The Visualization of Mathematics: Towards a Mathematical Exploratorium

Richard S. Palais

Let us help one another to see things better.— CLAUDE MONET

Introduction

Mathematicians have always used their "mind's eye" to visualize the abstract objects and processes that arise in all branches of mathematical research. But it is only in recent years that remarkable improvements in computer technology have made it easy to externalize these vague and subjective pictures that we "see" in our heads, replacing them with precise and objective visualizations that can be shared with others. This marriage of mathematics and computer science will be my topic in what follows, and I will refer to it as *mathematical visualization*.

The subject is of such recent vintage and in such a state of flux that it would be difficult to write a detailed account of its development or of the current state of the art. But there are two important threads of research that established the reputation of computer-generated visualizations as a serious tool in mathematical research. These are the explicit constructions of eversions of the sphere and of embedded, complete minimal surfaces of higher genus. The history of both of these is well documented, and I will retell some of it later in this article.

Richard Palais is professor emeritus of mathematics at Brandeis University. His e-mail address is palais@math.brandeis.edu.

This article is dedicated to the memory of Alfred Gray.

Because some illustrated figures become clearer or more impressive when viewed in color or when animated, the author has made available a Web version of the article with links to such enhanced graphics. It is to be found at: http://rsp.math.brandeis.edu/VisualizationOfMath.html.

However, my main reason for writing this article is not to dwell on past successes of mathematical visualization; rather, it is to consider the question, Where do we go from here? I have been working on a mathematical visualization program[1] for more than five years now. In the course of developing that program I have had some insights and made some observations that I believe may be of interest to a general audience, and I will try to explain some of them in this article. In particular, working on my program has forced me to think seriously about possibilities for interesting new applications of mathematical visualization, and I would like to mention one in particular that I hope others will find as exciting a prospect as I do: the creation of an online, interactive gallery of mathematical visualization and art that I call the "Mathematical Exploratorium".

Let me begin by reviewing some of the familiar applications of mathematical visualization techniques. One obvious use is as an educational tool to augment those carefully crafted plaster models of mathematical surfaces that inhabit display cases in many mathematics centers [Fi] and the line drawings of textbooks and in such wonderful classics as *Geometry and the Imagination* [HC]. The advantage of supplementing these and other such classic representations of mathematical objects by computer-generated images is not only that a computer allows one to produce such static displays quickly and easily, but in addition it then becomes straightforward to create rotation and mor-

[1] *The program is called 3D-Filmstrip, but I will refer to it simply as "my program" in this article. Later in the article I will explain how to obtain a copy for personal use.*

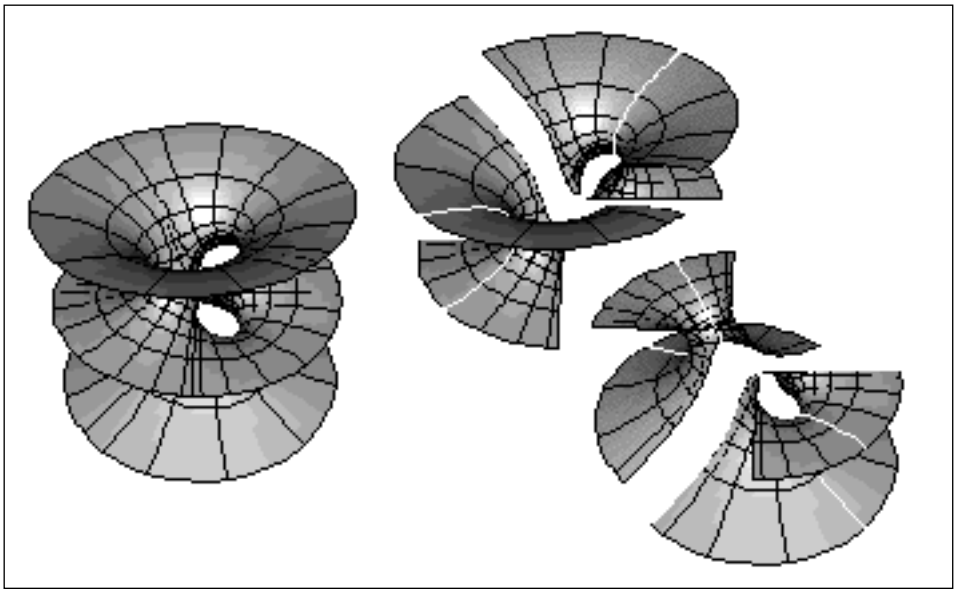

Figure 1. Symmetries of the Costa surface. The Costa surface (left) is cut by the three coordinate planes into eight congruent tiles, fundamental domains for the symmetry group. The horizontal plane cuts the Costa surface along two straight lines; the upper and the lower half are moved apart so that they do not overlap. The vertical planes are planes of reflectional symmetry and the symmetry lines are emphasized as gaps in the top and bottom part. The eight fundamental domains, one per octant, can each be represented as a graph, and Hoffman-Meek's theorem that the Costa surface is embedded follows easily from this fact.

phing animations that can bring the known mathematical landscape to life in unprecedented ways.

Even more exciting for the research mathematician are the possibilities that now exist to use mathematical visualization software to obtain fresh insights concerning complex and poorly understood mathematical objects. For example, giving an abstract mathematical object a geometric representation and then displaying it visually can sometimes reveal a new symmetry that was not apparent from the theoretical description. Just such a hidden symmetry, disclosed by visualization, played a key role in the Hoffman-Meeks proof of the embeddedness of the Costa minimal surface [H]. (See Figure 1.) Similarly, a morphing animation in which a particular visual feature of a family of objects remains fixed when certain parameters are changed can suggest the existence of a nonobvious invariant. The helicoid-catenoid morph that we discuss later is an example of this kind. (See Figure 2.)

Applied mathematicians find that the highly interactive nature of the images produced by recent mathematical visualization software allows them to do mathematical experiments with an ease never before possible. Since very few of the systems they deal with admit explicit, closed form solutions, this ability to investigate solutions visually has become an essential tool in many fields. For example, in studying fluid flow close to the onset of turbulence, the description of a velocity field in a small 3-dimensional region over a period of only a few seconds can generate trillions of floating point numbers. While there are statistical techniques for making sense of such huge data sets, displaying the velocity field visually is essential to get an insight into what is going on.

Also, scientists who need and use mathematics but are not completely at ease with abstract mathematical notations and formulas can often better understand the mathematical concepts they have to deal with if these concepts can be given a visual embodiment. Finally, there is no denying that mathematical visualization has a strong aesthetic appeal, even to the lay public—witness the remarkable success of coffee table picture books of fractal images!

