PLEASURES OF PLICATION

Brian Hayes

Among all the forms of sculpture, the most interesting from a mathematical point of view may well be paper folding. In support of this assertion I would point out that paper folding requires a distinctive leap of the imagination from Flatland into Roundworld, from two dimensions to three. A sculptor who carves stone or models clay begins with a medium that is already three-dimensional, but the paper folder must envision the solid object in the flat and formless sheet. This calls for strong geometric intuition.

A computer program now under development offers assistance in making the leap into a higher dimension. Given a three-dimensional object, the software attempts to generate a "folding net," a two-dimensional diagram that you can color or otherwise decorate, then print on paper, cut out and fold up to create a three-dimensional model. The program, called HyperGami, is the creation of Michael Eisenberg of the University of Colorado at Boulder and Ann Nishioka, a doctoral student at Colorado. So far it is being used mainly to explore paper models of polyhedra, which are objects that hold much mathematical interest of their own.

In addition to serving as a bridge between dimensions, HyperGami spans two worlds in another way as well. Part of its user interface relies on the direct manipulation of objects—the point-and-click paradigm common in most graphics software today. But there is also a programming-language interface, which allows operations on the solids and on their folding nets to be expressed algorithmically. Exploring ways of combining these two styles of computing was one of the original motivations for the project.

Solids and Nets

The best-known form of paper folding is the art of origami, long practiced in Japan, China and Korea, and with many enthusiasts elsewhere as well. In the traditional form of origami, objects must be folded from a single square sheet of paper. Furthermore, folding is the only operation allowed; there is no marking, measuring, cutting or gluing. In classical origami you should be able to unfold the finished object and recover the original sheet of paper.

The name HyperGami pays tribute to the art of origami, but the program in its present form is best suited to somewhat less rigorous folding challenges, which admit tools forbidden to the origami purist—namely scissors and paste. The folding nets produced by the program are arrangements of connected polygons that have to be cut out of the sheet and then glued along corresponding edges to form a three-dimensional model. Although these operations lie outside the customs of origami, they draw on another old tradition. The idea of a folding net goes back at least as far as Albrecht Dürer, the German Renaissance engraver, who incorporated polyhedra into a number of his works. And of course some of the polyhedra are themselves important emblems of classical Greek mathematics.

When the HyperGami software is started up, it displays three main windows and several auxiliary palettes. One of the windows shows an orthographic projection of whatever three-dimensional object is currently under construction, and a second window displays the corresponding flattened folding net. The third window gives access to the programming environment, where procedures that act on the graphic objects are entered and executed. The palettes provide a variety of tools and controls for building and decorating polyhedra. For example, one palette consists of buttons that generate various simple polyhedra as the starting point for a construction project. Another palette specifies colors and pen widths for use in decorating the objects.

You can do quite a lot with the HyperGami system just by issuing direct commands from the palettes and from menus, without doing any programming. Clicking on a button instantly creates both a three-dimensional view and a folding net for any of several simple polyhedra, including the five Platonic solids (the regular tetrahedron, cube, octahedron, dodecahedron and icosahedron). With other buttons the three-dimensional view can be rotated about x, y and z axes. With a click of the mouse the individual polygons that

Brian Hayes is a former editor of American Scientist. *Address: 211 Dacian Avenue, Durham, NC 27701. Internet: bhayes@mercury.interpath.net.*

make up the surface of the polyhedron can be filled with a selected color or pattern. Any such decoration applied to the folding net can then be transferred to the three-dimensional view. A pen tool can be made to draw simultaneously on both the net and the solid, which helps a great deal in figuring out which polygons in the net correspond to which faces of the solid.

Other interactive commands alter the geometry of both the net and the solid. By applying a linear mapping, you can stretch or compress a polyhedron along any axis. For example, a regular icosahedron (which could be inscribed inside a sphere) can be deformed into a prolate and highly nonregular icosahedron (which could be inscribed inside an ellipsoid). The program attempts to draw a folding net for the distorted solid, so that you can build a paper model.

