AUTHOR	Thomas, David A., Ed.
TITLE	Scientific Visualization in Mathematics and Science Teaching.
INSTITUTION	Association for the Advancement of Computing in Education, Charlottesville, VA.
REPORT NO	ISBN-1-880094-09-6
PUB DATE	95
NOTE	293p.
AVAILABLE FROM	AACE, P.O. Box 2966, Charlottesville, VA 22902.
PUB TYPE	Books (010) -- Guides - Classroom Use - Teaching Guides (For Teacher) (052) -- Collected Works - General (020)
EDRS PRICE	MF01/PC12 Plus Postage.
DESCRIPTORS	Calculators; Computer Software; *Computer Uses in Education; *Educational Technology; Elementary Secondary Education; Graphs; *Mathematics Instruction; *Science Instruction; *Technology Education; *Visualization
IDENTIFIERS	Graphing Utilities

ABSTRACT

Science and mathematics educators are expected to use existing educational technologies effectively and to keep informed about emerging technologies that might become important educational tools in the not-so-distant future. This monograph offers some help in that regard by highlighting a number of existing and emerging educational technologies. Chapters are: (1) "The Power of Visualization: The Impact of Graphing Technology on the Secondary Mathematics Curriculum," L. E. Yunker; (2) "Using Graphing Calculators to Teach High School Mathematics," L. Kaber & K. Longhart; (3) "Advanced Technologies as Educational Tools in Science: Concepts, Applications, and Issues," D. D. Kumar, P. J. Smith, S. L. Helgeson, & A. L. White; (4) "Videodisc Technology: Applications for Science Teaching," D. R. Lavoie; (5) "Computer Visualization: New Window on Mathematics," D. A. Thomas & M. Mitchell; (6) "Visualizing Computer Science," R. J. Ross; (7) "Getting Started With Supercomputing: An Approach for High School Students," D. W. Hyatt; (8) "Scientific Visualization in Chemistry, Better Living through Chemistry, Better Chemistry through Pictures: Scientific Visualization for Secondary Chemistry Students," R. R. Gotwals, Jr.; (9) "The National Education Supercomputer Program," R. Enderton & B. Lindow; (10) "New Mexico High School Supercomputing Challenge," M. S. Foster; (11) "Sharing Multiple Complementary Representations in the Teaching of Science," N. H. Sabelli & I. S. Livshits; (12) "Education and Collaboration in an Evolving Digital Culture," D. J. Cox; (13) "The Hypergraphics Honors Seminar at Illinois," G. K. Francis; and (14) "A Syllabus For Scientific Visualization," A. Pang. (MKR)

Scientific Visualization in Mathematics and Science Teaching

edited by David A. Thomas

Association for the Advancement of Computing in Education

Chapter 13

The Hypergraphics Honors Seminar at Illinois

George K. Francis

One edition of Math 198, the Freshman Honors Seminar in Mathematics at the University of Illinois at Urbana-Champaign, is an intensive introduction to real time interactive geometrical programming. Its name, "Hypergraphics," connects to David Brisson's (1978) proposed synthesis of art and mathematics for the purpose of revealing the mysteries of space beyond the confines of our 3-dimensional perception (Banchoff, 1990).

The course is designed for novices, but experienced programmers are welcome, provided they contract for an individual study project commensurate with their skills. Most beginners also reach a level of competence by the middle of the course to complete a project of their own. Students work on Apple IIgs and Silicon Graphics Iris computers. They program in BASIC, Forth, and C. Of course, these languages are augmented by graphics packages. For the first, &-GRAFIX is a machine language extension of Applesoft BASIC which was written by students in the UIMATH.APPLE Lab over the past four years (Sandvig, 1990). The ISYS Forth compiler is the product of a local software engineer (Illyes, 1988). It was developed, to a large extent, with the needs of the Apple Lab in mind. The graphics library on the Iris, known as gl, is such an effective resource that it is possible to learn enough basic C to write a respectable real-time interactive computer animation project during just one semester.

The students in this elective course are generally members of the University of Illinois Campus Honors Program, which selects 500 bright students from a population of 27,000 undergraduates. Thus, the members of my classroom compare well with the students taking similar courses at private universities. For example, a somewhat similar course at Princeton was taught and reported on by Conway, Doyle, and Thurston (1991).

In this chapter I shall describe my course in some technical detail so that the reader may not only profit from my experience but may weigh the basis of my opinions regarding hightech education. Respecting the principle that a mathematics paper should always contain "something old, something new, something borrowed, and something true," I include the complete, annotated, 250-line source-code for illiSnail, a real-time interactive computer animation (RTICA) which my students use, study, and modify on the Iris 4D/25TG computers in the Renaissance Experimental Laboratory (REL) of the National Center for Supercomputing Applications (NCSA) of the University of Illinois at Urbana-Champaign (UIUC).

Portrait of an Honors Student

Let me begin, by way of an anecdotal documentation, by sketching the activities of a recent, not atypical student. Pablo was technically a freshman but came to college with an excellent preparation. Together with his prodigious talent, this allowed him to compete with the juniors and seniors for first place in the class of fifteen. Pablo's first "essay" was a whimsical animation in &-GRAFIX of swimming fish blowing bubbles. His second was the best of only three solutions for an assignment to write a concise, recursive program in Forth that draws a Sierpinski Triangle. The class studies a series of very simple programs which are small enough to fit into one's mental "hip-pocket" and can be played on every computer. Pablo's "commentary" on the one for the lesson on Logistic Chaos was to modify it and so draw the well known bifurcation diagram of this famous dynamical system.

Two years ago, my teaching assistant, Glenn Chappell, had brought with him a superb piece of pedagogical software. His program is written in BASIC and 65816 machine language. It is, in fact, an interpreter for a tiny language, CSL, which Glenn invented for simulating the cellular automata popularized by Kee Dewdney in the pages of the Scientific American. Pablo completed the assigned experiments with CSL, comparing them intelligently to their older Forth versions. In my graduate Geometrical Graphics course on the Irises, we develop RTICAs which have a feature for recording the user's activity in a script which can then be played back automatically. Although such student projects are often user-hostile, they deal with interesting topics that appeal to the honors students. The "movie-making" feature makes it possible to incorporate them into lessons for Math 198. Pablo's cohort used the Snailhunt RTICA to explore a certain Möbius band located on the 3-dimensional hypersphere in 4-space (Francis, 1990). (This RTICA is discussed in detail in a later section of this chapter.) Pablo's movie and brief documentation showed an unusual level of curiosity and good mathematical free association. He manipulated the program to produce fanciful shapes reminiscent of public art on the Daley Plaza in Chicago. Working out the answers to the conjectures he made could have become his semester project. But he chose a much more ambitious one: to implement Carter Bay's 3-dimensional version of Conway's Game of Life as described in Dewdney's (1988) popular book, *The Armchair Universe*.

Pablo's project was the most notable achievement in that class. He wrote his own RTICA of a 3-dimensional Life automaton. With only a high school AP-Pascal course behind him, he learned and mastered C/Unix/gl on the Iris and wrote a program few of the students in my graduate course could write. The more ambitious student projects often receive extensive help from the staff. Pablo managed all that pretty much on his own.

Pedagogical Notes

Although Math 198 is similar to Thurston's Math 199 at Princeton (Conway, Doyle, & Thurston, 1991), there are enough differences to warrant a closer comparison of these two courses. In fact, my Math 198 more closely resembles the two-week intensive course, "Geometry and the Imagination," taught by John Conway, Peter Doyle, Jane Gilman, and William Thurston at the Geometry Center, summer 1991. Their course was followed by a ten-week research and training program for some of the high school and college students who took the short course. This permitted the completion of substantial projects, some of which included computer visualization (Marden, 1991).

Math 198 at Illinois is a 3 credit course, though all, including the instructor and the teaching assistant, spend far more time on this course than is customary for an undergraduate course with 3 assigned contact hours per week. Quality education at a large, state supported university requires such dedication and extra effort.

The Project

Each student has a project to complete. The project presentation has an oral component. This takes place during the final week of the semester, seminar-style and with cake and soft-drink refreshments. Typically, the student explains what the program is about and how to operate it. An abbreviated version of hands-on demonstration earlier in the course is followed (sometimes preceded) by a 10-20 minute lecture at the white-board. The demonstration and segments of the mini-lecture are videotaped. The demo taping is occasionally staged and repeated to improve its quality. However, little of the mini-lecture and none of the discussion is taped, to encourage both presenter and audience to express themselves freely. Our unexpected experience is that, unlike the author's generation, members of today's TV generation show practically no camera shyness or stage fright. Also, they know how the videotapes will be used: They are shown at the Honors House on special occasions—for instance, to visiting parents of prospective participants. Older students, who are helping out during these orientation sessions, are delighted to watch their colleagues "perform" their final for Math 198. Last year's tapes are also shown in class to explain to the students what is expected of them.

The Grade

A second major difference from the Princeton course is how the final grade is determined. A course like Math 198 is unsuitable for either a pass/fail or a standard grading scheme. The temptation to procrastinate or merely audit such a course is too great, especially for the freshman and sophomore. On the other hand, inhomogeneity of preparation, experience, and motivation precludes competitive examinations and comparative evaluation. Instead, each student receives a "contract-grade" according to the following announced and periodically repeated formula. Once the student has completed the basic assignments and tutorials, he has earned a "gentleman's C" for the course. All that needs to be done for an A is to complete the semester project. A student whose project is well started but incomplete receives a B. This can be changed to an A once the missing work is submitted.

