COMPUTER RECREATIONS

A program for rotating hypercubes induces four-dimensional dementia

by A. K. Dewdney

"My husband has disappeared into thin air, and I think you had something to do with it!" The woman on the telephone was Cheryl, and she was clearly upset. Her husband Magi, my microcomputer amanuensis at the University of Western Ontario, had apparently vanished while viewing a computer program I had suggested he write. The program rotates a four-dimensional analogue of a cube called a hypercube and projects it on a display screen. Cheryl went on in agitation: "There's a weird pattern of lines on the monitor and his clothes are lying in a heap near the chair. He must have been wearing these strange colored glasses made out of cardboard. And look at this—his socks are still in his shoes!"

Here, it seemed to me, was an obvious case of four-dimensional dementia. Victims become convinced they have stepped out of ordinary space and entered a higher-dimensional reality invisible to others. The delusion that one has disappeared can be so powerful that others take part in it: the victim can enter a room full of people and seem invisible to all. Fortunately Magi's case has a happy ending; I shall save it for last. In the meantime I submit the hypercube program to the wider public with what I hope is a responsible warning: Readers likely to fall prey to Magi's dementia are urged not to write the program or to view its output on a display screen. Potential victims include anyone with a history of obsession about the higher dimensions or anyone who is even occasionally tempted by the prospect of unknown realities.

The fourth dimension has been a vehicle for physical and metaphysical speculation at least since the 19th century. The idea of a fourth, physical dimension culminated in Einstein's theories of special and general relativity; space and time together make up a four-dimensional continuum in which all real events are timelessly frozen. This view of the universe may be undergoing dimensional modifications; the so-called Kaluza-Klein theories introduce seven or more new dimensions in the form of miniature hyperbubbles attached to every point of spacetime [see "The Hidden Dimensions of Spacetime," by Daniel Z. Freedman and Peter van Nieuwenhuizen; SCIENTIFIC AMERICAN, March, 1985].

The fourth dimension that I have come to know and love is the child of mathematics. Readers in ordinary rooms have a three-dimensional coordinate system suspended overhead. Three walls meet in each corner of the room, and from that corner radiate three lines, each of which is the meeting place of a pair of walls. Each line is perpendicular to the other two lines. Can the reader imagine a fourth line that is perpendicular to all three lines? Probably not, but that is what mathematicians require in setting up the purely mental construct called four-dimensional space. You now have the chance to explore this space in a personal way and without danger to your person. You have only to write the program I call HYPERCUBE.

HYPERCUBE can trace its origins to a film produced in the mid-1960's by A. Michael Noll, then at Bell Laboratories, that depicts the two-dimensional shadows of four-dimensional objects moving in four-dimensional hyperspace. The program as it now stands, however, was developed by Thomas Banchoff and his colleagues in the Computer Graphics Laboratory at Brown University, and my inspiration for this column comes from the fascinating images it generates [*see illustrations on pages 19, 21 and 22*]. Banchoff, who is a professor of mathematics, directs the visual exploration of higher-dimensional surfaces and spaces as a complement to his writing and research as a geometer. In 1978 he and Charles Strauss produced a 9½-minute computer-generated color film that has since become a classic in the mathematical underground: *The Hypercube: Projections and Slicing.* (The film can be obtained from the International Film Bureau, Inc., 332 South Michigan Avenue, Chicago, Ill. 60604.) Banchoff is also probably the leading expert on the life and work of Edwin A. Abbott, the English clergyman and teacher who in 1884 wrote *Flatland,* a tale of imagined life in two dimensions.

Banchoff and his colleagues have devised striking images that illustrate properties of four-dimensional objects. The images on page 19, for example, depict the rotation of a four-dimensional hypercube in four-dimensional space. To appreciate the images consider the shadow cast by an ordinary cube on a plane: the shadow can resemble a square inside a square. If the appropriate faces of the cube are shaded, the shadow is a square with a square hole in it [*see bottom illustration on page 20*].

Similarly, when a hypercube is illuminated from a point "above" ordinary space in the fourth dimension, the three-dimensional "shadow" cast by the hypercube can resemble a cube inside a cube. The inner cube is surrounded by six six-sided polyhedrons that can be regarded as distorted cubes. The four distorted cubes adjacent to the sides of the inner cube fit together to form the solid figure whose surface is the boxlike torus shown in Banoff's images. The other two distorted cubes, the inner cube and the outer cube also form a solid torus, which is not shown. As the hypercube rotates, the square hole in the visible torus seems to move toward the viewer. Those who write the program HYPERCUBE will see similar changes, albeit not so realistic or continuous.