Another built-in operation is vertex truncation: lopping off all the vertices and thereby replacing them with faces. Truncating the 30 vertices of an icosahedron yields a new object with 60 vertices, 12 pentagonal faces and 20 hexagonal faces, all arranged with the symmetry of a soccer ball. This truncated icosahedron is also the structure of buckminsterfullerene, the C_{60} carbon molecule. After truncating a polyhedron, HyperGami again tries to generate a folding net.

I say that the program *tries* to generate a folding net because the success of the attempt is not guaranteed. If you take a folded paper polyhedron and cut along selected edges, you will eventually reach a state in which no two polygons are joined along more than one edge. At this point the paper can be made to lie flat, but the result is not necessarily a valid folding net, because some of the polygons may overlap. For some nonconvex polyhedra no single-sheet folding net exists. (A polyhedron is nonconvex if it has indentations—or more formally if you can find two points on its surface connected by a straight line that passes outside the surface.) Does every convex polyhedron have a nonoverlapping net? The question is apparently an open problem in computational geometry (Croft *et al.* 1991, p. 73). Even when a net is known to exist, however, finding it can be a combinatorial headache. The number of candidate nets grows exponentially with the number of faces in the polyhedron.

HyperGami's net-generating algorithm begins by choosing a single face as the starting point for the net, then adds adjacent faces one by one to exposed edges. Heuristic rules built into the program favor nets that are considered easier to fold; the user can adjust the rules when necessary. Although the algorithm is not guaranteed to work in all cases, Eisenberg and Nishioka report that it has not yet failed to find a net for any convex solid, and it also works for many nonconvex ones.

To actually build the polyhedra, you need a printer, and preferably a color printer. Color ink-jet printers, which have recently become quite inexpensive, work well for this purpose. Eisenberg and Nishioka remark that HyperGami "represents