I have never had to give the gentleman's C, and the drop rate is about 1 to 2 students per class of 15-20 students. Of course, everyone is individually evaluated to facilitate writing recommendations, which many students in the Campus Honors Program eventually request. Also, generous praise and encouragement are offered privately as needed.

The plan for the project is negotiated with the instructor. The pre-proposa' proposal, progress report, and (rarely) preliminary draft are carefully monitored and commented on. The principle guiding the choice and ambitiousness of the project is for the student to apply newly acquired mathematical and computing skills. The project is complete insofar as the program works, has been publicly presented, and the written documentation is acceptable. The latter includes a one-page operating instruction, a careful specification of the hardware configuration, a narrative essay with bibliography on the mathematics, a hand-annotated printout of the program, and a technical note on computational difficulties that were solved or remain to be solved by the next student building on the present project. The class materials are to a large extent the work of previous students, often with emphasis on their shortcomings. Thus some of the best projects each year are corrected continuations and extensions of previous projects. The project is treated as a contractual agreement to produce a certain piece of work by a dea lline.

Manipulatives

The Princeton course makes excellent use of geometrical artifacts, including mirrors and construction kits. We make no extensive use of physical models or experiments, except on the computers and, to a much lesser extent, with video equipment. This is entirely the consequence of our severely limited physical facilities at Illinois.

The Journal

In the Princeton course the student keeps a bound journal into which assignments, class notes, and other appropriate items are either pasted (if complet d on a word-processor) or entered by hand. While I have always encouraged my students to keep a 3-ring notebook for handouts, clean copies of their class notes, homework, and tests, I had assigned the keeping of an "intellectual journal" only last year after learning about its use in the Princeton course. In the fall we assigned journal keeping to some 100 students, who were preparing to become elementary school teachers, in the lecture/lab course on "Experimental Arithmetic" (Francis, 1992) which also has a programming project component. The logistics of this multisection course, the immaturity of the students, and our neglect to collect the journals regularly and monitor their content, led to the failure of this first experiment. On examination at the end of the semester, 90 percent of the journals were nothing more than daily diaries which recorded the trials and tribulations of an unfamiliar and difficult lab course. Except for the psychological benefits of catharsis for the students and scathing criticism for the instructional staff, these journals were a waste of resources. Quite the contrary was the case for the journals kept by the Math 198 students the following spring. I explained from the start the Princeton model of the journal, including its pros and cons, and invited my students to experiment with their own format. The only requirement was that the hard-bound journals (no fair tearing out pages) had ample margins and blank even-pages for comments and corrections. The journals were collected and read two or three times during the semester and commented on, copiously in some cases. A written exchange developed between student and teacher in a few journals, an almost Victorian dialogue of glosses. On the first round, about a third of the students had nearly empty journals or started writing a diary instead. Most of these had mended their way by the time of the second reading. Some of the journals were astoundingly good right from the start. Of course, one must not forget the exceptionally verbal, high honors cohort taking Math 198.

The journals proved to have unexpected benefits also for the professor, not the least of which was the fact that a 10x15 inch, black, hardbound journal is difficult to lose or misplace. Secondly, it provided an opportunity for individual instruction. Erroneous or incomplete journal entries prompted me to write a mini-lesson right into the journal. This information and elaboration beyond the class instruction went directly to the interested student, without wasting other people's time. Finally, cumulative progress could be monitored over the semester without having to decipher the numerically encoded entries of a gradebook. I even got a second chance to correct my own misleading "corrections" on rereading them at a later time.

Facilities

Math 198 has been taught for three years in succession in an essentially identical fashion. This particular configuration of hardware, software, students, instructors and content was based on the experience with different configurations of related courses taught under the auspices of the UIMATH.APPLE Lab since 1983. It was therefore uniquely suited to its academic environment. Without similar experimentation and fine tuning it is unlikely that such a course can be successfully taught in another environment. Nevertheless, we hope that a careful description of its configuration below will be of use also to someone planning such a course under their own circumstances.

The Student Cohort

Math 198 is for students in the Campus Honors Program, but others with comparable credentials may also take the course. The University of Illinois Campus Honors Program admits, on a competitive basis, circa 500 students from an undergraduate student body of ca 27,000. Or, approximately 100 students from over 600 applicants join the program each fall. (The difference is made up of students who join the program at a later time.) The quality, motivation, and preparation of these students therefore is not dissimilar from those for whom the Princeton course was designed. Math 198 is officially an elective for freshmen. Its curriculum is the instructor's choice. Since the course under the present discussion, and its predecessors, is the only version of Math 198 which treats programming graphics computers, no ambiguity results from referring to it simply as Math 198.

In a class of 15-20 students, a third tend to be freshmen, the others range over all three remaining years. Typically, five are novices with respect to computer programming, four are so proficient that they volunteer to help train the novices, and the remainder can program in at least one language on at least one computer. Occasionally, a freshman is among the computer proficient, but juniors or seniors who are computer novices are discouraged from taking the course. The students are usually science or engineering majors, though there are always one or two from the humanities or the fine arts. There have never been any students from agriculture, commerce, or the social sciences. Two or three students are women.

Not surprisingly, all students have had calculus, trigonometry, and analytic geometry, at least in high school. The majority are concurrently enrolled in a middle level course in differential equations, linear algebra, geometry, physics, or computer science, so that their projects are almost invariably related to topics studied in these related courses.

Instruction

The professor and his teaching assistant both meet the students for an average of five hours a week in lecture and lab. The course carries 3 credits, but the schedule is so arranged that the lab component is contiguous with the formal instruction. The topics are strongly modularized so that they can be variously selected and arranged to maximize their relevancy to the students' current interest and capacity. A 5-10 minute introduction is followed by a 15-30 minute hands-on tutorial on the computer. This can be in the form of operating, analyzing, modifying, and experimenting with a pocket program (see below) on the Apple IIgs. On the Iris it usually means operating an RTICA to perform a particular set of experiments and recording the outcomes (observations) on a work sheet or in a notebook.

Hardware

The computers used at the high and the low end are standardized. Each student has access to a fully networked Personal Iris 4D/25TG in the Renaissance Experimental Laboratory (REL) of the National Center for Supercomputing Applications (NCSA). These Irises have 24 bit RGB framebuffers with a 24 bit hardware Z-buffer and graphics accelerators. This configuration suffices for real-time interaction with fully animated, rendered and lighted scenes containing 1 to 5 separate graphics objects.

Half the class sessions take place in REL, the other half in the Apple Lab. There the students use 20 independent, 9-year-old Apple IIe computers which have been upgraded to be a IIgs with (effectively) 4-bit RGB color or 4-bit gray- scale Z-buffer.

In addition, many students own popular micro computers, and the University provides access to Macintosh and IBM sites. While these sites are very useful for word processing, they are uniformly useless for Math 198. Students with computers in their rooms are permitted (weaker students are encouraged) to structure their projects in a way that maximizes their using personal equipment for graphics and mathematical algorithms. Of course, it remains their responsibility make sure that the project can be demonstrated to the entire class on some computer in a publicly accessible location.

Software

On the Iris we use a proprietary but close variant of the Unix operating system, the MIPS C-compiler, and the standard graphics library supplied by Silicon Graphics. All students use real-time interactive computer animations (RTICA) written by and for students in the graduate course I teach in REL, or as projects by previous undergraduates in Math 198. (An example of such a program is listed and discussed in the last section of this chapter.)Those with sufficient experience (beginning with Pascal proficiency) are encouraged to master the rudiments of an editor (vi, emacs, or jot) to try their hand at programming. For novices in C, Unix, and gl, there is a graduated collection of programs to study and modify. For the more advanced student (a few of whom already have some experience programming an Iris) there are projects by previous students to study, improve, and frequently rebuild from scratch.

For the Apple IIgs we have a considerable accumulation of student and instructor developed software, most of it in 65816 machine language, which extends the native Applesoft BASIC resident in the IIgs ROM. The most popular of these is Sandvig's gs.amper.new. This contains a large number of graphics primitives which fit into an ordinary Applesoft program much the same way as calls to the Iris graphics library fit into a C-program. Much more versatile is Robert Illyes' (1988) Forth compiler, ISYSFORTH/GS, which is specifically adapted and optimized for Apple IIgs graphics. Both languages were developed with the needs of the Apple Lab in mind, and both permit us to use the Apple IIgs as a simple, understandable "toy-version" of a "grown-up" graphics computer like the Iris.

Applesoft BASIC extended with &-GRAFIX is commonly used for projects by people who have never programmed mathematics before. Forth is mostly used to tease the last ounce of performance out of our Apples, and to introduce certain concepts such as cellular automata, recursion and object oriented programming in an analytical fashion. In Forth, the student can reach every corner of the computer and even mess with the machine-stack, something which it is not recommended they do on Macs, IBMs and Irises.

On their private machines, the students generally program in TURBOPascal and a variety of C-compilers. A popular project for someone who just acquired an brand new 386 or 486 box with a good but user-hostile graphics card is to write their own high level graphics library, sometimes with machine-language routines for certain primitives (lines, fills, rotations, for example). It should be emphasized here that in the instructional part of the course, geometry is assisted by computer graphics. But, in the projects, it is usually the other way around.

Content

We have developed a number of different techniques of introducing various topics into the course. Here we shall report on only two of these, the pocket program for any personal computer, and an RTICA on the Iris graphics workstation.