The images on pages 21 and 22 are from a forthcoming film by Banchoff and his colleagues Hüseyin Koçak, David Laidlaw and David Margolis: *The Hypersphere: Foliation and Projections.* The hypersphere is a far more complex object than the hypercube, and I shall not describe it in detail. Nevertheless, one can begin to appreciate the images by considering an ordinary sphere. If the sphere is initially at rest on a plane tangent to its south pole and a light is fixed at the initial position of its north pole, the shadow cast on the plane by the lines of latitude is a series of concentric circles [*see bottom illustration on page 20*]. If the sphere is rotated while the light is kept fixed, the images of the circles may become nonconcentric, and the image of any circle that passes through the source of light is a straight line.

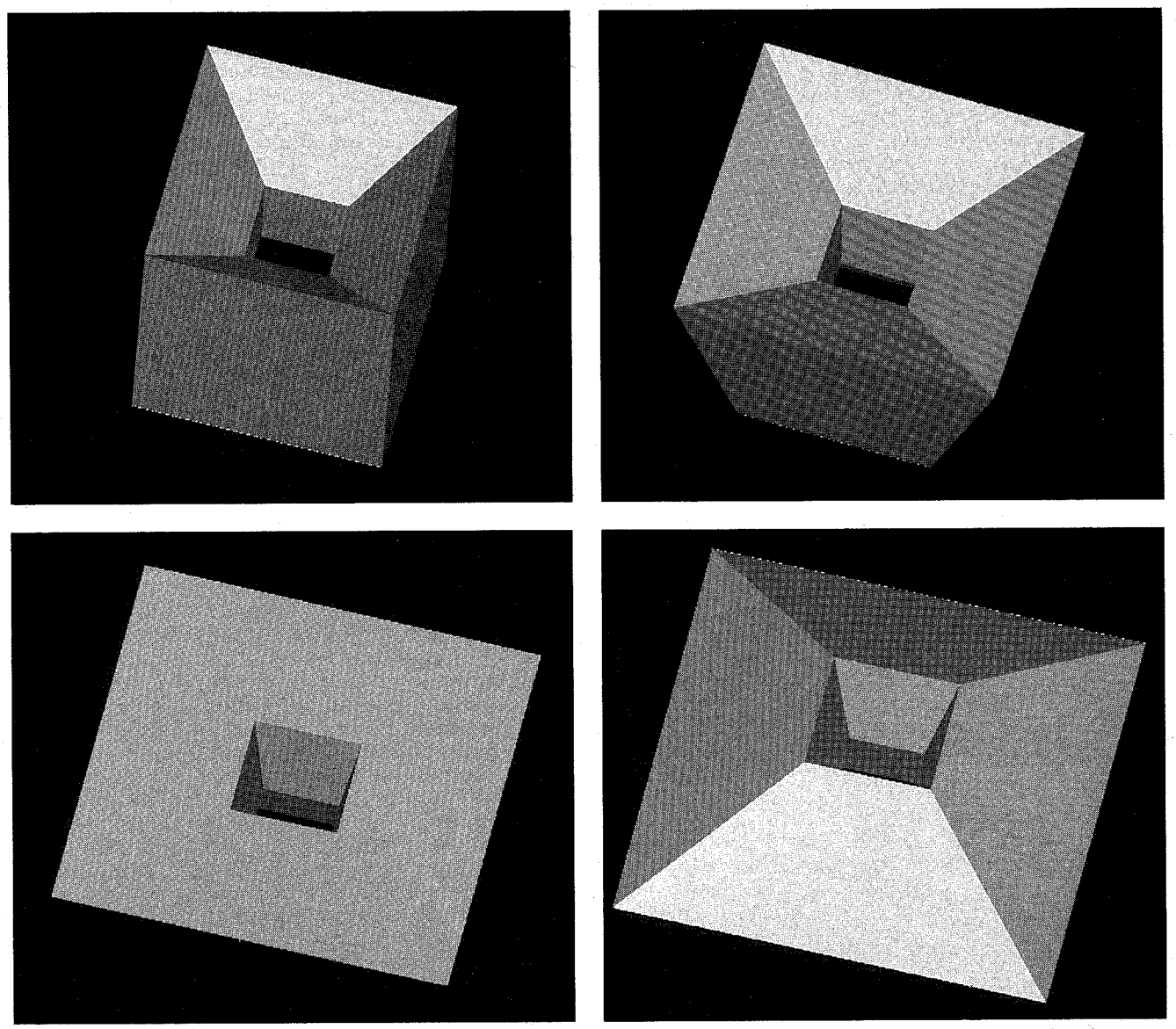
Similarly, the three-dimensional "shadow" cast by a hypersphere can be viewed as a series of concentric toruses [*see illustration on page 21*]. The toruses are made more readily visible in Banchoff's images by cutting away parts of one torus along strips that wind around it. When the hypersphere is rotated, the toruses appear to swell up and sweep past one another. Any torus that passes through the source of light becomes infinitely large [*see illustration on page 22*].

