From *This Explains Everything: Deep, Beautiful, and Elegant Theories of How the World Works*, John Brockman, ed., New York: Harper Perennial, 2013. Pp. 269-272.

Why Do Movies Move?

Alvy Ray Smith

Movies are not smooth. The time between frames is empty. The camera records only twenty-four snapshots of each second of time flow, and discards everything that happens between frames—but we perceive it anyway. We see stills, but we perceive motion. How can we explain this? We can ask the same question about digital movies, videos, and videogames—in fact, all modern digital media—so the explanation is rather important, and one of my favorites.

Hoary old "persistence of vision" can't be the explanation. It's real, but it only explains why you don't see the emptiness between the frames. If an actor or an animated character moves between frames then—by persistence of vision—you should see him in both positions: two Humphrey Bogarts, two Buzz Lightyears. In fact, your retinas do see both, one fading out as the other comes in—each frame is projected long enough to ensure this. It's what your brain does with the retinas' information that determines whether you perceive two Bogarts in two different positions or one Bogart moving.

On its own the brain perceives the motion of an edge, but only if it moves not too far, and not too fast, from the first frame to the second one. Like persistence of vision, this is a real effect, called apparent motion. It's interesting but it's not the explanation I like so much. Classic cel animation—of the old ink on celluloid variety—relies on the apparent-motion phenomenon. The old animators knew intuitively how to keep the successive frames of a movement inside its "not too far, not too fast" boundaries. If they needed to exceed those limits, they had tricks to help us perceive the motion—like actual speed lines and a POOF of dust to mark the rapid descent of Wile E. Coyote as he steps unexpectedly off a mesa in hot pursuit of that truly wily Road Runner.

Exceed the apparent motion limits—without those animators' tricks—and the results are ugly. You may have seen old school stop-motion animation—such as Ray Harryhausen's classic sword-fighting skeletons in *Jason and the Argonauts*—that is plagued by an unpleasant jerking motion of the characters. You're seeing double, at least—several edges of a skeleton at the same time—and correctly interpret it as motion, but painfully so. The edges stutter, or "judder," or "strobe" across the screen—the words reflect the pain inflicted by staccato motion.

Why don't live-action movies stutter? (Imagine directing Uma Thurman to stay within "not too far, not too fast" limits.) Why don't computer animated movies a la Pixar judder? And, for contrast, why do videogames, alas, strobe horribly today? All are sequences of discrete frames. There's a general explanation that works for all three. It's called "motion blur," and it's simple and pretty.

Here's what a real movie camera does. The frame it records is not a sample at a single instant, like a Road Runner or a Harryhausen frame. Rather the camera shutter is open for a short while, called the exposure time. A moving object is moving during that short interval, of course, and is thus smeared slightly across the frame during the exposure time. It's like what happens when you try to take a long-exposure still photo of your child throwing a ball and his arm is just a blur. But a

bug in a still photograph turns out to be a feature for movies. Without the blur all movies would look as jumpy as Harryhausen's skeletons—unless Uma miraculously stayed within limits.

A scientific explanation can become a technological solution. For digital movies—like *Toy Story*—the solution to avoid strobing was derived from the explanation for live-action: Deliberately smear a moving object across a frame along its path of motion. So a character's swinging arm must be blurred along the arc the arm traces as it pivots around its shoulder joint. And the other arm *independently* must be blurred along *its* arc, often in the opposite direction to the first arm. All that had to be done was to figure out how to do what a camera does with a computer—and, importantly, how to do it efficiently. Live-action movies get motion blur for free, but it costs *a lot* for digital movies. The solution—by the group now known as Pixar—paved the way to the first digital movie. Motion blur was the crucial breakthrough.

In effect, motion blur shows your brain the path a movement is taking, and also its magnitude—a longer blur means a faster motion. Instead of discarding the temporal information about motion between frames, we store it spatially in the frames as a blur. A succession of such frames overlapping a bit—because of persistence of vision—thus paints out a motion in a distinctive enough way that the brain can do the full inference.

Pixar throws thousands of computers at a movie—spending sometimes more than thirty hours on a single frame. On the other hand, a videogame—essentially a digital movie restricted to real time—has to deliver a new frame every thirtieth of a second. It was only seventeen years ago that the inexorable increase in computation speed per unit dollar (described by Moore's Law) made motion-blurred digital movies feasible. Videogames simply haven't arrived yet in 2012. They can't compute fast enough to motion blur. Some give it a feeble try, but the feel of the play lessens so dramatically that gamers turn it off and suffer the judder instead. But Moore's Law still applies, so soon—five years? ten?—even videogames will motion blur properly and fully enter the modern world.

Best of all, motion blur is just one example of a potent general explanation called The Sampling Theorem. It works when the samples are called frames, taken regularly in time to make a movie, or when they are called pixels, taken regularly in space to make an image. It works for digital audio too. In a nutshell, the explanation of smooth motion from unsmooth movies expands to explain the modern media world—why it's even possible. But that would take a longer explanation.