Pocket Graphics Programs

These are simple programs that do non-simple things. They also contain working examples of useful techniques. A pocket program expresses one idea in as economical a way as possible within a given language. It is a program one carries around in one's mental pocket, to be produced on demand, and implemented on whatever computer is at hand. It is primarily a didactic instrument. The dozen or so pocket program in Math 198 first of all serve to introduce the student to the major themes of the course. Together, they provide the skeleton on which to hang the concept of computer geometry. Finally, they focus the effort of learning a new language or mastering a new graphics card by way of something concrete to translate or implement. Let me illustrate what I mean by describing two of the pocket programs on adjacent topics.

The Sierpinski Gasket

This is a planar Cantor set obtained by recursively removing the central quarter of a triangle. Connecting the midpoints of the sides of a triangle decomposes the area into four congruent triangles similar to the parent triangle. Inflicting this excision to each of the three daughter triangles, and their offspring in turn, leaves a figure whose fractal dimension is easily demonstrated. Self similarity shows that doubling the side-size of the Gasket triples the measurable content of the figure. If, by analogy to lines, squares, cubes and tesseracts, "dimension" is defined to be that power of 2 the content of a figure must be multiplied by in order for it to equal the content of a similar figure obtained by doubling its linear scale, then

$$1 < d = \log_2(3) < 2$$

Here is a curious way of generating a Sierpinski Gasket due to Michael Barnsley. For the traditional description, see Banchoff (1990, p. 32). Barnsley (1988, p. 179) invented it to illustrate an Iterated Function System. It can be expressed in eight lines of classical BASIC.

```
10 REM SIERPINSKI
20 X=100:Y=50
30 CLS : REM INIT GRAPHICS
40 FOR I= 0 TO 2:READ XF(I),YF(I):NEXT
50 DATA 0,32, 100,0, 100,63
60 J=INT(3*RND(1))
70 X=(X+XF(J))/2:Y=(Y+YF(J))/2
80 PSET (X,Y) : REM PLOT POINT
90 GOTO 60
```

This program moves a point to a new position which is half way from its current position towards a randomly chosen one of three fixed points. This stochastic dynamical system has a fractal attractor.

This program uses only two graphics primitives: initializing graphics on the Radio Shack TRS-80 Model 100 with the clear-screen command on line 30, and plotting the point (X,Y) on line 80. It is appropriate that the vertices of the triangle be given as an array, line 40, because this simplifies how the choice, line 60, is made each time through the loop. Since line 40 can be written as

```
40 READ XF(0), YF(0), XF(1), YF(1), XF(2), YF(2)
```

it is also a gentle introduction to a counted loop. The compact READ/DATA format is convenient here to express the triangle as a geometrical object in terms of its display-list.

Even at this basic level the program above invites experimentation. Replacing the fixed initial position, line 20, by a user INPUT statement, and later by a random choice, underscores the fact that the gasket is a universal attractor. Altering the number of vertices, lines 40-50, and the proportion away from 1/2 in line 70, leads to some remarkable discoveries. For example, last spring Monica Plisch discovered variants of this iterated function system, whose attractors were other well known fractal figures, such as Koch's Snowflake. A small generalization, using a color computer, leads to a surprising variation discovered spontaneously by many students. It makes an instructive difference whether one sets the color to number J on line 75 or on line 85. The former identifies the three sub-gaskets. The latter, recording the color of the previous choice, gives a visual clue of how iterated function systems work in the first place. Barnsley's maple leaf (1988, p.108) is not too far away at this stage of the tutorial.

On the other hand, on the Iris it is an easy generalization to 3- and 4- dimensional iterated function systems, which provides a nice motivation to a more formal treatment of a semigroup of contracting affine transformations.

Pocket Dynamical Systems

The exact number and identity of pocket programs used in any given instance of Math 198 is not fixed. Only their use and purpose remains the same from year to year. A large collection of concepts, most of them new to the student, are more effectively taught by way of examples than in a systematic syllabus of abstractions. Training in the vocabulary of comparative anatomy is not needed to enjoy a visit to the zoo. One needs an intelligent arrangement of live examples from each of the major zoological classes, together with brief descriptions that do not neglect the homologies between different species.

More than half of the students in Math 198 major in the traditional (hard) sciences. Thus the notion of a dynamical system is one of the themes to be developed thoroughly. With Hirsch and Smale (1974, p.159) we favor this definition: "A dynamical system is a way of describing the passage in time of all points of a given space ...." The Sierpinski Gasket iterated function system is a kind of dynamical system, albeit a stochastic one. A dynamical system moves a point to a new position according to a rule which depends only on the coordinates of the point being moved. In the present case there are three rules to choose from, and the program uses a generator of pseudo-random numbers to choose which of the three rules to follow each time. When there is only one rule, one says that the dynamical system is deterministic. One calls the stream of fractions returned by successive calls to RND(1) "pseudo-random" because they only seem random to us because we are ignorant of the algorithm that produces them. Of course, this algorithm is designed to mimic a true random number generator as well as the programmer can manage.

The Sierpinski Gasket is the attractor of this dynamical system. Informally speaking, this means that it is a set of points towards which each trajectory (or orbit) tends. The succession of positions taken by a point under the influence of a dynamical system is called the orbit or trajectory of its original initial position. The attractor should also be an invariant set, that is, the trajectory of a point in the set never leaves it. Here is a more traditional example of a dynamical system.

The Lorenz Mask

The next pocket program is also a dynamical system insofar as a rule inside an eternal loop (lines 60,70, and 80) moves the point (X,Y,Z) initially set on line 10, along a trajectory. This time, however, the dynamical system is frankly deterministic; there is no call to a pseudo-random number generator because there is only one rule to choose from.

```
 5 REM LORENZ
10 X=0.9:Y=0.1:Z=0:CLS
20 READ XO,YO,UX,UY,E,N,D
25 DATA 120,32,1,-1,.05,-50,.02
30 XR=XO-N : XL=XO+N
40 YP=YO+Y*UY:REM LOOP HERE
50 PSET(XL+(X-E*Z)*UX,YP)
55 PSET(XR+(X+E*Z)*UY,YP)
60 X1=X+10*(Y-X)*D
70 Y1=Y+(28*X-X*Z-Y)*D
80 Z1=Z+(X*Y-8*Z/3)*D
90 X=X1:Y=Y1:Z=Z1:GOTO 40
```

It is a 3-dimensional dynamical system because the point being moved has 3 components. This raises the problem of how to represent 3-dimensional data in the two dimensions of a picture. On a slow computer, with a coarse-grained, monochrome video display, the best way to do this is by means of stereo-pairs. On faster machines with more advanced graphics primitives there are other ways of achieving the illusion of 3-dimensional plasticity. Here is the list in the order of presentation in Math 198: perspective, depth-cueing, motion, z-buffering, shading.

Stereo-pairs are viewed with the help of devices which insure that your right eye sees one image, while your left eye sees an image of the same scene from a slightly different angle. Our visual system is very forgiving. It is not necessary to compute these two views very accurately, which would slow things down even more. Here we use a small shear, lines 50 and 55, to approximate binocular vision. A shear is a distortion which moves a rectangle to a parallelogram without changing its base or altitude. Think of a stack of playing cards pushed uniformly to one side.

In the absence of helpful optical devices it is easier to cross your eyes. So, shift the right-eyed view to the left, and vice versa. Cross your eyes by focusing at an object, e.g. pencil tip, roughly half way from your nose to the screen. Wait until you see three rather than four fuzzy images. Then wait until the middle one comes into sharp focus. On the printed page, the images are smaller and closer together. Here the view for the right eye is on the right. These can be viewed unaided by focusing your eyes at infinity (not crossed). One way of achieving this is to place your nose right up to the image until your eyes are relaxed (unconverged, unfocused). Then move the page back slowly until the fused, 3-D image jumps into focus. To reverse right with left in the program, change the sign of the nose offset, N, or the eye-shear fraction, E, but not both.

The shape you see developing is called the Lorenz Mask and it is a very popular example of a strange attractor. Strictly 2-dimensional dynamical systems do not need the complication of stereo-viewing. On the other hand, they don't have strange attractors. The Lorenz is also a favorite character in Nonlinear Mechanics because the rule that reads the velocity at a point is not a linear function of the coordinates of the point.

The iteration loop begins at line 40. This program uses both world and screen coordinates. On line 40, the vertical screen coordinate, YP is obtained by adding the fraction Y of the vertical unit UY to the vertical origin YO. This interprets the world coordinate Y as a fraction (proper and improper) of the fixed vertical displacement. It is an example of the axonometric projection from 3 to 2 space. Such a projection may be described informally as follows. Draw three line segments in the picture plane, the x,y,z-axes, from a common point, the origin. The axonometric image of a point (a,b,c) in 3-space is located at the end of a path that begins at the origin O, moves along the x-axis to point A for which the segment OA has the ratio a : 1 to the axis, then moves parallel to the y-axis a ratio b : 1, then a ratio c : 1 parallel to the z-axis.

In lines 50,55, the horizontal displacement from the left-origin, respectively right-origin, is computed. Starting from the true displacement, X, as seen by the cyclopean eye (in the middle of your forehead), which is the same for both eyes at the point you are looking at, it becomes progressively greater as the point recedes into the background, ie., as the z-coordinate becomes greater.

In lines 60-80, the world point (X,Y,Z) is moved the small fraction D along a displacement, the velocity vector, which itself depends on the current position. The reason this simplest of all numerical integration techniques is perfectly adequate here is that the dynamical system has a strong attractor. Even if at each step the computed point moves to another, nearby trajectory, it will converge to the attractor anyway.