Dimensional analogies are valuable tools in constructing and understanding four-dimensional phenomena. The hypercube, for example, is derived from the cube just as the cube is derived from the square. To get the cube from the square lift the square in a direction perpendicular to its plane, up to a height equal to its side [*see top illustration on next page*]. The new cube has eight vertexes, twice as many as the initial square, and 12 edges, four from the initial square, four from the final square that is lifted away from the initial square and four that arise when vertexes in the initial square are connected to their counterparts in the final square. The cube also has six square faces, one coincident with the initial square, one coincident with the final square and one erected between each of the four pairs of edges that make up the initial and final squares.

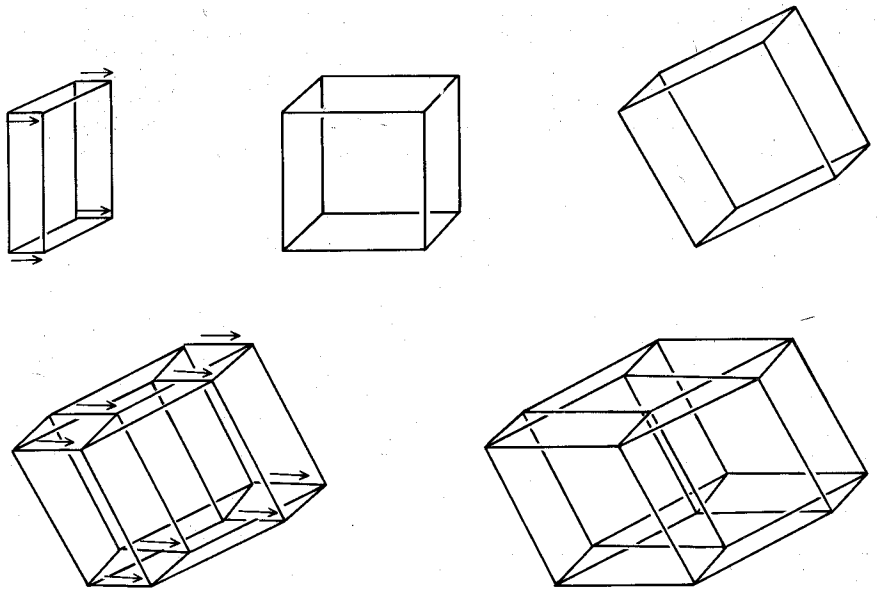
If one pretends for the moment that an additional dimension is available, the same operation can be repeated with the cube: "lift" the cube away from ordinary space in the direction of the extra dimension, out to a distance equal to the side of the cube [*see top illustration on next page*]. The result is a hypercube. But in what direction does the extra dimension lie? I cannot explain that. Even a photograph of me pointing into the fourth dimension would be utterly useless. My arm would simply appear to be missing.

Nevertheless, the number of vertexes, edges, faces and hyperfaces (ordinary cubes) that make up the hypercube can readily be counted. The number of vertexes is just the number of vertexes in the initial cube plus the number in the final cube, or 16. Each of the eight vertexes in the initial cube is joined by an edge to one of the eight vertexes in the final cube, and there are also 12 edges in each of the two cubes. Hence there are $8 + 12 + 12$, or 32, edges in the hypercube. One can also show that the hypercube has 24 ordinary faces and eight hyperfaces.

I am indebted to David Laidlaw for

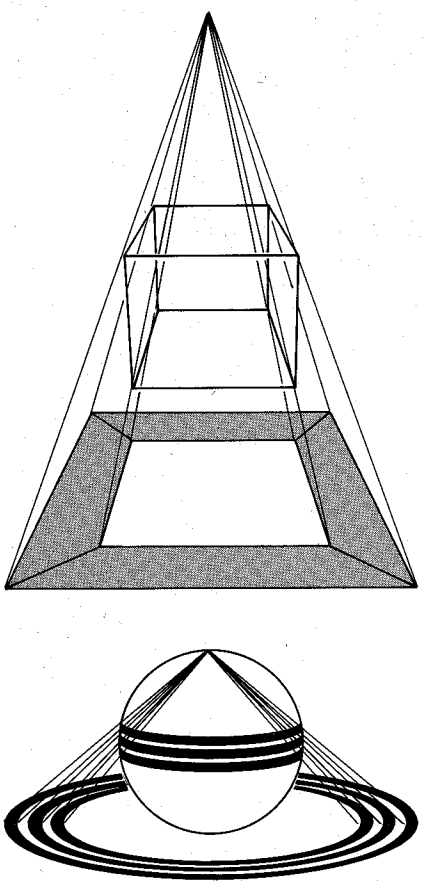


Rotation of a four-dimensional hypercube through dimensions 2 and 4, projected into ordinary three-dimensional space



