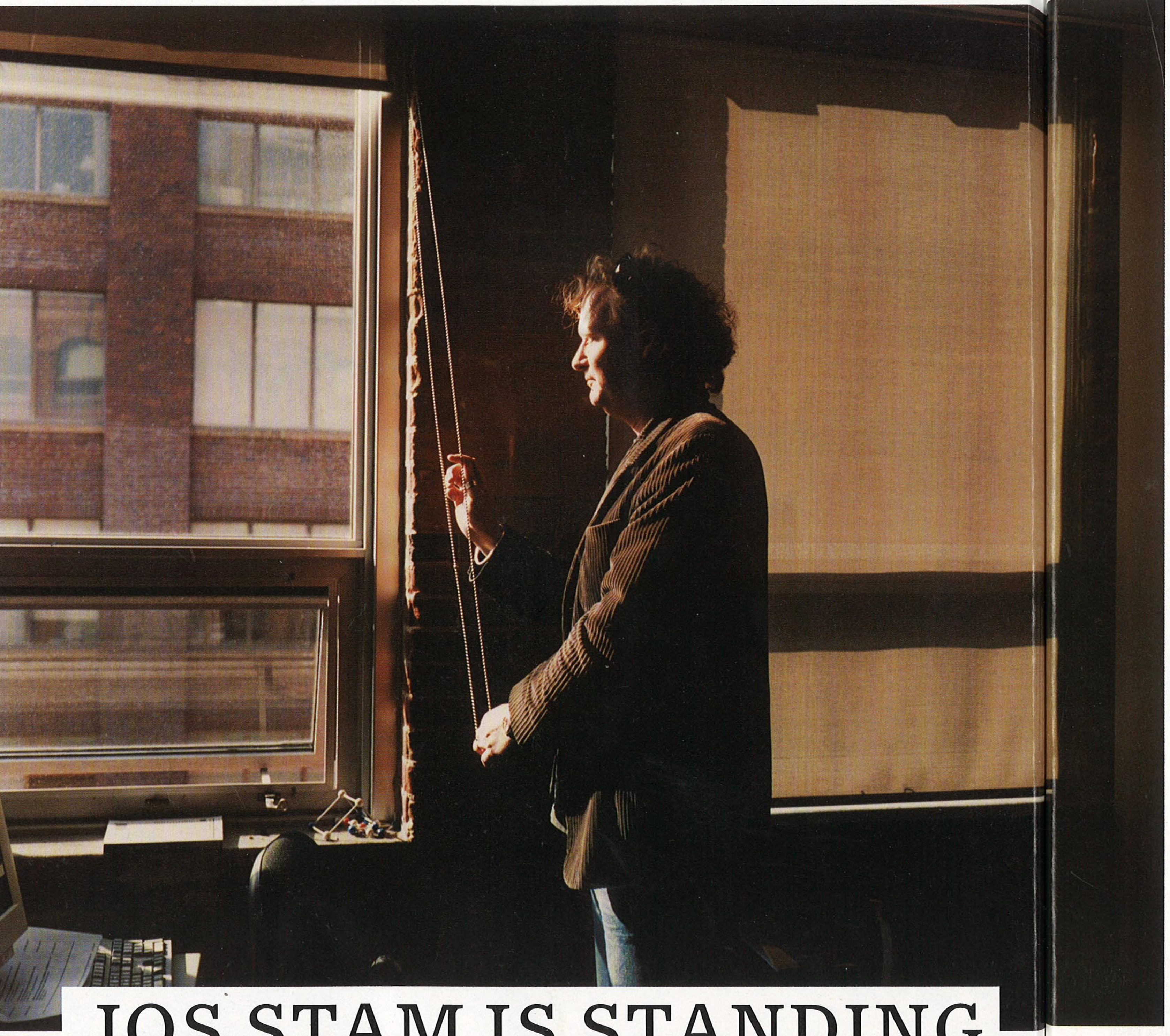


FORMULA FOR DISASTER

EXPLOSIONS. STORMS. WAVES. *CGI ace Jos Stam is creating a physics machine that can make special effects look absolutely, completely real.* **BY MICHAEL BEHAR**



Stam in his Toronto office: "I started coding just for the beauty of it."

