Creating Polyhedral Models by Computer

Michael Eisenberg and Ann Nishioka
Department of Computer Science and Institute of Cognitive Science
University of Colorado, Boulder
Boulder, CO USA 80309-0430

Abstract

This paper describes a computer application named *HyperGami* that permits users to design, explore, decorate, and study a rich variety of paper polyhedral models. In structure, HyperGami is a "programmable design environment", including both a direct manipulation interface as well as a domain-enriched programming environment based on the Scheme language; the application is thus designed to be accessible to students of geometry while providing challenging projects for long-term or expert users (such as professional mathematicians and designers). In the course of this paper, we describe the HyperGami interface and language; illustrate the construction of "customized polyhedra" of various sorts; discuss the results of our initial experiences using the system in working with middle-school students; and argue for the utility of embedding programming languages in educational design environments such as this one.

1. Introduction

Over the centuries, human beings have been fascinated by polyhedral models. Perhaps the earliest known examples of such models are architectural: the Egyptian pyramids, dating from about 2500 B.C. Less awe-inspiring (but nonetheless intriguing) examples include a small Etruscan dodecahedron made of soapstone (ca. 500 B.C.), and regular icosahedral objects—dating perhaps from the Roman period—that have been found in Egypt and have since been displayed at the British Museum (Malkevitch, 1988; Coxeter, 1973, p. 13). Renaissance artists were attracted to polyhedra: for instance, da Vinci depicted the regular solids (among others) in his illustrations for Fra Luca Pacioli's book *Divina Proportione*, while a famous portrait of Pacioli—rendered by Jacopo de Barbari—shows the mathematician with two solid polyhedral models, one on either side (Marinoni, 1974; Coxeter, 1988; Senechal, 1988).

Polyhedra needn't always be modelled in stone, like the pyramids: the more homely material of paper will also serve quite nicely. The invention of paper polyhedral models—generated from two-dimensional "folding nets"—is credited to the sixteenth-century artist Albrecht Dürer (Malkevitch, 1988); and since Dürer's time, paper has remained one of the most important media for the creation of mathematical solids. This report describes a computer program—named *HyperGami*—that reflects (and, we hope, fruitfully extends)

this venerable tradition of polyhedral modelling. In structure, HyperGami is a programmable design environment (Eisenberg and Fischer, 1994): a computational system that integrates direct manipulation interface tools and application-specific programming elements. The purpose of the HyperGami system is to provide an environment that is both accessible to children (of about middle school age) and sufficiently powerful and extensible to provide challenging and creative activities for professional artists and mathematicians.

The remainder of this paper presents a description of our current HyperGami system and how it may be used to create polyhedral models. The following section provides some background and pedagogical motivation for the program; in the third section, we explain the overall design of HyperGami and introduce both the language and interface elements of the system. Section 4 details a variety of methods for creating new "customized" polyhedra by extending simpler examples; while Section 5 presents two complete scenarios of polyhedron construction. Section 6 describes some of our observations from initial pilot tests of HyperGami with local elementary and middle school students; and finally, Section 7 focuses on future directions for research and development of HyperGami.

2. Motivation: Polyhedral Modelling in Mathematics Education

2.1 Polyhedral Models as "Mathematical Manipulatives"

Mathematics educators have long made use of polyhedral models, both as examples of specific geometric shapes and as tangible representatives of finite symmetry groups. One especially lovely illustration of this practice—a set of cardboard "classroom demonstration" models, now over two centuries old—is still on display in a cabinet at the University of Göttingen in Germany (Mühlhausen, 1993). In our own century, a variety of books have been published that include instructions for the creation of paper polyhedra: some examples are Holden (1971), Wenninger (1971), Schattschneider and Walker (1977), Jenkins and Wild (1990), and Hilton and Pedersen (1994), all of which include instructions and tips for model-builders (typically concentrating on the "standard" constructions—i.e., the regular and semiregular solids).

Though the educational rationale behind the construction of polyhedral models is rarely provided explicitly[1], it reflects the widespread view among mathematicians of the

[1] For an eloquent exception, see the final section in the essay by Senechal (1990).

importance of mathematical imagery (Hadamard, 1949; Fomenko, 1994). Hilbert and Cohn-Vossen's well-known text on geometry opens with the first author's observation:

> "In mathematics, as in any scientific research, we find two tendencies present. On the one hand, the tendency toward abstraction seeks to crystallize the *logical* relations inherent in the maze of material that is being studied.... On the other hand, the tendency toward *intuitive understanding* fosters a more immediate grasp of the objects one studies, a live *rapport* with them, so to speak, which stresses the concrete meaning of their relations.... In this book, it is our purpose to give a presentation of geometry... in its visual, intuitive aspects." (Hilbert and Cohn-Vossen (1952), p. iii; emphasis in the original.)

Polyhedral models are seen as tangible aids to this imagery process—as a means of "learning through the hands". The notion is that students can best understand three-dimensional geometry by actually holding and manipulating shapes: there is a level of "haptic perception" in this process that cannot be imitated even by the use of sophisticated three-dimensional computer graphics. This intuition is reflected, in other areas of mathematics education, by the widespread use of manipulatives (such as "number rods"); and it is given voice by the mathematician/physicist Stanislaw Ulam:

> "It is one thing to know about physics abstractly, and quite another to have a practical encounter with problems directly connected with experimental data... I found out that the main ability to have was a visual, and also an almost tactile, way to imagine the physical situations, rather than a merely logical picture of the problems..." (Quoted in Cooper (1989), p. 15.)

There is, moreover, some direct experimental support for the pedagogical value of constructing polyhedral models to be found in the well-known studies of Piaget and Inhelder (1948) on the development of geometric concepts in children. These researchers performed a series of experiments in which children of ages 5-12 were asked to look at paper models of various simple shapes (including a cube and pyramid) and to draw what the shapes would look like in their unfolded form; as reported,

> "The findings of this experiment present two main aspects which are equally important for the study of geometrical thought. The first of these aspects concerns the difficulties which the children encounter in trying to imagine the development of

> lateral surfaces. It would seem that below the age of 8 or 9 they perform in a controlled way the rotations which are such a constant feature in the spontaneous drawings they produce before the age of 6 or 7! The second aspect shows that imagining the rotation and development of surfaces depends largely on the actual process of unfolding solids, and the motor skills involved in such actions. *In particular, the child who is familiar with folding and unfolding paper shapes through his work at school is two or three years in advance of children who lack this experience*." (pp. 275-6; emphasis added)

2.2 Polyhedral Models as a Creative Medium

In the preceding section, we have argued that—from the purely abstract standpoint of building mathematical skills—there are sound reasons for introducing students to polyhedral models. But perhaps the most important reason for providing this introduction is that these objects possess a compelling and stately beauty, as suggested by Pedersen (1988):

> "I have these models in my office and students come in and beg to know how to make them. They never ask, 'What are they good for?' They know! And we know too." (p. 143)

Polyhedral models have a motivational value that is unique in mathematics education. They can act not only as tangible mathematical diagrams, but as objects almost approaching the status of artistic creations. Unlike other mathematical manipulatives, polyhedral models are typically displayed, museum-fashion (as implied by Pedersen's quote above); they can be decorated in ways that highlight (or de-emphasize) certain symmetry operations; they can be the products of collaborative construction; they can even be given away as presents (cf. Banchoff (1990), p. 14).

In this light, we begin to see that current materials for the creation of polyhedral models—while often quite beautiful or inventive—nevertheless leave room for improvement. "Modelling kits" typically provide either pre-built models or put-together pieces from which a relatively limited number of models may be composed. Likewise, books on polyhedral modelling include folding nets only for "standard" shapes: there is little guidance for creating shapes that happen not to be included among the set already provided. Moreover, to decorate a shape, the student must paint the net (or the completed polyhedron) by hand;

and such decorations cannot easily be reproduced, annotated, edited, traded, or stored. Conceivably, a student might employ a scanner to read a folding net into computer form (in which case the net could be decorated using commercial graphics programs); but even this process could only be employed with already-provided (i.e., "standard") nets, the decorative strategies employed could not easily make use of the inherent geometric structure of the nets, and there would be little or no support for predicting the appearance of the eventual folded solid from its net representation.

It is with these points in mind that HyperGami has been designed: one may think of this system as a kind of "CAD tool for polyhedral modelling." HyperGami permits users to create for themselves an endless variety of "custom polyhedra" (in addition, of course, to the "standard" shapes); these models are generally shown in both their three- and two-dimensional (folding net) forms, and the latter form may be decorated by using a collection of direct manipulation or programming techniques. We now turn to a more detailed description of the HyperGami system itself.

3. HyperGami: a Brief Tour

HyperGami is written by the authors in the MacScheme[2] dialect of Lisp, and runs on all color Apple Macintosh machines with at least 10 megabytes of memory. When the system is brought up, it presents the user with a number of windows, as shown in Figure 1. Briefly, the **TwoD** and **ThreeD** windows are used to depict two-dimensional and (for some polyhedra) three-dimensional views of a constructible shape; the **Polyhedra** window provides an initial "starting sampler" of solids; the **Nets/Solids** window provides a variety of controls (to be described, as needed, later in this paper); and the **Paint** window provides standard direct manipulation tools for (e.g.) choosing color and pen-width. Finally, the **transcript** window is an interface to a "domain-enriched" interpreter for the Scheme programming language (Eisenberg, Hartheimer, and Clinger, 1990); as we will discuss in more detail, HyperGami's version of Scheme has been enhanced with a set of special-purpose "embedded sub-languages" for creating, editing, and decorating models under construction.

---

[2]Lightship Software, Palo Alto, CA.