How a plane generates a cube and a cube generates a hypercube

an explanation of HYPERCUBE. The version of the program I shall describe represents a hypercube by showing only its vertexes and edges. Moreover, the view the program generates does not necessarily depict a cube inside a cube; instead the view depends on how HYPERCUBE is implemented and on how it is run. Every time the hypercube in the program is rotated the vertexes swing into new positions and a new, oddly confusing view of the object results. With continued experimentation, however, the views begin to make a strange kind of sense, and one feels on the threshold of something awesomely spacious and inviting.



Projections of the cube and the sphere

The 16 vertexes of the hypercube in the program are numbered from 0 to 15 according to a simple scheme. If each number is rewritten in binary form and converted into an array of four bits, a miniature coordinate system emerges. The binary digits of 13, for example, are 1 (that is, one 8), 1 (one 4), 0 (zero 2's) and 1 (one 1). The binary number can then be written as the array (1,1,0,1), which almost gives a practical coordinate system for the initial position of the hypercube. (It is not a position that resembles a cube inside another cube.) To convert the binary array into useful coordinates, change the 0's to -1's and multiply each member of the array by a number large enough to generate an image of practical size on the display screen of the computer. If the multiplier is 10, for example, the coordinates of vertex 13 are $(10,10,-10,10)$.

Dimensions seem to creep in everywhere as HYPERCUBE is written. A two-dimensional matrix, or array, called *vert* preserves the vertexes as they are initially defined. Since there are 16 vertexes with four coordinates each, *vert* is a 16-by-4 matrix of 64 numbers; *vert*(i,j) is the jth coordinate of the ith vertex. The program HYPERCUBE holds the matrix *vert* inviolate; *vert* is defined at the beginning of the program and its contents are then transferred to a second 16-by-4 matrix called *cube*. The matrix *cube* can be thought of as a working matrix; its contents are continually altered by the rotations carried out in the program.

HYPERCUBE is divided into three major sections following the initialization of *vert:* the selection of the desired rotation of the hypercube, the calculation of the coordinates of the rotated hypercube and the display of the result on the monitor. If the rotating object were three-dimensional, one could select the rotation by specifying the orientation of the axis of rotation and the angle of the rotation about the axis. For a rotating four-dimensional object, however, picking an axis of rotation does not determine a rotating plane: remember that there are two nonequivalent directions perpendicular to a given plane. On the other hand, even in four-dimensional hyperspace it remains true, as it does in ordinary space, that a rotation can affect just two dimensions at a time. If a three-dimensional object is rotated, two of its dimensions swing into each other while the third dimension remains fixed. Similarly, when a four-dimensional object is rotated, two dimensions change direction in the space while the other two remain fixed.

There are many ways a four-dimensional object can be rotated to a new position. It turns out, however, that any position can be reached by applying a sequence of rotations limited to motions within the planes defined by the coordinate axes of the surrounding four-dimensional space. There are four coordinate axes in a four-dimensional space, numbered, say, from 1 to 4, and there are six ways any two of them can be combined. Hence there are six planes within a four-dimensional space determined by the coordinate axes: plane 1–2, the plane determined by axes 1 and 2, plane 1–3, plane 1–4, plane 2–3, plane 2–4 and plane 3–4.

For each of the six planes there is a corresponding kind of rotation, which can be specified by a 4-by-4 square matrix of 16 numbers. The six rotation matrixes are named *rot12, rot13, rot14, rot23, rot24* and *rot34*. The user of the program must type in the name of the kind of matrix selected and the angle of rotation the matrix will generate. For example, typing "rot23" followed by "60" would cause a rotation of 60 degrees within the plane defined by the second and the third axes.

Suppose one wants to confine the rotation of the hypercube to the third and fourth dimensions, the most mysterious rotation of all. The rotation matrix *rot34* is applied. Its entries are 0's, 1's and three other numbers

distributed according to the following pattern:

$$\begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & -b & a \end{matrix}$$

The angle of the desired rotation in degrees is selected and stored in the variable *ang,* and the numbers *a* and *b* depend on *ang: a* is equal to *cos*(*ang*) and *b* is equal to *sin*(*ang*), where *cos* and *sin* are the trigonometric functions sine and cosine.

