iris UNIVERSE

spring 1987

the IRIS community magazine

welcome to the IRIS UNIVERSE

This is the premier issue of the IRIS community magazine. It is designed to foster the exchange of pictures, programs and personal contacts. We invite you to contribute your ideas to the IRIS universe; in fact, its quality depends on you.

This issue kicks off with two articles that discuss Computer Graphics in Mathematics and Geometric Modeling . We are happy to feature a number of substantial software packages available for the IRIS, listings of community activities, and to inform you about opportunities to tune-into the IRIS network.

The first time around, our publishing skills are still somewhat green. But we made every effort to print at least a few images by direct digital printing processes. The color separation for the cover pictures was created from a bitmapped file transferred to a Scitex. The ray-traced Alpha_1 picture was converted into PostScript, shipped to a Macintosh disc, and printed on a Linotronic photo-typesetter. Dietmar Saupe's b/w images were printed on a Laserwriter and photographically half-toned. These pictures serve as prototypes of the principles of digital prints. With your support and creativity there is more to come in the near future.

The next issue will be published in time for SIGGRAPH'87. We will focus on new approaches to education, animation, direct hardcopy output, and the Supercomputer connection. Your contributions are welcome. By the way, to subscribe to the IRIS Universe, just drop us a line or give us a call.

best greetings
your editors

cover pic

Butterfly © 1987 John W. Peterson, University of Utah

The butterfly was created using the Alpha_1 geometric modeling system and the Utah Raster Toolkit, both developed at the University of Utah Computer Science Department (see articles elsewhere in this issue). The body of the butterfly was modeled with Alpha_1 using non-uniform rational B-spline surfaces. The wing textures were made from digitized black and white ink drawings, the colors added digitally using the toolkit's compositor. The butterfly was generated using a ray-tracing algorithm, with the computations performed in about three hours on a network of several workstations throughout the department. The shadow was added as a post-process - a matte for the butterfly was used to cut out the dark green texture. This was then blurred with a Gaussian filter, and composited together with the original butterfly image. Thanks to Rod Bogart and Spencer Thomas for assistance in processing the textures.

backcover pic

Potential of the Mandelbrot Set
© 1986 Dietmar Saupe,
University of California at Santa Cruz

Potential of a satellite Mandelbrot set with a corresponding Julia set on the "moon" in the background. (See Dietmar's article *Computer Graphics in Mathematics* in this issue for a detailed explanation of the genesis of this picture. Also see *The Beauty of Fractals* by H.O. Peitgen and P.H. Richter for more of Dietmar's colorful fractal pictures. Benoit Mandelbrot proclaimed them to be "the best looking fractals".)

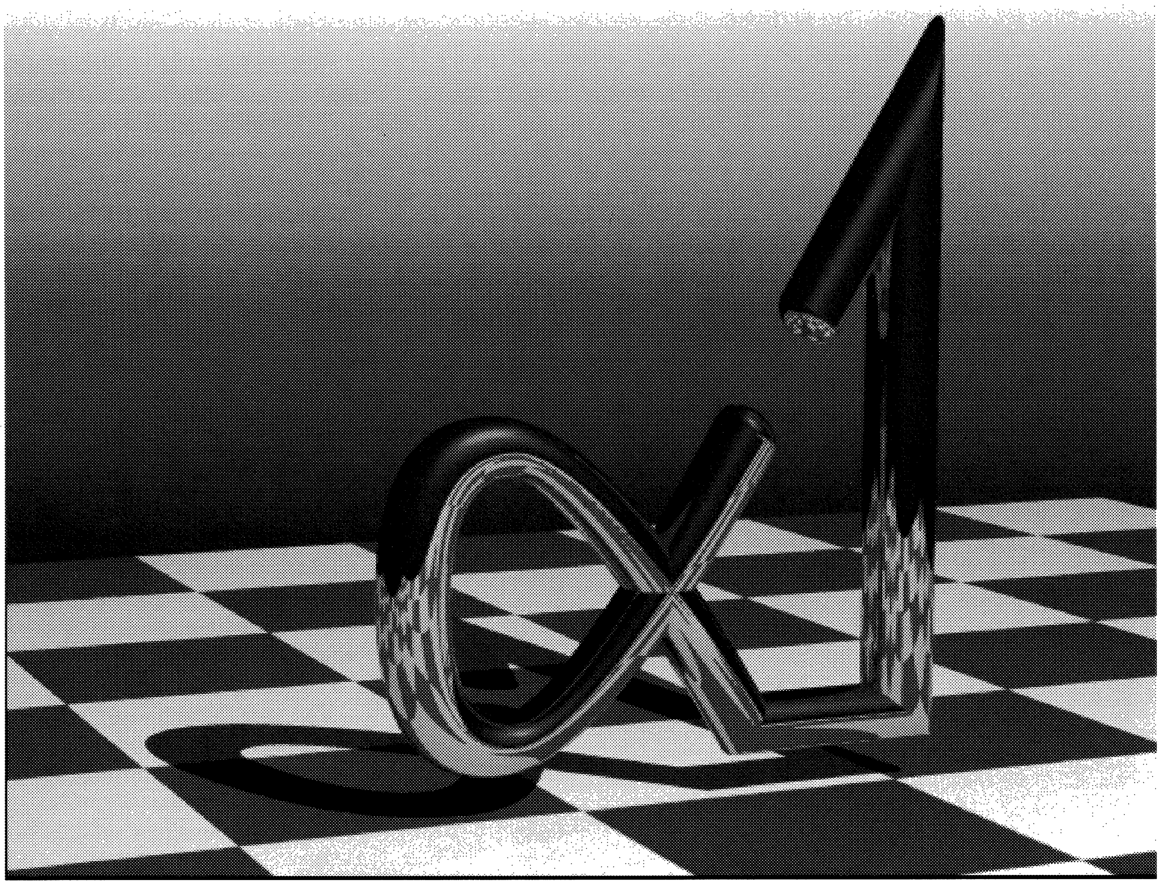
contents

IRIS UNIVERSE ©1987
editors:
Frank Dietrich
University of Utah
Dept. of Communication
Salt Lake City, UT 84112
(801) 581-5875
<dietrich@gr.utah.edu>

Zsuzsanna Molnar
Silicon Graphics Inc.
2011 Stierlin Road
Mountain View, CA 94043
(415) 962-3315
<ucbvax ! sgi ! zsuzsa>

text & images © 1987 by individual authors.

SiliconGraphics
Computer Systems



alpha_1: modeling with nonuniform rational b-splines

by Russel Fish, University of Utah

Alpha_1 is a *Geometric Modeling System* which integrates mechanical design, analysis, and manufacturing around a common description of part geometry. It started as a research testbed system, to demonstrate some innovative ideas in representation and modeling. The Alpha_1 system design provides for growth and change. A substantial modeling system has been built to professional software standards over a period of several years. The Alpha_1 project in the Computer Science Department of the University of Utah has had a stable core of about 6 senior members and major funding from the National Science Foundation, DARPA, ONR, BRL, ARO, and several industrial sponsors. The Alpha_1 modeling system is now becoming a means of disseminating our research results to a wider user community.