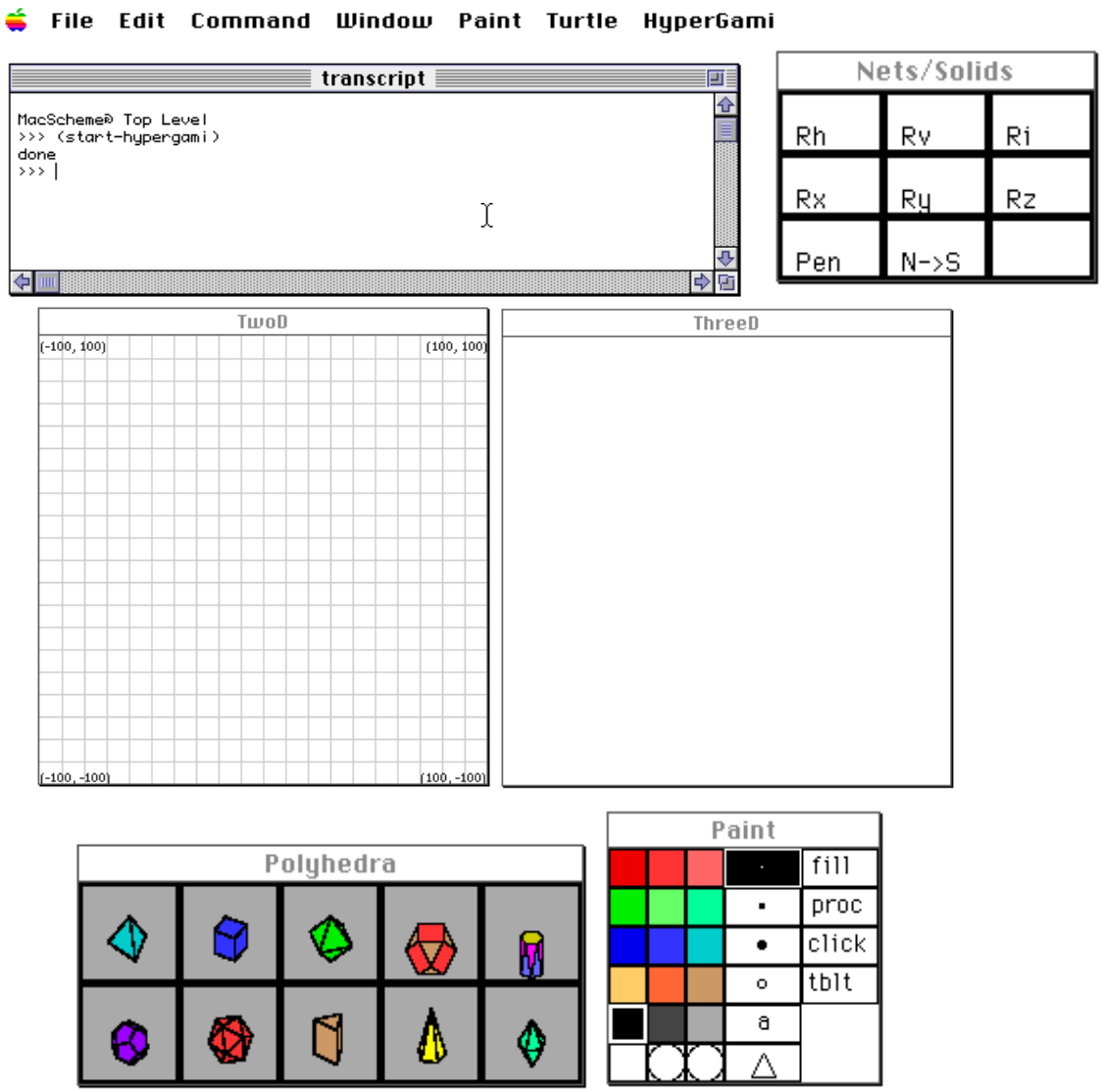

Figure 1. The initial HyperGami interface. The **TwoD** and **ThreeD** windows (center row) are used to display polyhedral nets and solids in construction; the **Polyhedra** window (bottom left) allows the user to select initial shapes; the **Nets/Solids** (top right) window provides controls for viewing solids and for "linking" net and solid representations; the **Paint** window provides standard direct manipulation paint tools. Finally, the **transcript** window (at top) is an interpreter for an "application-enriched" version of the Scheme programming language. The three menu titles at right are specific to HyperGami; the first four text menu titles are standard menus in the MacScheme system.

The icons in the **Polyhedra** window of Figure 1 represent an initial palette of "simple" polyhedra from which to choose: the five regular, or Platonic, solids[3]; the cuboctahedron

[3]These are the tetrahedron, cube, octahedron, dodecahedron, and icosahedron, all visible toward the left of the **Polyhedra** window in Figure 1.

(a semi-regular, or Archimedean, solid); prisms; antiprisms; pyramids; and bipyramids. (The last four of these selections employ regular polygons as bases—for instance, we can create a pyramid of any height based on a regular pentagon.) As will be seen shortly, the user can create a far wider variety of models than suggested by this initial set; but just to provide a sense of the HyperGami interface, we can pursue a short scenario illustrating how the system may be used to create a simple model of a Platonic solid. If the user selects (e.g.) the octahedron icon from the **Polyhedra** window, she sees two views of the octahedron as shown in Figure 2: a two-dimensional "folding net" view of the solid, and a

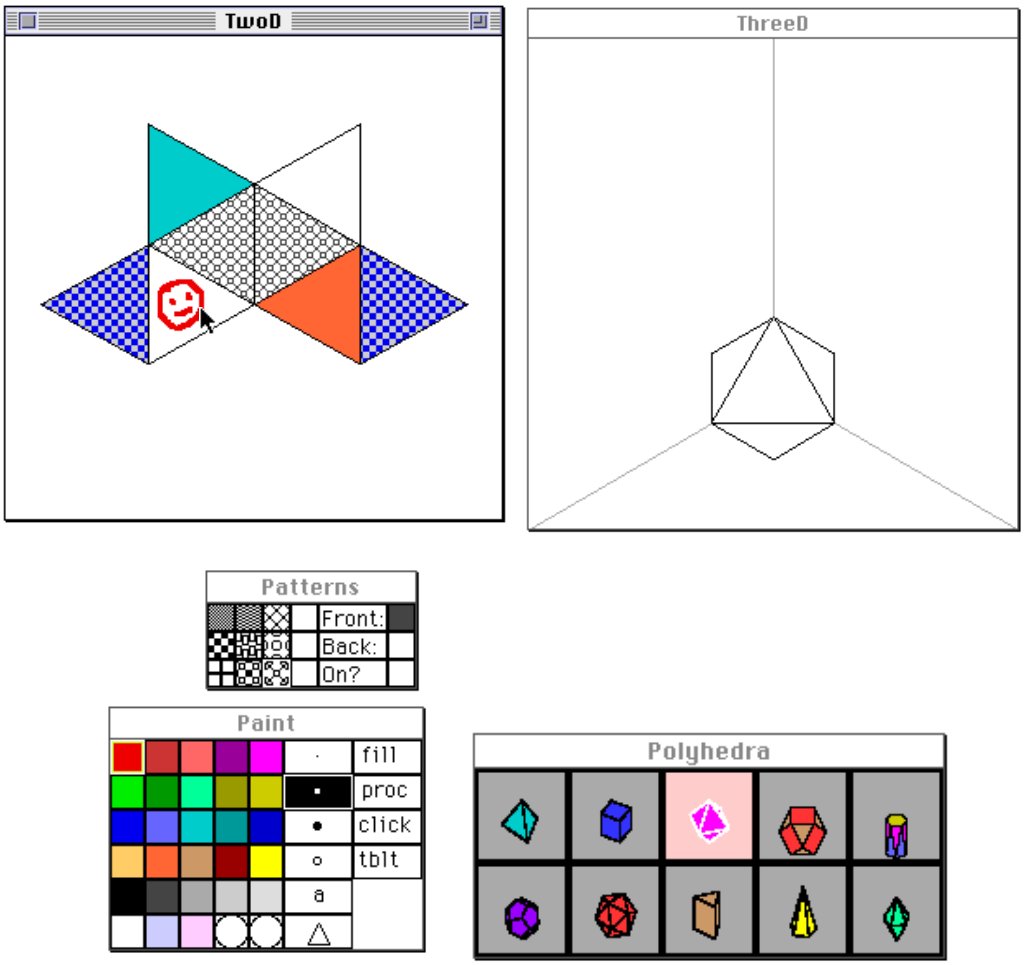

Figure 2. Here, the user has selected the octahedron icon from the **Polyhedra** palette; the system responds by presenting a two-dimensional net for the octahedron in the **TwoD** window and a view of the solid shape in the **ThreeD** window. The user can now employ the **Paint** and **Patterns** windows to decorate the net (in this example, the former window has been expanded and the latter has been opened via menu choices); she has also elected to decorate one triangular face "by hand", using the mouse. If the user is happy with the decorated net, she may now print it out and fold it into an octahedron.

three-dimensional view (in the **ThreeD** window). Now, by employing various tools from the **Paint** window, the user may decorate the folding net (as shown in Figure 2); she may fill the triangular faces of the net with solid colors or multicolored patterns (chosen from a menu-accessible "Patterns" window). Once the user has decorated the folding net to her liking, she may output the contents of the **TwoD** window to a color printer and fold the net into an attractive multicolored model.

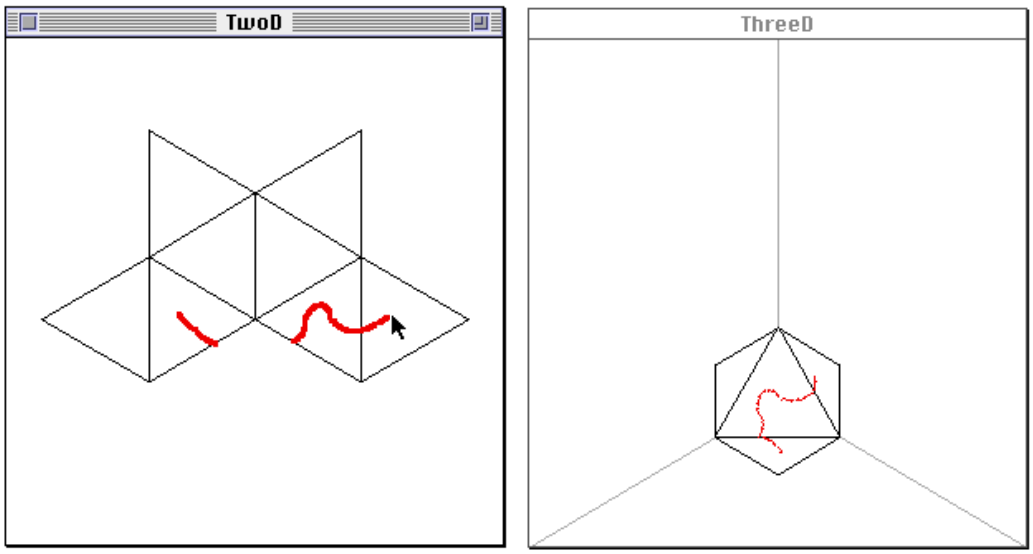

Figure 3. "Linked" two- and three-dimensional representations in HyperGami. By selecting the "Pen" mode choice in the **Nets/Solids** window, we can activate a mode in which, by drawing with the mouse on the net (in the **TwoD** window) we can view the corresponding line as it appears on the solid (in the **ThreeD** window). Note that in this example, the user has drawn one continuous line by dragging the mouse in the **TwoD** window; when the mouse moves beyond the net boundaries no line is drawn, but the line returns when the mouse passes over the net once more.