The rule for generating the other five rotation matrixes is simple. The *a*'s appear on the main diagonal of each matrix in positions that correspond to the dimensions affected by the rotation. The *b*'s appear at all the other intersections of rows and columns that correspond to the rotating dimensions. All other entries on the main diagonal are 1's, and the rest of the entries in the matrix are 0's. For example, *rot13* is the following matrix:

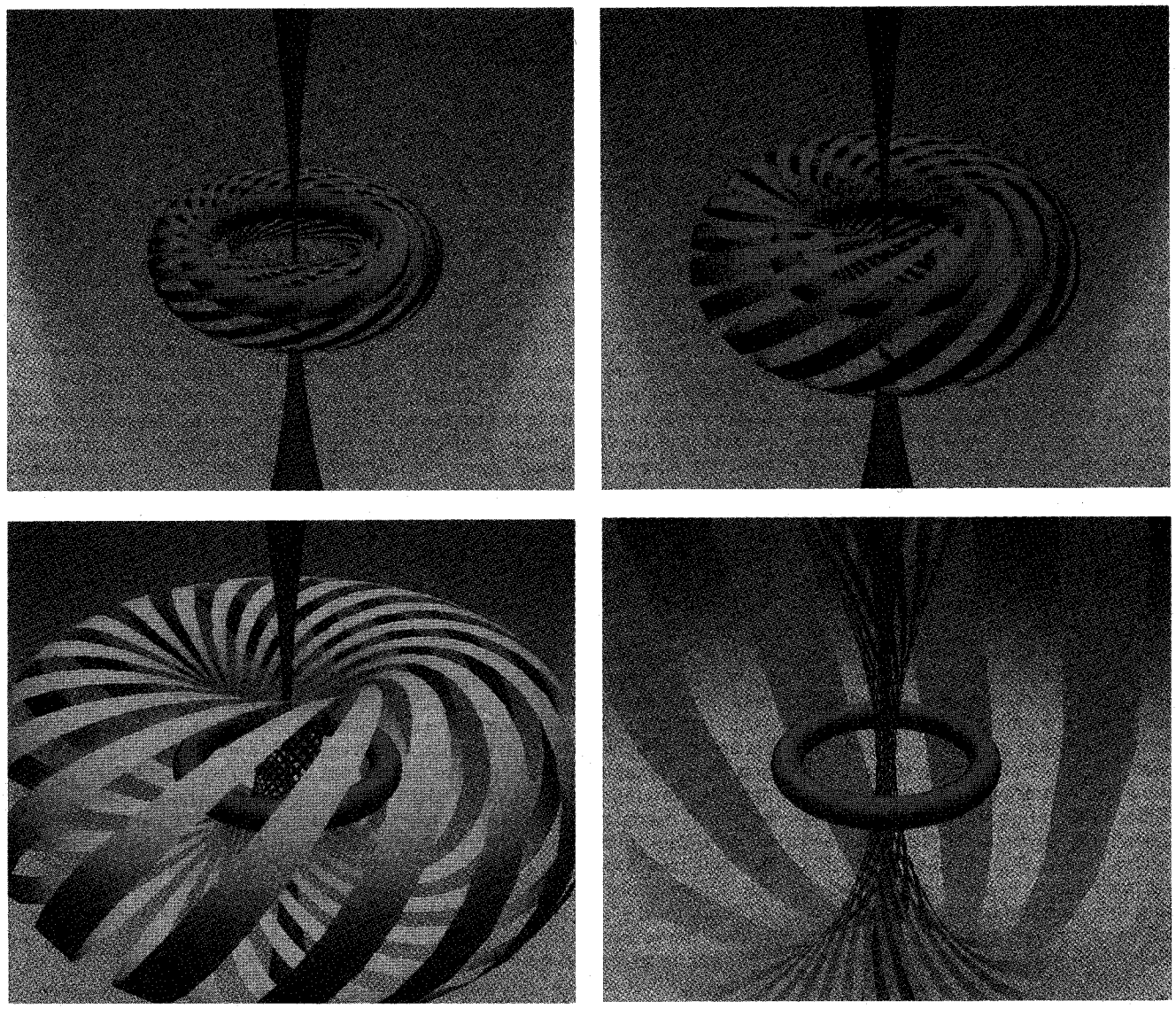
$$\begin{matrix} a & 0 & b & 0 \\ 0 & 1 & 0 & 0 \\ -b & 0 & a & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

When the desired rotation matrix has been selected, it is assigned by HYPERCUBE to a special matrix, *rote.* The assignment can be made conveniently by employing a double loop, the inner loop for the sequence of numbers across a row of the matrix and the outer loop for the sequence of rows.