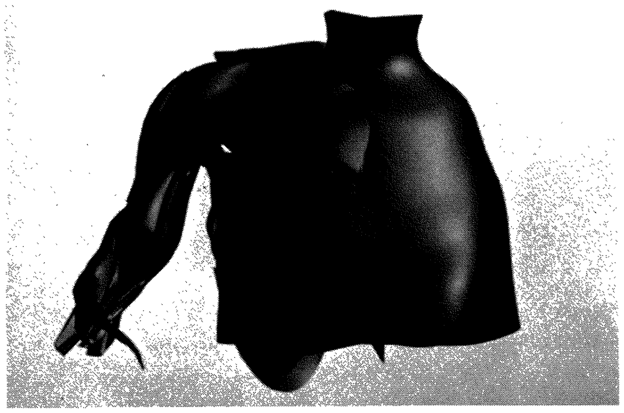
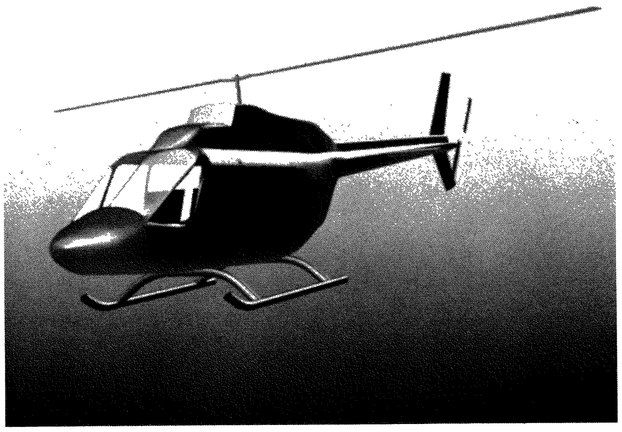
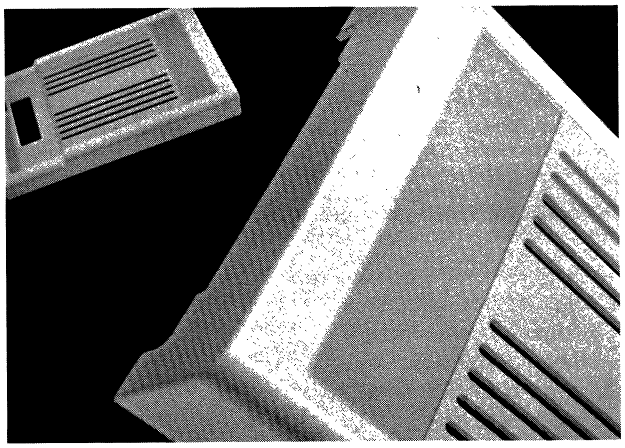
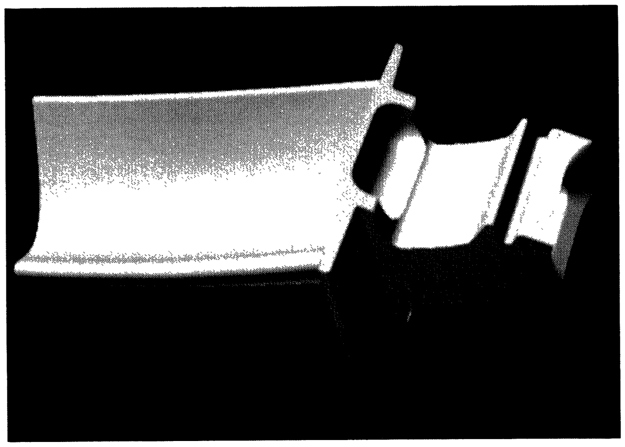
combining solid modeling with sculptured surface shapes

The ideas of several modeling "cultures" come together in Alpha_1. Solid modeling brings the idea of using complete, three dimensional descriptions of mechanical parts to directly support engineering analysis and manufacturing processes. There is also the idea of building up complex solids from simpler ones with *Constructive Solid Geometry (CSG)* operations such as (regularized) union, intersection, and difference. While many solid modeling systems are only able to deal with combinations of simple shapes such as spheres, cylinders and polyhedral shapes, Alpha_1 allows those shapes and also general sculptured shapes. *Free-form surface representations* were developed to meet the obvious needs of aerospace and automotive design for "curvy" surfaces.

Less obviously powerful free-form surface representations can concisely describe the flat surfaces and sharp and rounded corners which occur in mechanical parts together with areas which have curvier shapes. Alpha_1 uses NURBS as its *boundary representation* of solids.

what's a NURB?

NonUniform Rational B-Splines (NURBS) is a powerful free-form curve and surface representation. In fact, most of the other forms are subsets of NURBS. Designers using Alpha_1 can ignore much or all of the surface representation details if they desire, and so do not have to become spline experts. (The designer's view is covered in the next section.) We will skip over NURBS mathematics here, and concentrate on what makes them special. The most important point is that you

can exactly represent any piecewise polynomial or conic section curve with NURBS.

There are many polynomial or piecewise polynomial schemes known, all of which give different control "handles" which are an improvement over the raw polynomial coefficients. Non-rational B-splines, uniform B-splines, interpolating splines, tensioned forms like Beta-Splines, Bezier curves, and so on are included in the NURBS representation because of their piecewise polynomial basis. One difficulty with polynomials is that they don't exactly represent conic sections other than parabolas. When you want exact circular arcs, say for surfaces of revolution, the polynomial has to have W coordinates supplied as well as the usual X,Y or X,Y,Z, and then later the W is divided through. This ratio is what is referred to by the *rational* part of the NURBS acronym. In addition, NURBS exactly represent all other conic sections (ellipses and hyperbolas) as well as circles and parabolas.

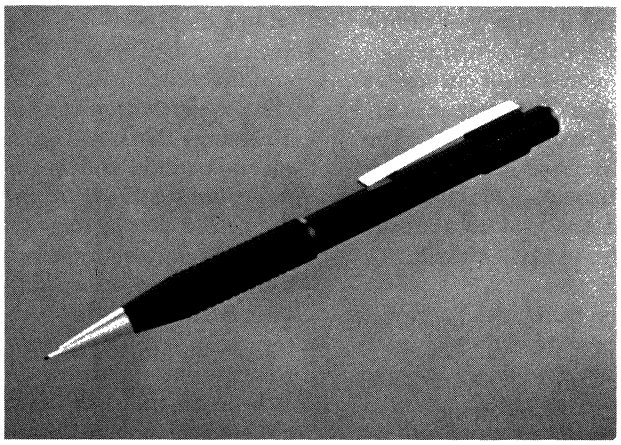
The nice thing about spline curves is that they allow you to treat a collection of pieces of various polynomials as a single entity, maintaining continuity of tangent directions, curvatures, and so on without working at it. Furthermore, NURBS allow you to introduce sharp corners and curvature changes into a continuous curve. (For example, the transition from a flat part of a wall into a rounded corner.) That's the *nonuniform* part of the NURBS acronym, and it refers to the B-spline *knot vector*. NURBS in Alpha_1 are not limited to cubics. Linear, quadratic, cubic, or higher order splines can be used as needed. The order should be kept as low as possible for a given shape to keep the representation concise (polynomials of higher order need more coefficients, which translates to more NURBS control points). Alpha_1 includes an order-raising algorithm to allow exactly combining curves of different orders when desired.