JOS STAM IS STANDING

on a pearl-white beach under a cloudless sky. He is visiting his parents, who are vacationing in Faro, a medieval town on Portugal's Algarve coast. Stam, a 41-year-old computer scientist specializing in 3-D graphics, doesn't look at the world the way the rest of us do. Reality is a binary riddle to be cracked, a series of fleeting images best appreciated after they've been rendered into 1s and 0s. Even here, watching the waves hit a beach in Portugal, his thoughts drift, as they always do, toward numbers. He begins scribbling in a small black notebook filled with mathematical interpretations of everything he sees. ¶ Stam is a Nordic Goliath, a neck-craning 6′ 8″, with blond hair, pale green eyes, a deeply cleft chin, and hands the size of bear paws. He wrote the software behind many of the visual effects in modern Hollywood films—he is one of the few programmers to have won an Oscar—yet he's all too aware that no software can re-create the aquatic spectacle before him. Computers can simulate simple fluid motion, but on their own they still can't reproduce the complexity of a breaking wave.

PHOTOGRAPH BY Jason Nocito

From Jos Stam's notebook, an equation for the velocity of fluids:

$$\frac{\partial u}{\partial t} + (u \cdot \nabla) u = -\nabla p + \frac{1}{R} \nabla^2 u + f \qquad \nabla \cdot u = 0$$

And one for the density of smoke:

$$\frac{\partial \rho}{\partial t} + (u \cdot \nabla) \rho = K \nabla^2 \rho + S$$

Sure, *Titanic* and *The Perfect Storm* had digitally created oceans. But those effects depended on the tedious melding of multiple rudimentary computer simulations. Ten years later, no software can produce believable effects that don't also require untold hours of manual tweaking—and any time additional components are layered in by hand, the finished effect is less realistic. Stam calls the creation of a believable crashing wave, in all its multidimensional complexity, "the holy grail of computer animation." And he may be closer than anyone to finding it.

WHEN I FIRST meet Stam to discuss his work, at a tapas bar near his office in downtown Toronto, I admit right off that I'm a bit confused. "Aren't graphics programs already doing physics-based animation?" Answer: sort of. He glances around the room. He points out flickering candles, a sloshing glass of wine, and the billowy pleats and folds on the blouse of our waitress, who has just delivered a plate of lobster confit. All are formidable "problems," he explains, using the innocuous term that graphics coders reserve for the most daunting challenges. Stam himself has already devised an algorithm that crafts digital smoke with startling realism (it was used in *The Lord of the Rings* and *War of the Worlds*). But to create a digital wave or flickering flame that can realistically interact with other objects and forces (including rocky shorelines or light breezes) would require a CG-effects system that truly behaves in accordance with *all* the laws of physics. Such systems are still in their infancy—they're used to animate the simpler cartoon physics of videogames and certain discrete elements of movies—but it's not clear if any processor or software program will ever be powerful enough to mimic reality at the click of a mouse.

Stam is wearing designer blue jeans, purple lace-up combat boots, and a black T-shirt beneath a retro corduroy sport jacket. (When did coders stop being geeks?) He's so tall, he looks like he's about to fall out of his chair. He clutches the table and leans toward me. "Watching those waves really made me appreciate how hard it is to animate something that complex," he says of the beach in Portugal. "I'm fascinated by the mix of water, sand, and froth. But how would you model that? Is there an equation that accounts for all of it?"

Current animation software can't handle a lot of elements interacting within a single shot. A ship ablaze on a stormy sea, lashed by gale-force wind and rain, would be impossible to animate completely with existing software. The animations for the waves, ship, flames, wind, and rain have to be solved individually, then painstakingly blended and layered element by element into each frame of film. This can take a team of animators at a visual effects house many months and cost hundreds of thousands, if not millions, of dollars. Stam has a better idea: Teach visual effects software all the fundamental laws of physics and let it do the grunt work for you. Think of it as a unified field theory for animation—the animator can simply plug in the variables:

WIND SPEED:	25 KNOTS
WAVE HEIGHT:	7 FEET
SHIP'S MASS:	46,000 TONS
OCEAN DEPTH:	7,000 FEET

Additional details might include air and water temperature, time of day (for lighting effects), wind direction, source of ignition ... you get the idea. With the parameters set, hit Enter and voilà—the software would crunch the numbers and spit out the finished scene. At least, that's the theory. In practice, while algorithms for individual components (fire, water) already exist, integrating them all together has proven to be hideously complicated.