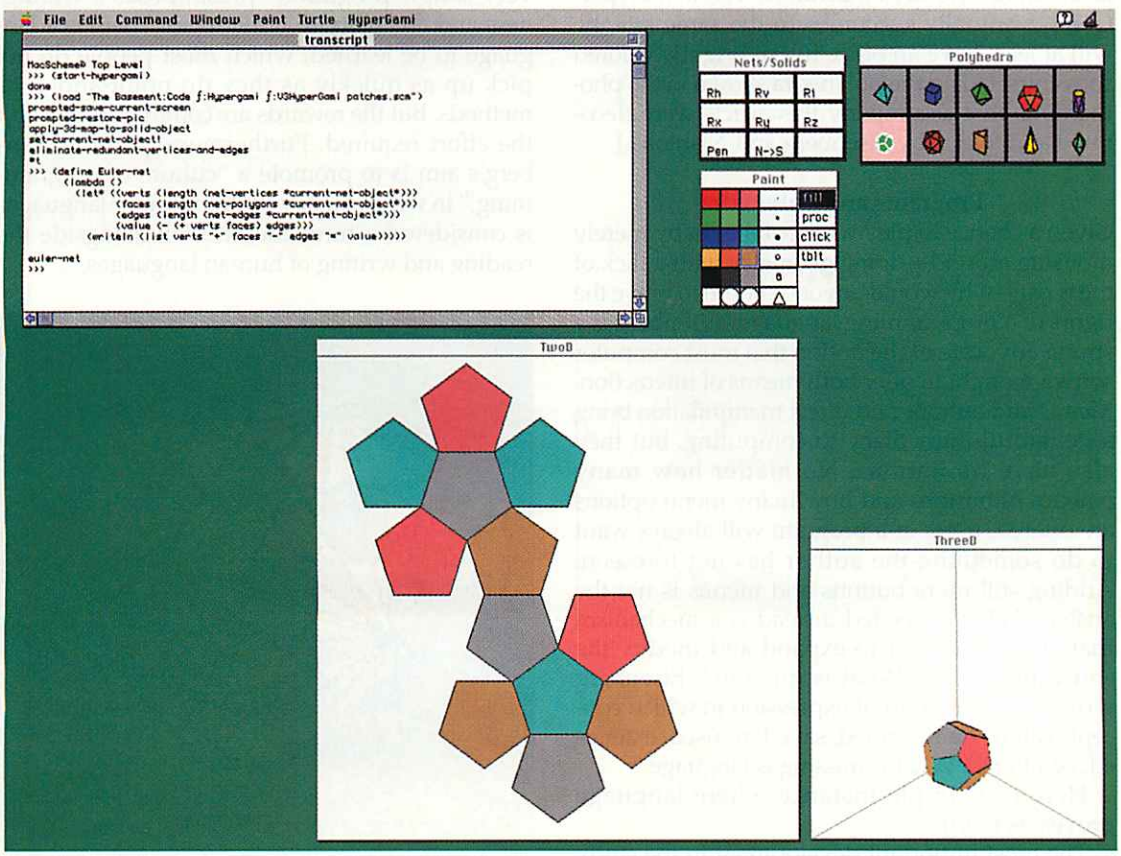

Figure 1. The HyperGami system at work. The polyhedron under construction is a four-colored dodecahedron.

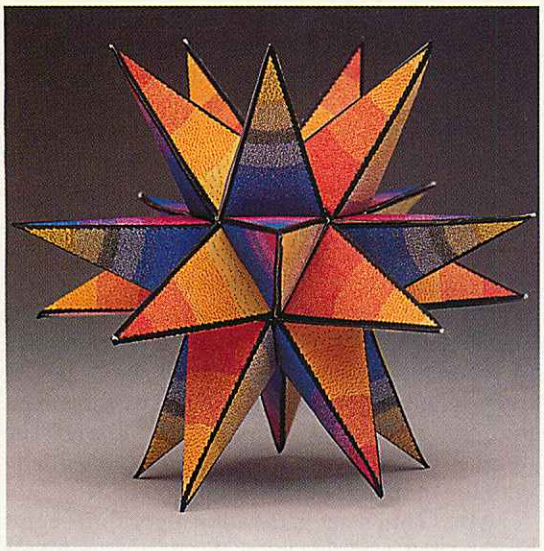

Figure 2. A great stellated dodecahedron. (Photographs by Jim Sink.)

an attempt to expand the role of the color printer—rather than being merely an output device, the printer now becomes something a little closer to a 'mathematical toy shop.'" For Eisenberg the emphasis on making tangible artifacts reflects his disappointment that earlier software projects have proved so ephemeral. As an author of programs for the Tandy Color Computer, he has seen his work become inaccessible in a span of just a few years. The software itself is still perfectly serviceable, but no one has a machine to run it. If HyperGami eventually succumbs to the same fate, he will at least have an office full of brightly colored souvenirs. (All of the polyhedra shown in the photographs that accompany this article were decorated and folded by Eisenberg and Nishioka.)

Programs and Polyhedra

Given a chance to play with polyhedra by merely mousing around—doing geometry with a flick of the wrist—why would anyone choose to brave the rigors of a programming language? Eisenberg is a strong advocate of the notion that most computer software ought to offer both means of interaction. Menus and buttons and direct manipulation bring a delightful immediacy to computing, but they also have limitations. No matter how many palettes of buttons and how many menu options are offered, users of a program will always want to do something the author has not foreseen. Adding still more buttons and menus is not the answer. What is needed instead is a mechanism that allows the user to expand and modify the software at will. "What is missing," Eisenberg writes, "is a medium of expression in which concepts can be built, named, saved, re-used, extended, combined. What is missing is language."