Even this relatively straightforward scenario poses some mathematical challenges to the HyperGami user. The task of predicting how a two-dimensional pattern will fold into a three-dimensional solid is a difficult exercise in spatial visualization.[4] HyperGami thus includes several tools that facilitate the cognitive process of mapping a two-dimensional folding net to its eventual three-dimensional form; one such tool is represented by the Pen button in the **Nets/Solids** window. By selecting this button, the user can employ a mode (shown in Figure 3) in which the pen acts as a "real-time link" between the **TwoD** and **ThreeD** windows: that is, when the pen is used to draw on the folding net in the **TwoD** window, the user simultaneously sees the corresponding line drawn on the solid shape in

[4]Indeed, paper-folding tasks have been studied as psychological measures of mental imagery (cf. the earlier citation of Piaget and Inhelder, 1948; see also Shepard and Feng, 1972).

the **ThreeD** window. This technique allows the user to get a sense of how polygons in the net correspond to the faces of the folded solid; and it gives rise to some interesting interactive challenges (such as trying to draw a closed curve on the solid by drawing lines on the two-dimensional net).

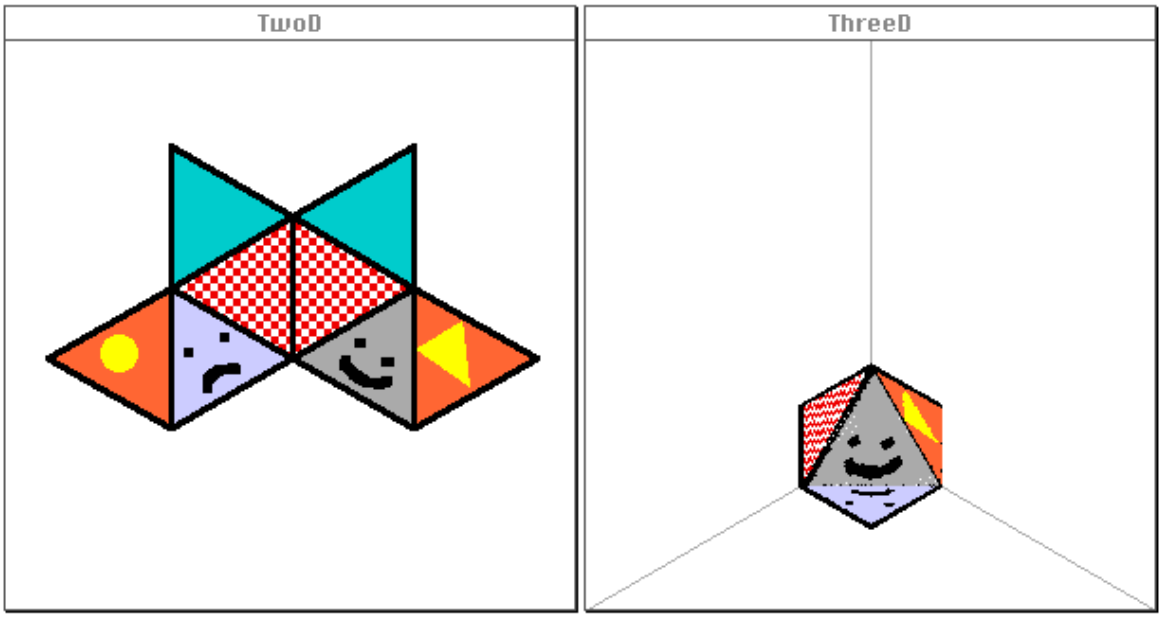

Figure 4. Here, the user has decorated the octahedral net with a variety of designs as shown at left; by using the "N->S" button in the **Nets/Solids** window, she instructs the program to "paint" the net decoration onto the solid view at right.

Another useful technique allows the HyperGami user to transfer any arbitrary decoration from a two-dimensional net to its corresponding solid view. By selecting the "N->S" (for "net-to-solid") button in the **Nets/Solids** window, the user instructs HyperGami to "paint" the **ThreeD** solid view with the decorations created in the **TwoD** window. As an example, then, suppose that the user has decorated her octahedral net as shown in Figure 4; if she now wishes to see how the eventual octahedron will look (without actually going to the trouble of printing, cutting, and folding the solid), she can use the "N->S" button to get a preview of the octahedron's appearance. Though this painting process is somewhat slow and not perfectly accurate (it proceeds on a pixel-by-pixel basis, transferring "net pixels" to "solid view pixels"), it has nonetheless proved to be of tremendous utility in providing an informative picture of the model under construction.

The "enriched" Scheme language supplied with the system includes a number of what might be called "embedded sub-languages"—that is, user-accessible packages of

procedures and objects for working within some particular task area.[5] Broadly speaking, these sub-languages fall into two categories: those for creating decorative patterns for the nets, and those for creating, manipulating, and editing the geometric structures both of two-dimensional (net) and three-dimensional (solid) objects. A full treatment of these elements is well beyond the scope of this paper, but we present a brief example of the language in use just to indicate how custom decorations may be approached within the system; the fourth section below will present examples of the net-and-solid-manipulation elements of the language. (Further detail about the decorative elements of the system may be found in Eisenberg (1991).)

Suppose, then, that our user wishes to create turtle-graphics patterns with which to decorate her octahedral model. As a first step, she creates two simple graphics procedures (using HyperGami's turtle-graphics package) to create a pattern of rotated octagons:

```
(define (octagon side)
  (repeat 8 (forward side) (right 45)))

(define (octagon-pattern side)
  (repeat 8 (octagon side) (right 45)))
```

These procedures are based on standard Logo turtle primitives (Abelson and diSessa, 1980); the `octagon-pattern` procedure, when called on a numeric argument, instructs the programmable pen (or "turtle") to draw eight octagons of the given side-length, with a rotation of 45 degrees between successive octagons.

Our user might now employ her newly-created procedures to decorate regions within the net. A straightforward (but less than optimal) method for doing this would be to "drag" the turtle about the net, positioning it within each triangular region[6]; once positioned, the user evaluates a Scheme expression to create the desired pattern with a reasonable size:

```
(octagon-pattern 0.15)
```

---

[5]Cf. the discussion of "metalinguistic abstraction" in Abelson and Sussman with Sussman (1985). That text also employs Scheme as its illustrative language; and it may be worth a brief aside here about the potentially controversial use of Scheme as the language underlying HyperGami (as opposed to some other choice, such as Basic or Logo). Our belief is that Scheme is in many ways an excellent foundation for an end-user language, combining as it does a powerful functional programming style (including higher-order procedures) and a simple—if, to some, less than aesthetically appealing—syntax. A thorough discussion of the issues involved would take us too far afield; for more discussion on this question, see Eisenberg (1991).

[6]HyperGami includes a "turtle-drag" mode for the mouse which allows the user to move the turtle to any desired position in the **TwoD** window.

The size of the pattern (here, 0.15) may be determined by experimentation: the user could, for instance employ the procedure a few times with different sizes (and with the turtle in "pen-up", or non-drawing, mode) until she is satisfied with the result. Figure 5a, at left, shows a snapshot of this technique in operation.

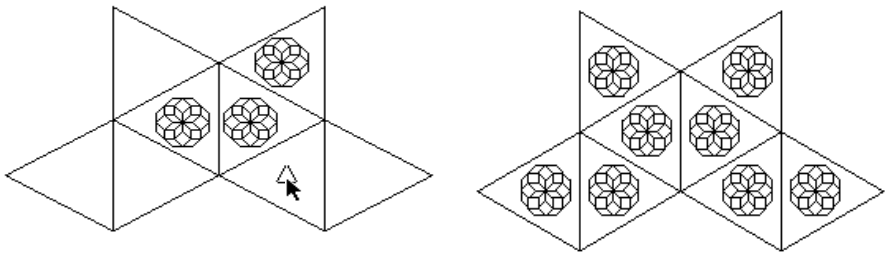

Figure 5a (left). Using turtle graphics patterns to decorate the octahedral net. The user drags the turtle to the center of each triangle and then invokes a newly-written turtle-graphics procedure.
Figure 5b (right). Here, instead of dragging the turtle by hand, we have employed a more advanced technique, writing a procedure which will move the turtle to the center of each triangle and draw the appropriate pattern.

As described here, the strategy for decorating the octahedron is perfectly reasonable[7]; but just to suggest a more advanced use of the HyperGami language (and to suggest the range of tools available within the language), we might look once more at Figure 5a and note that positioning the turtle at the center of each triangle "by eye" is a difficult task. One improvement on this idea, then, would be to write a procedure which—given a center point of a triangular region—will position the turtle at that point and create the desired pattern:

```
(define (draw-octagon-pattern-at-point point)
  (move-turtle-to-point point)
  (octagon-pattern 0.15))
```

Here, the `move-turtle-to-point` primitive positions the turtle at the given starting point; then the already-created `octagon-pattern` procedure draws the desired pattern. But in order to use this new procedure, we would need (for each triangle in the net) the center point of that triangular face. One of HyperGami's built-in net-manipulation primitives (named `current-net-polygon-midpoints`) in fact provides a list of

[7]Indeed, this past fall we collaborated with one of our middle-school students in a project that she initiated—namely, decorating a dodecahedron with turtle-drawn polygons. The project made use of essentially the same method described in the text.

center points of each polygon.[8] Thus, we could create our decoration by the following means (employing the standard Scheme `for-each` procedure):

```
(for-each draw-octagon-pattern-at-point
          (current-net-polygon-midpoints))
```

Stated in prose: for each of the points in the list produced by `current-net-polygon-midpoints`, we call the `draw-octagon-pattern-at-point` procedure on the given point. The result is shown in Figure 5b.