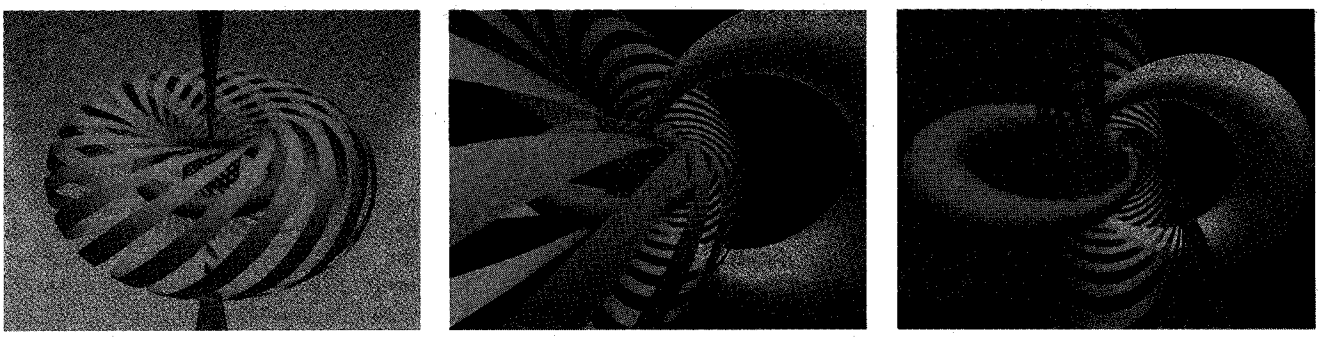
The calculation of the coordinates of the rotated hypercube is done by "multiplying" the matrix *cube* by the selected rotation matrix *rote.* The product of the two matrixes is stored temporarily in a third matrix called *temp,* and it gives the coordinates of the rotated hypercube. That product is found according to the rules of matrix multiplication, a standard operation on matrixes of numbers. It results from an orderly orgy of multiplications embedded in three nested loops.

Temp, like *vert* and *cube,* is a 16-by-4 matrix of 64 numbers: it gives four coordinates for each of the 16 vertexes of the rotated hypercube. Each number *temp*(*i,j*) in the matrix is designated by a pair of indexes *i* and *j.* For example, *temp*(13,3) is the third coordinate of the 13th vertex of the rotated hypercube. Its value is the sum of four products of numbers drawn in a precisely defined way from the matrix *cube* and from the matrix *rote* [*see illustration on page 23*]. The innermost loop in the program can therefore be indexed by the letter *k,* which runs from 1 to 4, and that loop returns the value of one entry of *temp.*

The *k* loop is then placed inside the



Sequence of nested toruses, analogous to the latitude lines on a sphere, projected from the hypersphere into three dimensions



Projected motion of toruses as the hypersphere is rotated, analogous to the projected motion of latitude lines during the rotation of a sphere

intermediate loop that has index *j*. The *j* loop computes all four coordinates of the *i*th vertex of the rotated hypercube according to the procedure I have just outlined; in other words, it fills in all four entries in the *i*th row of *temp*. Finally the *j* loop is placed inside the outermost loop, which has the index *i*. The *i* loop calculates all 16 rows of *temp*, and when it is completed, *temp* gives the coordinates of all the vertexes of the rotated hypercube, resplendent in its new position. In order to display it on the computer monitor one more double loop is needed that replaces the old position coordinates of the hypercube in *cube* with the newly calculated coordinates from *temp*.

A hypercube has four dimensions but a display screen has only two. It is therefore convenient to stipulate that the first two dimensions, or coordinates, of the hypercube correspond to the screen coordinates. The simplest method for dealing with the third and fourth dimensions of the hypercube is to ignore them. The display technique I shall describe does just that, but it can be enhanced—and the resulting object can be projected in nearly demonic complexity—by making both the third and fourth dimensions somewhat more apparent.

To display a skeletal version of the hypercube, the program need only display its edges. Since the hypercube has 32 edges, the display section of HYPERCUBE need only draw the appropriate lines between 32 pairs of vertexes. But in what order? There are almost infinitely many possibilities, and so the answer is perhaps a matter of aesthetic and personal choice. Nevertheless, it is hard to resist drawing the edges as an Euler trail, after the mathematician Leonhard Euler. A pencil can trace such a trail on paper without being lifted from the paper and without tracing any line more than once. Consecutive edges of the hypercube drawn as an Euler trail have a common vertex.

Round, through, up and down races the Euler trail as it is drawn through the vertexes. Here is one that strikes me as quite pretty, given by the numbered vertexes of the hypercube connected in the following sequence: 0, 1, 3, 2, 6, 14, 10, 8, 9, 11, 3, 7, 15, 14, 12, 13, 9, 1, 5, 7, 6, 4, 12, 8, 0, 4, 5, 13, 15, 11, 10, 2, 0. These vertexes are stored in an array called *trail* with index *i;* the *i*th vertex in the sequence of 33 vertexes is designated *trail*(*i*). For each value of *i* there are instructions for looking up the first and second coordinates of both *trail*(*i*) and *trail*($i + 1$). The line-drawing command in one's programming language must then be invoked to connect the two points. The lookup and the line drawing are embedded in a single loop with index *i,* which draws a line from each vertex in the sequence to the next.