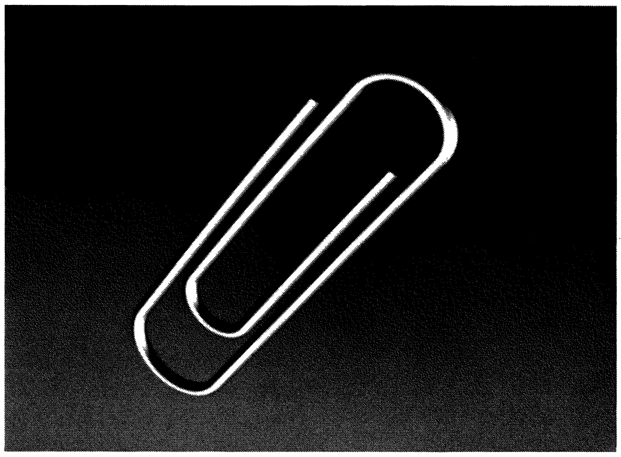
The current technology is to turn curve representations such as NURBS into surfaces using *tensor products*. The easy thing to remember is that anything we say about curves applies equally well to tensor product surfaces, which can be thought of as two continuous sets of cross-curves, going in both directions of a rectangle which is formed into the desired shape. True surface schemes such as tensor products are different from *lofting* schemes, where a set of separate curves is combined in some ad-hoc way that may allow evaluating points on the surface, but may not allow you to determine the shapes of curves in the cross direction. In some lofting schemes you can't even represent the shapes of curves between the the lofted curves. In Alpha_1, the kinds of curves combined into the two directions of a NURBS surface are independent. For example, the surface of a cylinder could be represented by an open linear spline in the flat direction, crossed with a periodic quadratic rational spline in the round direction.

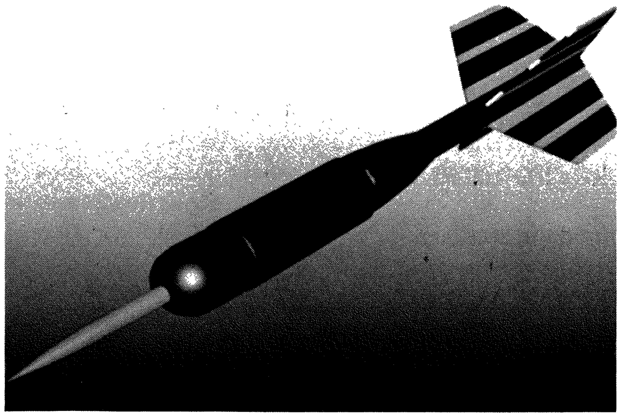
modeling with NURBS

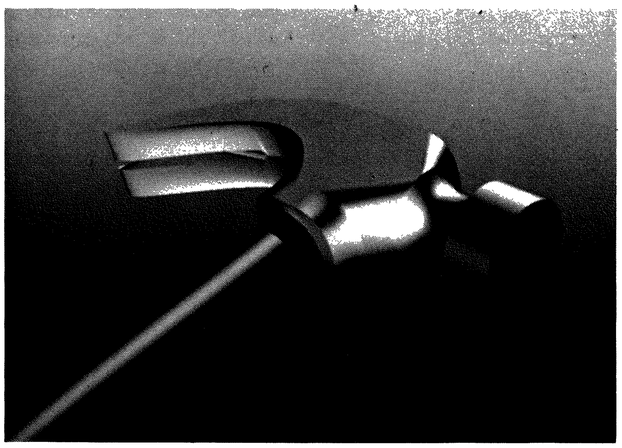
One of the basic ideas of Alpha_1 is that designers need the flexibility of NURBS representations, but should not have to think about mathematical representations when they are modeling. The simple-minded approach to NURBS forces designers to directly specify NURBS in terms of control points, knot vectors, end conditions, and so on, at a level which is intimate with the NURBS mathematics. Directly dragging the control points around in space demonstrates a display device well, but we don't think it is sufficient to help a designer translate his or her idea of a shape into a good model. To allow designers to concentrate on modeling, Alpha_1 has a layered library of object types and design operations, starting with simple plane and 3-D geometry such as dimensions, points, lines, arcs, and circles. Then we jump up to NURBS with some intuitive curve and surface constructors such as profiles, surfaces of revolution, extrusions, filling in between four edge curves, and so on. Modification operations such as bend, warp, stretch, and taper can be used to adjust the initially constructed NURBS into the desired shapes.

Some constructors, such as surface of revolution and extrusion, produce a *shell* of surfaces representing a solid. Other times, it is more natural to construct a set of surfaces independently and then join them into a shell. A shell object is just a structure allowing you to treat a set of surfaces with adjacencies declared between them as a single object. Other structure objects in Alpha_1 include *groups* which wrap sets of objects as a single object, and *instances* which allow positioning copies of objects. Solid shells can be combined by using the CSG union, intersection, and difference operators in an expression ob-









ject. Since boundary representations of solids are combined by trimming surfaces along intersection curves, we have extended the CSG operators to handle partially bounded solids. The solids must be defined in the areas relevant to the intersection curves, but the designer doesn't have to define irrelevant surfaces to close a shell.

Finally, Alpha_1 includes the notion of *parametric object types,* which allow users to introduce new types of objects into the system. Parametric object types can be defined or redefined "on the fly" during modeling sessions. A procedure for constructing the shape description of a part from a few parameters is given once, and then employed as each new instance of the object type is created or modified. In fact, the CSG "primitive" solid object types (sphere, cylinder, block, and so on) are implemented this way. In addition, Alpha_1 offers "rounded edge" blocks and cylinders. All of them construct NURBS shells as their geometry. Expressing design *features* (slots, holes, flanges, ...) as parametric types allows concentrating on the manufacturing of a part, with the geometry following along behind. *Families of parts* or product lines represented as parametric types give an even higher-level approach.

the alpha_1 interactive modeling environment

We like to provide immediate graphical feedback as a side-effect of constructing a model in Alpha_1. With current technology, that usually means showing an interactive line drawing of the objects, hopefully with depth-cued lines. We also like to make the designer's view of the model constantly available for manipulation as part of the graphics environment. Twisting knobs or dragging a mouse to change the view seems to be less distracting than issuing viewing commands.

Alpha_1 has two user interfaces for generating modeling commands, serving different purposes. The first is a command language interface which grew out of our programming environment. Modeling commands are edited in a display oriented text editor, such as EMACS running in a window, and executed line-by-line by the modeling system running as a sub-process. To change the model, a modeling command is edited and re-executed. The modeling system takes care of propagating the change through other objects which have referenced the object being changed. A second user interface, currently under development, interprets a stream of events generated from graphics windows. A designer selects commands from menus, picks objects and indicates positions in windows, and so on. The graphical user interface invokes the same functions as the command language user interface and works in the same modeling environment, so they can be used together.

shaded images for visualizing models

One form of output from an Alpha_1 model is shaded images. The designer gets denser feedback from a shaded image than from line-drawing images. In fact, there is enough information in high-quality shaded images to serve as a partial substitute for the traditional process of building physical prototypes from drawings. The design process for complex parts is greatly speeded as a result. Alpha_1 includes utility programs for high-quality shaded rendering. There is always a trade-off of time versus quality, however. *Scanline Z-buffer* rendering produces an image in a few minutes, with relatively realistic lighting, surface shading, and transparency. *Ray tracing* takes longer, but adds realistic features such as shadows, reflections, and refraction through transparent objects. Alpha_1 rendering utilities produce *Run Length Encoded (RLE)* image files. The *Utah Raster Toolkit* contains tools for combining RLE images and displaying them on a variety of frame buffers. The Utah Raster Toolkit is publicly available and also provided along with the Alpha_1 and BRL-CAD distributions. (See pg. 7 & 8 in the Universe.)