The Third Dimension

Effective management of the depth illusion is a major theme in Math 198 since one aspect of the course is to "perceive" 4-dimensional reality in its 3-dimensional "shadows." When only one method is used to indicate depth, any momentary ambiguity spoils the illusion, blinking while looking at a Necker cube, for example. So a second method, depth-cueing for example, is good insurance. This means that points further back are drawn more dimly than those in front. The Apple IIgs has 4-bit color pixels; that is, a pixel may be assigned one of 16 shades of gray proportional to the distance of the point from the viewer. Cary Sandvig's graphics package has a number of pixel-operations built into it. A pixel-operation is simply the ability of storing the result of a logical operation between the color number about to be assigned to a pixel and the number that is already there. Standard point plotters just overwrite the old pixel. The pixel function that replaces two numbers by their maximum is the one used here to simulate depth-cueing. Another pixel function, the exclusive-or function, is used for simulating separate pixel planes, for example a cursor that can pass through a picture without alteration.

The next, mechanically more demanding object, is a depth-cued line. This is useful to improve the legibility of a rotating cube, or hypercube. For this we switch to ISYSForth because it compiles code that is fast enough to rotate simple wire-frames composed of user-built line-segments. That is, the student learning the Bresenham line drawing algorithm can implement it in Forth together with personalized pixel operations. Very simple polygonal surfaces can be rendered in a way that simulates Z-buffering by programming scan-lines that fill triangles. The pixel operations make it possible to program a limited but recognizable texture map simulation.

Most Math 198 students are eager to skip over these details and move to the Iris where such graphics are library primitives. Some, however, take the opportunity to become more deeply involved with graphics primitives and, for their class project, produced an ML and C based graphics package for their own personal 386 based home computers.

This is as good a place as any to defend Math 198 against the charge of being a course which teaches computer engineering without a license. There is mathematics in everything, and the study of anything sufficiently interesting to bright students becomes mathematics when the epistemological approach itself is analytical rather than merely practical and goal oriented.

Hypergraphics on the Iris

At this point in the course interest and attention begin to bifurcate. Cellular automata and Mandelbrot sets can still be done on the Apples using the (by now familiar) languages of &-GRAFIX, CSL and ISYSForth. Some students now experiment with possible projects using these methods. All students migrate to the Iris lab for a 3-week introduction to geometrical graphics. We next discuss the source code for illiSnail. This example is typical of the RTICAs we use for anything from a 30 minute hands-on demo to a two week summer workshop for math teachers.

The Curriculum

In the first lab session the students learn to control the animation. For illiSnail this entails flying through a Möbius band so stretched that its boundary lies in a plane. It looks vaguely snail-like. I first saw a wire mesh model of this surface hanging from the ceiling of Bernard Morin's office in Strasbourg. According to Larry Siebenmann (1982) the engineer Michel Pintard made a wire model like this in the 1930s. Pintard had studied topology with Hadamard. I was deeply impressed by the beautiful, computer generated 16 mm film of this surface made by Dan Asimov and Doug Lerner (1984) at Lawrence Livermore National Laboratory with a Cray-1 supercomputer. But I had to wait for the Iris 4D to write a real-time, interactive computer animation. In fact, this surface is an interesting example of a significant class of ruled, minimal surfaces in spaces with elliptic geometry, such as the 3-sphere in 4-space. The RTICA is capable of generating several other significant surfaces, such as Steiner's Roman Surface the Clifford Torus and Lawson's Minimal Kleinbottle, a portion of which constitutes Brehm's Trefoil Knotbox. The exercises are listed at the end of the chapter. (See also Color Prints 11 and 12.)

In the second session, students learn the geometry of these surfaces and their homotopies. The third session is a survey of the internal operation of the program. It serves as an introduction to geometric computer graphics. In the fourth and last common session, all students use the RTICA (or minor modifications of it) to produce a brief (1-2 minute) animation by recording their manipulations in a script file. This is the "lab-report" recording the results of their exploration into hypergraphics.

The Program

The program listed here is actually a condensation into a single file of several RTICAs of graded difficulty and sophistication. It was designed on a Personal Iris 4D/25G using Irix 3.3.1. It was recompiled on some other systems using Irix 4.0 to increase our confidence that, with some minor tinkering, this code will run on any Iris. There are still a few bugs in it, only some of which are intentional. It is an old Navajo custom to weave an error into every blanket to forestall the temptation to imagine the work to be perfect.

For the sake of brevity we omitted several useful and instructive features, in particular Chris Hartman's script writer and object maker. Both produce textfiles. An illiScript captures the key-presses and so recreates the animation by reading it back into the same RTICA. An illiObject is the display list for a particular stage of a homotopy of the surface. It is intended to be used by a more elaborate and sophisticated surface viewing program. The reader may obtain the source code for illiSnail from the author. Videotapes showing solutions to the exercises below, and other experiments, may be obtained from the NCSA (Francis, Chappell, & Hartman, 1994).

This program is written in "vanilla-C," using only a few of the most useful functions in the Iris graphics library. No attempt was made to write the program in exemplary C. Nor does its style conform to standard rules of "pretty printing." In Math 198 we treat programs like proofs, in which the visual space occupied by a symbol is roughly proportional to its mathematical importance. The program is meant to be studied and "unpacked" slowly before it is modified or rewritten. In particular, students are encouraged to practice using the editor of their choice on the Iris to rewrite the code in their favorite style. One time, a student translated a C-program into Fortran in order to understand it. Ironically, it had been translated from Fortran to C for the purpose of teaching it to the class. Common programming problems often have more than one solution in C. For any given problem, the absolutely optimal or most elegant solutions was generally not used, mainly because I probably don't know it. The student is certainly welcome to teach the teacher a trick or two. So, without further apologies, here is a print-out of the code which I shall document with just sufficient detail to be of profit to anyone with access to an Iris computer with the standard ANSI-C compiler and the shared libraries in its directories.

```
/* George Francis, Glenn Chappell and Chris Hartman */
 * Mathematics Department and NCSA */
 * (C) 1992 Board of Trustees */
/* Unversity of Illinois, Urbana, Illinois 61801 */
/* e-mail  gfrancis@math.uiuc.edu */
/* descendant of knotbox.c, moebius.c, cubevert.c, fly7.c */
/* this version 11/20/92 */
#include <gl.h>          /* graphics library */
#include <device.h>      /* device library */
#include <math.h>        /* mathematical library */

#define  MAX(x,y)          ((x<y)?y:x)
#define  MIN(x,y)          ((x<y)?x:y)
#define  ABS(x)            ((x<0)?-x:x)
#define  DG                M_PI/180.
#define  C(t)              fcos(t*DG)
#define  S(t)              fsin(t*DG)
#define  FOR(a,b,c)        for(a=b;a<c;a++)
#define  IF(K)             if(getbutton(K))
#define  SOAK(K)           while(getbutton(K))
#define  TOGGLE(K,f)       IF(K){f=1-f; SOAK(K);}
#define  IFSHIFT           if(getbutton(LEFTSHIFTKEY)||getbutton(RIGHTSHIFTKEY))
#define  PRESS(K,A,b)      IF(K){IFSHIFT{A;}else{b;}}
#define  PRES_S(K,A,b)     IF(K){IFSHIFT{A;}else{b;};SOAK(K);}
#define  LABEL(x,y,W,u)    sprintf(phrase,W,u); cmov2(x,y); charstr(phrase)
#define  DOT(aa,bb)        (aa[0]*bb[0]+aa[1]*bb[1]+aa[2]*bb[2])
#define  NRM(aa)           fsqrt(DOT(aa,aa))

int grnd, rndr, cube, thick, win, fly, binoc, msg, gap, brt,
    th0,th1,dth,cdth,fdth,ta0,ta1,dta,cdta,fdta.
float lux[3]={1.,2.,3.}, lu[3], id[4][4], aff[4][4],
    alfa,beta,lima,amp,pwr,maus,nose, mysiz,focal,speed,far ;
char phrase[256];
float vq[8][3]; /* Kommrush cube in main */
int fr[]= {0,1,5,1,3,7,3,2,6,2,0,4,5,7,6,4};
long znear,zfar.
```

```
deFaults(){int ii,jj; /* in the key of Z */
/* surface patch */
   alfa = 2; beta = 1; lima=0.; gap = 1;
   th0 = 90 ; th1 = 269; cdth=18; fdth= 6;
   ta0 = 90 ; ta1 = 270; cdta=18; fdta= 6;
   dta = cdta; dth = cdth;
/* flags */
   rndr=1; cube=0; win=2; msg=1; thick=4; binoc=0;
/* flying  */
   maus=.01; speed=.04; fly=0; mysiz=.02, focal=2., far=9.5;
/* rendering*/
   grnd = 0; brt = 255; amb= .3; pwr= 16.; nose=.06;
/* reset the affine matrix */
   FOR(ii,0,4)FOR(jj,0,4)aff[ii][jj]=id[ii][jj];
/* and put the object into the background */
   aff[3][2]= -4.2;
/* z-buffer and depthcueing parameters */
  znear= 0xf; zfar = 0x7fffff;
}
DeFaults()  /* if you want another set of them */
}
arguments(argc,argv) int argc; char **argv; /* Pat Hanrahan, 1989 */
      {while(--argc){ ++argv; if(argv[0][0]=='-') switch( argv[0][1]){
      case 'w': win  = atoi(argv[1]); argv++;argc--;break;
      case 'g': grnd = atoi(argv[1]); argv++;argc--;break;
      case 'p': alfa = atoi(argv[1]);          /* Moebius band a=2,b=1 */
                beta = atoi(argv[2]);          /* Clifford torus 2   2 */
                argv += 2; argc -= 2;break; /* Knotbox         2   3 */
      case 'l': lux[0] = atof(argv[1]);      /* light source direction */
                lux[1] = atof(argv[2]);      /* 0. 0. 1. for headlight */
                lux[2] = atof(argv[3]);      /* 1. 2. 3. is default    */
                argv += 3;argc -= 3;break;
}}   /* to add commandline arguments ad libitum  mimic the syntax */

int paint(lmb,dog,cat)float lmb; int dog,cat;{ int rr,gg,bb; float spec;
                                          /* Chris Hartman, 1991 */
    rr = dog;
    gg = abs(cat - 128) + 64;
    bb = brt - cat;
    lmb = MAX(lmb,amb);
    spec=MIN(255,brt*( 1. + pwr - pwr*lmb));  /* Ray Idaszak, 1989 */
    rr = MAX(lmb*rr, spec);
    gg = MAX(lmb*gg, spec);
    bb = MAX(lmb*bb, spec);
    return( (bb<<16) + (gg<<8)+rr);
}
surf(vv,th,ta) float vv[3]; int th,ta;{float ll, xx,yy,zz,ww,xl;
      ll  = C(lima) - S(lima)*C(ta); /* limaconic homotopy */
      xx  = C(alfa*th)*C(ta)*ll;      /* Sudanese Moebius Band */
      yy  = S(alfa*th)*C(ta)*ll;      /* Sue Goodman & Dan Asimov, ca 1980*/
      zz  = C(beta*th)*S(ta)*ll;      /* Larry Siebenmann, 1982 */
      ww  = S(beta*th)*S(ta)*ll;      /* Blaine Lawson, 1970 */
      xl =  .7071*(xx-ww);            /* nearly polar  projection */
      ww = .7071*(xx+ww);
      vv[0] = xl*7/(10+9*ww);
      vv[1] = yy*7/(10+9*ww);
      vv[2] = zz*7/(10+9*ww);
}
drawsurf(){int th, ta, dog; float vv[3], aa[3], vo[3], nn[3], lmb;
  for(th=th0; th < th1; th += dth){
    bgntmesh();
        dog = 255*(th-th0)/(th1-th0); cpack( paint(.8,dog, 0));
        surf(vo,th,ta0); v3f(vo);                      /* first vertex */
        surf(aa,th+dth-gap,ta0); v3f(aa);              /* first rung */
     for(ta = ta0+dta; ta <= ta1; ta += dta){
        surf(vv,th,ta ); /* normal on vo aa vv */
        nn[0]=(aa[1]-vv[1])*(vo[2]-vv[2])-(aa[2]-vv[2])*(vo[1]-vv[1]);
        nn[1]=(aa[2]-vv[2])*(vo[0]-vv[0])-(aa[0]-vv[0])*(vo[2]-vv[2]);
        nn[2]=(aa[0]-vv[0])*(vo[1]-vv[1])-(aa[1]-vv[1])*(vo[0]-vv[0]);
        lmb = DOT(lu,nn)/NRM(nn); if(lmb<0.)lmb = -lmb;
        cpack( paint(lmb, dog,255*(ta-ta0)/(ta1-ta0)));
        v3f(vv); surf(aa,th+dth-gap,ta); v3f(aa);
        vo[0]=vv[0]; vo[1]=vv[1]; vo[2]=vv[2];
```