Mathematical Visualization $\notin$ Computer Graphics

One important lesson I have learned from my own experience is that mathematical visualization programming should not be approached as just a special case of 3-dimensional (3D) graphics programming. While the two share concepts and algorithms, their goals and methods are quite distinct. Indeed, there are peculiarities inherent in displaying mathematical objects and processes that if properly taken into account can greatly simplify programming tasks and lead to algorithms more efficient than the standard techniques of 3D graphics programming. Conversely, if one ignores these special features and, for example, displays a mathematical surface with software techniques designed for showing the boundary of a real-world solid object, many essential features of the surface that a mathematician is interested in observing will end up hidden. The mathematician's fine categorization of surfaces into parametric, implicit, algebraic, pseudo-spherical, minimal, constant mean curvature, Riemann surfaces, etc., becomes blurred by the computer graphics notion of surface, and one quickly learns that not only are off-the-shelf computer graphics methods inadequate for creating and displaying all of these various types of surfaces but also that a special method designed to optimize the display of one type of mathematical surface may not be appropriate for others. One corollary of this is that it is not a good strategy to base mathematical visualization on some small fixed number of predefined, high-level graphics routines and expect that one will be able to shoehorn in all varieties of mathematical objects. Of course, one needs a number of low-level graphics primitives to get going, but instead of the Pro-

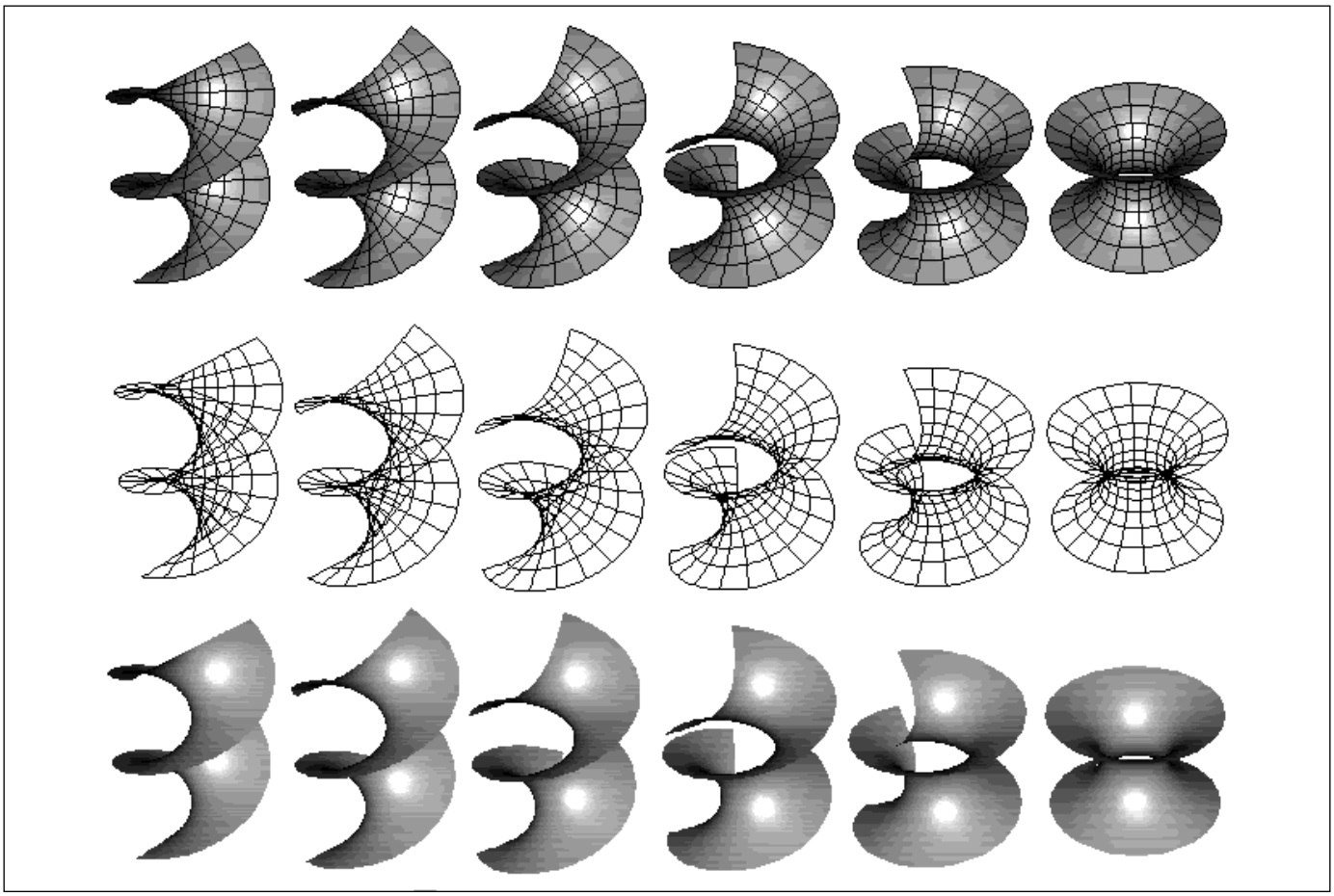

Figure 2. Helicoid-Catenoid Morph. Shown here (using three rendering methods) are six frames of the associate family morph joining the helicoid and catenoid minimal surfaces. Patch rendering (top) and wireframe rendering (middle) expose the isometric quality of the deformation, while ceramic rendering conceals it. Note how the automatic hidden lines feature of the painter's algorithm makes the patch version visually superior to the wireframe one.

crustean approach, attempting to fit each mathematical object to one of a few high-level display methods, it is better to use the low-level routines to design optimal display algorithms for each special kind of mathematical situation. This entails more effort for the programmer, but the superior results warrant the extra effort. A second corollary is that one or more mathematicians must play a central and ongoing role in the planning and development of any serious mathematical visualization software project. I consider myself fairly knowledgeable in differential geometry, and for much of the basic programming of the curves and surfaces parts of my program I played both the role of programmer and of mathematical consultant. But I found that it was absolutely essential for me to work closely with experts (Hermann Karcher and Chuu-Lian Terng respectively) to do a professional job in programming the creation and display of minimal surfaces and pseudospherical surfaces.

In what follows I will, as above, often illustrate some point by referring to the visualization of geometric objects like curves, surfaces, and polyhedra. I choose such examples mainly because they are highly intuitive and so require less explanation. But it is important to realize that almost all of the same points could be made in relation to the display of conformal mappings, solutions of ordinary and partial differential equations, or visualizations associated to almost any other category of mathematical object.

Multiobject vs. Single-Object Graphics Worlds

I claimed above that mathematical visualization has features that set it apart from general computer graphics and that requires some special techniques and algorithms. In this and the next section I will give two simple examples that illustrate this point.