During modeling sessions with Alpha_1, we like to pop up shaded image windows under the window manager on a design workstation. Shaded images are also useful in documenting completed models. Workstations are starting to provide shading hardware which partially fills the gap between interactive line-drawing images of models and rendering utility programs. Often, you can get a low-quality shaded image in a small number of seconds. This is still not fast enough to gain understanding of a shape by rotating it in real time as you twist a knob, so line drawings still have a niche. The next generation of workstations will come closer to being able to generate interactive shaded images, and Alpha_1 will put that capability to good use.

future

Alpha_1 is a dynamic, evolving system. We use its extension capabilities in our development, as well as placing them in the hands of users for customization. Research and development continues on many topics including the graphical user interface, process planning, and N/C machining and Finite Element interfaces from Alpha_1 models.

Another interesting line of development involves adding variants of NURBS into the Alpha_1 modeling environment as new object types. *Multivariate splines* allow some freedom from the "rectangular topologies" of tensor product surfaces. *Trimmed surfaces* are another approach to hiding parametric rectangles. In essence, Alpha_1 already trims surfaces in the evaluation of CSG expressions, and we merely need to make the trimming operations available at a smaller granularity to the designer.

We are starting to distribute Alpha_1 to make the tools we have developed available for others to use. A small company has been formed in cooperation with the University of Utah to further the development and distribution of Alpha_1. Alpha_1 is now in preliminary distribution to selected university, industrial, and government sites.
Please address queries on Alpha_1 to:

Richard F. Riesenfeld
Engineering Geometry Systems
2685 Eagle Way,
Salt Lake City, Utah 84108
<rfr@cs.utah.edu> (ARPA)
{inhp4,decvax}! utah-cs! rfr (UUCP)

Ballistic Research Laboratory solid modeling system and ray-tracing benchmark package

Release 1.15 (9-Jan-87)

The BRL CAD Package is a solid modeling system, with ray-tracer, framebuffer library, and lots of image tools. We have tested this software on Silicon Graphics' IRIS's, VAXen running 4.2 and 4.3 BSD, Gould PowerNodes, Sun Workstations, Cray XMP, Cray-2, Alliant FX/8, and Pyramids. (Benchmark comparisons as well as the benchmark itself, are included as part of the software distribution, so you can see how your favorite machine stacks up). The software included in this package is only the tip of the iceberg. The next release is expected to contain a serious image- and signal-processing toolkit to complement the existing tools. Hopefully, many of you will find this a worthwhile software base to build upon, and will contribute your favorite graphics tools to round out the collection. This distribution does not include the various BRL-specific model analysis tools such as GIFT, SAR, SLAVE, VAST, etc.

communication and enhancements

New releases of this software will be issued roughly semi-annually. Information about new releases will be routinely provided to recipients of this software. Request a subscription by sending to <CAD-REQUEST @ BRL.ARPA>. If your address changes, please let us know, so we can update our records. You are invited to participate in the BRL CAD software mailing list, called <CAD @ BRL.ARPA>. Bug reports and discussions of new features are the main topics; volume of messages has been light (so far). If you find bugs, please report your experiences. *While BRL makes no offer of support, we are most interested in hearing about your experiences.* If you develop additional software for the BRL-CAD environment that you would be willing to share, please send it to us for inclusion in the next release.

The software is distributed free of charge under certain conditions. For information and to obtain a copy of the distribution, you should contact:

Mike Muuss
Leader, Advanced Computer Systems Team
U.S. Army
Ballistic Research Laboratory
APG, MD. 21005-5066 - USA
<mike@BRL.ARPA>

SGI demos

If you would like the source code to some of your favorite IRIS demos: dog, flight, radar, shadow, heme, insect, jet, shuttle, robot, surfcar and zshadecar, contact Monica Schulze at Silicon Graphics, 415-962-3320 or ucbvax!sgi!monica. She will send you a demo source agreement and ask you to send in a blank tape for copying.

There are a couple of other places to find IRIS code examples. There is a sample code section in the IRIS User's Guide. The IRIS Programming Tutorial contains source code examples including one program, *viewport*, with extensive comments on programming within the window manager. Finally, each software release has the source to a mex and non-mex arch demo.

BRL-CAD contents

libsysv	Some System-V compatability routines
mged	A solid-model editor
librt	A solid-model ray-tracing library
rt	A ray-tracing lighting model, for rendering
db	Several solid-model databases, in ASCII form
conv	ASCII/binary database converters
bench	Scripts to drive the RT benchmark
pix	Reference images for the RT benchmark, in ASCII form
libpkg	A "message-passing" interface to TCP network links
libfb	A generic frame-buffer library
rfbd	TCP server for remote frame-buffer access
libtermi	A library to handle terminal mode setting
libplot3	A public-domain 2-D and 3-D UNIX-Plot library
librle	A Run-Length-Encoding library (originally from UofUtah)
util	Zillions of image-handling utilities, as tools
fbed	Frame-buffer image editor
vdeck	Convert mged models to GIFT-format card decks.
dmdfb	libfb support for layers in Teletype 5620 DMD terminal

utah raster toolkit

The Utah Raster Toolkit is a collection of programs for manipulating and composing raster images. These tools are based on the Unix concepts of pipes and filters, and operate on images in much the same way as the standard Unix tools operate on textual data. Some of the commands available in the package include:

an image compositor;
  for downfiltering images;
  for repositioning images;
tools
  for rotating and stretching images;
  for providing various backgrounds;
  for manipulating individual color
    and matte channels;
  for manipulating color tables; and
  for converting images from 24 to 8
    bits

The Raster Toolkit uses a special run length encoding (RLE) format for storing images and interfacing between the various programs. This reduces the disk space requirements for picture storage and provides a standard header containing descriptive information about an image. Some of the tools are able to work directly with the compressed picture data, increasing their efficiency. The RLE format allows for additional data, such as Z values, to be stored with the image. A library of C routines is provided for reading and writing the RLE image format, making the toolkit easy to extend.

In order to display RLE images, a number of programs are provided for displaying the pictures on various devices including the IRIS. These display programs all read from standard input, so they are conveniently used as the end of the raster toolkit pipeline. The common interfaces of the RLE format and the subroutine library make it easy to interface the toolkit to a wide variety of image sources and displays.

Additional routines are available for generating dither matrices (e.g. for display programms running on devices with less than 24 bits of color), and for converting images to PostScript. The raster toolkit also includes Unix man pages for the library and commands, some sample images, and additional documentation.

See the paper by Peterson, Bogart, and Thomas in the Proc. of the 3rd USENIX Graphics Workshop 1986. It describes the individual tools, gives several examples of their use and how they work together. Additional topics that arise in combining images-- how to combine color table information from multiple sources, are discussed.

distribution

For ARPAnet sites, the Toolkit may be obtained via anonymous FTP to the site *utah-cs,* in the file *pub/toolkit.tar.* Sites not on the ARPAnet can obtain the Raster Toolkit on a 9-track, 1600 bpi tar format tape by sending check or money order for $200.00 payable to the CS Dept. to:

Utah Raster Toolkit, Loretta Cruse
University of Utah
Computer Science Department
Salt Lake City, UT 84112

Courtesy of Mike Muuss at BRL, the Raster Toolkit is also included as contributed software in the BRL-CAD distribution.