BEST COPY AVAILABLE

```
      }
     endtmesh();
   }
}
drawhoop(ta,dt)int ta,dt{int th; float vv[3];
   cpack(0x44ffff);
   bgntmesh();
       for(th=th0;th<th1-dth; th += dth){
          surf(vv,th,ta );v3f(vv);surf(vv,th,ta+dt);v3f(vv); }
   endtmesh();
}
drawcube(){ int ii; /* Steve Kommrusch, 1984 */
    lRGBrange(100,100,100,255,255,255,znear,zfar);
    linewidth(thick); depthcue(1);
    bgnline();
    FOR(ii,0,16)v3f(vq[fr ii]]);
    endline();
    depthcue(0);
}
drawstars(){ int ii,jj; float vv[3],tmp[4][4]; /* Glenn Chappell, 1991
   pushmatrix(); pushmatrix(); /* 2 copies of the projector *
   loadmatrix(aff); getmatrix(tmp);
   tmp[3][0]=tmp[3][1]=tmp[3][2]=0; /*pure rotation */
   popmatrix(); multmatrix(tmp);
   srandom(1);  /* the stars don't change, hence the 1 */
   bgnpoint(); cpack(0xffffff);
     FOR(ii,0,1000){
     FOR(jj,0,3) vv[jj] = random()/(float)0x40000000-1.0; v3f(vv); }
   endpoint();
   popmatrix(); zclear(); /* the way it was before */
}
messages(){ /* text information as heads-up display */
    viewport(0,1279,0,1023); ortho2(-1.25,1.25,-1.,1.);
    if(!binoc){ cursor(); cpack(0x888888); circ(0.,0.,.01);}
    cpack(grnd?0:0xffffff);
    LABEL(-.9, .95,"(ESC)ape (Z)ap (C)oarse (F)ine (G)ap= %d ",gap);
    LABEL(-.1, .95,"%d =(BAR)fly (B)inoc (R)ender (Q)ube (PRNTSCRN)",fly);
    LABEL(-.9,-.95,"illiSnail RTICA by George Francis, U Illinois, 1992",win);
    LABEL( .1,-.95,"  The shifted key inverts the action, usually.",win);
 /* frequently used control keys */
    LABEL(-.9, .85,"(M)aus   %g",maus);
    LABEL(-.9, .80,"far cli(P)er= %.2g",far);
    LABEL(-.9, .75,"push near clipper (i)n (o)ut %.2g", mysiz*focal);
    LABEL(-.9, .70,"f(O cal factor %.2g",focal);
    LABEL(-.9, .65,"my s(I)ze %.2g",mysiz);
    LABEL(-.9, .60,"(S)peed  %g",speed);
    LABEL(-.9, .55,"(L)imacon %g",lima);
/* rarely used control keys */
    LABEL(-.9, .90,"(f1)ambient %g",amb);
    LABEL(-.6, .90,"(f2)specular %g",pwr);
    LABEL(-.3, .90,"(N)ose %g",nose);
    LABEL(-.0, .90,"(f5)th0 %d",th0);
    LABEL(.25, .90,"(f6)th1 %d",th1);
    LABEL(.5 , .90,"(f7)ta0 %d",ta0);
    LABEL(.75, .90,"(f8)ta1 %d",ta1);
}
keyboard(){ /* control keys */
    TOGGLE(PRINTSCREENKEY,msg) /* messages and cursor vanish */
    TOGGLE(RKEY,rndr)          /* surface vanishes */
    TOGGLE(QKEY,cube)          /* stick cube appears */
    TOGGLE(SPACEKEY,fly)       /* flying mode */
    PRES_S(SKEY,speed += .01, speed -= .01) /* accelerate */
    PRESS(ZKEY, DeFaults(), deFaults())    /* zap changes */
    PRESS(IKEY, mysiz /= 1.1, mysiz *= 1.1) /* rescale the world */
    PRESS(OKEY, focal *= 1.1 , focal /= 1.1) /* telephoto */
    PRESS(PKEY, far *= 1.01 , far  /= 1.01)   /* beware of this */
    PRESS(MKEY, maus += .01 , maus -= .01 )    /* gentler, kinder mouse*/
    PRES_S(BKEY, binoc=nose=0, binoc=1; nose=.06) /* cross your eyes */
    PRESS(NKEY,  nose -= .01 , nose += .01 )   /* parallax adjuster */
```