Now for the visual (and psychological) complications. There are two standard methods for presenting the third dimension of the hypercube. The orthographic method simply ignores the third dimension, and all the vertexes are projected directly onto the flat surface of the display screen no matter how far they are behind it. In one-point perspective the vertexes are projected onto the screen as though they were shadows cast by a point source of light centered on the screen and some distance behind the hypercube. Viewing the shadows on the screen is equivalent to viewing the hypercube from behind, but visually it is indistinguishable from a front view.

To achieve the effect of one-point perspective in HYPERCUBE one assumes that the third coordinate of a vertex is equal to the distance between the vertex and the display screen, in the direction of the imaginary point source of light. By solving for the sides of proportional triangles the program determines a multiplier needed to convert the first two coordinates of a vertex into screen coordinates. For example, if the imaginary light source is 20 units behind the screen, a vertex at (5, −7, 11,8) can be projected onto the screen by multiplying each of the first two coordinates by 20 and dividing each result by 20 − 11, or 9.

I had dreaded including in the small space that remains a complete description of the process for creating stereoscopic images portraying the fourth dimension of the hypercube. There is a general technique for making stereoscopic images, and I hope to devote a future column to the subject. For the hypercube program, however, Banchoff and his colleagues have adopted a much simpler method. For each position of the hypercube make a new pair of images by applying *rot14* through an angle of three degrees in one direction and three degrees in the other. Dimension 1 is the direction parallel to the horizontal alignment of the viewer's eyes, and dimension 4 is the target of the exercise. The two small rotations nicely approximate the views of the hypercube from the eyes of the viewer: merely imagine the two lines of sight converging near the center of the hypercube at an angle of six degrees.

Readers who want to capture the thrills of 3D movies can make stereoscopic viewing glasses out of red and blue cellophane. In this case HYPERCUBE is run twice, once for each small rotation. The result of the first rotation is colored blue by the program and the result of the second is colored red. Readers need not be concerned about which is which if the eyeglasses are made to be invertible.

Personally I prefer not to struggle with cellophane, and I have learned to fuse stereoscopic pairs by shear force of will. The technique requires that the two rotated images be reduced in size and then translated to horizontally adjacent and nonoverlapping positions on the screen. They should be the same color, and so a monochrome screen is sufficient, and they should be no farther apart than the distance between the viewer's eyes. Do not stare at the images; look instead at some point between them and infinitely far beyond. The two hypercubes will appear to drift and jiggle toward each other like a pair of shy lovers until they fuse.

Even if the third and fourth dimensions get no special treatment, HYPER-

CUBE can generate images much like the ones shown in Banchoff's graphic sequence. With successive rotations through small angles in the third and fourth dimensions, readers may see the two crude toruses balloon, pinch off and regenerate much like their smoother cousins in the illustration of the rotating hypersphere.

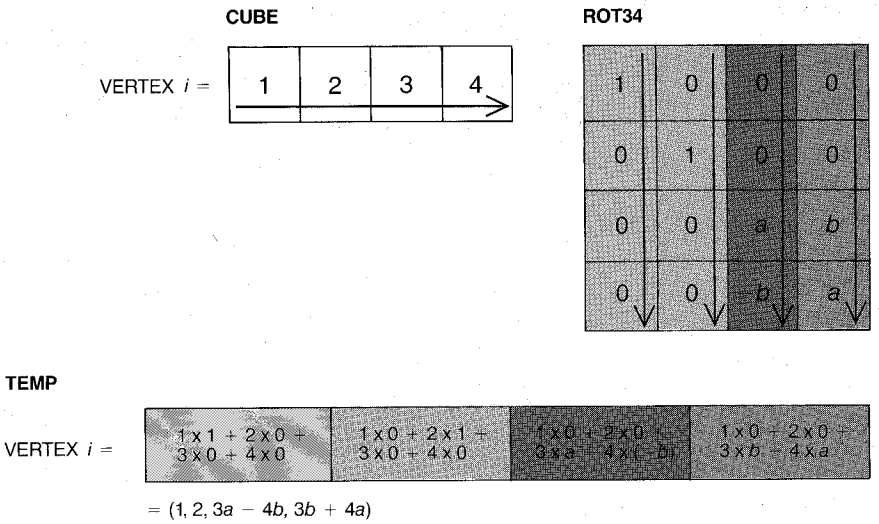
The program HYPERCUBE had obviously caused the disappearance of my friend Magi. The happy ending to his four-dimensional dementia came with a telephone call. Not surprisingly, he spoke of wondrous things. "You probably think I'm crazy," he said. (The phrase is always a sure tip-off.) "I've just been floating around in the fourth dimension. I saw a cross section of my house sweep by. Then I moved in close and tickled my cat's kidneys..."