If one examines a typical visual created by a 3D graphics program, say the lead-in to a nightly TV news program, one sees many different objects moving in disparate ways. A globe representing the earth spins around a vertical axis, while a logo zooms in as it simultaneously rotates about a horizontal axis, etc. This is an example of what I will refer to as a multiobject graphics world. The normal method for animating such a world is to create each 3D object in a "fiducial" location and ori-

entation and then associate to the object a 4×4 "update matrix" that, for each frame of the animation, will have the values needed to translate and rotate the object from its original position to the position appropriate for that frame. To create one frame of such an animation requires that each point of each object in this multiobject world be transformed by the matrix appropriate to its object. To create a real-time animation for a scene of any complexity using this method, one needs a very powerful computer—usually one with specialized graphics hardware.

On the other hand, if one examines a typical mathematical visualization, one sees that it consists of a single object (curve, surface, polyhedron, etc.) that is usually centered on the screen, and a rotation animation is almost always about the screen center. Let me refer to such a graphics setup as a single-object graphics world. Now, of course, one could treat such a setup as just a special case of an n object world, ignoring the fact that $n = 1$. But 1 is a rather special integer, and in fact there is a more efficient way to rotate a single-object world about an axis than applying the associated rotation matrix M to each of the points defining the object—namely, apply the matrix inverse of M to the viewing camera location and the three vectors defining its orientation. Visually this will have the same effect, but in general it will be considerably more efficient, and it can make it possible to do real-time rotation on simple desktop computers without special graphics hardware.

Offscreen vs. Onscreen Drawing

A standard rendering technique for displaying a surface on a computer monitor is the "painter's algorithm". The surface is represented as the union of "facets" (colored polygons, often triangles or rectangles). These facets are sorted by their distance from the viewing camera; then they are "painted" on the screen from back to front. This has the obvious advantage of automatically hiding those facets that are behind other facets.

Computer graphics experts nearly always couple the painter's algorithm with a second technique called "double-buffering" or "offscreen drawing". That is, they first draw the entire surface in a so-called "offscreen buffer", a block of computer memory that duplicates the video display memory. Only when the surface is complete is this buffer copied back to the video display. The result is that the completed surface suddenly appears on the monitor. The reason for double-buffering is that, in most situations, the user is not supposed to see the ugly sight of a partially painted surface.

But in certain situations, using offscreen drawing to display a mathematical surface is a programming offense that approaches a felony! Most interesting surfaces are highly complex and often immersed rather than imbedded. Viewed from any location, there will be several "sheets", and important details on far sheets will be obscured by the nearer sheets. Watching such a surface gradually being built up by the painter's algorithm can be a remarkably revealing experience, playing the role for a geometer akin to dissection for an anatomist.

Nevertheless, I receive occasional well-meaning e-mail messages from nonmathematicians with a knowledge of 3D computer graphics who have somehow come across my program. The message is always the same: my program is very nice, but I should really get hold of an elementary text on computer graphics and learn about double-buffering so that I can get rid of those silly partially drawn surfaces! (I usually respond that I do use double-buffering: after the surface is completely drawn on-screen, I copy the video RAM to an offscreen buffer that I use for screen updating. I suspect this completely backwards way of doing things convinces them I am hopeless, since the exchange usually ends there.)

Processes

It Is Important to Visualize Processes As Well As Objects

When I started developing my program, I felt that the main task of a mathematical visualization program was to display various mathematical objects. But as time has passed I have come to realize that this is only part of the story and perhaps much more important is the display of mathematical *processes*. I would be hard put to give a precise mathematical definition of what I mean here by "process"—one that would cover all the important cases that might arise—but, roughly speaking, I mean an animation that shows a related family of mathematical objects or else an object that arises by some procedure naturally associated to another object. Perhaps it is best to explain with a few examples.

Morphing

Morphing is one of the most important processes, so let me explain it first. Most mathematical objects occur in natural families that are described by certain parameters—also called moduli if we first divide out an appropriate group of automorphisms. For example, an ellipse can be described by five parameters, the coefficients of its implicit equation, and, if we divide out by the rigid motions of the plane, by two moduli, namely, the lengths of the two semi-axes. One initial goal for a mathematical theory of some new kind of object is usually a "classification theorem"—roughly speaking, discovering the space of moduli. The next step is a detailed investigation of the moduli space to see how various properties of the object depend on the moduli and to see what values of the moduli give rise to objects with special, interesting properties. For example, when its two semi-axes are equal, an ellipse is a circle and has a continuous group of

symmetries, while the generic ellipse has only a finite symmetry group.

If we can devise a good way of displaying an object graphically, including its dependence on moduli, then we can move along a curve in the moduli space and draw frames consisting of the graphical representation of the object at various points along the curve. If we then play these frames back in rapid succession, we get a movie of how the object changes as we change the moduli along the curve, and this is what I call a *morph*. Clearly this can be a powerful tool in investigating the moduli space. Often, even when the moduli space is infinite dimensional, it will contain special curves that provide interesting morphs.

For example, minimal surfaces come in one-parameter families (so-called *associate families*), all of whose members are isometric, though usually not congruent. Using the associated family parameter as a morphing parameter provides a particularly beautiful animation, one that in principle can be modeled in sheet metal. The helicoid and the catenoid belong to an associate family, and differential geometry books often show several frames of a morph between them. (See Figure 2.)

Similarly, the space of moduli for pseudospherical surfaces can be identified with the space of solutions of the Sine-Gordon partial differential equation. The latter contains certain n-parameter families (the pure n-soliton solutions) that correspond to particularly interesting surfaces. The 1-solitons correspond to the well-known Dini family, which contains the pseudosphere, and it was Chuu-Lian Terng's desire to see how properties of the corresponding surfaces changed as one morphed within the 2-soliton family that provided the original motivation for starting work on my program. (See Figure 3.)

The morphing process is such a powerful and revealing tool that whenever I add a new category of mathematical objects to the repertory of my program, I spend a lot of time thinking about and experimenting with creative ways to use morphs that are particularly adapted to that category. For example, I found that in displaying conformal maps, morphing along a carefully chosen path between a given map and the identity map is a particularly good way to reveal the structure of the map. And morphing is the obvious method for displaying the bifurcations of solutions of ordinary differential equations or for watching the onset of chaos as some key parameter is varied. A typical case of the latter is the well-known Lorenz equation that provided one of the motivations for early work on chaos. In fact, Lorenz used the primitive computer techniques available to him at that time to vary the Reynolds number and watch as an attracting fixed point turned into the "Lorenz Attractor".

More Processes

Let me quickly mention just a few other "processes" to illustrate further the scope of that term.