BEST COPY AVAILABLE

```
      PRES_S(CKEY, dth=dta += 1 , dth=cdth; dta = cdta) /* coarser mesh */
      PRES_S(FKEY, dth =MAX(1,--dth);dta=MAX(1,--dta), dth=fdth;dta=fdta)
      PRESS(LKEY, lima -= 1., lima += 1. )   /* Limacons of Pascal homotopy */
      PRES_S(GKEY, gap=0 , gap=MIN(dth,gap+1)) /* between the ribbons */
      PRESS(F1KEY, amb -= .01, amb += .01 )    /* ambient floor */
      PRES_S(F2KEY, pwr = 1. ,  pwr *= 2. )    /* specular ramp */
      PRESS(F5KEY,  th0 = MIN(++th0,th1) , th0--) /* source patch */
      PRESS(F6KEY,  th1 = MAX(--th1,th0) , th1++)
      PRESS(F7KEY,  ta0 = MIN(++ta0,ta1) , ta0--)
      PRESS(F8KEY,  ta1 = MAX(--ta1,ta0) , ta1++)
 }
main(argc,argv) int argc; char **argv; {int ii,jj,dx,dy; float temp;
/* Kronecker delta for the identity */
        FOR(ii,0,4)FOR(jj,0,4)id[ii][jj]=(ii==jj?1:0);
/* Steve Kommrush's cube by Gray-code */
        FOR(ii,0,8)FOR(jj,0,3)vq[ii][jj]=(ii&(1<<jj))?1.0:-1.0;
/* load up deFaults */
       deFaults();
/* but use Hanrahan's argumentor */
        arguments(argc,argv);
/* normalize light the light direction */
        temp=NRM(lux); FOR(ii,0,3) lux[ii] /= temp;
 /* decide on window style */
        switch( win){
        case 0:                                : break;
        case 1: prefposition(0,640,0,512); break;
        case 2: prefposition(0,1279,0,1023); break;
        }
 /* open the windows */
        winopen("illiSnail");
        zbuffer(1); doublebuffer(); RGBmode(); gconfig();
        lsetdepth(0,0x7fffff); /*this may be the default */
        zfunction(ZF_LESS); /* z-buffer reversed by GREATER */
/* keep Irix 4. from messing up your windows */
qdevice(LEFTMOUSE); qdevice(MIDDLEMOUSE); qdevice(RIGHTMOUSE);

while(!getbutton(ESCKEY)){   /* eternal loop */
          keyboard();
 /* Chappell's flyer */
            dx = getvaluator(MOUSEX) - 640; dx = ABS(dx)>5?dx:0;
            dy = getvaluator(MOUSEY) - 512; dy = ABS(dy)>5?dy:0;
            loadmatrix(id);
            if(!fly) translate(aff[3][0],aff[3][1],aff[3][2]);
            rot(-dy*maus, 'x'); rot(dx*maus,'y');
            PRESS(RIGHTMOUSE,  rot(-10,'z'),  rot(-1,'z'))
            PRESS(LEFTMOUSE,   rot( 10,'z'),  rot( 1,'z'))
            if(!fly) translate(-aff[3][0],-aff[3][1],-aff[3][2]);
            PRESS(MIDDLEMOUSE, translate(0,0,-speed),translate(0,0,speed));
            multmatrix(aff); getmatrix(aff);
/* rotate light source */
        FOR(ii,0,3){lu[ii]=0; FOR(jj,0,3) lu[ii] += aff[ii][jj]*lux[jj];}
/* draw a frame */
        cpack(grnd?0xffffff:0); clear(); zclear();

    if(binoc) viewport(0,640,256,768);     /*  right eye is left*/
/* cross-eyed shifted stereoscope  gkf 2.29.92 with gcc flyer */
     window(-mysiz*1.25,mysiz*1.25,-mysiz,mysiz,mysiz*focal,far);
     drawstars();
     translate(-nose,0,0); multmatrix(aff);
     drawhoop(ta0,-2); drawhoop(ta1,2);
     if(rndr)drawsurf(); if(cube)drawcube();

   if(binoc){
     viewport(640,1280,256,768);  /* left eye is right */
     window(-mysiz*1.25,mysiz*1.25,-mysiz,mysiz,mysiz*focal,far);
     drawstars();
     translate( nose,0,0); multmatrix(aff);
     drawhoop(ta0,-2); drawhoop(ta1, 2);
     if(rndr)drawsurf(); if(cube)drawcube();}
   cursoff(); if(msg)messages();
   swapbuffers();
   reshapeviewport();
  /* end while loop */ } /* end it all */
```

BEST COPY AVAILABLE

#include, #define, and variables.

The program is entirely self-contained, up to calling functions in the graphics, device and mathematical libraries. Next come a series of abbreviations for the compiler which improve the readability of the source code. The trig function macros, present here merely for notational convenience, are an occasion for discussing how to optimize real-time animations. There are still many graphics computers slower than an Iris where considerable improvement in real-time performance can be achieved by tabulating the values of transcendental functions.

For many sound but admittedly controversial reasons, we depart from standard programming practice in the matter of variable types, and other customs. The extra precision of "long" and "double" is rarely needed, and the names "int" and "float" are more mnemonic anyway. This is not, however, the place to defend or promote these departures. A reader who is offended by this rough but practical style of writing C is welcome to blue pencil my code as if it were a school-boy's effort.

deFault(), arguments()

Almost all variables which are or may someday be, interactively manipulated, or which are, or may be, used by more than one independent subroutine, are global. Their default values are assigned in a subroutine, which itself can be called repeatedly by the user during execution. There is a second set of defaults which the student can use to customize his own version of illiSnail.

The RTICA uses an exceedingly simple algorithm for reading arguments from the command line. A one letter flag announces new default value(s) for the desired parameter set. The student can easily add and subtract cases in the switch block without rebuilding the subroutine. Many years ago, when I explained to Pat Hanrahan that my students cannot spend much effort on learning input/output syntax in C, he designed this routine for them.

paint()

The surfaces the RTICA generates are all painted and lighted. That means two things. The color of a vertex is a geometrical attribute, for example a function of the surface parameters. The corresponding values of red, green and blue are attenuated proportional to the Lambert lighting model (Francis, 1987, pp. 61-64). This needs only the computation of the cosine between the normal direction and the direction of the light source. At the dim end, we clamp this attenuation at what corresponds to an "ambient" level. At the bright end, we ramp all three values steeply to pure white to give the illusion of a specular region. Later, at the cost of one 3x3 matrix multiplication per frame, we move the light source so that the specular high-light appears to be stationary as a surface rotates about some axis. This simplification of the standard Phong lighting model evolved from one which Ray Idaszak developed for the Etruscan Venus Project (Francis, 1987, Appendix). It is easy to explain and to apply, and it works quite well, especially for one-sided surfaces. Its merits and demerits are discussed elsewhere (Francis, 1991; Idaszak, 1988).

Chris Hartman mixed the present color palette in rich pastels suitable for videotaping. The meaning of the colors painted on the surface is quickly discovered once the user manipulates illiSnail to draw a fine-grained rectangular patch (see Exercise 1.1). In this subroutine the "dog" and "cat" chase each other through a color gamut as they map the surface parameter values into a mixture of red, green, and blue.

surf(vv,th,ta)

This function returns the position of the mapping.

$$v = f(\theta, \tau)$$

For another surface, the programmer need only change this function, and adjust the default parameter values. This code segment becomes less of a mystery once it is transformed into standard mathematical notation as follows.

We first map a $\pi \times \pi$ sized rectangle to 4-space, where it occupies a ıatch on one of the real algebraic, geodesically ruled, minimal, surfaces immersed in the 3-sphere, as described by Lawson (1970).

$$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} \cos(\alpha\theta) \\ \sin(\alpha\theta) \\ 0 \\ 0 \end{bmatrix} \cos\tau + \sin\tau \begin{bmatrix} 0 \\ 0 \\ \cos(\beta\theta) \\ \sin(\beta\theta) \end{bmatrix}$$

We rotate the 3-sphere in the xw-plane,

$$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \rightarrow \begin{bmatrix} \frac{x-w}{\sqrt{2}} \\ y \\ z \\ \frac{x+w}{\sqrt{2}} \end{bmatrix}$$

and finally project this to 3-space from a point just outside the 3-sphere to avoid accidental zero-division.

$$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \rightarrow \begin{bmatrix} \frac{7x}{10+9w} \\ \frac{7y}{10+9w} \\ \frac{7z}{10+9w} \end{bmatrix}$$

Multiplying [x,y,z,w] by the factor (cos (λ)-sin(λ)cos(τ)) before projection, has the effect of moving the semicircular τ-wires, for λ = 0 to 90°, through the Limaçons of Pascal, to full circles of half the diameter and all passing through the origin.

drawsurf(), drawhoop()

The surface itself is drawn as a succession of ribbons with a greater or smaller gap between them. Each ribbon, parametrized by τ, uses the triangular mesh function of the Iris graphics library. The gap is controlled by the G-key, the stepsize of θ and τ is controlled by the F-key and the C-key, which switch between a fine and a coarse mesh. The shifted F-key makes the mesh finer, the shifted C-key makes it coarser. The R-key toggles the rendered surface on and off. It is intended to switch to the wire-frame, a feature students are invited to install as an exercise. The yellow θ-ribbon, generated by the drawhoop() routine, is neither painted nor lighted to really illustrate the tmesh() syntax. The RGB-color primitive cpack(0xbbeedd) employs a hexadecimal encryption of the 3 color values. In this example, it yields bb=11*16+11=187 blue, ee=238 green, and dd=221 red, in that order.

drawcube(), drawstars()

The "unit" cube is in this RTICA for reference (its inside radius is 1) and to train the user in cross-eyed binocular viewing of the stereo images. The parameters of the binoculars can be interactively adjusted, and it is easier to use the familiar line-drawn cube than the unfamiliar surfaces to check the effect of such changes.

The cube itself is drawn as one continuous polygon following a vertex list. Recall that the hypercube is a significant actor in Math 198. Steve Kommrush, a student in my very first computer based edition of the course, left us with a beautifully simple algorithm for drawing the hypercube. As an elementary exercise in modifying illiSnail, the student is invited to turn the 3-cube into a 4-cube which can be rotated in all 6 planes, and thus implement Tom Banchoff's (1977) classic visualization of the hypercube.

Originally Glenn Chappell's stars were an amusing experiment, but their presence really helps keep one's sense of position while navigating the labyrinthine interior of the surfaces.

messages(), keyboard()

The next subroutine displays messages on the screen, for example the current value of parameters, and which keys to press to change them. The key-presses are interpreted by the next subroutine in one of three styles. Togglers alternate between two states. Cyclers are more sophisticated versions of this and operate like the buttons on a digital wristwatch. As students run out of keys to program, cyclers become popular despite their confusing logic. Since the keyboard is meant to be read between each frame, pressing a key can be interpreted incrementally as an "accelerator." The shifted key reverses the direction of change in the parameter. In some cases the changes are naturally big steps and one wants to force the user to think between presses. For this purpose we "soak" the key, so that it must be let up to take effect.