The example as presented thus far barely begins to suggest the rich collection of language elements that may be employed within HyperGami for decorative purposes. The program includes procedures for experimenting with dynamical systems (e.g., for creating fractal patterns via iterated function systems (Barnsley, 1988)); for creating new color objects via hue/saturation/value or RGB representations; for designing custom gradients and patterns (much like those found in commercial paint programs); and for "reading in" pictures, in Macintosh PICT form, into polygonal regions (thus permitting the user to create designs in other applications which may then be employed to decorate a net).[9] The expressive range of HyperGami's graphical language elements is suggested by some of the examples shown later in this paper; but before turning to other matters, it is worth pausing to note that the activity of decorating polyhedral models may be motivated by more than "mere" artistic expression. Both Coxeter (1973) and Wenninger (1971), for instance, suggest the use of coloring patterns as a means of illustrating particular mathematical principles for polyhedral models; to take a simple example, we might decorate the faces of our octahedron with four solid colors—two faces allotted to each color—so that each face is directly opposite the other face with the same color.

---

[8]The "center point" in this case is defined as the center of mass of a set of unit masses placed, one apiece, at the vertices of the polygon.

[9]These packages were largely developed for use in an earlier graphics application, "SchemePaint", upon which HyperGami has built. For more detail about SchemePaint and its associated language elements, see Eisenberg (1991).

4. Operations for the Creation of Customized Polyhedra

4.1 The Basic Polyhedra; Generating Nets from New Solids

As noted in the previous section, the initial HyperGami interface presents the user with some "standard" initial choices for polyhedra: the five Platonic solids, the cuboctahedron, pyramids, bipyramids, prisms, and antiprisms. When the user selects an icon for one of these choices from the **Polyhedra** window, the system creates a two-dimensional net for the selected shape in the **TwoD** window and a three-dimensional view of the solid in the **ThreeD** window. In the case of prisms, pyramids, and antiprisms, the system first presents the user with a dialog box asking for certain relevant parameters (e.g., the number of vertices in the regular polygon at the base of the pyramid, the height of the pyramid, and so forth) before constructing the two views.

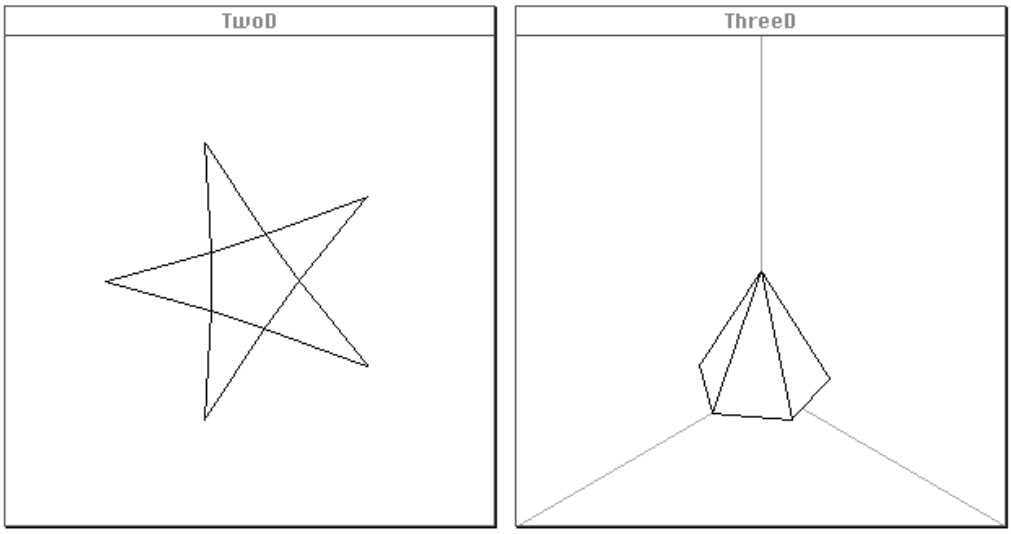

Figure 6. The net and solid views for a pentagonal pyramid of height 2 (the circumradius of the pentagonal base is 1). Here, HyperGami has generated the net "on the fly", after first creating the desired solid.

In the case of some models—the Platonic solids, for example—HyperGami maintains "prestored" nets and solids for rapid access. Thus, when the user selects the octahedron icon (as in the example of the previous section), the program simply displays the (two-dimensional) polygons of the net and the (three-dimensional) polygons of the solid. More interestingly, however, HyperGami is in many cases able to generate nets from their

corresponding solids. This is in fact what happens when the user selects one of the parametrizable solid icons from the **Polyhedra** window: for instance, when the user asks to create a pentagonal-based pyramid of height 2, the system generates, on the fly, the net shown in Figure 6.

The algorithm by which the system generates two-dimensional nets from corresponding solids is simple in outline. Basically, the algorithm is a parametrized search routine which first locates a face of the three-dimensional solid; places that face, "flat", on the Euclidean plane; and then attempts to place similarly "flattened" neighboring polygons on the Euclidean plane corresponding to still-unaccounted-for faces of the solid. The path of the algorithm is suggested by the successive "snapshots" shown in Figure 7a; here, a net is generated for a square antiprism.

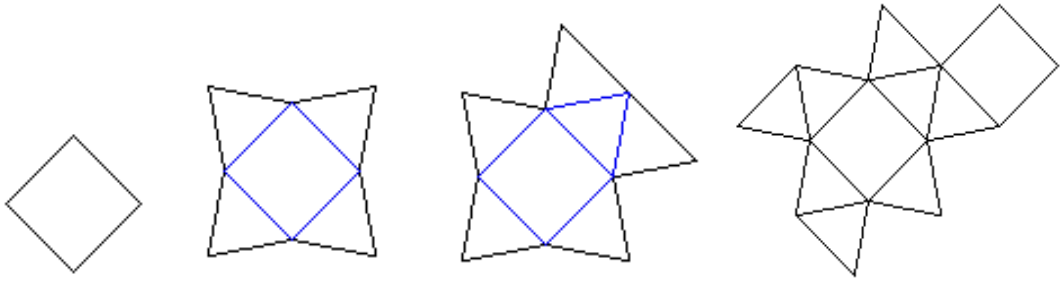

Figure 7a. Successive stages (from left to right) in the generation of the net for a square antiprism. The initial square (left) is surrounded by four triangles (second from left), one of which is in turn bordered by two additional triangles (third from left). The final net is shown at right.

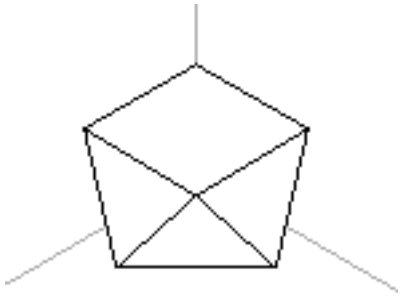

Figure 7b. The square antiprism whose net is generated above.

It is perhaps worth mentioning some of the strengths and limitations of HyperGami's net-generation algorithm.[10] On the positive side, it has a high empirical rate of success; thus far, it has not failed to generate a net for any convex polyhedron and has in fact generated nets for numerous complex nonconvex solids. On the other hand, if the algorithm fails to find a workable net for a nonconvex solid, this is not a proof that no such net exists (though there are in fact solids for which it is impossible to construct a single connected folding net). The mathematical problem of determining whether a given arbitrary solid can be "unfolded" to form a single two-dimensional net is a complicated one (cf. Croft, Falconer, and Guy, 1991, pp. 73-75); HyperGami's current technique is in the nature of a stopgap solution—but a highly useful stopgap solution, as the examples that follow demonstrate.

It should also be mentioned that HyperGami allows the user to create, directly via language expressions, nets that are less constrained than those generated "automatically" from solids. For instance, the user can create new nets that correspond only to portions of eventual constructed solids; these nets may be especially useful in the creation of complex forms, but currently the program cannot link such "partial" nets to solid objects. In practice, this means that complex nets may be used for the creation of solids, but the user cannot view the eventual solids in three-dimensional form on the screen.

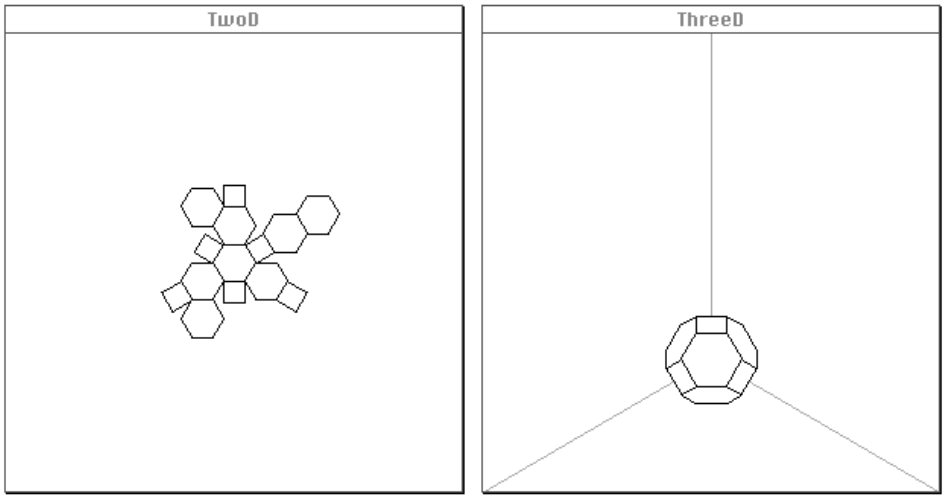

Figure 8. A truncated octahedron (at right) and its corresponding folding net.

[10] We are currently preparing an internal report including a more thorough technical description of the algorithm and its variants (Eisenberg 1996).

4.2 Operations on Solids:Vertex Truncation, Capping of Faces, and Application of Linear Maps

Once a starting solid object has been created in HyperGami, the user is supplied with several operations for creating variants of that solid. The first such operation, vertex truncation, permits the user to "chop off" a vertex of a solid (as if by cutting through the solid near the vertex with a whittling knife). Figure 8 depicts an example: here, an octahedron is truncated at every vertex to create a "truncated octahedron"[11]. The system allows the user to perform simple truncations via menu (in this case, the user specifies the amount by which each neighboring edge will be "sliced" near each vertex in the solid[12]); more selective vertex truncation is accomplished by using HyperGami's language elements. In the latter case, for example, we might find the solid vertices whose z-coordinate is positive and truncate the solid at those vertices solely.