Stam likes hideous complications. He works for Autodesk, the software company that produces Maya, the world's leading 3-D modeling software. Maya has shaped nearly every CG visual effect you've seen since 1998—from the first *Matrix* to the newest *Spider-Man*.

With his current project, a new module for Maya called Nucleus, Stam is getting closer than ever to his goal. Stam's formulas allow two or three elements to interact as they would in the real word—smoke with wind, or fabric with solid objects. What hasn't been accomplished yet is a system that can encompass and express dynamically *all* of the earthly elements and physical laws at once. Stam wants software that can play God with pixels. Right now, he's only in the early stages, but Nucleus is taking the special effects world a step closer to Stam's dream of a unified physics-driven system that can produce eye-popping visual effects all by itself.

STAM, WHO GREW UP in Switzerland, didn't care much for math as a kid. He was more interested in art. "My hero as a teenager was Salvador Dali," he says. He spent his spare hours doing airbrush interpretations of classic Dali pieces, such as *The Persistence of Memory*, complete with melting clocks and gooey waterfowl. He even sold a few and shows me some photos of the paintings he still keeps in his office. We're sitting at his desk, or what I assume is a desk. I can't see it beneath the clutter. His workspace looks like the aftermath of a bomb detonated in a paper mill.

"I always thought I would be an artist or art historian," Stam says. But his older brother, Sim, would have none of it. After a relentless big-bro-knows-best campaign, Sim convinced Stam, then 14, to take a class in computer programming.

"He kept telling me I could make money doing it," Stam says. "I couldn't even type, so it took me forever to enter a program. But once I got the concept of what a program was and what you could make it do, it was a revelation. I stopped painting and started coding just for the beauty of it, the pure joy. It was all so logical to me, like poetry."

By the age of 18, he'd made up his mind to double major in computer science and mathematics at the University of Geneva. In 1987, he got an Amiga 1000, the first PC to feature a set of processors engineered for full-color animation. He wrote a *Pac-Man* knock off and made a "ray tracer" program for rendering 3-D images. "I was totally hooked," he says. In 1994, while earning a PhD from the University of Toronto, Stam got a call from the CG software firm Alias Systems (now part of Autodesk). Alias wanted to license some code from a paper he'd written and incorporate it into the company's next release of PowerAnimator, the predecessor to Maya. PowerAnimator had already generated the first-ever digital 3-D water effect, in *The Abyss*, and it brought to life the metal-morphing cyborg cop in *Terminator 2: Judgment Day*. Alias also gave Stam a job.

His first big assignment was tackling some problems with modeling surfaces. Real-world objects, especially curvy, multifaceted ones like clouds, mountains, plants, and animals, have an infinite number of measurable points. "You can only represent finite numbers in a computer," Stam says. So what numbers do you choose to model? A technique called subdivision modeling winnows down the choices by cleaving a smooth surface into polygons, the building blocks for 3-D images. The polygons are divided, then divided again, until they're small enough to encompass the object in a fine mesh grid that a computer can use to fool the human eye into believing it's seeing curves where there are none. "Think of it as always chopping off the corners of polygons until they appear smooth," he explains. Stam was tasked with making the process more efficient. Four weeks after getting the assignment, he came up with a faster and more elegant way to use subdivision modeling. "The result won me the Oscar," says Stam, who received an Academy Award for Technical Achievement in February 2005.

"The best animators are a combination of computer scientist and fine artist. They have the eye of an animator but the brain of a hardcore technologist. That's a rare commodity in this business."

THESE DAYS, NUCLEUS takes up much of his time. The first module, called nCloth, simulates how various fabrics—cotton, leather, silk, rayon—fold, curl, drape, and crease over a rigid surface. "The physics going into nCloth look great," says Kim Libreri, VP of advanced strategy at Digital Domain, a Hollywood effects house, and visual effects supervisor for the *Matrix* films. Stam and his research partner, Duncan Brinsmead, a principal scientist at Autodesk, show me a demo of nCloth. Brinsmead launches a short animation they call "Dancing Ballerina." As she pirouettes, the dancer's satiny purple costume twirls and bounces with uncanny verisimilitude—the digitized cloth is *interacting* naturally with the body beneath it, without the need for any additional rendering. Change the movement of the body and the fabric adjusts accordingly.