Here is a simple instance where language proves its worth:

The most important development in the study of polyhedra since antiquity was Leonhard Euler's 1750 discovery of an invariant relation between the number of faces, edges and vertices in a polyhedron. The relation is expressed in the equation:

$$faces - edges + vertices = 2.$$

Adrien-Marie Legendre and others later proved that the equation holds true for any polyhedron that is topologically equivalent to a sphere. Exploring this relation by manually counting the faces, edges and vertices of a model is tedious and error-prone. (Maybe that's why it remained unnoticed for more than 2,000 years.) Doing the counting with a mouse on a computer screen is even more awkward. On the other hand, a procedure to do the arithmetic can be written in a line or two.

A command to calculate the Euler characteristic could have been built into the HyperGami software and provided as a menu choice, but there is no end of other functions one might then demand. For example, having just computed *(faces – edges + vertices)* for a three-dimensional polyhedron, the urge is irresistible to apply the same formula to the two-dimensional net. Again, writing a procedure to solve the problem is trivial. (The result is not immediately obvious. There is a one-to-one mapping between the faces of the solid and the polygons of the net, but some edges and vertices are duplicated when the polyhedron is cut open. Is the Euler characteristic the same for all nets? Is the constant in the equation still 2? Some answers are given at the end of this article.)

Solving problems by writing programs—even very simple programs—presupposes a willingness and an ability to write them. There is a language to be learned, which most people do not pick up as quickly as they do point-and-click methods. But the rewards are commensurate with the effort required. Furthermore, part of Eisenberg's aim is to promote a "culture of programming," in which facility with computer languages is considered a fundamental skill alongside the reading and writing of human languages.

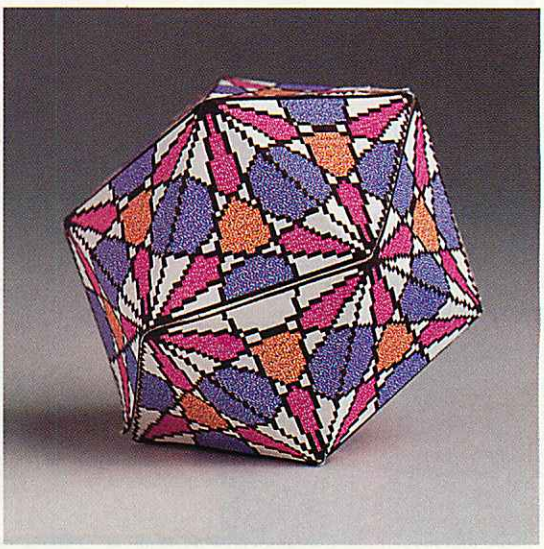

Figure 3. A regular icosahedron.

The programming language underlying HyperGami is Scheme, a dialect of Lisp. Scheme was developed in the late 1970s, mainly at the Massachusetts Institute of Technology; it is a small language, often chosen for introductory computer-science courses, but it is also a highly expressive one. Unlike the rudimentary "macro" or scripting languages built into some programs, Scheme includes all the facilities of a general-purpose programming environment, such as constructs for defining variables, procedures and composite data structures. The entire HyperGami system is written in Scheme, and the full power of the Scheme interpreter is available to the user. (The software relies on features of a specific Scheme implementation, called MacScheme, which runs only on the Apple Macintosh computer, and thus HyperGami too is confined to the Macintosh.)

The version of Scheme presented to the HyperGami user is a superset of the standard language, enriched with various amenities for manipulating polyhedra. Data structures representing faces, edges and vertices are pre-defined, and there are numerous built-in procedures that operate on the current solid or the current folding net.

One major element added to HyperGami's Scheme is a "turtle graphics" system for decorating folding nets. Turtle graphics is usually associated with the Logo programming language (which emerged from the same MIT community as Scheme), but it can be implemented in any language. The original turtle was a hardware device that crawled over a sheet of paper carrying a pen; now a virtual turtle moves around the computer screen under program control, drawing a line as it goes. The advantage of drawing with a turtle rather than drawing by hand is that turtle-graphics routines can be encapsulated in procedures, which then become components of more elaborate procedures, and so on. In this way intricate patterns are assembled from simple commands.