Although the Raster Toolkit software is copyrighted, it may be freely redistributed on a "GNU-like" basis. For further technical information on the Raster Toolkit, send mail to:

toolkit-request@cs.utah.edu
(ARPA)
{ihnp4, decvax} ! utah-cs ! toolkit
(UUCP)

gnu emacs

by Tim Moore
<tbmoore@athena.mit.edu>

Gnu Emacs is a text editor that is available for most Unix systems. It is a fantastic editor for several reasons. As a screen editor it provides mnemonic key strokes to do almost any editing command one could think of, such as cursor movement, cutting and pasting, insertion of files, etc. It offers specialized modes for things like writing programs in different languages or writing papers using *TEX* or *scribe.* Most of the editor is written in Lisp, and one can easily write extensions to Gnu Emacs in this Lisp. If you can think of a function that Emacs doesn't have, add it! Finally, Emacs is free, which is pretty amazing. Richard Stallman, the author, is head of the Free Software Foundation and is dedicated to spreading high quality software at close to zero cost.

For basic text editing, key strokes such as 'C-f' ('Control f') and 'C-b' move the cursor forward and backward one character. More complicated commands are based on the fundamental key strokes. For example, 'M-f' ('Escape f') moves the cursor one word forward. Many different keystrokes provide most of the basic text mode functions. However, if you forget the keystroke, or want to execute a function that doesn't have a key binding, you can type 'M-x' and then the function name. For example 'M-x auto-fill-mode' puts Emacs in auto-fill mode, the mode this writeup was typed in. If you really forget the key binding, Emacs has *extensive* help facilities, including an interactive on-line tutorial.

For editing programs and other stuff such as sending and recieving mail, Emacs has more specialized modes. When editing C or Lisp (and now FORTRAN) programs the editor can indent lines according to the program's structure. It also provides key bindings to

editor

move around the program, i.e. the next function or block, and functions to things such as aligning comments. You can compile programs from inside Emacs. If the compilation is unsuccessful, Emacs will parse the error messages and jump to the corresponding file and line.

The large majority of Emacs is written in Lisp. This makes it very easy to customize the editor by either changing key bindings or writing new functions in Lisp. In order to set the indentation level in C mode to 4 spaces, you could type "(setq c-indent-level 4)". You can add any customizations into a startup file, making their execution automatic. Extensive customization is possible through Lisp. When X window features were added to Emacs, most of the code was written in Lisp with a very small amount of C support. Emacs provides a Lisp interaction mode to interactively evaluate constructs in Lisp.

Gnu Emacs is free, or very cheap if you buy it on tape. This is makes it very attractive for Unix users. However, even if it cost several hundred dollars it would still be just as attractive, due to its richness of features. It is available via anonymous FTP to

prep.ai.mit.edu.

See /u2/emacs/ GETTING.GNU.SOFTWARE on that machine for details on copying. If this route is unavailable to you, write to:

Free Software Foundation
1000 Mass Ave
Cambridge, MA 02138

The configuration files necessary to get GNU Emacs running on an IRIS 3030 are in the info-IRIS directory at Sumex-aim.

info-IRIS network

Info-IRIS is a network interest group for users of Silicon Graphics IRIS workstations. It is a means for IRIS users to communicate with, and learn about, other groups using these machines. Within this arena, people can share hints they've discovered, useful tools they've developed, problems they've encountered or overcome, and descriptions of work they're doing.

The list is set up as a "distribution list" of subscriber addresses, so that any message sent to Info-IRIS is simply copied and redistributed to each name on the list. The list is maintained on an ARPANET host at Stanford University, but is accessible to users on the other major networks as well. There are currently approximately 100 subscribers from ARPANET, CSNET, BITNET, MFENET, MAILNET and UUCP, with some sites keeping their own local distribution lists. An archive of messages is also kept, so that new users can see what issues have been discussed in the past.

A recent addition has been the setting up of a repository for public-domain software available for the IRIS that has been developed by Info-IRIS subscribers. These could be utilities, bug-fixes, demos, fonts, anything that the author feels is of general interest to the IRIS audience. The programs are stored in a directory on the Stanford host, and are available to users either through FTP or physical mail (magnetic cartridge).

To join the list, or simply to get more information, send a message to:

Info-Iris-Request
@Sumex-Aim.Stanford.Edu *(ARPA)*

{ucbvax,phri}!sumex-aim.arpa!
Info-Iris-Request *(UUCP)*

For other networks, consult your local postmaster on routing to the ARPANET or send physical mail to:

John Brugge
Knowledge Systems Lab
Stanford University
701 Welch Rd., Bldg. C
Palo Alto, CA 94304

IRIS bulletin board

If you would like to access the latest information about the IRIS and its user community, dial the IRIS Bulletin Board--an on-line, no charge service for all IRIS users with information on the following subjects: general IRIS and current release information, listing of critical bugs, bug submissions to SGI, high tech programming tips, listings of User Software Exchange, current Geometry Partners and Special Interest Groups.

The customer bulletin board is a menu-based program that lets you communicate informally with other IRIS users and with Silicon Graphics. You access the bulletin board through a 1200 baud modem. Just dial (415) 965-0303 and log into the system. To receive a login name and a password, call:

Monica Schulze
Silicon Graphics, Inc.
Technical Marketing Dept.
(415) 962-3320
E-mail to: ucbvax ! sgi ! monica.

IRIS interaction at SIG-CHI

by Mason Woo, Silicon Graphics Inc.

Be on the watch for IRIS superworkstations in the Great White North. The popular 3-D Exploratorium, SGI's electronic classroom, is coming to the SIGCHI+GI conference in Toronto from April 7-9. The Toronto Hilton Harbour Castle is serving as host to the CHI+GI 1987 conference (Human Factors in Computing Systems and Graphics Interface). Silicon Graphics will be running hourly electronic classroom sessions throughout the entire conference. Attend and experiment with 3-D, color, wireframe and shaded graphics in motion with interactive educational software. In addition, 7 IRIS application demonstrations from the University of Waterloo, National Film Board of Canada, Alias Research Inc., and Silicon Graphics are scheduled to appear. These demonstrations are described below. We hope you'll join us in Toronto!

spline-based curve matching experiments

demonstrator: Eric Bosch, University of Waterloo

Splines have been used effectively in computer graphics to model smooth two and three dimensional shapes. This demonstration focuses on a computer-arbitrated experiment which investigates whether there is an undiscovered hierarchy of effectiveness in spline methods. The data gathered from a spline-matching pilot study are discussed.

interactive definition, editing and contouring of surfaces

demonstrator: Robert Dickinson, Univ. of Waterloo

This system demonstrates the definition and interactive editing of spline approximations to functions of two variables. The techniques demonstrated are applicable to the analysis of field problems in engineering, free-form surface design, and other sciences. The system can interpolate and preprocess point data, and edit single-valued functions and spline surfaces. Both interpolation and editing utilize a contouring subsystem designed specifically for B-spline surfaces.

conman: a graphical data-flow environment

demonstrators: Paul Haeberli, Rob Myers, SGI

This demonstration will show a data-flow environment which knits independent windowed programs together to build flexible, applied configurations at the user's interactive, graphical behest. In the data-flow environment, each process in a set of interconnected Unix processes, is given up to eight input ports and eight output ports. A data-flow connection manager Conman, allows the user, on the fly, to connect an output port of one process to an input port of any other process.