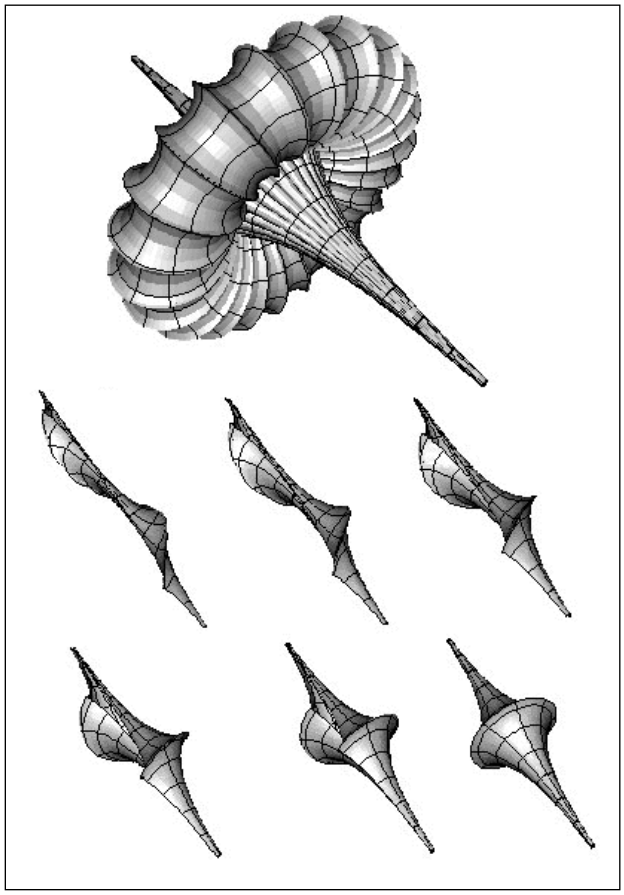

Figure 3. Pseudospherical Surfaces. Solutions of the Sine-Gordon Equation (SGE), $u_{tt} - u_{xx} = \sin(u)$, correspond one-to-one with surfaces in R^3 that model Lobachevsky's hyperbolic geometry. SGE is a soliton equation, and at top we see the surface corresponding to a time-periodic solution with soliton number 2 called The Breather. Below that is a six frame morph through the Dini family of surfaces, corresponding to the 1-parameter family of SGE 1-solitons.

If a plane curve is given, it is revealing to show an animation in which the "osculating circles" are drawn at a point that moves along the curve, the centers of curvature tracing out the evolute of the curve as the animation proceeds. In fact, there are many such classical processes that associate other curves with a given curve (pedals, strophoids, epicycloids, parallel curves, etc.), most of which become much easier to understand and illustrate with a computer.

For a space curve, an interesting process is the construction of a "tube" about the curve. This construction involves choosing a framing for the normal bundle to the curve, usually the "Frenet frame", and the tube serves to reveal the important (but usually invisible) framing. One's first impulse is to choose a tube with a round cross-section on aesthetic grounds, but to see the framing clearly, one should use a tube with a square cross-section. Once they see the point, mathematicians will almost

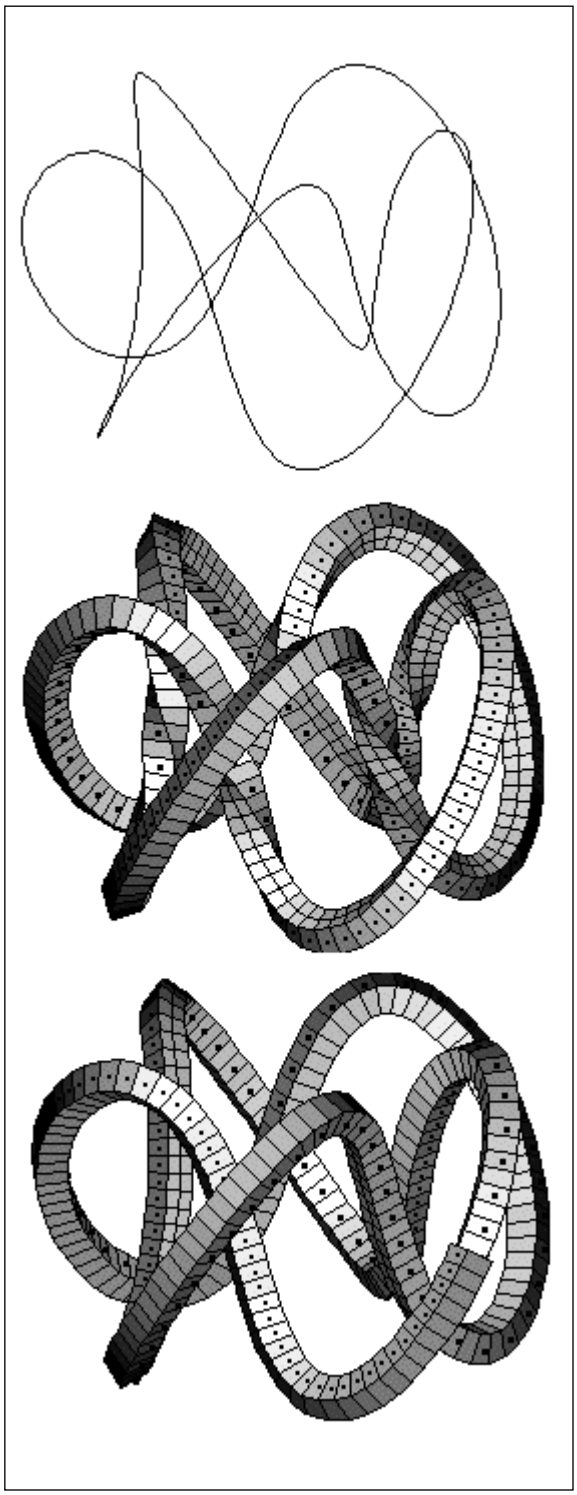

Figure 4. Tubes about a Torus Knot. A projection of the 2, 5-torus knot, and of two tubes about it with square cross-section. The upper tube uses the Frenet framing of the normal bundle; the other uses parallel framing. Notice how the tubes expose the 3-dimensional nature of the knot. One can see the Frenet frame twist more rapidly in the inside of the torus, and the eye also detects the nontrivial holonomy of the parallel framing, shown by the tube not matching up in the lower right.

always prefer the latter. Most people are usually very surprised to see how fast the Frenet frame may twist where the curvature is small. It is also interesting to switch from the Frenet frame to a parallel framing of the normal bundle. In this case there is no twisting, and the framing indeed looks parallel. But now the holonomy becomes strikingly visible: in going around a closed curve, the frame does not usually return to its starting value. (See Figure 4.)

For a surface, important processes are the construction of its focal sets and parallel surfaces.

For a polyhedron, two interesting processes are the constructions of its stellation and its truncation (the latter is what converts a regular icosahedron into a buckyball), and I find it instructive to morph between the untruncated and truncated forms.