There are dozens of lengthy equations that go into this type of simulation: algorithms for the cloth, for the dancer's body, for light, and for shadows. And then there are algorithms that address the interplay between fabric, body, light, and shadows. You'd need more algorithms if she were dancing in the wind or the rain or in a smoky cabaret. And it's not enough to just add more computers. To build fast physics simulators that can push CG visual effects to the next level, animators have to rely on mathematical finesse, not processing brawn. "The best animators are a combination of computer scientist and fine artist," Libreri says. "They have the eye of an animator but the brain of a hardcore technologist. That's a rare commodity in this business."

It may be that Stam, who once made extra cash drawing caricature sketches of friends and family, is uniquely suited to figuring out exactly where the math ends and the art begins. He believes that flawless visual effects aren't a matter of just simulating everything down to the last megapixel. "What does it mean to be realistic?" Stam asks.

"The art is picking equations good enough to fool the moviegoer," says Peter Schröder, a professor of computer science at Caltech. "What Jos is doing with Nucleus is asking whether we can find some common mathematical principles without going all the way to something that is ridiculous from a computational point of view. In principle, we should be able to describe everything in a CG simulation with quantum math, because it's all just atoms bouncing around doing their thing. But at that level, it's not computationally feasible."

And anyway, the human eye doesn't need to see that much. Even without the complete picture, our brains fill in the missing information. "More detail is not always the key," Stam says. "Look at Rembrandt closely and it's just big blobs of paint. Sometimes you have to exaggerate things to make them more real."

MAYA'S MAIN COMPETITOR is a promising technology from Stanford University called PhysBAM (physics-based modeling), developed by computer science professor Ron Fedkiw. A consultant to Industrial Light & Magic on the making of *Terminator 3: Rise of the Machines, Star Wars Episode III: Revenge of the Sith,* and *Poseidon*, Fedkiw points me to a short CG sequence on his Web site that shows ice cubes tumbling into a glass of shimmering water, then sloshing around until they're dissolved. It doesn't seem that impressive until you realize that the simulation is done exclusively with algorithms that know how ice, water, and light interact naturally. Once the animation is initiated, the animator is completely hands-off. The result is not only realistic but also utterly random—different every time—exactly as if you tossed a handful of ice cubes into your scotch and jiggled the glass until they melted.

For filmmakers, random outcomes are not a good thing. "In *Poseidon* we had to do this big splash of water on the decks," Libreri recalls. "But we couldn't get the simulator to do what the director wanted." It turned out that the simulator worked perfectly: A 200-foot wave slamming into a cruise ship is going to do whatever the hell *it* wants. The irony is that Nucleus and PhysBAM may not make the work of a filmmaker any easier. Caltech's Schröder explains: "Let's say the director wants a shot where you let go of a cloth, and he wants that cloth to land on a particular branch in a particular tree. In the real world, what are the chances it will land on that branch? Basically, zero." And there's the rub: A perfect CG simulator for the real world will replicate precisely what happens in the real world: chaos.

On one of five computers in his office, Stam calls up a short smoke simulation. The aim is to get rising wisps to bend and twist in a predetermined direction but still look natural. He enters start and end points, then applies a subtle virtual nudge to the smoke for the interstitial frames. The simulation makes numerous attempts to form the desired shape and match the final keyframe—in this case, a smoky letter C. If it fails, it automatically restarts itself. After 30 minutes and about a thousand cycles, the smoke naturally wafts into a C.

As it turns out, a little chaos goes a long way. "Over-controlling simulations can ruin all the beautiful physics," Fedkiw told me. "We've found that less control, better algorithms, and a different breed of artist is the key."