Another useful operation to perform on solids is to apply various types of affine maps to them to produce new rotated, stretched, or "sheared" polyhedra. For instance, suppose we have generated the standard dodecahedron from the initial set of possibilities. We now create a three-dimensional map that "stretches" shapes in the x-dimension only:

```
(define stretch-x-map
  (make-3d-map (x y z)
    (* 1.5 x)
    y
    z))
```

The syntax of HyperGami's `make-3d-map` primitive permits the user to create arbitrary linear three-dimensional maps. We can now create a new "stretched" dodecahedron:

```
(define stretched-dodecahedron
  (apply-3d-map-to-solid-object stretch-x-map *current-solid-object*))
```

The new shape is shown in Figure 9b (the original dodecahedron is shown in Figure 9a); the folding net for the shape (generated by the program) is shown in Figure 9c.

---

[11]In fact, we have chosen the extent of truncation to produce a semiregular solid whose faces are regular hexagons and squares.

[12]This choice of parameter was motivated by ease of specification, allowing the rapid generation of (e.g.) Archimedean solids from the Platonic solids. On the other hand, truncating vertices in this manner may not produce "true" polyhedra: in particular, truncating an arbitrary vertex by subtracting a fixed length from all incident edges is not guaranteed to produce a planar face as a result. Our program can in fact inform the user whether a truncation has produced an invalid result; the program also includes an alternative "technically correct" method of truncation that is somewhat harder to use but does not result in solids with nonplanar faces.

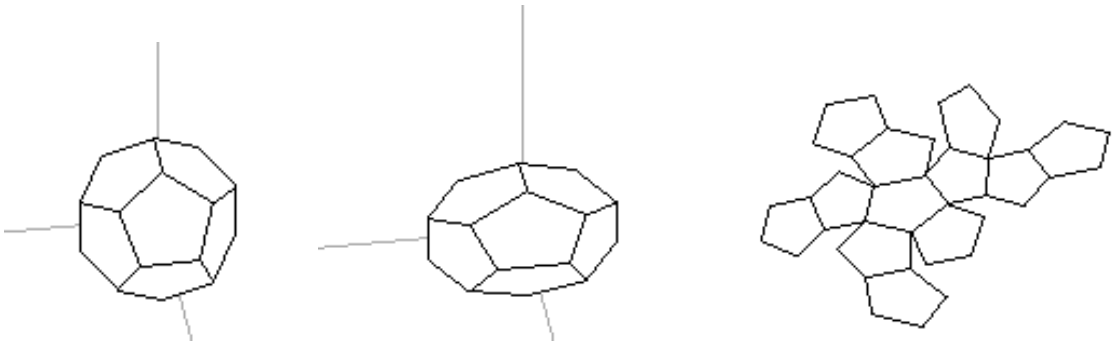

Figure 9a (left). A dodecahedron
Figure 9b (center). The result of "stretching" the dodecahedron out along one dimension.
Figure 9c (right). The folding net for the "stretched dodecahedron."

Finally, it is possible to use HyperGami's language elements to place "caps" on faces: that is, adding a new vertex ***v*** somewhere "outside" a face ***f*** of the solid and creating new triangular faces between ***v*** and each of the original edges of ***f***. The most typically-employed form of capping is one in which new vertices are added above the midpoint of the face, at a specified distance ***h*** from the original face. An example is shown in Figure 10. At the left of the figure, we have a pentagonal bipyramid; at center, the result of placing cap vertices above each of the faces of the bipyramid (in this instance, to a height of 0.3, where the circumradius of the central pentagon is 1); and at right, the folding net for the capped bipyramid is shown (as calculated by our program).

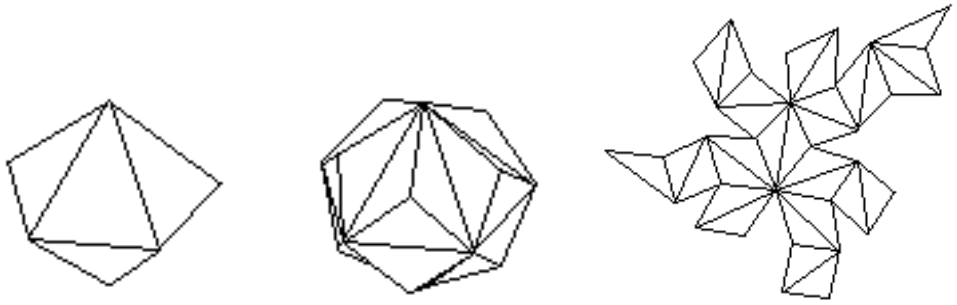

Figure 10a (left): a pentagonal bipyramid.
Figure 10b (center): The bipyramid has been "capped" at every face.
Figure 10c (right): A folding net, generated by the program, for the capped bipyramid. Note that the system finds a viable net even though the solid is nonconvex.

5. Creating Decorated "Customized" Polyhedra: Two Scenarios

In this section, we bring together the various isolated techniques described in the previous two sections to illustrate how HyperGami may be used to create complete, decorated polyhedra. In our first example, we use one of the built-in polyhedral nets within HyperGami (in this case, for an icosahedron) and employ language techniques to decorate it in a way that is not only attractive, but in addition highlights a particular mathematical interpretation of the solid. In the second example, we start with perhaps the simplest solid of all—the cube—and perform some operations on it to construct a "specialty" polyhedral model.

5.1 Decorating an Icosahedron

Suppose we initially select the icosahedron icon from the **Polyhedra** palette; the system presents us with a folding net and solid view of the icosahedron, as shown in Figure 11. In this instance, we may decide that we do not wish to vary the solid (it is an extremely attractivemodel[13]), but instead simply wish to decorate the solid. Our decoration will be one that calls attention to the fact that the icosahedron is in fact a "bicapped pentagonal antiprism": by focusing on a central "ring" of ten triangles, one can see that this ring is "capped" on either end by sets of five triangles. Our decoration will therefore employ two basic patterns: one for the "caps" and one for the "ring".

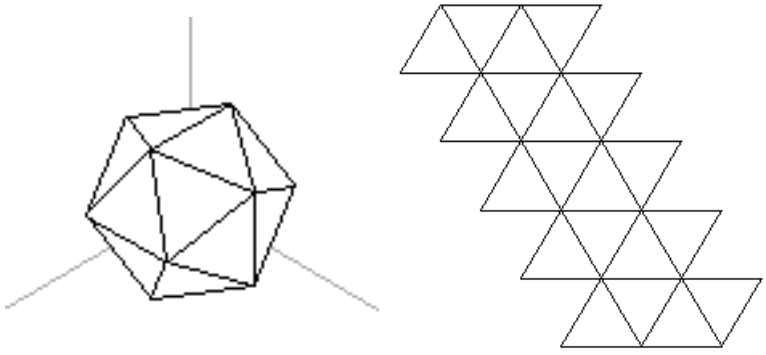

Figure 11. The icosahedron (left) and its folding net.

We begin by creating a procedure that randomly chooses between three colors:

[13]The icosahedron is in fact the solid used for the logo of the Mathematical Association of America.

```
(define (choose-red-green-or-blue)
  (let ((choice (random 3)))
    (if (= choice 0) red (if (= choice 1) blue green))))
```

We can now use a primitive in HyperGami, `set-mouse-function!`, to make this procedure the "filling" routine for any area on the screen (details of this process are given in Appendix A). By filling a triangle with this procedure, we effectively produce a "speckle" pattern in which red, green, and blue are randomly sprinkled within the triangle.

We would now like to use this procedure to fill triangles corresponding to a "central ring" of the icosahedron; but how can we tell which triangles might correspond to a "cap" and which to a "ring"? To answer this question, we use the linked representations (as shown in Figure 12), drawing a line around a particular "cap" of the icosahedron (in this case a "cap" pointed toward the viewer). Armed with this insight about which triangles in the net correspond to a cap, we can now easily see which remaining triangles correspond to the "ring" underneath that cap. We then employ our new procedure to decorate the ring, as shown in Figure 13.

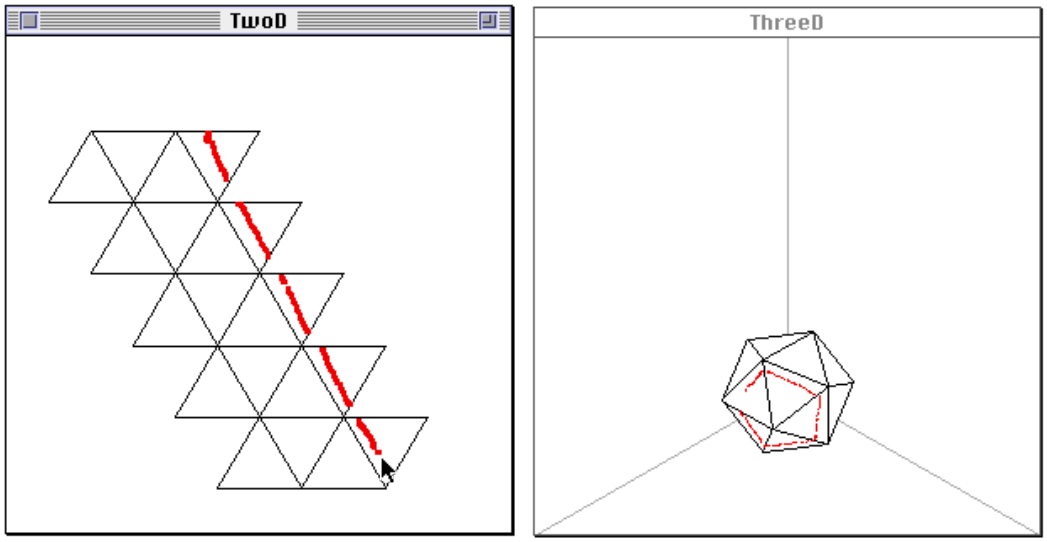

Figure 12. We use the linked two- and three-dimensional representations of the icosahedron to identify a solid "cap" (pointed toward the viewer) corresponding to the five net triangles toward the right of the two-dimensional pattern.

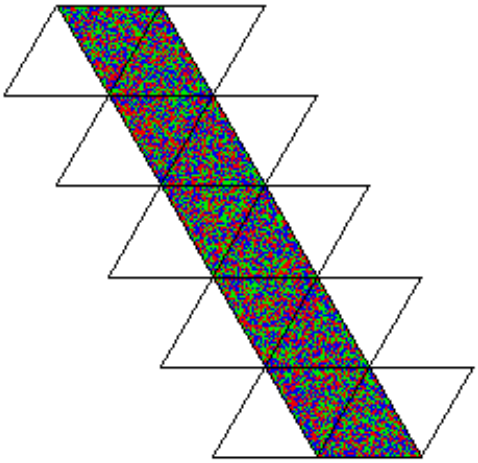

Figure 13. The "central ring" underneath the cap drawn in Figure 12 is now decorated with a random pattern of red, green, and blue pixels.