Mathematics vs. Art

One should not confuse mathematical visualization with mathematical art. By the latter I am referring to the work of talented graphic artists and sculptors whose principal subject matter originates in the world of mathematics. Everyone has seen the fascinating and beautiful mathematical drawings of M. C. Escher [Sc], the famous Dutch graphic artist of the first half of this century. More recently, the Russian mathematician and artist Anatoliĭ Fomenko has enriched our mathematical heritage with spectacular images of surreal vistas, drawn from the depths of his own inspired imagination, that illustrate and illuminate complex mathematical concepts [Fo]. Currently a new generation of artists is finding inspiration from the platonic world of mathematics. Prominent among these are the sculptors Helaman Ferguson, Charles Perry, and Brent Collins. All of them use mathematical visualization software to create the objects that underlie their sculpture, but like Escher before them, they then impress their own artistic vision on the mathematical raw material from which they start.

One seeming distinction between a mathematical visualization graphic and a piece of mathematical art is the apparent difference in time and difficulty it takes to produce them. The former is usually generated completely automatically, often in only a few moments of computer time, while the latter often takes days or even weeks of skilled handwork by the artist, perhaps preceded by an even longer period of thoughtful planning. But this way of seeing things distorts a deeper reality. The serious work of planning and creating a mathematical visualization graphic really takes place when the algorithms are developed and coded, and for complicated objects this can be a long and arduous piece of research. It is the programmer, not the computer, that creates a mathematical visualization.

The real difference between the two lies in their ultimate goals. In the creation of mathematical art, mathematics is a starting point, but art controls—"artistic license" is granted the artist to deviate from perfect fidelity to the mathematics and to use other aesthetic principles to emphasize aspects of reality that the artist is trying to show us.

But in the creation of a mathematical visualization, the controlling principle should always be to show as clearly as possible the underlying mathematical qualities and properties of the objects being visualized. The temptation to "pretty up" a visualization should be resisted, particularly if mathematical information gets lost in the process. One example of this principle was mentioned above, namely, using square rather than circular cross-sections for tubes around space-curves in order to make visible the framing of the normal bundle.

Here is another example, this time from surface theory. Examination of a great many computer-generated surface visualizations will show that they almost all fall into one of three main types that I will refer to as wire-frame, patch, and ceramic. The term "wire-frame surface" is self-explanatory. In a patch rendering, one still displays the wire-frame skeleton but in addition fills in each of the rectangular patches with a color that mimics the way white paint on the surface would reflect light from several light sources with different positions and colors. If these positions and colors are chosen with care, a patch mode rendering will give a realistic 3-dimensional appearance to the surface. In a ceramic rendering of a surface, the wire-frame is removed and only the colored patches are displayed. If the patches are small enough, the color of the surface will appear to vary smoothly, and the resulting rendering is again realistic.

Now, a nonmathematician may feel that the wire-frame skeleton is extraneous and that the ceramic version looks more beautiful. But beauty, it is said, is in the eye of the beholder, and to the eye of a geometer it is the wire-frame[2] or patch version that frequently looks more beautiful, since it conveys extra mathematical information that is discarded along with the wire-frame mesh. In fact, if chosen with care, the mesh will be an orthogonal net that displays the conformal structure or even the Riemannian metric induced from the immersion of the surface into R^3. A dramatic way to illustrate this is to watch in succession three versions of the helicoid-catenoid morph, using first wire-frame, then patch, then ceramic rendering. The crucial fact that one wants to illustrate, namely, that an associate family deformation is isometric, fairly jumps out of the screen in the first two versions but is completely hidden with a ceramic rendering (Figure 2). My point is not that a patch rendering is necessarily always better than a ceramic one, but rather that when using software to create a mathematical visualization, the ideal should be to carefully tailor all available options to display best the mathematical features that need emphasizing in a particular situation, and aesthetics should not be permitted to override mathematical considerations.

The Mathematical Exploratorium

It is no secret that the incredible quantity of information on the World Wide Web is as yet poorly organized and is not easily classified as to relevance and quality. Trying to separate nuggets of serious value from all the dross can be a frustrating experience. Asking any of the various Web search engines to provide a list of Web sites that contain references to almost any imaginable topic will result in bushels of possibly relevant Web addresses (i.e., URLs), but sifting through them to find the few that are of serious interest is usually an arduous and time-consuming task.

I think of my drawings as if they were photographs of a strange but real world.
—Anatoliĭ Fomenko

Over the past year I have been diligently searching the Internet for sources of mathematical visualizations,[3] both as preparation for writing this article and for use in another project in which I am involved. It was a pleasant surprise to see how many things one can find already, and this corpus is growing rapidly. Of course it is not of uniform quality—some is amateurish and slapdash—but there is also much of professional quality. As I gradually arranged this material for my own immediate purposes, I began to realize what a useful resource could be created by carefully organizing all the best-quality visualizations and animations of mathematical objects and processes, cataloging and documenting them to form an online virtual museum of mathematics that I refer to as The Mathematical Exploratorium. Let

[2] *To be sure, wire-frame rendering lacks 3-dimensionality, but that defect can be easily overcome by using various stereo vision techniques. The name of my program, 3D-Filmstrip, was chosen because it emphasizes stereo rendering of 3D objects using the* anaglyph *method (i.e., using red/green glasses).*

[3] *I have created a gallery of 3D-Filmstrip visualizations and animations and placed it on the Web. The main catalog is at the Web address* `http://rsp.math.brandeis.edu/3D-Filmstrip_html/Galleries/Catalogs/MainCatalog.html`, *and that catalog also contains links to many of the best examples of mathematical visualization that I know of on the Web.*

me explain what I have in mind in a little more detail.

The Mathematical Exploratorium would be divided into "wings". There would be a Surface Wing, a Polyhedron Wing, a Fractal Wing, a Tiling Wing, an Ordinary Differential Equation Wing, etc. There would also be a wing devoted to a Museum School, where there would be software packages for the creation of visualizations as well as documentation and tutorials explaining their use. Each wing would be divided into galleries: for example, the Surface Wing would have a Pseudospherical Surface Gallery, a Minimal Surface Gallery, and so on, and some galleries would be further divided into alcoves.

There would be a main catalog that would list in an abbreviated format all the "holdings" of the Exploratorium, and each wing and gallery would have its own more detailed catalog, with thumbnail previews of all the objects it contains. Of course, these catalogs would be written in html (the "hypertext markup language"), and clicking on the name of an object or an animation would bring it up on the computer screen.

Each visualization would be accompanied by a short label giving its identity, its creator, and other items of a bibliographic nature. In addition, the label would contain a link to detailed mathematical documentation of the object being visualized—discoverer, special properties, mathematical theorems it illustrates, relations to other objects, interesting ways to morph it, etc. Similarly, each wing and gallery would have documentation that gives a quick overview of the mathematical area it covers and references to one or more monographs that cover the subject in detail.