Suppose you want to call attention to some of the symmetries of a polyhedron by drawing a line from the center of each face to the midpoint of each edge. Routines for finding the center of a face and the midpoint of an edge are already present in the HyperGami system, and so you can begin by writing a procedure that places the turtle at the center of a given polygon and draws a line to the midpoint of a specified edge. This procedure is then invoked repeatedly by a higher-level procedure that draws radial lines to all the edges of a single polygon. Finally, yet another procedure applies the radial-line routine to all the polygons in a net. If you design the procedures carefully, they will work equally well for polygons with any number of edges and for polyhedra with any number of faces.

Some of the most interesting HyperGami programs (in my view) are those that explore the connections between the two-dimensional net and the three-dimensional solid. For instance, consider a program for four-coloring a polyhedron—that is, coloring each face with one of four colors in such a way that no two adjacent faces are the same color. (Four colors are guaranteed to be enough for any polyhedron topologically equivalent to a sphere. Therefore you can check that the coloring is possible by calculating Euler's characteristic.) The aim is to paint the colors onto the folding net, but it is not enough that the net itself be properly four-colored. Faces that are distant and unconnected in the net may turn out to be adjacent in the polyhedron, so that they cannot be assigned the same color. The coloring procedure must do its checking for conflicts in the solid, then perform the actual coloring in the net.

Programs and Nonpolyhedra

Writing programs about polyhedra is valuable not only for what it can accomplish but also for what it can teach. When you start manipulating polyhedra algorithmically, you are compelled to think about their geometry in new ways.

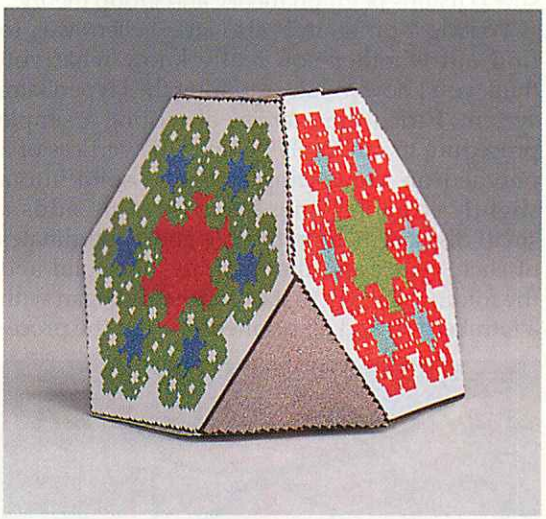

Figure 4. A truncated tetrahedron.

Suppose you tried to describe a regular dodecahedron to a friend who had never seen one. You might say something like, "It has 12 faces, all of them regular pentagons, joined along their edges so that three faces meet at every vertex." This verbal sketch might suffice for a friend with exceptional geometric intuition, but it is not nearly enough to induce a computer to draw a picture of a dodecahedron and generate a folding net. The description leaves too much undefined. (What's a face? What's an edge? What's a vertex?) Furthermore, what it describes is a kind of Platonic ideal of a dodecahedron, floating somewhere in space. Drawing a picture requires choosing a specific dodecahedron, with a definite size and position and orientation.

The description of a dodecahedron that works best for a computer is one you would be unlikely to choose to communicate to a friend. It begins with a listing of the coordinates of the vertices. The list says, in effect: Define a point at $\{x = 0, y = 0, z = 0\}$, define another point at $\{x = 1, y = 0, z = 0\}$, define a third point at $\{x = 1.309, y = 0.951, z = 0\}$, and so on for 20 points. Then a

second list—an adjacency matrix—indicates which of the vertices are connected by edges. Finally a third list specifies which groups of vertices define planar faces.