There are, of course, many other ways of controlling an RTICA; pull-down menus with sliders and buttons are the most popular. I invite all of my students to compare the effort and reward of a heads-up display with two handed pilotry favored by flight-simulators to the alternative of controlling everything with the mouse and having verbose menus interrupt the animation. Soon most agree that ten-fingered users quickly learn do many things automatically and together, without a need for distracting writing in the field of view. The messages, and even the mouse cursor, can be turned off with the function key marked "Print Screen." I often joke with ardent defenders of pull-down menus and slider-bars that I would not care to be a passenger in a commercial jet flown by a pilot clicking a mouse to select control values from a pull-down menu.

main(argc, argv)

This brings us to the main block of the program which consists of a setup sequence and an eternal loop from which one can escape with the escape key. Students are initially discouraged from changing this part of the program because it contains the hardware specific calls in the correct order. However, an understanding of its operation in general terms helps one to perform the experiments and to interpret the sometimes baffling outcomes.

In the setup, the identity matrix is built with a Kronecker delta defined in terms of the ternary operator of C. This prepares the student for its use in Steve Kommrush's very clever construction of the unit cube. Next, the default values are assigned to the parameters, and perhaps modified by Hanrahan's routine. For example, a new light source, perhaps a headlight for the flier, can be given on the command line. Its unit direction is calculated by the program. The student might build in several lights, or a local light source as an exercise.

The current program is simplified to work properly only with a full Iris screen of size 1280x1024. Indeed, it must be suitably adjusted to work on the small Indigo screen. The command line choice of three window styles is a start in this adjustment. On the other hand, if a smaller window is needed without recompiling, execute this Unix line:

```
iris% illiSnail—w 0
```

Now we are in the loop. The mouse-syntax in this RTICA is an adaptation of that invented by Glenn Chappell in his much more sophisticated geometrical viewing program, illiFly. The intention there as here is to give the illusion of piloting a small space capsule in and around a mysterious topological object in empty space. The capsule can move forward and backward, and orbit sideways around the object. The porthole can change its focal length for wide angle or narrow angle viewing, and the entire world can change its apparent size relative to the capsule. All this is actually a plausible rationalization of the effects one can achieve using the Iris graphics library primitive for perspective projection.

At the heart of every RTICA is some way of coupling the motion of the mouse with motion in a subgroup of geometric transformation of the world coordinates. The one used here has evolved through many years of student and instructor experimentation, and is one of several we encourage new students to improve upon. Once all the function specific to the Iris graphics library are translated into standard multilinear algebra, the present version can be shown to be both obvious and nearly optimal. However, we cannot do that here. A definitive exposition of these matters is in preparation (Francis, Chappell, & Hartman, 1994).

In broad terms then, a displacement of the mouse from the center of the screen (marked by a gray bullseye) is translated into a small modification of an affine transformation of 3-space. Recall that the Iris geometry pipeline operates as if the homogeneous coordinates of each vertex are multiplied by a succession of 4x4 matrices. At any given moment, this succession may be associated into a product of just two matrices. The first represents a member of the 3-dimensional affine group, which is a semi-direct product of GL(3,R) and R^3. (For practical purposes, think of the Euclidean group of rotations and translations.)

The second matrix represents a projective transformation which expresses linear perspective. The Iris graphics library takes a resolutely pre-Copernican view, placing the eye at the origin of the world coordinate system, and looking "backwards" into the negative z-direction. A rectangular window is placed a positive distance from the eye, and everything visible is clipped to lie in the frustum of a cone between the projection plane and a far clipping plane. There are two keys in illiSnail which control the projection matrix. The O-key changes the focal length of the view. You may think of a zoom lens. Increasing the focal parameter has a telephoto effect; decreasing simulates a wide angle lens. The latter is useful for viewing the inside of the tunnels formed by the surfaces.

In order to fly through these tunnels one has to eliminate the effect of the frontal clipping plane. The I-key does this without changing the linear perspective or the area occupied by the object on the screen. On the other hand, pressing the O-key and the I-key together, changes only the scale of the viewing window, without changing how near to the eye or the object the clipping plane is located. As you fly closer to an object, it is possible with these controls to slice frontal windows into the surface for looking in, or shrinking your apparent size so as to fly around inside.

The two states of the rotor, toggled by the space-bar and echoed on the message board, are called "flying" and "orbiting." In the former state, the axis of rotation is through the observer. Thus the space pod moves to where the mouse cursor is pointing. In the latter, it passes through the object and it appears to turn in the direction of the mouse movement as is customary for trackball rotors (Francis & Kauffman, 1994; Hanson, 1992). Press the middle mouse button to move forward at the speed adjusted by the S-key. Shift-mouse reverses the direction, while shift-S decreases the velocity. The sensitivity of the mouse can be changed with the M-key.

Binocular vision is induced, approximately, by shifting the entire scene to one side and the other before projection. For cross-eyed viewing, toggle the B-key: The right image is sent to the left viewport and vice versa. The "nose" parameter, on N-key, adjusts the binocular parallax, so that a negative value produces stereopairs for parallel viewing. Stereopairs help the user to discriminate certain surface features more accurately, and to discover programming errors during code modifications. We do not recommend crossing your eyes for any length of time.

Exercise 1.1

Here are 3 easy experiments to perform on the 3 surfaces in illiSnail. The F5—F8 keys change the range of the two surface parameters. Press the shift-key and the F7, F8 pair simultaneously to retract the Möbius band to its more familiar position of a ribbon with half-a-twist in it. Note how pressing the F and C keys switches from a fine to a coarse grained triangulation of the surface. The shifted F-key refines the triangulation, the shifted C-key coarsens it. Be aware that key presses are not buffered in a queue. All keys are polled after each frame. So if a frame takes a while, the key action is slow. In this way, the visible effect of an action is the confirmation that a key has been pressed, and no inexplicable sequence of queued up actions can happen when no keys are pressed. Retracting the other surface parameter (shift F5 F6) yields a rectangular patch which is good for studying the color scheme.

Exercise 1.2

Now press all four keys, F5–8,to restore the Möbius band and stretch it out so that its border becomes a plane circle and the "diameters" are again semicircles. If you are in a hurry, the Z-key zaps all changes and restores the original settings. Hold the L-key down and watch these semicircles close to full circles. The border of the Möbius band shrinks to a point, producing the cross-cap model of the real projective plane (Banchoff, 1978; Francis, 1987, Ch. 5). The G-key controls the width of a gap between successive meridional strips that make up the surface. Shift-G zeros the gap. Note how these ribbons are Gouraud shaded in one direction, but not the other. This improves binocular convergence as well as simplifying the code.

Exercise 1.3

For the third experiment provide the flier with a "headlight" by executing

```
iris% illiSnail–u 0.0 0.0 3.0
```

from the Unix command line. Next, rotate the snail, switch to flying mode (space-bar). Try to fly through the twisting tunnel without sliding through the walls. Once inside the snail shell, you may wish to slow-down (S-key) and widen your field of view (O-key). Release the mouse button to stand still and look around.

Exercise 2

Now switch surfaces.

```
iris% illiSnail–p 2 2–u 1. 2. 10.
```

This yields a (nearly) stereographic projection of the Clifford Torus from the round 3-sphere to flat 3-space. Repeating the first experiment demonstrates how the torus may be regarded as a closed, two-sided ribbon with one twist in it, stretched out until the edges come together along a circle. Flying through both holes of a torus is predictably easy. Exploring the limaçonic homotopy meaningfully here is more of a challenge. Note that Hanrahan's subroutine can change more than one case of default parameters. We installed "headlights" too.

Exercise 3

```
iris% illiSnail–p 2 3
```

The final surface is the most difficult to understand. The first experiment applied to this surface shows how a Möbius band spanning a (yellow) trefoil knot has 3 half twists. Bending the knot so as to form a triple-circle (think of the knots that garden hoses tend to form) brings 3 sheets of the surface together along the same curve. This 2-dimensional cell complex is a smooth realization of what Ulrich Brehm (1991) calls a knotbox. If you succeed in flying through this object, your trajectory will be a trefoil knot. A reader initiated into the topological mysteries of knot complements will recognize this complex as a standard spine of the complement of the trefoil knot in the 3-sphere.

succession may be associated into a product of just two matrices. The first represents a member of the 3-dimensional affine group, which is a semi-direct product of GL(3,R) and R^3. (For practical purposes, think of the Euclidean group of rotations and translations.)

The second matrix represents a projective transformation which expresses linear perspective. The Iris graphics library takes a resolutely pre-Copernican view, placing the eye at the origin of the world coordinate system, and looking "backwards" into the negative z-direction. A rectangular window is placed a positive distance from the eye, and everything visible is clipped to lie in the frustum of a cone between the projection plane and a far clipping plane. There are two keys in illiSnail which control the projection matrix. The O-key changes the focal length of the view. You may think of a zoom lens. Increasing the focal parameter has a telephoto effect; decreasing simulates a wide angle lens. The latter is useful for viewing the inside of the tunnels formed by the surfaces.

In order to fly through these tunnels one has to eliminate the effect of the frontal clipping plane. The I-key does this without changing the linear perspective or the area occupied by the object on the screen. On the other hand, pressing the O-key and the I-key together, changes only the scale of the viewing window, without changing how near to the eye or the object the clipping plane is located. As you fly closer to an object, it is possible with these controls to slice frontal windows into the surface for looking in, or shrinking your apparent size so as to fly around inside.

The two states of the rotor, toggled by the space-bar and echoed on the message board, are called "flying" and "orbiting." In the former state, the axis of rotation is through the observer. Thus the space pod moves to where the mouse cursor is pointing. In the latter, it passes through the object and it appears to turn in the direction of the mouse movement as is customary for trackball rotors (Francis & Kauffman, 1994; Hanson, 1992). Press the middle mouse button to move forward at the speed adjusted by the S-key. Shift-mouse reverses the direction, while shift-S decreases the velocity. The sensitivity of the mouse can be changed with the M-key.