A Little History

Two problems in mathematics have helped push the state of the art in mathematical visualization—namely, the problems of everting the 2-sphere and of constructing new, embedded, complete minimal surfaces, especially higher-genus examples. In the case of eversion, the goal was to illuminate a process so complex that very few people, even experts, could picture the full details mentally. In the case of minimal surfaces, the visualizations actually helped point the way to rigorous mathematical proofs.

Everting the Sphere

Let $f : S^n \to R^{n+1}$ be a smooth map of the n-sphere into Euclidean space of dimension $n+1$. We recall that f is called an *immersion* if it is locally a nonsingular embedding or, equivalently (by the Implicit Function Theorem), if at each point p of the sphere the differential, Df_p, is injective. In this case, we can associate to f a self-mapping G_f of S^n (called the *Gauss map* of f) as follows. The (oriented) tangent space of S^n at p is mapped by Df_p onto an oriented n-dimensional subspace V of R^{n+1}, and $G_f(p)$ is that one of the two unit normals to V that extends the orientation of V to the standard orientation of R^{n+1}. We call the degree of G_f the *turning number* of f.

A homotopy f_t of $f = f_0$ is called a *regular* homotopy if each stage is an immersion and if, in addition, $f_t(x)$ is jointly smooth in t and x. In that case, Df_t is a homotopy and is easily seen to induce a homotopy G_{f_t} of Gauss maps so that the turning numbers of f_0 and f_1 will be equal. It is a simple exercise to compute that the turning number of the identity map is 1 while that of the antipodal map is $(-1)^n$.

An *eversion* of the n-sphere is by definition a regular homotopy between the identity map and the antipodal map—in effect it turns the n-sphere inside out without creasing it along the way. By what we have just seen, there can be no eversion of a circle (or any odd-dimensional sphere). But how about the 2-sphere? The turning number is not an obstruction, but can we really turn it inside out? For most differential topologists in the mid-1950s it seemed that the answer must be no, so there was considerable surprise—and even some disbelief—when Stephen Smale in his thesis [Sm] proved a general result having as a corollary that *any* two immersions of the 2-sphere in R^3 were regularly homotopic. Smale's proof was in principle constructive, but it was so complicated that it did not really provide an effective method for giving an explicit eversion. The first explicit eversion was apparently discovered by Arnold Shapiro in 1961. Shapiro never published it, but it was described (with illustrations) by Anthony Phillips in a 1966 *Scientific American* article [Ph] that first brought the sphere eversion problem to public attention.

All proposed explicit eversions have been so complicated that most people are able to understand how they work only by watching an animated visualization played many times over. The first reasonably simple eversion was discovered by Bernard Morin in 1967, and a number of stages of Morin's eversion were rendered into chicken-wire models by Charles Pugh. Nelson Max [MC] digitized the grid points of these models by making careful hand measurements of their locations and then using a computer to interpolate the resulting 3-dimensional grids, creating in this way a morphing animation in the form of a movie (*Turning a Sphere Inside Out*) that provided the final "seeing is believing" argument to convince any remaining doubters that the 2-sphere could indeed be everted. In two further films (*Regular Homotopies in the Plane, Parts I and II*) Max used visualization techniques to explain the statement and proof of the so-called Whitney-Graustein Theorem—the fact that equality of turning numbers is not only necessary but also sufficient for two smooth immersions of the circle in the plane to be regularly homotopic.

Many more computer-generated visualizations of eversions have been proposed since that first one. One, suggested by William Thurston, has been made into a beautiful video called *Outside In* [L]. It is available, accompanied by a highly readable brochure called *Making Waves*, written by Silvio Levy. The latter documents the making of the video and the mathematics behind it and also gives further details concerning the history of sphere eversions through 1995. A recent and very interesting sphere eversion that uses Brakke's Surface Evolver software is described in [Sc] and [FSKB].

Constructing Embedded Minimal Surfaces

The theory of minimal surfaces is a fascinating mixture of complex function theory, partial differential equations, and differential geometry, and for well over a century it has attracted the creative energies of successive generations of mathematicians. Recent activity has centered on the study of embedded, complete minimal surfaces of "finite topology" (i.e., conformal to a compact Riemann surface with a finite number of points removed). Two decades ago the only known examples of such minimal surfaces were the plane, the catenoid, and the helicoid, all of which were already known to the geometer J. Meusnier at the time of the American Revolution. The fact that no more had been discovered over such a long period led to the obvious conjecture that there were in fact no others. About thirty years ago Robert Osserman started investigating a somewhat more restrictive class of complete minimal surfaces, namely, those for which the total curvature (i.e., the integral of the Gaussian curvature) was finite. Osserman's investigations eventually led to very tight constraints for any possible new embedded example of such a surface. Another decade passed before Celsoe Costa, in his thesis, discovered an example of a finite-curvature minimal immersion of the square torus with three punctures that fit all the known constraints for it to be embedded. But the equations were so complicated that there seemed no way to approach the problem of providing the analytic details required for a rigorous demonstration that Costa's surface had no self-intersections.

David Hoffman heard about the Costa example in a telephone conversation with Osserman and quickly decided to use computer graphics techniques to attempt to visualize Costa's surface well enough to check whether it at least appeared to be embedded and, if so, then perhaps also to see some visual clues that might help prove embeddedness rigorously. Hoffman discussed his ideas with William Meeks, who also became excited about the possibility of using computers in such an innovative way. Working together with James Hoffman, an expert in computer graphics programming, they were able to carry out this program over the course of several weeks, during which they alternated between staring at computer-generated images and finding rigorous proofs for the surprising conjectures those pictures suggested. (See Figure 1.) David Hoffman's well-written article [H] describing this project makes wonderful reading. I can do no better than quote a little from his telling of the story:

> ... we were able to create pictures of the surface. They were imperfect ... [H]owever, Jim Hoffman and I could see after one long night of staring at orthogonal projections of the surface from a variety of viewpoints that it was free of self-intersections. Also, it was highly symmetric. This turned out to be the key to getting a proof of embeddedness. Within a week, the way to prove embeddedness was worked out. During that time we used computer graphics as a guide to "verify" certain conjectures about the geometry of the surface. We were able to go back and forth between the equations and the images. The pictures were extremely useful as a guide to the analysis.

The article ends with these words:

> The computer-created model is not restricted to the role of illustrating the end product of mathematical understanding, as the plaster models are. They can be part of the process of doing mathematics.