Finally, we would like to use the turtle to decorate each of the remaining cap triangles. To do this, we will write a procedure much like the one shown earlier: in this case we move the turtle to each triangle in the net, and then wait for user input to see if this is indeed a "cap" triangle. If it is, the turtle now draws a simple pattern from the center of the triangle to each of the vertices; if not, the turtle moves on to the next triangle. (The complete procedure is shown in Appendix A.) The resulting final decorated net is shown in Figure 14a; this may now be printed out and folded into the solid shown in the photograph in Figure 14b at right.

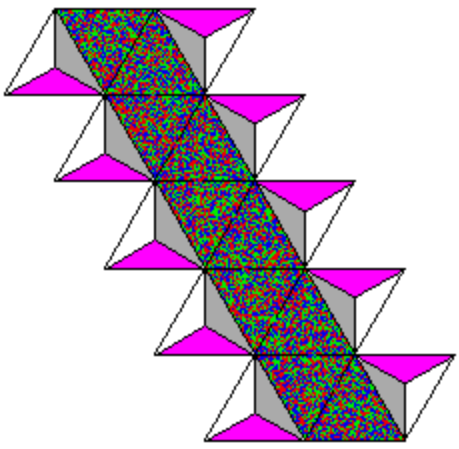

Figure 14a (left). The final folding net for our decorated icosahedron.
Figure 14b (right). The folded solid.

5.2 Creating a New Polyhedron

In the previous section, a number of solid-manipulation techniques were introduced—vertex truncation, application of linear maps, and face capping. These operations may likewise be composed; in this scenario, we begin with a cube, and then apply several operations in succession. Suppose, then, that we begin with the cube of side-length 2 shown in Figure 15a. We proceed to truncate this cube at each vertex, "chopping" each edge by 0.67 to produce the solid in Figure 15b; we then apply the same "stretch-x" linear map seen in the previous section to the truncated cube to produce the shape in Figure 15c.

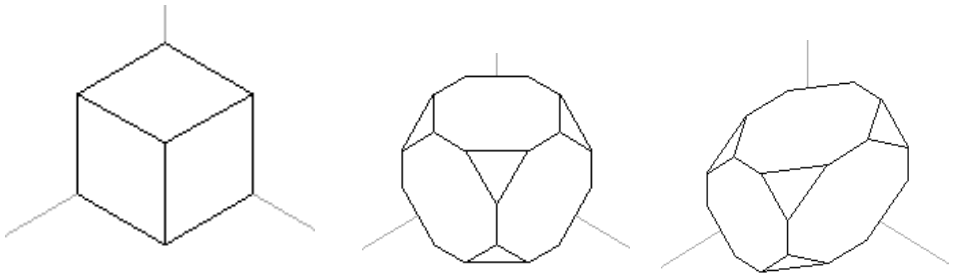

Figure 15a (left). A cube.
Figure 15b (center). The result of truncating each vertex of the cube.
Figure 15c (right). Stretching the truncated cube along the x-dimension.

Finally, we would like to take the solid in Figure 15c and cap only two of its faces, corresponding to the "unstretched" octagons parallel to the *yz* plane. HyperGami's language primitives allow us to identify these two polygons in the solid and to cap these two polygons only (the details of this process are given in Appendix A); the resulting shape is depicted in Figure 16a, and the corresponding net in Figure 16b. Note that the solid that we have produced retains only some of the symmetry operations of the original cube (for instance, unlike the cube, this new shape has no three-fold axis of rotational symmetry).

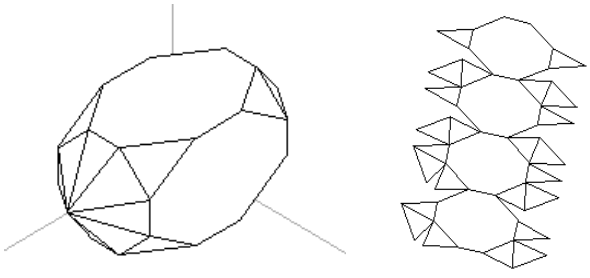

Figure 16a (left). Each "unstretched" octagon of the solid in Figure 15c has been capped.
Figure 16b (right). A folding net for the shape in Figure 16a.

Finally, we decorate the folding net (here, using simple patterns and solid colors) to produce the decorated net in Figure 17a; we print this out and fold it into the shape photographed in Figure 17b.

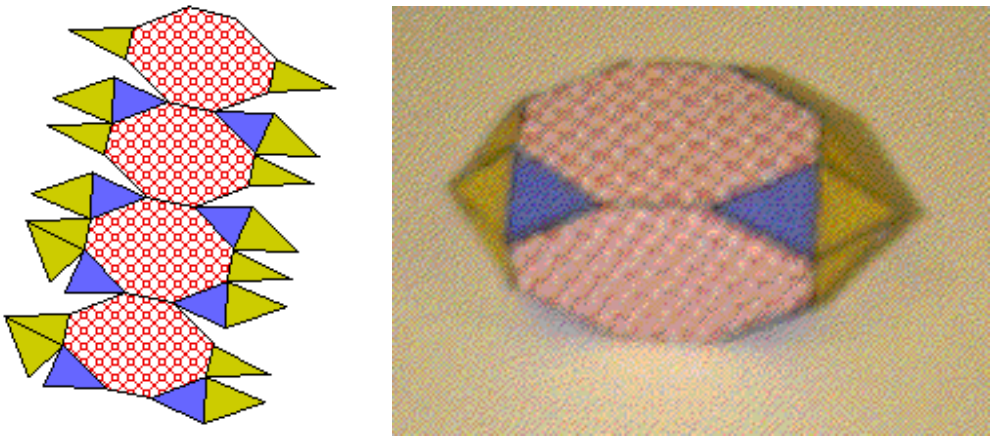

Figure 17a (left). The decorated net for the solid in Figure 16b.
Figure 17b (right). The folded solid.

6. HyperGami in Use by Students

During the past academic year (1994-5), we began our initial pilot studies of HyperGami with ten local schoolchildren ranging in age from eight to thirteen. These students were volunteers with interest in computers, paper crafts, or both; they worked as individuals or in pairs, attending weekly or bi-weekly sessions ranging in length from one to three hours. In this section we describe the activities undertaken by the children, and we summarize what we believe are the key issues that arise when HyperGami is used in an educational setting.

6.1 Pilot Project Activities

Not all of the children's HyperGami projects involved polyhedral models; as the name of the program suggests, it may also be used to decorate folding nets for origami figures (Eisenberg and Nishioka, 1994), and some of the projects that the children undertook involved the creation of (e.g.) decorated origami frogs and turtles. On the other hand, all of the children did choose, at some point, to undertake the construction of polyhedral figures. One thirteen-year old girl worked in collaboration with us to construct a dodecahedron decorated with turtle patterns; the strategy employed was much like the one illustrated in

Figure 5a. That same student collaborated with an eight-year-old to construct a decorated great stellated dodecahedron by first folding together a "core" icosahedron and then printing out twenty individually-decorated "cap" triangular pyramids which could be glued to the central core; the students in fact worked at this project for more than half the day, and then continued their work at home, in their eagerness to see the completed product.

It should be noted that the construction of a shape like the great stellated dodecahedron makes an excellent group project, allowing as it does for individual students to create their own "personalized" pyramidal caps which may then be put together in constructing the final shape. The eight-year-old participating in the construction of the figure enjoyed the fact that they "... didn't just make one paper" but made several different designs for the caps of the figure; and she made note of the fact that she and her friend "both had turns" in designing the construction.

In addition to decorating and constructing standard polyhedral forms, our students have recently begun to use some of the more advanced features of the system to build their own "custom" solids along the lines discussed in the previous sections. One student created a "stretched icosidodecahedron"; another student experimented with an elongated icosahedron to create an "alien head"; another used a variety of shapes (including antiprisms and a "stretched dodecahedron") to create a polyhedral sculpture of a brontosaurus; and yet another has employed antiprisms and a truncated icosahedron in the course of making a paper hippopotamus. In these last two cases, then, the students' work encourages them to think of polyhedra not only as aesthetically pleasing andmathematically rich units in themselves, but also as structural "building blocks" for multipart paper sculptures; such sculptures, like the great stellated dodecahedron mentioned above, have also proven to be especially good candidates for collaborative or group constructions (inasmuch as they often consist of distinct parts on which various students can work). We will return to this topic briefly in the next section.

In the process of decorating their polyhedra, students also employed some of the programming language features of the system. A favorite activity—and gentle introduction to some of the language concepts—involved the creation of "customized" colors. The students wrote Scheme procedures to numerically specify different amounts of red, green, and blue color components (on a scale from 0 to 1); they then used this procedure to color specified areas of their folding nets. Some of the more advanced students wrote procedures

similar to the "speckling" pattern discussed in the previous section, and one pair worked with procedures to create "rainbow" color gradients.

Besides turtle graphics and color-generation procedures, students have also made use of several other language constructs in the system. Two students used procedures to create customized "pattern fills" (expressed as Scheme lists of ones and zeroes, corresponding to the foreground and background elements of the pattern). Several students also used Scheme procedures for placing strings at specified locations to add text to their polyhedral sculptures; in this way they added decorations such as names on polyhedral buildings. On the interface side, several students have employed the "linked two- and three-dimensional pen" technique (described in Section 3 earlier) in order to place decorative elements at specific desired locations in their constructions.

Figure 18. HyperGami constructions by students. From left to right: a "stretched bipyramid", a "capped cuboctahedron", a decorated icosahedron, a paper sculpture of a brontosaurus on a log, and a "stretched dodecahedron". The constructions in this photograph were created by four students (ranging in age from 11 to 13) working with us during their first semester of tutorials in HyperGami.

Perhaps the most important dimension along which to assess the children's activity is the affective one. As noted earlier in this paper—and as confirmed by our experience—polyhedral models are capable of evoking feelings of delight, pride, and even (in the case of paper sculptures) affection from their creators. One student who created a "capped

cuboctahedron" (by attaching pyramids to a cuboctahedral core) informed us that it had been on display at her home; an eleven-year-old student gave personal names to her constructions; a thirteen-year-old student has shown his creations to his teacher at school; yet another student has employed his constructions (including a stella octangula) as presents to various adults. Few other mathematical manipulatives—and little in the purely "virtual" world of computer creations—could be capable of eliciting such behavior.