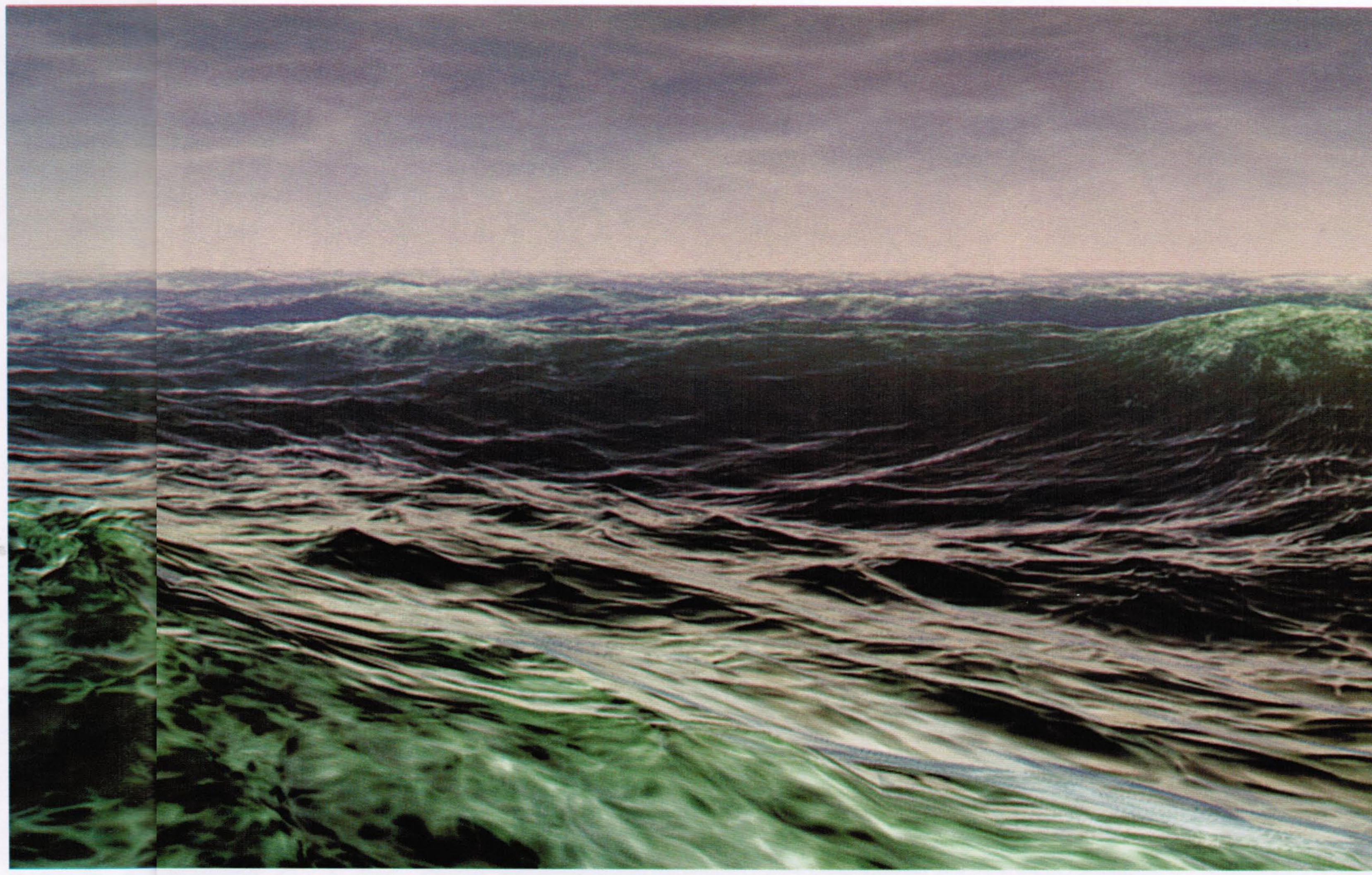
Does Stam ever worry that dazzling special effects will become too common and thus spoil their impact? "Filmmakers will just keep making it harder, doubling or tripling the size of a scene," he says. "But that's good for us because it keeps us in business."

I notice that there are short diagrammatical equations scribbled in red and black dry-erase marker on a floor-to-ceiling window that separates Stam's office from the main hallway. After a second, I realize that although the figures can be read by people outside the office, they have actually been written on the inside of the window—backward. It's a bit of mathematical macho designed to impress. "And they're solved for seven dimensions," Stam points out.

"Why seven," I ask. "Isn't three enough?"

"Oh, just for fun," he says.

MICHAEL BEHAR (michael@michaelbehar.com) *wrote about underwater logging in issue 15.02.*


Stam's Maya software renders waves so realistic that filmmakers used it for movies like *Spider-Man 3* and *The Day After Tomorrow.*