interactive kinetic control of keyframe computer interpolation

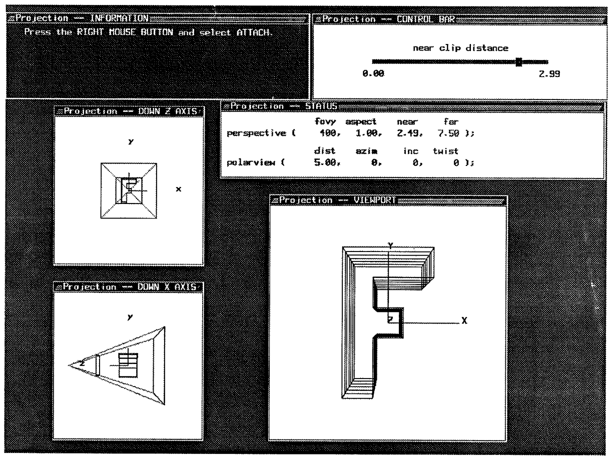
demonstrator: Ines Hardtke, National Film Board of Canada

The experimental system is a mini keyframe interpolation system designed to test two different methods of kinetic control over the inbetweening process. Given a script and some key drawings, the program interpolates the sequence and allows the animator to interactively change the timing of the resulting motion in two ways: a) through the use of a "time-line"; b) through the use of a "speed-line."

palette: an RGBA paint program

demonstrator: Terry Higgins, National Film Board of Canada

"RGBA" refers to images whose pixels include an opacity or "alpha" component in addition to red, green or blue. This makes it possible to composite images in a way that preserves the anti-aliasing of


Dynamic Environment for 3-D Graphics Programming

```
Press the RIGHT MOUSE BUTTON and select ATTACH.
```

```
                fovy  aspect   near    far
perspective (   400,   1.00,   2.49,  7.50 );
                dist    azim    inc   twist
polarview (     5.00,     0,     0,      0 );
```

Projection -- INFORMATION
Projection -- CONTROL BAR
near clip distance
0.00
2.99
Projection -- DOWN Z AXIS
Projection -- STATUS
Projection -- VIEWPORT
Projection -- DOWN X AXIS
y
x

edges and translucency of areas, with beneficial consequences for computer-assisted cel animation, special-purpose renderers, and interactive digital "cut and paste" applications. Palette is the first published design of an RGBA paint system.

alias/1 user interface

demonstrator: Dave Springer, Alias Research Inc.

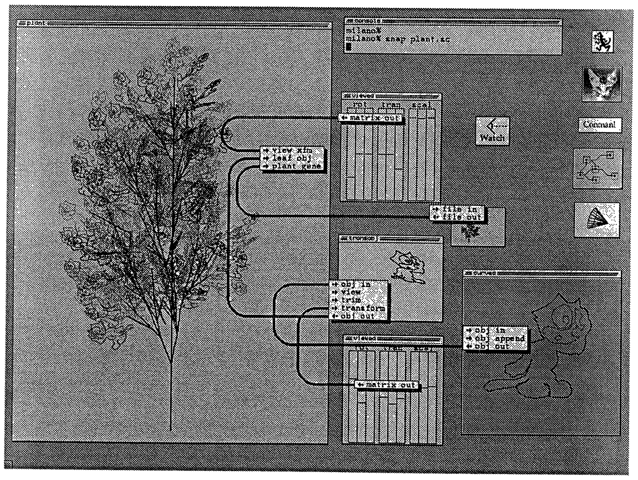
Alias/1 is a commercially available system for three-dimensional graphical design, animation and video/film production. The user interface to this system was designed and developed for industrial designers and graphics artists who have little or no computer training. With Alias/1 a designer can interactively construct 3-D solid models, assign movement and other dynamics, view a wire-frame representation of the animation instantly, and render the scene using high-resolution, anti-aliased, ray-casting techniques.

dynamic educational environments for 3-d graphics programming

demonstrator: Mason Woo, Silicon Graphics Inc.

The demonstration will focus on the user interface implementation for a series of on-line educational tools for teaching 3-D graphics. These "Graphics Labs" were developed to link the visualization of 3-D graphics and the mathematical values of a graphics routine. The demonstration supplements the presentation of the ongoing 3-D Exploratorium electronic classroom shown at this conference.

Conman: Graphical Data-Flow Environment



```
milano%
milano% snap plant.sc
```

IRIS user groups

two northeast user groups

RCA is hosting a meeting for IRIS users in the New York, New Jersey, Connecticut and Pennsylvania area. It will take place at RCA/ESD in Moorestown, New Jersey on Wednesday, April 22 from 10 a.m. to 2 p.m. For more information contact Curtis Bell or Ed Chancy at RCA, 609-722-4248 or 722-4264, or Tim Sherbak or Donna DeVinko at the Silicon Graphics' Paramus New Jersey office, 201-599-2172.

Bill Linnane of MIT Lincoln Labs is organizing the second user group meeting in the Boston area. The meeting is tentatively scheduled for April 29, at the Silicon Graphics' Eastern Region office, 128 Technology Center, Waltham, Mass. 02154. Contact Bill at MIT Lincoln Lab, Box 73, M/S L-109, Lexington, Mass. 02173, phone 617-863-5500 ext 4849 for more details.

2nd international IRIS user group meeting

The 2nd International IRIS User Group Meeting will be at the SIGGRAPH '87 conference, during the week of Jul y 27 in Anaheim, California. The goals of last years' meeting ,attended by over 150 IRIS users, was to promote exchange among IRIS users and to provide a feedback channel to Silicon Graphics. Mike Muuss of Army Ballistics Research Lab presented the BRL-CAD software, Paul Haeberli of Silicon Graphics demoed a prototype of his graphical dataflow environment, Conman, Jim Clark spoke about the future directions of Silicon Graphics, and the meeting ended with a question and answer session and reception.

If you have suggestions for workshop topics you'd like to participate in or other activities you'd like to see at the upcoming User Meeting at Siggraph '87, please contact Zsuzsanna Molnar at Silicon Graphics, 415-962-3315.

special interest group: real time simulation

If you are working in the field of visual simulation, this group is forming to discuss issues such as cockpit displays for flight simulation--symbology, sensor & radar data, instructor & role playing displays--C3 displays, and IRIS communication with host computers. See the notefile on the IRIS Bulletin Board or contact:

Dr. Sarah E. Stead
Bell Helicopter Textron, Dept. 87
P. O. Box 482
Fort Worth, TX 76101
(817) 280-6157

animation in Berlin

The first two. computer animation companies in Berlin, West Germany, *D.I.N.O. Productions* and *mental image,* are IRIS users. We expect to see visual wonders fueled by Berlin's renowned scientific heritage and its contempory art scene that took New York and Los Angeles by storm. The *Wild Expressionists* are coming to computer graphics...

europe's broadcasting giant

Broadcasting in several languages from one of the smallest European countries, Radio Television Luxembourg reaches the largest audience all over Europe. For many years RTL was one of the few European commercial stations in an otherwise government controlled broadcasting scene. Recently, together with the French company SESA, RTL launched the *Centre Europeen de Recherches d'Images de Synthese* CERISE. CERISE is part of the project EUREKA, a high tech research and development program founded by 18 European countries in 1985. IRIS workstations running Wavefront software are being used for initial production purposes and image research.

the interactive image

An exciting new project is being developed by the faculty and students of the Electronic Visualization Laboratory at the University of Illinois at Chicago for Chicago's Museum of Science and Industry. Called *The Interactive Image,* it is a progressive interactive computer graphics exhibition that will demonstrate the latest advancements in science and technology to a large and enthusiastic lay audience.