Software for Mathematical Visualization

My program, 3D-Filmstrip, is a mathematical visualization tool for Macintosh computers that is written in Object Pascal. The principal goal that has guided me in its development has been to make available a wide variety of interesting mathematical visualizations from many areas of mathematics, using an interface that is easily accessible, even to new users and nonprogrammers. One simply chooses an object from a pull-down menu (or describes a "user object" by entering a few algebraic formulas) and then immediately sees a default view of that object. There are several menus for customizing the view in various ways and another for creating animations. For a more complete description, including full documentation in hypertext (HTML) format, visit the 3D-Filmstrip home page on the Web {3dfs}.[4] The home page also has a link to a gallery of visualizations and QuickTime animations produced using the program. Macintosh users can obtain a copy for their personal use by

[4] *References in curly brackets refer to Universal Resource Locators (URLs) that will be found in the references at the end of the article.*

Online Mathematics Visualization Software

{3dfs} 3D-Filmstrip Home Page, http://rsp.math.brandeis.edu/public_html

{Evol} Ken Brakke's Surface Evolver Home Page, http://www.susqu.edu/facstaff/b/brakke/evolver/evolver.html

{Geom} Geomview Home Page, http://www.geom.umn.edu/software/download/geomview.html

{Grp} Grape Home Page, http://www-sfb288.math.tu-berlin.de/~konrad/grape/grape.html

{Oor} Oorange Home Page, http://www-sfb288.math.tu-berlin.de/oorange

{Snap} SnapPea Home Page, http://www.geom.umn.edu/software/download/snappea.html

{Sup} Superficies FTP Site, ftp://topologia.geomet.uv.es/pub/montesin/Superficies_Folder/

{Surf} Surf Home Page, http://www.mathematik.uni-mainz.de/AlgebraischeGeometrie/surf/surf.shtml

{MLb} Mathworks (Matlab) Home Page, http://www.mathworks.com/products/matlab/

{Mpl} Maple Home Page, http://www.maplesoft.on.ca/

{Wri} Wolfram Research (Mathematica) Home Page, http://www.wri.com/

downloading it from a link on the home page or using an ftp client aimed at:

ftp://rsp.math.brandeis.edu/pub/

As a developer of a particular mathematical visualization software package, I think it would be inappropriate in an article such as this for me to review other "competing" packages, so I will restrict myself to listing some of the better-known ones, with a few descriptive remarks about each.

There are a number of commercial software packages with mathematical visualization capabilities. Of these, perhaps the best known are The Three M's: Matlab {MLb}, Maple {Mpl}, and Mathematica {Wri}. The standard licenses for these programs are expensive, but there are also inexpensive student versions available, and many universities have site licenses. These are not primarily mathematical visualization programs. Maple and Mathematica are symbolic manipulation programs, and Matlab is a numerical analysis program, but all three have very good graphic backends for displaying the results of their computations, making them excellent platforms for mathematical visualization. One minor drawback is that each of these programs has its own programming language that a user must learn in order to do anything nontrivial with them. But these are very high-level interpreted languages, and they are considerably easier to learn and to use than the standard compiled languages. An important point in their favor is that there are versions of each for Macintosh, Wintel, and various flavors of UNIX, and software developed for any of these platforms is readily transportable to the others.

Geomview {Geom} is not really a mathematical visualization program in itself, but rather, as its name suggests, a viewing program. To use it, the user must create a 2D or 3D object in a prescribed format, either by writing a program to do so or by using some program (such as Surface Evolver; see below) that has a Geomview interface built in. Grape {Grp} and Oorange {Oor} are similar to GeomView, but they provide more tightly integrated facilities for the creation of the mathematical content to be displayed. Competent C programmers used to working on a UNIX workstation will probably find that using one of these programs provides the easiest approach to creating sophisticated mathematical visualizations on their own. But Mac or Wintel users and those without programming experience using a compiled language will probably be more comfortable working with one of the commercial programs mentioned above.

I should also mention some special-purpose mathematical visualization programs. There are many programs for displaying solutions of ordinary differential equations and analyzing them for various dynamical systems-related properties (closed orbits, limit cycles, etc.). Indeed, the use of such programs is rapidly becoming an essential component in the teaching of this subject. Some of these programs are stand-alone, while others are written to be used in conjunction with Matlab, Maple, or Mathematica.

Creating visualizations of implicitly defined curves and surfaces leads to many interesting problems, both for the mathematician and for the programmer. The need to solve the equations involved numerically is one major difficulty. Constructing reliable and efficient algorithms for finding all the solutions when there are no restrictions on permitted singularities is not a completely solved problem. This is so even for the important special case of interest to algebraic geometers, namely, when the objects in question are defined as the solutions of polynomial equations. Another difficulty is that implicitly defined surfaces do not come equipped with a natural grid, and so special techniques (such as so-called "ray-tracing" methods) must be used to render them. Because of these challenges (and the importance of algebraic geometry), it should come as no surprise that there are a number of programs that specialize in displaying implicit surfaces. On the Macintosh there is Angel Montesinos Amilibia's Superficies program {Sup}. This has the kind of intuitive user interface one expects from a Macintosh program; and in addition to displaying a surface from a user-supplied implicit equation, it will also draw geodesics, asymptotic lines, and curvature lines on the surface.[5] In the UNIX world, there is a program called Surf, written by Stephen Endraß {Surf}. (Endraß has made his source code available under the GNU license.)

[5] *I would like to thank Angel Montesinos Amilibia for permitting me to use some of his algorithms and code for handling implicit curves and surfaces in 3D-Filmstrip.*

Two other notable special-purpose programs are Jeff Weeks's SnapPea {Snap}, for creating and investigating 3-dimensional hyperbolic geometries, and Ken Brakke's Surface Evolver {Evol}, for investigating the evolution of surfaces under various "energy"-minimizing gradient flows.

Why Write Mathematical Visualization Software?

At this point I should confess that one of my goals is to encourage others to become involved in building the Mathematical Exploratorium. The creation of mathematical visualization software and content is a relatively new and growing area, full of opportunities to make significant, original contributions. I work on mathematical visualization mainly because I enjoy the challenge of making abstract mathematical concepts "come alive" by implementing them in software. But more than that, I think of it as a new form of publication. Part of the obligation (and joy) of the academic life is giving some form of permanent expression to the ideas we have thought hard about. That, after all, is what we mean by publication.

Traditionally, mathematicians have satisfied this obligation by writing books and research articles, and these will no doubt continue to be the primary form of mathematical communication. A program is not a substitute for a theorem, and an assistant professor worried about publishing enough papers to qualify for tenure should probably think twice before becoming involved in a time-consuming software project. Nevertheless, as mathematics gets ever more complex, it becomes increasingly important to have good tools to supplement our intuition and for communicating our intuitive ideas to others.