This description of a dodecahedron is so opaque and inaccessible that the polyhedron itself seems to have fled the scene altogether, leaving behind nothing but a trail of numbers. Where is the geometry? What has become of the object's elegant symmetries? The complaints have merit. Few people can look at the coordinates of a polyhedron and see its form, any more than they can glance at the score of a symphony and hear its harmonies. And yet the disembodied, numerical description of a polyhedron also has much to recommend it. As already noted, it makes easy work of calculating quantities such as the Euler characteristic. More important, if staring at the adjacency matrix leads you away from geometry, it draws you into the equally beautiful (though more abstract) realms of graph theory and group theory.

Writing a program is also an excellent way to find out whether you really know what you think you know. While exploring the HyperGami system, I tried the exercise of writing a simple procedure to randomly "jiggle" the vertices of a polyhedron, converting a regular solid into a slightly irregular one. The procedure adds a small displacement to the x, y and z coordinates of each vertex and then redraws the solid and the folding net. My first experiments began with a tetrahedron and an octahedron, and the procedure seemed to work fine. But then I tested it on a cube, and discovered that what I had created was not a polyhedron at all. Although the folding net was made up entirely of straight-sided quadrilaterals, and although the edges of these polygons could be locally fitted together at each vertex, assembling the entire structure yielded a solid with curved rather than planar faces.

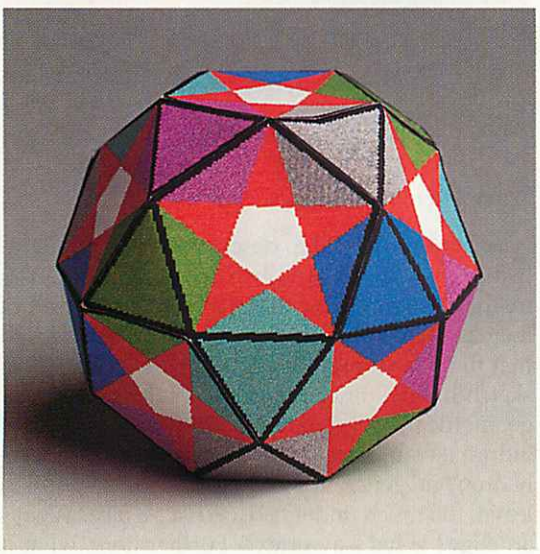

Figure 5. An icosidodecahedron.

I find it interesting to reflect on where I went astray. Thinking of polyhedra in terms of Cartesian coordinates gives primacy to the vertices. In the numeric representation it is the vertices that seem to anchor the solid; the edges and faces are derivative features entirely determined by the position of the vertices. Given this view, it is an easy step (for a lazy mind) to suppose that a vertex can be moved at will, as long as the edges and faces are made to follow along. The supposition is true in two dimensions, which is why the folding net still looks okay after the perturbation. But three-dimensional geometry is more tightly constrained. Vertices can be pushed around freely only in polyhedra whose faces are all triangles. (And even in such "deltahedra" there are a few pathological cases to watch out for.)

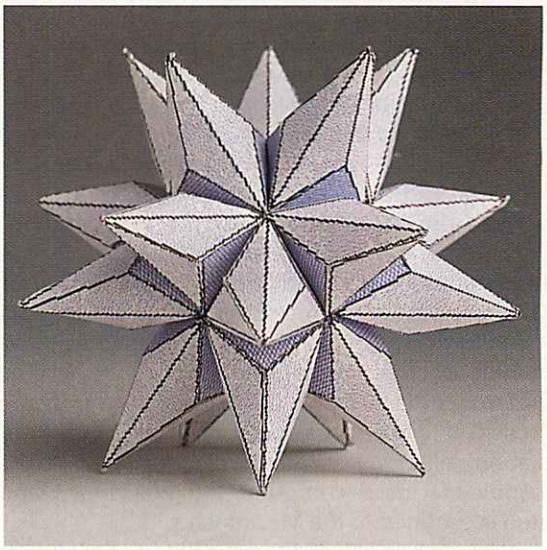

Figure 6. A variant of the great stellated dodecahedron.