Tom DeFanti, past chair of ACM/SIGGRAPH and editor of the SIGGRAPH Video Review, will be responsible for linking an IRIS to the CRAY X-MP at the National Center for Supercomputing Applications at the University of Illinois at Urbana-Champain for the interactive installation *The Fourth Dimension.* This installation will illustrate the value of color-enhanced scientific data displayed on graphics workstations. Audiences will experience first-hand the computational power and speed of today's supercomputers, and will discover how 4D diagrams and other complex images help the scientist in analyzing millions of data points, such as an astronomical star cluster.

Dan Sandin, inventor of the Sandin image processor and recipient of several Guggenheim and Rockefeller grants, will stage an exhibit of *PHS*Colograms that combine *P*hotography, *H*olography, *S*culpture, Computer Graphics and video to produce startling 3D images. Unlike holograms, Phscolograms are in full color. They range in size from 11 x 14 to 32 x 48 inches and, displayed as backlit transparencies, appear to jump across the room. Images are mathematically generated by defining parameters to control viewing angle, color and shading. Resulting images are photographed directly off workstation monitors and several of them are processed into one Phscologram.

Other exhibits planned include *Zanimation, Interactive Fractal Space, Laser Space, Escher,* and a computer controlled *TV Switcher Panel.* Interactive videodisks from SIGGRAPH's Artshow will complete the show.

The first phase of this educational (yet entertaining) show is scheduled for October 24, 1987 through January 10, 1988, a larger, more elaborate show is planned to take place 12 to 18 months later. The show is organized by SIGGRAPH veteran Maxine D. Brown. She can be reached at:

Maxine Brown
University of Illinois at Chicago
Electronic Visualization Laboratory
EECS
POB 4348
Chicago, IL. 60680
(312) 996-3002

the puzzling story of Silicon Graphics' logo

by Frank Dietrich

In August 1982, when Silicon Graphics was a company of only a handful of Stanford engineers, just founded to transfer new research results into the high-tech market place, the need for a company logo came up. Jim Clark, inventor of the Geometry Engines, VLSI processors that are optimized to rapidly compute the bread-and-butter job of transforming, clipping, scaling and projecting 3-dimensional objects onto a 2-dimenional display, had heard about Scott Kim's design talents.

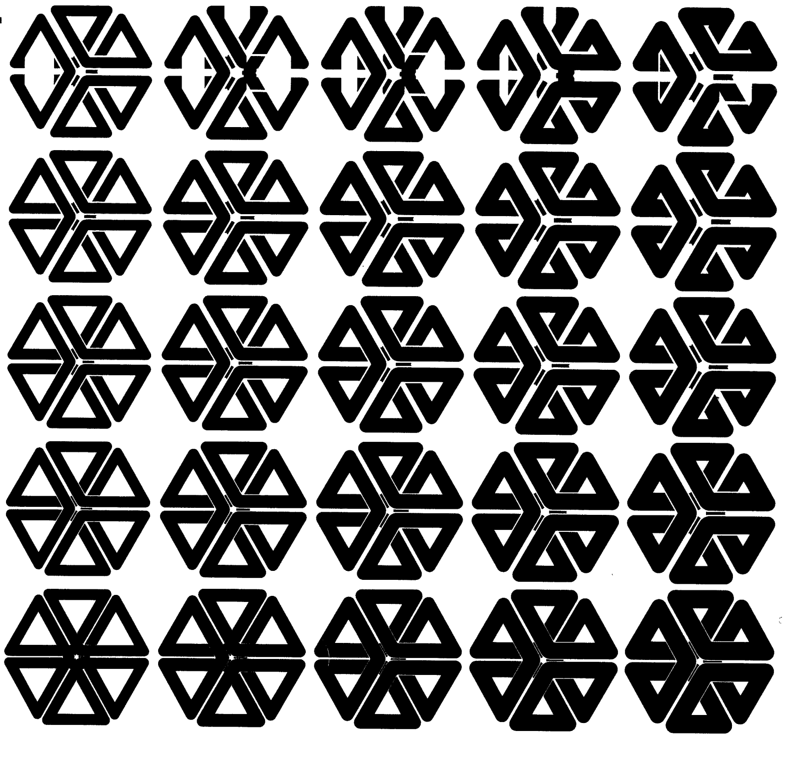
a visual thinker for a visual technology

At the time, Scott Kim was already well known at Stanford University and in the computer graphics community for his puzzling calligraphic *inversions*, hand written words that could be read upside down or right to left. (In the meantime, these *Inversions* have been published as a book with the same title by MIT Press. Soon, they will be made available as an interactive program for the Macintosh.) Scott Kim combines many skills in one single person as if he himself is one of the n-dimensional tessaracts that he designs, only revealing some of the multitude of his dimensions at a time, then switching state and showing another side. He is a puzzle master, inventive calligrapher, computer graphics designer and computer scientist. Currently, Scott is putting the finishing touches on his PhD thesis on a visual programming system that he is writing under Donald Knuth at Stanford.

No one single description does him justice because his creative spirit evades definition. If I would have to find a term to encompass all his professional activities, I would call him a *visual thinker*. During our interview, Scott used his notepad extensively to elegantly sketch his thoughts, to guide his cognitive process, to test his mental models, and to communicate his conclusions. If it is impossible to draw a satisfying portrait of Scott in these few lines, let me just describe the paths his thinking traveled when confronted with the task of designing a logo for Silicon Graphics. Kim felt challenged to create a visual representation that could symbolize the main concept of the three-dimensional dynamics of the VLSI technology that was to set new standards in computer graphics.

digital design of a logo

Scott Kim's guiding motto was "to design something engaging intellectually." It is noteworthy that he fused method and message in one synergetic process. By using the page description language JAM, the precursor of PostScript that was under development at Xerox at that time, Scott started a design process where "pen never touched paper." Scott felt it was "an appropriate use of computers to write a program instead." Earlier he had already programmed the well known spiral of *Infinity* in JAM. The main question was, what the logo should depict. After several attempts to use the company's name in form of an inversion, Kim disregarded this direction altogether. He thought more about the technology encapsulated in the Geometry Engines and finally focused on 3-dimensionality as the topic that needed to be concisely visualized. Scott chose the basic building block of 3-dimensionalty, the cube, and defined its outside vertices by elegantly looping an endless tube around. By rounding the corners of the cube, he created "a room without sharp edges, a room without walls." After experimenting with several versions of the Necker cube that can be seen as being oriented



either into a page or coming out of it, Kim settled on a different type of optical illusion, one between two and three dimensions. This was achieved by rotating the cube precisely so that the resulting 2-dimensional hexagon could be read as a geometric pattern of a stylized flower. With this mindbending move he instilled ambiguity into the visual meaning of the design. Now the logo oscillates between connotations of technology and organic growth, it perceptionally hovers between two and three dimensions without ever settling down into a simply explainable world. In short, the logo dynamically changes meaning, yet it preserves a clearly recognizable appearance.