6.2 Observations of Children Using HyperGami

Our observations of students using HyperGami during the past semesters have alerted us to a number of issues (and promising directions for further study). In some cases, these issues relate to questions of how the use of a tool such as HyperGami may be affected by (and may in turn affect) children's development of spatial imagery. For instance, we conducted interviews with our students focusing on tasks of mental rotation and surface development. In these interviews—similar in spirit to the studies by Piaget and Inhelder described earlier—students were shown a flat folding net generated in HyperGami and then were asked to select the corresponding polyhedron from a collection of three-dimensional paper shapes without physically folding the net. The results of these interviews show differences in approaches taken by younger and older children to identify the appropriate solid. In particular, younger children (ages 8 and 9) used a combination of problem-solving techniques including both the attempt to mentally visualize the folded shape as well as the use of visual cues such as matching polygons on the nets of the polygons on the three-dimensional figures. By contrast, none of the older students (up to age 13) who participated in the study claimed to do any mental papter folding, but instead isolated the correct shape by purely empirical means. One of the most systematic students first counted the number of sides of one of the polygons on the net; then carefully counted the number of polygons on the net to see if they matched the number of polygons on the solid; and then repeated this process for each figure presented.

The activity of constructing paper polyhedra with HyperGami is one that requires not only skill in spatial visualization, but also manual dexterity in cutting, folding, and gluing the net to arrive at the finished shape. Some of the children participating in this year's pilot study felt that assembling the paper sculptures was the most difficult part of the construction process. There was a marked difference in dexterity between the eight-year-olds and the twelve- and thirteen-year-olds in the study; when constructing polyhedra or origami in pair situations, younger students sometimes struggled to keep up with the older ones.

However, this problem did not prove to be a big obstacle since older students often helped younger ones through the more difficult steps.

Another point of interest in terms of the construction process is that less patient children quickly learned that rushing through polyhedra building led to dented faces, soggy paper, and edges that did not fit together. It was sometimes a struggle for the children to temper their eagerness to see the finished product with the patience to assemble individual components in a robust manner. Not every project ended in success: one attempt (by a thirteen-year-old) to produce a construction of a paper octahedron placed inside a folded transparency cube never quite generated a "clean" result (in large part because of the difficulty of folding transparencies and matching the sizes of the two components). Making finished HyperGami constructions thus proves to be a matter of both abstract work—dealing with representations (and often language expressions) at the computer screen—and craftsmanship in working with an actual physical medium. Nonetheless, because of the way that it combines real world experiences with a virtual computer environment, we would argue that HyperGami in this sense runs happily counter to the "faster is better" philosophy embedded in many of today's educational software packages. In the words of artists Kasahara and Takahama (1987), working with paper encourages "the sense of breadth and ease and the willingness to learn by taking detours... that society is now in danger of losing."

Going beyond our continuing tutorial work with children, we plan to initiate work using the system with classroom teachers during the coming year as well. In addition, an undergraduate student interested in crystallography has conducted a project using the program to create solids based on the shapes of crystals. (For example, the truncated octahedron shown in Figure 8 is in fact a space-filling solid, so that one can create larger "packings" of such shapes; cf. the discussion of crystals in Chieh (1988).) Our current plans also include the incorporation of HyperGami polyhedra into a children's book now in development (as described in the following section).

7. Related, Ongoing, and Future Work

The development of HyperGami reflects the influence of a number of lines of work. In particular, the system's strategy of integrating "real-world" artifacts with specification by programming constructs owes much to the example of LEGO/Logo (Resnick, 1988), an educational system that permits students to write Logo language expressions that dictate the

behavior of mobile Lego constructions. It should also be mentioned that a number of additional efforts in integrating computer programs with paper constructions have recently been developed. Lang (1994), for instance, describes a computer program that assists in the calculation of crease patterns for intricate or novel origami figures; and a popular CD-ROM has been developed to teach principles of paper airplane construction (WordPerfect MainStreet, 1994).

HyperGami shares many of the concerns and interests of these varying efforts, but also exhibits its own distinct characteristics. Unlike LEGO/Logo, HyperGami has little to offer in the way of simulation or dynamic specification (the paper constructions, unlike Lego constructions, are not assumed to be equipped with motors and sensors); but because HyperGami constructions are made in paper they exhibit a flexibility and richness of form unique to that medium. Moreover, the decorative element of paper constructions—the ability to create designs and patterns for solid forms—lends yet another dimension to this form of constructive activity.

Perhaps the most important similarity between HyperGami and LEGO/Logo is the incorporation of an interactive programming environment to lend expressiveness to the design activity. In this respect, HyperGami differs from other current geometric construction kits (whether these kits are books of nets, CD-ROM construction packages, or sets of plastic parts); the language elements of the system facilitate not only customized decoration of nets but the construction of a variety of new polyhedra, as described earlier. In turn, the ability to construct new "specialty" polyhedra has led us toward new projects and activities, including the construction of more complex paper sculptures from smaller polyhedral parts ("ori-hedra"). The guiding notion behind these constructions is that they present polyhedra as "building blocks", in a manner less abstract than the standard mathematical treatment; this can be a potential boon to those students who may feel themselves more at home in artistic than mathematical activities.[14]

Figure 19 illustrates the construction of an "ori-hedron" resembling a caterpillar. In Figures 19a and 19b, we see the nets for two polyhedra used to construct the caterpillar: at left, a bicapped hexagonal prism (these will be used for the segments of the caterpillar's body),

[14]To our knowledge, the first and only such use of polyhedral building blocks to construct larger paper sculptures is due to Suzuki (1987). Suzuki's constructions, unlike ours, are essentially undecorated; and they use relatively few specific polyhedral building-block types, rather than newly-created "specialty" polyhedra.

and at right a prism constructed for an "elongated" octagonal base (these will be used for the caterpillar's sneakers). Figure 19c shows how multiple copies of these nets can be put together to create an overall "caterpillarhedron" made of a long series of body segments and shoes.[15]

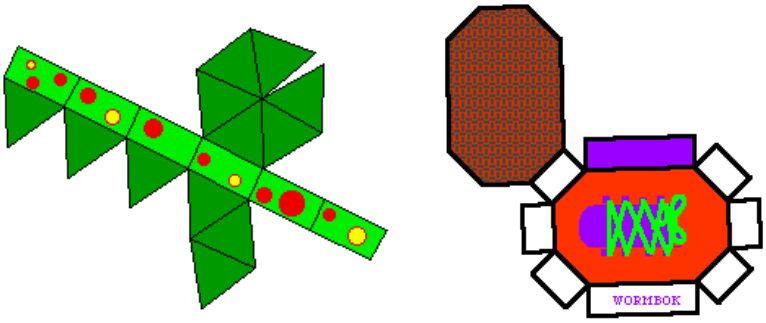

Figure 19a (left). The net for the caterpillar's body segments.
Figure 19b (right). The net for the caterpillar's sneakers.

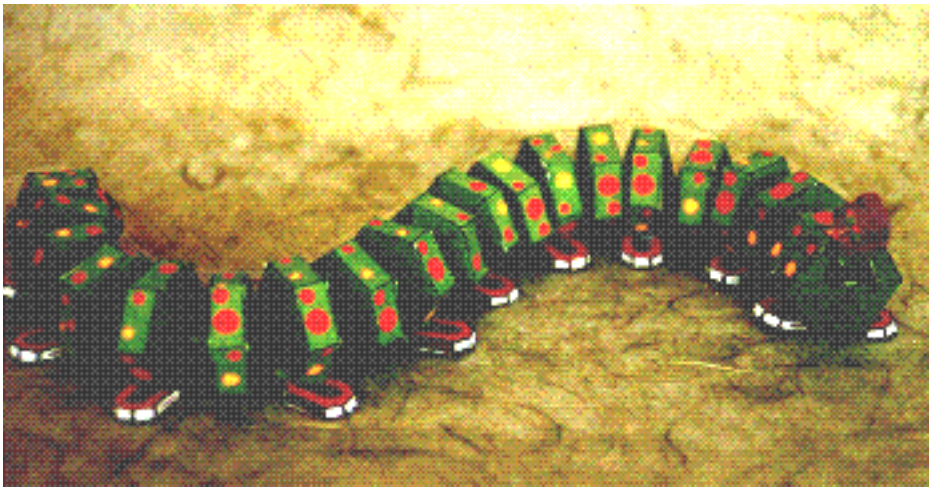

Figure 19c. The caterpillarhedron, out for a stroll.

We are currently developing a "children's mathematics book" including a number of ori-hedra developed in HyperGami (along with text explaining how these forms were created using the software); this would serve as a sourcebook of ideas for students interested in creating their own polyhedral paper sculptures. We have some confidence in the motivational aspect of this activity: as noted in the previous section, some of our current students have become engaged in ori-hedral projects.

[15]The caterpillar's eyes—just for completeness of description—are made of tiny tetrahedra, and his antennae are short strips of paper. The body segments for the sculpture are strung together with fishing line.

Looking past these ongoing projects, there are vast numbers of research questions suggested by the current stage of development of HyperGami. First—in the area of program development—the current system is very much a "work-in-progress", with numerous potential avenues of improvement. For instance, the program does not include a "reverse linkage" between solids and nets; e.g., one cannot reverse the directionality of Figure 3 by drawing a line on the solid figure in the **ThreeD** window while observing how such a line would appear on the net. Yet another area for investigation would involve the creation (and access) of net/solid databases, and how such databases should be structured. (For instance, suppose we create a new solid like the one in Figure 16. How might we locate solids "similar" to this one—should these be solids likewise derived from a cube, solids with only octagonal and triangular elements, or solids whose folding nets resemble, in some fashion, the net that we just constructed?) On a more mundane level, we are currently developing much-needed documentation and support materials (with a particular focus on classroom activities) to accompany the program.

Our own belief is that a program such as HyperGami offers fertile ground for studying—and assisting—the development of skills of spatial visualization, as suggested by the interviews mentioned in the previous section (and as implied by the Piaget-Inhelder results). One addition to the program that we have also begun to investigate involves the creation of computational "critics" (along the lines described by Fischer *et al.* (1991)) that can guide students with heuristic techniques for visualizing and interpreting solids as they are created. Such critics would be less directive than an online "geometry lesson plan", but would be capable of interpreting the student's ongoing work with the goal of suggesting related mathematical topics for students to explore.