Until recently graphics systems powerful enough to do interesting geometric modeling existed only in a few centers that could afford the expensive combination of hardware and software that such systems required. Moreover, these systems required special proprietary hardware and drivers so that they could not run even slowly on the standard Macintosh and PC-type workstations that have become ubiquitous in academia. But the supercomputer of a decade ago was no more powerful than today's high-end Macs and PCs, and very respectable mathematical visualization programs can now be written for these machines. The time has clearly arrived to make the powerful geometric modeling algorithms that have been developed in recent years more widely available to the whole mathematical community. It is not easy. The problem is not just to translate the code, but also to create programs with good user interfaces that are easy to use for someone other than the programmer. I am hoping that 3D-Filmstrip will serve as an early example of this kind of mathematical visualization program, one that will stimulate others to create ever better software for giving life to our mathematical imaginings.

Acknowledgments

Many people have contributed to my education in mathematical visualization, but I would like to express very special thanks to Hermann Karcher, from whom I learned most of what I know. Hermann is a true pioneer in the field. He was creating amazing visualizations on his Atari a dozen years ago, before the subject became fashionable. He is also one of the leading experts in the modern rebirth of minimal surface theory that resulted from the discovery of the Costa surface. The feature of 3D-Filmstrip that many people find most appealing is the large repertory of minimal surfaces it can display, both classic surfaces and recently discovered ones. All the sophisticated mathematical algorithms required to build and render these surfaces came from Hermann, and I shall always remember with pleasure the long programming sessions as we wrote the code that made the screen light up with one after another of these beautiful objects. Hermann's wife, Traudel, is responsible for perhaps the single most striking visualization in 3D-Filmstrip: it models the remarkable gyrations of a charged particle in the Van Allen Belt as it moves under the influence of the Earth's dipole magnetic field.

References

Articles

[Ab] J. Abouaf, Variations of perfection: The Séquin-Collins sculpture generator, *IEEE Computer Graphics and Applications* **18** (1998).

[CHH] M. Callahan, D. Hoffman, and J. Hoffman, Computer graphics tools for the study of minimal surfaces, *Comm. ACM* **31** (1988), 641–661.

[FSKB] G. Francis, J. Sullivan, R. Kusner, K. Brakke, C. Hartman, and G. Chappell, The minimax sphere eversion, *Visualization and Mathematics* (Berlin-Dahlem, 1995), Springer, 1997, pp. 3–20.

[HMF] A. Hanson, T. Munzner, and G. Francis, Interactive methods for visualizable geometry, *IEEE Computer* **27** (1994), 78–83; `http://graphics.stanford.edu/papers/visgeom/`.

[H] D. Hoffman, Computer-aided discovery of new embedded minimal surfaces, *Mathematical Intelligencer* **9** (1987).

[HWK] D. Hoffman, F. Wei, and H. Karcher, The genus one helicoid and the minimal surfaces that led to its discovery, *Global Analysis in Modern Mathematics* (K. Uhlenbeck, ed.), Publish or Perish Press, Houston, TX, 1993, pp. 119–170.

[MC] N. Max and W. Clifford, Computer animation of the sphere eversion, *ACM J. of Comput. Graphics* **9** (1975), 32–39; films available from International Film Bureau, 332 S. Michigan Ave., Chicago, IL 60604.

[Ph] A. Phillips, Turning a surface inside out, *Scientific American* (May 1966), 112–120.

[PLM] M. Phillips, S. Levy, and T. Munzner, Geomview: An interactive geometry viewer, *Notices* **40** (October 1993).

[PP] U. PINKALL and K. POLTHIER, Computing discrete minimal surfaces and their conjugates, *Experimental Mathematics* **2** (1993).
[Sc] Sphere does elegant gymnastics in new video, *Science* **281** (July 31, 1998), 634.
[Sm] S. SMALE, A classification of immersions of the two-sphere, *Trans. Amer. Math. Soc.* **90** (1958), 281–290.

Books

[B] T. F. BANCHOFF, *Beyond the Third Dimension*, Scientific American Library, New York, 1990.
[DHKW] U. DIERKES, S. HILDEBRANDT, A. KÜSTER, and O. WOHLRAB, *Minimal Surfaces*, Springer-Verlag, Berlin and New York, 1992, two volumes.
[EG] D. EPSTEIN and C. GUNN, *Supplement to Not Knot*, Jones and Bartlett, 1991.
[Fe] C. FERGUSON, *Helaman Ferguson: Mathematics in Stone and Bronze*, Meridian Creative Group, 1994.
[Fi] G. FISCHER, *Mathematical Models, Vols. I and II*, Vieweg, Braunschweig and Wiesbaden, 1986
[Fo] A. FOMENKO, *Mathematical Impressions*, American Mathematical Society, Providence, RI, 1990.
[Fr] G. K. FRANCIS, *A Topological Picturebook*, Springer-Verlag, New York, 1987.
[G] A. GRAY, *Modern Differential Geometry of Curves and Surfaces*, CRC Press, Boca Raton, FL, 1993.
[HC] D. HILBERT and S. COHN-VOSSEN, *Geometry and the Imagination*, Chelsea, New York, 1952. Translation of *Anshauliche Geometrie* (1932).
[L] S. LEVY, *Making Waves*, A. K. Peters, 1995.
[Sc] D. SCHATTSCHNEIDER, *Visions of Symmetry*, Freeman, San Francisco, CA, 1990.
[W] J. WEEKS, *The Shape of Space*, Marcel Dekker, New York, 1985.

About the Cover

On the cover we see a portion of a minimal surface, the whole of which extends to infinity. Numerical data for parametrizing the surface were created using MATLAB routines written by H. Karcher, and then imported into 3D-Filmstrip for rendering.

A minimal surface is a mathematical model of a soap film, and this fact can be used to compute it numerically: if a closed wire (to be dipped into liquid soap) is described by a mathematical curve, then one can compute the spanning minimal surface numerically by simulating a characteristic property of a soap film, namely that it attempts to "pull itself together" (i.e., minimize its area).

Karcher's numerical algorithm uses a different approach. It starts from the so-called Weierstrass representation of a minimal surface, coming from the field of complex analysis. This has the advantage that the input data (two meromorphic functions on the parameter domain) are closely related to the global geometry of the surface. Moreover, it leads to an efficient and numerically stable algorithm for computing the coordinates of all points of the surface.

The particular surface on the cover is known by the name "Karcher's JE Saddle Towers". (JE refers to a particular Jacobi elliptic function that is part of the Weierstrass data.) This surface exhibits a $Z \oplus Z$ group of translational symmetries. Both generators are visually evident—one is a vertical translation and the other is a translation in the direction of the horizontal straight line that cuts the figure in half. (Rotation by 180 degrees about that line is another symmetry of the surface.) The two sides of the surface have been given slightly different reflectivities. The top of the lower horizontal wing appears lighter than the top of the wing above it because these are on different sides of the surface.

—H. Karcher and R. Palais