from *SIAM News*, Volume 36, Number 9, November 2003

Hooked on Meshing, Researcher Creates Award-Winning Triangulation Program

By Sara Robinson

A physicist, an engineer, and a mathematician, one story goes, were all staying in the same hotel. In the middle of the night, the engineer awakened to the smell of smoke: Fire had broken out. He found a fire extinguisher, doused the flames, and went back to bed. An hour later, the fire rekindled, this time waking the physicist. "We have people who take care of this sort of thing," the physicist muttered to himself; he then woke the engineer and had him put out the fire, and went back to bed. An hour later, the fire rekindled yet again, awakening the mathematician, who looked at the extinguisher and said: "This reduces to a known solution." Then he went back to bed.

Even applied mathematicians are often pegged as developers of useless theories. But as the tale of Jonathan Shewchuk and a computer program called Triangle shows, when researchers are familiar with both the theory and the needs of practitioners, theory can have a dramatic effect on practice.

Triangle, created by Shewchuk, is a tool for generating two-dimensional meshes, a task that has long been critical to widely ranging applications, such as the solution of partial differential equations, terrain modeling, and computer graphics. The meshing problem is this: Given a region of the plane, a surface, or a volume, divide it into finite chunks, called elements, that have "nice" properties. For a "finite element" solution to a PDE, representing a physical property like airflow, computer programs will find approximate solutions that are piecewise linear or polynomial on the elements. For computer graphics applications, such as rendering a room, a mesh follows the contours of shadows and creates a basis for a shading model.

Meshes became a part of engineering practice in the 1970s, but commercial generators were expensive, slow, and often unreliable, failing on inputs involving complex geometry. For academic simulations, reliable public-domain mesh generators were unheard-of. Graduate students working on the project would typically tailor a primitive mesh generator to their immediate needs, explains Shewchuk, an assistant professor at the University of California, Berkeley.

"People thought of mesh generation as a little obstacle they had to get out of the way and didn't foresee that it was going to be the hardest part of the problem," Shewchuk says.

Triangle was among the first of a collection of general-purpose programs that make use of a decade's worth of theoretical advances in mesh generation. As its basis, it uses previously existing algorithms that are provably optimal in certain respects and are guaranteed to work on a large class of domains.

Jonathan Shewchuk, creator of the two-dimensional mesh-generation tool Triangle, received the Wilkinson Prize for Numerical Software at ICIAM 03. He's shown here at the 2002 International Meshing Roundtable under a triangular mesh produced not by Triangle but by the Object-Oriented Remeshing Toolkit. For a sample of a mesh that was *generated by Triangle, see page 8.*

Shewchuk improved the existing algorithms, making them more effective in practice. He also figured out a way around floating-point errors that doesn't significantly add to the running time. The resulting program is computationally efficient, robust, and reliable, demonstrating that meshes that are mathematically optimal can be made practically optimal as well.

"What made theory interesting to practitioners is that when there was no proof theoretically, programs would crash on tricky domains," says David Eppstein, a professor of computer science at the University of California, Irvine.

Indeed, Shewchuk's program has proved to be popular among practitioners. Since 1995, when it was first posted on the mathematical software repository Netlib, Triangle has been downloaded more than 75,000 times. Shewchuk has also licensed the program for inclusion in 11 commercial software packages.

In July, at ICIAM 03, Shewchuk received the Wilkinson Prize for Numerical Software, for the design and development of Triangle. Sponsored by Argonne National Laboratory, the National Physical Laboratory, and the Numerical Algorithms Group, the prize is awarded every four years to the entry that best addresses all phases in the preparation of high-quality numerical software.

A Change of Direction

Entering the graduate program in computer science at Carnegie Mellon University in 1990, Shewchuk was certain that he wanted to work in numerical analysis, focusing on parallel computing. He chose David O'Halloran as his thesis adviser and joined a group that was developing earthquake simulators for a civil engineering project known as Quake. Because the simulator had to solve Navier's equations of elastic behavior, the researchers used large meshes.

Shewchuk's first task was to design parallel solvers that would partition the meshes so the simulator could run on parallel

processors. With no freely available general-purpose meshing tools available, graduate students on the project had done the usual thing—writing a primitive meshing program. The techniques they used could generate only simple, block-shaped meshes, however, and the project researchers knew that they would eventually need something better. "Mesh generation was a constant theme running through the background of the project," Shewchuk recalls.

When Shewchuk was two years into the project, Gary Miller, a parallel algorithm expert at Carnegie Mellon, passed along a paper that would change Shewchuk's research direction. The paper, "Mesh Generation and an Optimal Triangulation," by Marshall Bern and Eppstein, was a survey of recent theoretical advances in mesh generation, some of it the work of the authors.