Finally—and perhaps in a more futuristic vein—HyperGami (and similar "real-world-based" systems such as LEGO/Logo) suggests to us the possibility of incorporating computers in a variety of craft activities; conceivably, just as computers may be used to animate LEGO creatures or design paper sculptures, they may likewise provide new media of expression for students interested in woodworking, puppetry, glassblowing, or scientific instrumentation design. Our hope is that the issues raised in this still early effort will inform the development of design environments for artifacts beyond those constructible on paper.

Acknowledgments
We are indebted to the ideas, suggestions, and encouragement of numerous people. Hal Abelson, Jonathan Cohen, Wally Feurzeig, Eugene Hoffman, Michael Mills, and Jonathan Rees have been especially influential. Gerhard Fischer, Kumiyo Nakakoji, Julie DiBiase, and the members of the Center for LifeLong Learning and Design at the University of Colorado have been important sources of ideas, examples, and scholarship, as well as great conversation. Will Clinger, Zach Nies, and Ed Post generously helped in extending our MacScheme environment. We would also like to thank the two anonymous referees of the first draft of this paper; their comments motivated important revisions for the current version.

The work of the first author is supported in part by NSF grants RED-9253425 and a Young Investigator award IRI-9258684. The second author is supported by a fellowship from the National Physical Science Consortium. The work described in Section 7 is supported in part by the National Science Foundation and the Advanced Research Projects Agency under Cooperative Agreement No. CDA-940860. Finally, we would like to thank Apple Computer, Inc. for donating the machines with which this research was conducted.

Bibliography

Abelson, H. and diSessa, A. (1980). *Turtle Geometry*. Cambridge, MA: MIT Press.

Abelson, H. and Sussman, G. J. with Sussman, J. (1985). *Structure and Interpretation of Computer Programs*. Cambridge, MA: MIT Press.

Banchoff, T. (1990). Dimension. In Steen, L. A. (Ed.), *On the Shoulders of Giants: New Approaches to Numeracy*, pp. 11-59.Washington, D.C.: National Academy Press.

Barnsley, M. (1988). *Fractals Everywhere*. Boston: Academic.

Chieh, C. (1988). Polyhedra and crystal structures. In M. Senechal and G. Fleck (Eds.), S*haping Space: A Polyhedral Approach*, pp. 93-105. Boston: Birkhäuser.

Cooper, N. G., Ed. (1989) *From Cardinals to Chaos*. Cambridge: Cambridge University Press.

Coxeter, H.S.M. (1973) *Regular Polytopes*. New York: Dover.

Coxeter, H.S.M. (1988). Regular and semiregular polyhedra. In M. Senechal and G. Fleck (Eds.), S*haping Space: A Polyhedral Approach*, pp. 67-79. Boston: Birkhäuser.

Croft, H.; Falconer, K.; and Guy, R. (1991). *Unsolved Problems in Geometry*. New York: Springer-Verlag.

Eisenberg, M. (1996). A functional algebra for polyhedra, and its use in generating folding nets. (In preparation.)

Eisenberg, M. (1991). Programmable applications: interpreter meets interface. *MIT AI Lab Memo* 1325. Massachusetts Institute of Technology, Cambridge, MA.

Eisenberg, M. and Fischer, G. (1994). Programmable Design Environments. In *SIGCHI '94 Proceedings*, pp. 431-437. Boston, MA.

Eisenberg, M.; Hartheimer, A.; and Clinger, W. (1990). *Programming in MacScheme*. Cambridge, MA: MIT Press.

Eisenberg, M. and Nishioka, A. (1994). HyperGami: A Computational System for Creating Decorated Paper Constructions. To appear in P*roceedings of Second International Meeting of Origami Science and Scientific Origami*. Otsu, Shiga, Japan.

Fischer, G., Lemke, A.C., Mastaglio, T. and Morch, A. (1991). The Role of Critiquing in Cooperative Problem Solving. *ACM Transactions on Information Systems*, 9:2, pp. 123-151.

Fomenko, A. (1994). *Visual Geometry and Topology*. Berlin: Springer-Verlag.

Hadamard, J. (1949). *The Psychology of Invention in the Mathematical Field*. New York: Dover.

Hilbert, D. and Cohn-Vossen, S. (1952). *Geometry and the Imagination*. New York: Chelsea.

Hilton, P. and Pedersen, J. (1994). *Build Your Own Polyhedra*. Menlo Park, CA: Addison-Wesley.

Holden, A. (1971). *Shapes, Space, and Symmetry*. New York: Dover.

Jenkins, G. and Wild, A. (1990). *Make Shapes Series No. 1*. Norfolk, UK: Tarquin Publications.

Kasahara, K. and Takahama, T. (1987). *Origami for the Connoisseur*. Tokyo: Japan Publications, Inc.

Lang, R. (1994). Mathematical Algorithms for Origami Design. *Symmetry: Culture and Science*, 5:2, pp. 115-152.

Malkevitch, J. (1988). Milestones in the history of polyhedra. In M. Senechal and G. Fleck (Eds.), S*haping Space: A Polyhedral Approach*, pp. 80-92. Boston: Birkhäuser.

Marinoni, A. (1974). The writer: Leonardo's literary legacy. In L. Reti (Ed.), *The Unknown Leonardo*, pp. 56-85. (1990 edition.) New York: Harry N. Abrams.

Mühlhausen, E. (1993). "Riemann Surface—Crocheted in Four Colors." *Mathematical Intelligencer*, 15:3, pp. 49-53.

Pedersen, J. (1988). Why study polyhedra? In M. Senechal and G. Fleck (Eds.), S*haping Space: A Polyhedral Approach*, pp. 133-147. Boston: Birkhäuser.

Piaget, J. and Inhelder, B. (1948). *The Child's Conception of Space*. New York: W. W. Norton & Company.

Resnick, M. (1989). Lego, Logo, and life. In C. Langton (Ed.), *Artificial Life*, pp. 397-406. Reading, MA: Addison-Wesley.

Schattschneider, D. and Walker, W. (1977). *M. C. Escher Kaleidocycles*. Corte Madera, CA: Pomegranate Artbooks.

Senechal, M. (1988). A Visit to the Polyhedron Kingdom. In M. Senechal and G. Fleck (Eds.), S*haping Space: A Polyhedral Approach*, pp. 3-43. Boston: Birkhäuser.

Senechal, M. (1990). Shape. In Steen, L. A. (Ed.), *On the Shoulders of Giants: New Approaches to Numeracy*, pp. 11-59.Washington, D.C.: National Academy Press.

Shepard, R. and Feng, C. (1972). A Chronometric Study of Mental Paper Folding. Reprinted in R. Shepard and L. Cooper (Eds), *Mental Images and Their Transformations*, pp. 191-206. Cambridge, MA: MIT Press.

Suzuki, E. (1987). *Omoshiro ku kan e no sho tai. (Unique Creativity Utilizing Polyhedra.)* In Japanese. Tokyo: Teruko Nakagawa.

Wenninger, M. (1971). *Polyhedron Models*. New York: Cambridge University Press.

WordPerfect MainStreet (1994). *Paper Planes* (CD-ROM). Provo, UT: Novell, Inc.

Appendix A.

In this section we present the HyperGami/Scheme language expressions used to construct the two sample polyhedra in Section 5.

A1. In constructing the decorated icosahedron, we use the color-choosing procedure shown in the text:

```
(define (choose-red-green-or-blue)
  (let ((choice (random 3)))
    (if (= choice 0) red (if (= choice 1) blue green))))
```

By using the `set-mouse-function!` HyperGami primitive (which takes as its argument a procedure from position to color), we can make the new random color-selection procedure accessible as a fill routine:

```
(set-mouse-function! (lambda (x y) (choose-red-green-or-blue)))
```

Now, by first selecting the "proc" mode and then "fill" mode in the **Paint** window, we can fill any arbitrary region according to the given procedure. This produces the "ring" decoration shown in Figure 13.

To produce the lines within triangles, we use the following two turtle procedures. The first places the turtle at the center of a triangle and asks if we wish to draw central lines; if the user types in "Y", we then call `draw-central-lines-from-center`, which moves the turtle from its central point back and forth to each of the triangle vertices.

```
(define (draw-central-lines-if-cap triangle)
  (move-turtle-to-point (polygon-midpoint triangle))
  (newline)
  (display "Is this a cap triangle? ")
  (let ((answer (read)))
    (if (eq? answer 'y)
        (draw-central-lines-from-center triangle)))
    ))

(define (draw-central-lines-from-center triangle)
  (for-each
    (lambda (vertex)
      (pd)
      (setpos (xcor vertex) (ycor vertex))
      (pu)
      (move-turtle-to-point (polygon-midpoint triangle)))
    (vertices triangle)))
```

Now, for each polygon, in the net, we call the `draw-central-lines-if-cap` procedure; this call generates the pattern in Figure 14 (which we then filled by hand).

```
(for-each draw-central-lines-if-cap (current-net-polygons))
```

A2. To create the caps on the shape in Figure 16, we first identified those polygons in the solid which were octagons; the call below filters out those polygons with 9 vertex elements (in HyperGami, closed polygons are stored with the same vertex both as first and last

element in a list of vertices—hence we seek those polygons with 9 instead of 8 elements in their vertex lists).

```
(define *octagons*
  (filter (lambda (3dpoly)
            (= (length (vertices 3dpoly)) 9))
          (current-solid-polygons)))
```

We now filter out only those octagons whose vertices have points with entirely positive or negative x-coordinaes (the others cross over the yz-plane):

```
(define *end-octagons*
  (filter (lambda (octagon)
            (or (true-of-all?
                 (lambda (v) (> (xcor v) 0))
                 (vertices octagon))
                (true-of-all?
                 (lambda (v) (< (xcor v) 0))
                 (vertices octagon))))
          *octagons*))
```

And finally, we cap the two polygons. This can be done in one step, but we show a two-step process here. First we create an intermediate solid object in which the first of our two target octagons is capped; we then cap the second octagon on the newly-created intermediate solid. The result is the doubly-capped solid in Figure 16. (Note that `*stretchcube*`, in the first expression below, is the name of the "stretched truncated cube" in Figure 15c.)

```
(define *capped-shape1*
  (add-cap-vertex *stretchcube* (first *end-octagons*) 0.8))

(define *capped-shape2*
  (add-cap-vertex *capped-shape1* (second *end-octagons*) 0.8))
```