A correct program for jiggling vertices would have to check all modified faces for planarity, and triangulate them where necessary. On the other hand, my erroneous program is not a total waste. Some of the curved-surface solids it created are intriguing objects in their own right and may merit further investigation. (They induce wonderful illusions of depth perception.)

Much else in HyperGami also remains to be explored. Here are a few projects I lacked the time or the ability to pursue, which someone else might consider taking on:

The "dual" of a polyhedron is created by replacing every vertex with a face and every face with a vertex. The dual of a cube is an octahedron, and the dual of a dodecahedron is an icosahedron. It should be straightforward to write a HyperGami procedure that would handle these elementary cases. A procedure to find duals of *all* polyhedra would be much more difficult. For some solids the dual is not uniquely defined.

Toroidal polyhedra might present an interesting challenge to the net-generating algorithm. The smallest known toroidal polyhedron (Gardner 1978), which has just seven faces, probably cannot be decomposed into a one-piece folding net without overlaps, but larger examples should be more

tractable. Toroidal polyhedra are also a useful check on routines for calculating Euler's characteristic. And you earn extra credit if your procedure for four-coloring an ordinary polyhedron produces a correct seven-coloring of a torus, or better still a four-coloring (Croft *et al.* 1991, p. 75).

A project with a more practical focus would be to write a routine for adding gluing tabs to a folding net. The tabs must be arranged so that wherever two edges join in the folded solid, there is exactly one tab. Peter Hilton and Jean Pedersen suggest putting a tab on every other edge on the perimeter of the folding net. This rule will never cause an error (where a pair of joined edges have either no tabs or two tabs), but the resulting arrangement is not always optimal for folding. Furthermore, the shapes of the tabs sometimes need careful trimming to avoid overlaps and obstructions.

Notes

HyperGami requires a Macintosh computer with at least 10 megabytes of memory. A color monitor and a color printer are obviously helpful. The software is very much a work-in-progress, not a finished product. Some functions are not yet implemented; the documentation is incomplete; many bugs have yet to be eliminated. Readers who would like to experiment with the software in spite of these caveats should send an inquiry to Michael Eisenberg, Department of Computer Science and Institute of Cognitive Science, Campus Box 430, University of Colorado, Boulder, Colorado 80309-0430. Internet address: duck@sigi.cs.colorado.edu

Here is one possible analysis of the Euler characteristic of a two-dimensional net: Think about cutting open a polyhedron along the edges to create a net. The first slit, extending between two vertices (but not including the vertices), increases the number of edges by 1 and leaves the number of faces and vertices unchanged. Thereafter, extending the slit through a vertex and up to (but not including) the next vertex always adds 1 to the count of both edges and vertices. Hence the overall effect of n cuts is to add n edges and $n - 1$ vertices, so that the Euler characteristic for the net is: $faces - edges + vertices = 1$.

Bibliography

Croft, Hallard T., Kenneth J. Falconer and Richard K. Guy. 1991. *Unsolved Problems in Geometry*. New York: Springer-Verlag.

Eisenberg, Michael. 1991. Programmable applications: Interpreter meets interface. Massachusetts Institute of Technology Laboratory for Artificial Intelligence. A.I. Memo No. 1325.

Eisenberg, Michael, and Ann Nishioka. 1994. HyperGami: A computational system for creating decorated paper constructions. Presented at the Origami Science Meeting, Otsu, Japan.

Eisenberg, Michael, and Ann Nishioka. 1996. Creating polyhedral models by computer. To appear in *Journal of Computers in Mathematics and Science Teaching*.

Gardner, Martin. 1978. Mathematical games. *Scientific American*, November, pp. 22–32.

Hilton, Peter, and Jean Pedersen. 1994. *Build Your Own Polyhedra*. Reading, Mass.: Addison-Wesley.

Senechal, Marjorie, and George Fleck, eds. 1988. *Shaping Space: A Polyhedral Approach*. Boston: Birkhäuser.