Binocular vision is induced, approximately, by shifting the entire scene to one side and the other before projection. For cross-eyed viewing, toggle the B-key: The right image is sent to the left viewport and vice versa. The "nose" parameter, on N-key, adjusts the binocular parallax, so that a negative value produces stereopairs for parallel viewing. Stereopairs help the user to discriminate certain surface features more accurately, and to discover programming errors during code modifications. We do not recommend crossing your eyes for any length of time.

Exercise 1.1

Here are 3 easy experiments to perform on the 3 surfaces in illiSnail. The F5—F8 keys change the range of the two surface parameters. Press the shift-key and the F7, F8 pair simultaneously to retract the Möbius band to its more familiar position of a ribbon with half-a-twist in it. Note how pressing the F and C keys switches from a fine to a coarse grained triangulation of the surface. The shifted F-key refines the triangulation, the shifted C-key coarsens it. Be aware that key presses are not buffered in a queue. All keys are polled after each frame. So if a frame takes a while, the key action is slow. In this way, the visible effect of an action is the confirmation that a key has been pressed, and no inexplicable sequence of queued up actions can happen when no keys are pressed. Retracting the other surface parameter (shift F5 F6) yields a rectangular patch which is good for studying the color scheme.

Exercise 1.2

Now press all four keys, F5–8,to restore the Möbius band and stretch it out so that its border becomes a plane circle and the "diameters" are again semicircles. If you are in a hurry, the Z-key zaps all changes and restores the original settings. Hold the L-key down and watch these semicircles close to full circles. The border of the Möbius band shrinks to a point, producing the cross-cap model of the real projective plane (Banchoff, 1978; Francis, 1987, Ch. 5). The G-key controls the width of a gap between successive meridional strips that make up the surface. Shift-G zeros the gap. Note how these ribbons are Gouraud shaded in one direction, but not the other. This improves binocular convergence as well as simplifying the code.

Exercise 1.3

For the third experiment provide the flier with a "headlight" by executing

```
iris% illiSnail–u 0.0 0.0 3.0
```

from the Unix command line. Next, rotate the snail, switch to flying mode (space-bar). Try to fly through the twisting tunnel without sliding through the walls. Once inside the snail shell, you may wish to slow-down (S-key) and widen your field of view (O-key). Release the mouse button to stand still and look around.

Exercise 2

Now switch surfaces.

```
iris% illiSnail–p 2 2–u 1. 2. 10.
```

This yields a (nearly) stereographic projection of the Clifford Torus from the round 3-sphere to flat 3-space. Repeating the first experiment demonstrates how the torus may be regarded as a closed, two-sided ribbon with one twist in it, stretched out until the edges come together along a circle. Flying through both holes of a torus is predictably easy. Exploring the limaçonic homotopy meaningfully here is more of a challenge. Note that Hanrahan's subroutine can change more than one case of default parameters. We installed "headlights" too.

Exercise 3

```
iris% illiSnail–p 2 3
```

The final surface is the most difficult to understand. The first experiment applied to this surface shows how a Möbius band spanning a (yellow) trefoil knot has 3 half twists. Bending the knot so as to form a triple-circle (think of the knots that garden hoses tend to form) brings 3 sheets of the surface together along the same curve. This 2-dimensional cell complex is a smooth realization of what Ulrich Brehm (1991) calls a knotbox. If you succeed in flying through this object, your trajectory will be a trefoil knot. A reader initiated into the topological mysteries of knot complements will recognize this complex as a standard spine of the complement of the trefoil knot in the 3-sphere.

Bonus Exercise

A rewarding programming exercise would be to enable the user to control the values of the α and β parameters interactively, say on A and B keys. This way one can observe the twisting of the band and the transitions between these three surfaces (and many other surfaces) more conveniently.

The software discussed here and documentation for running on your own Silicon Graphics computer is available through anonymous ftp from the author (gfrancis@math.uiuc.edu) by executing

```
iris% ftp 128.174.111.12
```

and logging in as anonymous.

References

Asimov, D. & Lerner, D. (1984). The Sudanese Möbius Band, 3 min. videotape in *SIGGRAPH Video Review, 17.*

Banchoff, T.F. (1990). *Beyond the third dimension: Geometry, computer graphics, and higher dimensions.* New York: W. H. Freeman.

Banchoff, T.F. (1977). *The Hypercube: Projections and slicing* [9.5 min narrated, color film]. Chicago: International Film Bureau.

Banchoff, T.F. (1978). *The Veronese surface* [12 min. color film]. Providence, RI: Mathematics Department, Brown University.

Barnsley, M.F. (1988). *Fractals everywhere.* New York: Academic Press.

Brehm, U. (1991). Videotaped interview at the Technische University Berlin.

Brisson, D.W. (Ed.)(1978). Hypergraphics, visualizing complex relationships in art, science, and technology. *AAAS Selected Symposium 24.* **Boulder, CO: Westview Press.**

Conway, J., Doyle, P., & Thurston, W. (1991). Geometry and the imagination, Mathematics 199 at Princeton. *Research Report GCG27.* The Geometry Center, 1300 So. 2nd St., Minneapolis, MN 55454.

Dewdney, A.K. (1988). *The armchair universe.* New York: W. H. Freeman.

Francis, G.K. (1987). *A topological picturebook.* New York: Springer-Verlag.

Francis, G.K. (1990). The snailhunt. To appear in A. Bowyer (Ed.), *Mathematics of surfaces IV; Conf. proc. of the Inst. Math. Applic.*, Bath, England.

Francis, G.K. (1991). The Etruscan Venus. In P. Concus, R. Finn, & D. Hoffman (Eds.), *Geometric analysis and computer graphics.* New York: Springer-Verlag.

Francis, G.K. (1992) *Visual mathematics, Part I: Experimental arithmetic; Part II: Hypergraphics; Part III: Geometrical graphics, 1991.* Class notes and lab manuals. : UpClose Printing Copies, Champaign IL.

Francis, G.K., & Kauffman, L.H. (1994). Air on the Dirac strings. In W. Abikoff, J. Birman, & K. Kuiken (Eds.), *The mathematical legacy of Wilhelm Magnus.* Contemp. Math Series, Amer. Math. Soc., Providence.

Francis, G.K., Chappell, G., & Hartman, C. (1994). *A Post-Eucldean walkabout.* In the CAVE virtual reality theater (VROOM), at ACM SIGGRAPH 94, Orlando. Mosaiac documentation available on Internet from the National Center for Supercomputing Applications, U. Illinois.

Gunn, C., et al. (1990). Undergraduate research experiences, summer 1990. *Research Report GCG19*, The Geometry Supercomputing Project, U. Minnesota, Minneapolis.

Hanson, A. (1992). The rolling ball: Applications of a method for controlling three degrees of freedom using two-dimensional input devices. In D. Kirk (Ed.), *Graphics gems III*. New York: Academic Press.

Hanson, A., Munzner,T., & Francis, G. (1994). Interactive methods for visualizable Geometry. In special (July) issue of *IEEE Computer*, edited by Arie Kaufman.

Hirsch, M.W., & Smale, S. (1974). *Differential equations, dynamical systems, and linear algebra*. New York: Academic Press.

Idaszak, R.L. (1988). A method for shading 3D single-sided surfaces. *IRIS Universe*, pp. 9-11.

Illyes, R.F. (1988). *ISYSForth*. Champaign, IL: Illyes Systems.

Lawson, H.B., Jr.(1979). Complete minimal surfaces in S^3. *Ann. Math.*, *92*(2), 333-374.

Marden, A (1991). Director. *Undergraduate research experiences, summer 1991*; Research Report GCG33, The Geometry Center, 1300 South 2nd Street, Minneapolis, MN 55454.

Sandvig, C. (1990). *GS.AMPER.NEW: A 65816 ML graphics package for the Apple IIgs*. Technical Report, UIMATH.APPLE Lab, Mathematics Department, University of Illinois, 1990.

Siebenmann, L. (1982). Le Gobelet de Möbius. [privately circulated script] Brouillon, France.

Color Print 9

Color Print 10

Color Print 11

Color Print 12

NCSA

Chapter 13

Color Print 11: The Sudanese Möbius band, lower right, is the initial form generated by the real-time interactive computer animation illiSnail. It is related to the "standard" form of the Möbius band, lower right, as the spherical bead, upper right, is to an untwisted annular band spanning two unlinked (yellow) circles. Retract the circular ribs connecting opposite points on the curve(s). These shapes appear in the solution to Exercise 1.1 at the end of Chapter 13 of this book.

Color Print 12: A once-twisted annulus spans two linked circles, upper left. Stretch the ribs connecting opposite points on the curves to full circles and thus generate a torus, upper right, by a succession of pairwise linked circles. This is a round shadow of the flat, Clifford torus in 4-space. At the lower right is a Möbius band with 3 half-twists, spanning a yellow trefoil knot. Move the knot to a circle and obtain a shadow of Lawson's minimal Kleinbottle in 4-space, which is also a smooth form of Brehm's knotbox. These shapes appear in the solution to Exercises 2 and 3 at the end of Chapter 13 of this book.

Color Prints 11 and 12 were composed by Chris Hartman in Adobe Photoshop with frames from a Silicon Graphics Iris 4D/300VGX and printed on a Tektronix Phaser II color printer in the Renaissance Experimental Laboratories of the NCSA, University of Illinois. ©George Francis, Chris Hartman, and Glenn Chappell, NCSA, University of Illinois, 1993.