I will spare the reader any further details of the conversation. Suffice it that I persuaded Magi to run HYPERCUBE no more and to keep further explorations entirely on the intellectual plane. He has followed my advice, he says, and now he professes to have made many marvelous discoveries through his artificially amplified insight. For example, he has come up with two posers that seem worth passing along.

Think for a moment about the following sequence of objects: a unit line, a unit square, a unit cube and so on. The nth member of the sequence is the n-dimensional analogue of the cube. Now try two mental experiments on the objects: draw the diagonal to the n-dimensional cube and inscribe an n-dimensional sphere within the n-dimensional cube. The diagonal stretches from one corner to the opposite one; what happens to its length as the number n becomes progressively larger? What happens to the volume of the n-dimensional sphere, again as n becomes progressively larger? Magi's answers seem hardly sane; I shall give them in next month's column.

In my January column I described two programs, CLUSTER and SUPERCLUSTER, that simulate the evolution of a star cluster. It heartens me to think that in at least a few thousand homes they have led to a new form of entertainment, temporarily edging out television. No doubt some of these armchair universes are unfolding as they should, but others may be developing problems. The fault is not in our stars but in ourselves.

Brian Davis of Ann Arbor, Mich., and Peter Fortescue of La Jolla, Calif., had trouble with the acceleration equations in SUPERCLUSTER. The difficulties are fixed, I believe, by replacing the force f in the equation on page 13 with the acceleration a due to the force. To get a divide f by the mass of the attracted star. Andrew M. Odlyzko of AT&T Bell Laboratories pointed out that the position coordinates in the table on page 15 are in multiples of 1,000 astronomical units (A.U.), not single A.U.'s. Our own universe will now unfold correctly.



How a vertex of the hypercube is rotated from the third dimension into the fourth

When one views the live action of CLUSTER or SUPERCLUSTER on a display monitor, it is sometimes hard to tell which stars are in the foreground and which are farther back. Albert C. English of Delray Beach, Fla., and Peter Stearns of Lodi, Calif., have written special display programs that generate two images of clusters side by side, one as seen by the right eye and one as seen by the left. Readers able to manage the tricks of stereoscopic display will be able to view the clusters as they view the hypercube: in breathtaking depth.

Several readers had already written programs similar to SUPERCLUSTER, but they had applied the programs to our own solar system. The same application would also be feasible with SUPERCLUSTER. Those with the gumption can look up the mass, position and velocity of the 10 major bodies in the solar system for some reference time. One can then arrange to view the evolution of the entire system from above: wait a few minutes for the year 2000. Geoffrey L. Phillips of St. Louis, Mo., wrote a simulation for the earth-moon system that includes a small, massless space vehicle. Launching it from the earth in such a way that it begins to orbit the moon is no easy feat. Advanced practitioners might try launching a Voyager spacecraft on a grand tour of the gas giants that ends as it leaves the solar system.

William A. Hoff of Champaign, Ill., computed the time increment for the simulation dynamically by setting a variable called *dvmax* at the beginning of the program. In the course of the calculations of stellar motion the program always finds the maximum acceleration *amax* of a star. The next time increment is *dvmax* divided by *amax*. The technique prevents any velocity from exceeding *dvmax*.

In last month's column I gave an I.Q. minitest and posed several questions about numerical sequences. The first problem on the minitest is a good example of the ambiguity typically found in such problems. The problem was to complete the sequence 3, 7, 16, 35, ... Each term minus twice the preceding term gives the sequence 1, 2, 3, the second row of a pyramid. By this reasoning the missing term must be twice 35 plus 4, or 74. On the other hand, if a simple difference pyramid is constructed with three rows, the third row gives the sequence 5, 10, and it seems reasonable to complete the sequence with 15. The missing term must then be 69, but the programs described in the column would have missed this answer. The other answers to the test: *H* is the missing letter; the missing word is "up"; the odd man out is "identity"; the unscrambled name of the town not in Italy is Madrid, and the correct visual analogy is number 2.

The two numerical sequences on page 12 are completed by 350 and 22 respectively. The first sequence on page 10 can be solved by applying a generalized difference rule, with k equal to 3, and then a generalized quotient rule; the missing term is 324. The second sequence ought to defeat all but the most patient puzzle solvers who did not try to write SE Q. It can be solved by two quotient rules; the value of k in the first rule is 5. The missing term is $-65{,}551$.