A Mathematical Approach

Throughout the 1970s and much of the 80s, mesh generation was seen as a minor task, peripheral to the main project. Nevertheless, publications on meshing began to appear in the engineering literature. Eventually, the problem caught the attention of theoreticians. In 1988, Brenda Baker, Eric Grosse, and Conor Rafferty, all of Bell Labs, published the first meshing algorithm that provided shape guarantees on the elements. Specifically, given an input polygon in the plane, the Baker–Grosse–Rafferty algorithm covers it with a mesh of triangles whose angles are guaranteed to be between 13 and 90 degrees as long as the polygon has no angles smaller than 13 degrees. This guarantee was significant for applications, given the convergence and stability problems of finite element methods when the underlying mesh contains overly skinny triangles.

Shortly thereafter, Bern, Eppstein, and John Gilbert published the first mesh-generation algorithm that provided guarantees not only on the quality but also on the number of the triangles. Whereas the Baker–Grosse–Rafferty algorithm begins by placing a uniform square grid over the polygon, the Bern–Eppstein–Gilbert algorithm replaces the uniform grid with a quadtree, a recursive subdivision into squares of varying sizes. This innovation enabled the algorithm to produce larger triangles in areas with large features while keeping the aspect ratio of the triangles bounded.

At about the same time, Cornell researcher L. Paul Chew published an algorithm based on Delaunay refinement, a completely different technique that results in a Delaunay triangulation. Delaunay triangulations—which have the property that no triangle has a vertex of the triangulation inside its circumscribing circle—have nice optimality properties and are particularly desirable meshes for finite element methods.

Chew's algorithm triangulates a given polygon into a mesh in which all angles are between 30 and 120 degrees. The algorithm produces triangles that are roughly the same size (a uniform mesh), and their number is guaranteed to be within a constant factor of optimal for uniform meshes. Uniform meshes can have far more than the optimal number of triangles, however.

The new algorithms—particularly the Bern–Eppstein–Gilbert technique, which was presented at the 1990 IEEE Symposium on the Foundations of Computer Science—sparked the interest of computational geometers. Their goal, Shewchuk says, was to have algorithms that are guaranteed to work on a broad class of domains and yield meshes of a guaranteed quality in a reasonable running time.

The algorithms mentioned in Bern and Eppstein's survey had some significant limitations, however. Meshes produced with the quadtree approach had far more triangles than those produced by heuristic methods. The Chew approach was sometimes better but could produce only uniform meshes.

Shortly after Shewchuk read the Bern–Eppstein survey paper, a fellow graduate student directed his attention to a new theoretical result, presented at the 1993 ACM–SIAM Symposium on Discrete Algorithms. In the paper, Jim Ruppert, then a graduate student at the University of California, Berkeley, used ideas from Bern–Eppstein–Gilbert to extend Chew's Delaunay refinement algorithm. The Ruppert algorithm produces a triangulation of any planar straight-line graph (a graph, whose edges are straight lines, that can be embedded in the plane without crossings). As long as the input angles are greater than 90 degrees, the mesh triangles are guaranteed to have angles between α and $180 - 2\alpha$, for α chosen between 0 and 20 degrees. The triangles vary in size according to the features of the initial graph, and the mesh is guaranteed to be size-optimal to within a constant factor, depending on α.

"I was completely electrified by [the paper] because it was so right," Shewchuk says. "I became obsessed and decided to implement it immediately." After 10 days cooped up in his office, translating the theory into code, he had a new program, the prototype of Triangle. It had some robustness problems, Shewchuk says, but the new program usually worked and produced "great meshes." Shewchuk was thrilled.

Writing the program hooked him on the field, and Shewchuk eventually switched his thesis topic to mesh generation. "I came to realize after that that although parallel computing was interesting, computational geometry was much more so," he says. He kept the same formal adviser and stayed on the Quake project but took on Gary Miller as a co-adviser.

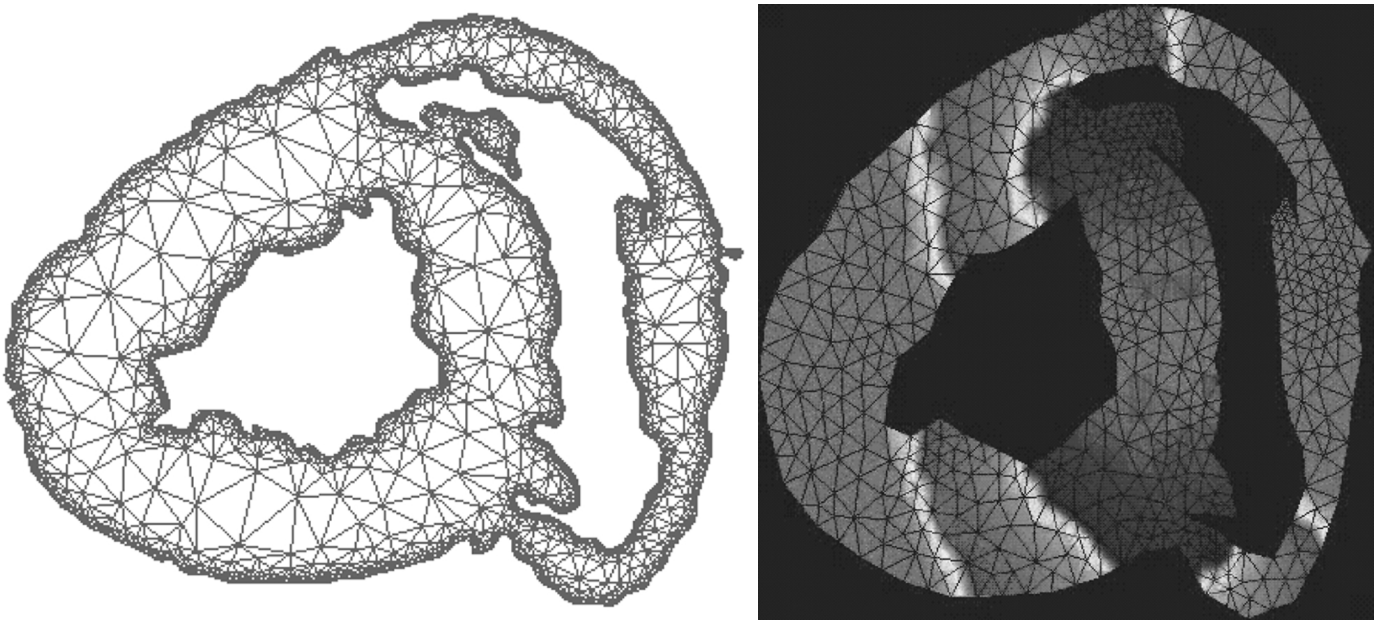
After releasing the first version of Triangle in 1995, Shewchuk continued to improve the program. He won CMU's 1997 dissertation award and, after a year as a postdoc at CMU, was hired as an assistant professor at Berkeley.

Triangle

For the first version of Triangle, Shewchuk followed Ruppert's approach, but with two algorithmic innovations. The first was to implement the fastest known method for producing a Delaunay triangulation for an initial domain. The second was to address problems with floating-point arithmetic.

The algorithm repeatedly uses an orientation test to see whether three input points lie clockwise relative to the coordinate system. The test comes down to computing the sign of a polynomial; because of roundoff error, however, computers frequently get the wrong answer when the value is close to zero. As a result, the program would sometimes come up with mutually inconsistent results and crash; at other times, it would produce triangulations that were nonsensical.

Shewchuk's solution was to use exact arithmetic in his program, which he did by devising an arithmetic simulator in software that



Triangle-produced meshes have been used to produce simulations of the propagation of electric current in the myocardium. Courtesy of John Pormann of the NSF Engineering Research Center for Emerging Cardiovascular Technologies.

can manage arbitrarily many digits. Exact arithmetic has other problems, however. Programs that use it are typically up to 1000 times slower.

Shewchuk was able to address this problem as well, by developing better exact arithmetic libraries for computing signs of polynomials. His approach is to do the computations adaptively, proceeding in the calculation only as far as needed to determine the sign of the polynomial. Although this technique didn't allow him to prove that his algorithm is fast on all inputs, for the vast majority of computations the technique works well, slowing the computation by no more than a factor of two and a half in worst-case tests.

After months of coding, Shewchuk released his program. The response was overwhelming: Hundreds of e-mail messages came from researchers using the program for finite element methods; several dozen respondents were working in computer graphics, and several dozen more on terrain databases and other geographic information databases, some from the Air Force and the Army. A common theme in the letters was delight in a program that didn't crash, Shewchuk says.

A later version of Triangle—the one for which he received the Wilkinson Prize—fixed the worst theoretical and practical flaw of Chew's and Ruppert's algorithms: their tendency to fail on domains with small input angles. A mesh generator cannot remove these input angles and (provably) must often create additional small angles in the triangles that mesh the domain. Shewchuk's solution was an algorithm that is guaranteed to create small-angle triangles only in the neighborhood of the small input angles.

On to Pyramid and Dynamic Meshing

Triangle is now in its fifth version, and Shewchuk is in the final stages of work on a program for three-dimensional meshes. Over the last few years, he has devised what he deems a fast and simple algorithm for three-dimensional mesh generation that works well for small angles. He has yet to implement it in his program, which will be called Pyramid.

Shewchuk says that his experience with Triangle gave him respect for the difficulty of translating theoretical ideas into a working computer program. "Making the program relatively user-proof with good error methods so that it would be flexible and wouldn't crash was much harder than doing the math and took a lot more time," he says. "There's a reason it's hard to move research into the real world—it takes a lot of work to go the extra mile to make something really usable and foolproof."

Undaunted by these difficulties, Shewchuk and Berkeley colleague James O'Brien are now at work on a new challenge, which they call the Snot project: modeling visco-plastic substances, such as slime, mucous, and syrup, using meshes that dynamically change with time yet retain their quality. "I'm trying to get a theoretical handle on it, and it's a problem that fights back hard," Shewchuk says.

Sara Robinson is a freelance writer based in Pasadena, California.