JAM made important interactive and interpretative features available to Scott that we see demonstrated in the illustration above. The picture shows several versions of the logo that are mathematically derived in sequence.The illustration is an example of the 880 dpi hardcopy laserprints that Scott used as proofs throughout the process. At this point, Scott felt it necessary to bring in "another pair of eyes", the designer Suzanne West.

the Swiss approach to design

Suzanne West followed the basic, but powerful rule of Swiss design, "find the simpliest solution whose visual potential can never be outgrown." Her artistic credo told her that beauty cannot be quantified. Therefore, she insisted on finding the aesthetically most convincing version of the logo not by proceeding systematically mathematically, but by proceeding systematically in visual terms. Again, the interactive qualities of JAM came into play when Suzanne continued to fine-tune the parameters to create a unified image that would maintain the proper "spaceflow" as a figure as well as in the carved-out background, and that could survive abuses through different printing processes. In particular, she had to consider the proportions and the graphical treatment of curvature, angles, in addition to the negative distance and weight of the lines within the composition. Modifications of the final proof were made to account for the bleeding that occurs in silk screening, and camera-ready variations were designed for different scales of the logo, from the embossed business card to the huge trade show display. Finally, colors needed to be selected. Suzanne developed two alternative color schemes. A calm blue universally appeases and is a rather "safe" color, whereas her purple option was more "exciting and adventuresome." Suzanne was suprised to see that Silicon Graphics' executives right away decided on the purple alternative to become the corporate color.

The visualization of an idea that is technology specific was accomplished by an interdisciplinary collection of thought-experiments, computerized tools, and a designer's sense for aesthetics. In my mind I already see Silicon Graphics' logo as an abstract character in a future computer animation, gradually evolving over time, and thereby visualizing growth and a dynamically changing technology.

computer graphics in mathematics: viewing parametrized surfaces

Dietmar Saupe
Department of Mathematics
University of California, Santa Cruz
Santa Cruz, CA 95064
<saupe@ucscc.ucsc.edu>

and: Institut für Dynamische Systeme
Fachbereich Mathematik
Universität Bremen
D-2800 Bremen-33
West Germany

1. what can computer graphics do for mathematics?

Computers and in particular computer graphics have long become valuable tools for researchers in engineering and in the natural sciences. Molecular modeling, CAD/CAM and image processing are just some of the many disciplines in which computer graphics has been established as an intrinsic and indispensable element. But it is also offering significant potential to workers in the more abstract realms of mathematics, and in fact it already has provided insights that led to major discoveries most notably in dynamical systems and differential geometry. In 1964 the meteorologist E. N. Lorenz discovered the first chaotic attractor in an attempt to solve "the problem of deducing the climate from the governing equations" [Lor64]. Although he did not solve the problem, his computer graphical discovery of a strange attractor which is now named after him, inspired a whole generation of physicists, mathematicians and other scientists to study systems that previously had been discarded as being too complex and inaccessible.

Only in 1980 B. Mandelbrot at IBM and his coworkers ran some innocent computer experiments motivated by the 70 years old deep theory of the French mathematicians G. Julia and P. Fatou [Man80]. They studied families of iterated rational maps constructed in such a way that some of them exhibit chaos similar to the chaos found in the Lorenz attractor. What he found, was a somewhat fuzzy looking set with a few apparent "specks of dirt" around it which however failed to vanish in an attempt to clean up the picture by using better computer equipment. Little did he know that soon his name would be attached to this unusual set which now has emerged to become the "most complicated object in mathematics" as well as the favorite object in computer programs written by countless amateur and many professional scientists around the world. The "specks of dirt" as we now know, of course, were little satellite Mandelbrot sets connected by infinitely thin strands to the main body of the Mandelbrot set. For an interesting account of the history of the discovery of the Mandelbrot set, see Mandelbrot's contribution [Man86] in *The Beauty of Fractals* by H. O. Peitgen and P. Richter [Pei86].

Where the above two examples fall into the category of dynamical systems, the computer graphical discovery of a new so called minimal surface in 1985 has excited the world of differential geometry. At Rice University in Houston D. Hoffman and W. Meeks investigated a particular surface in three dimensional space of a certain topological type. It was already known to be minimal and complete, i.e. roughly speaking it is a surface of least area and without boundary. Computer graphics confirmed their conjecture, that it is also a surface without self-intersections, and in the following they were able to prove that fact, which makes that surface the third known example in its category besides the helicoid and the catenoid and the first new one to be found in over 200 years [Hof85, Pet85].

the only way to learn mathematics is to experience it

The above examples show that the real power of computer graphics comes in where it allows to visualize constructs that otherwise exist only in the pure form of mathematical abstractions. Another striking example is displayed on the cover of this year's January issue of *The Mathematical Intelligencer* : the hypersphere. The hypersphere is the straight forward generalization of the usual sphere in the four dimensional Euclidean space. Yet it is so hard to imagine, since our ways of thinking are accustomed to the only three dimensional space that surrounds us. However, taking certain cuts of the hypersphere and their stereographic projections facilitates computer graphical viewing and aids in understanding the geometry of the underlying object [Koc87]. Below we will return to the hypersphere discussing a particularly interesting surface that "lives" in it: the Klein bottle.

A computer graphics workstation such as the IRIS 3030 is an ideal tool for the experimental investigation of such mathematical objects. It may well provide the testing grounds for conjectures, and, when used wisely may suggest a wealth of new ideas and problems. A decade ago there were probably less than a handful of sophisticated graphics devices to be found in the mathematics departments of this country. Now the potential of computer graphics has transformed into a need, and new academic programs in computational mathematics including mathematical computer graphics are mushrooming at universities in many countries. But the benefits of computer graphics in mathematics are not restricted to research alone. Viewing of curves, surfaces, etc. in two or three dimensions and the interactive, realtime simulation of dynamical systems may provide the student with intuitive and lively insights in the theory that previously could have been communicated only via dry lectures and books. The author is convinced that in the not so far future traditional courses in analysis, differential equations, geometry, numerical analysis will be complemented and enriched by hands-on computer graphics projects. It has offen been stressed by teachers that "the only way to learn mathematics is to do mathematics". Now we are tempted to append "... and to experience it"!

2. viewing parametrized surfaces

It is out of the scope of this paper to discuss a computer graphical application from the frontiers of research and refer the reader to the literature listed below. More modestly we restrict to a simple yet interesting problem, namely the interactive display of parametrized surfaces, a task ideally suited for the IRIS workstation. Of course, parametrized surfaces are used widely in computer graphics, and we do not intend to contribute new techniques for the already initiated computer graphics user. In the following we will briefly describe the concept of parametrized surfaces and their interactive display. Moreover, we will demonstrate some of the diverse examples along with formulas such that the interested reader may well reproduce most of our pictures. This application has proven to be very useful as part of an introductory graphics course for mathematics students and as a demonstration tool in a differential geometry course at the University of California, Santa Cruz.

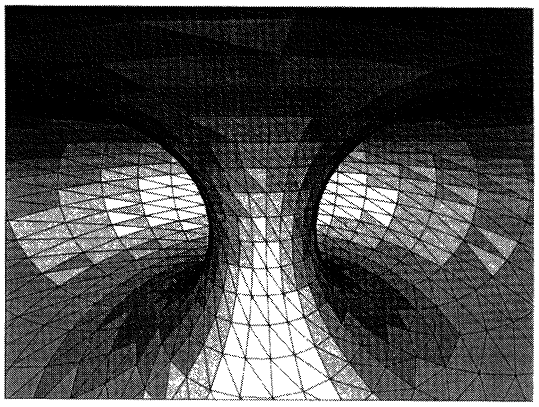
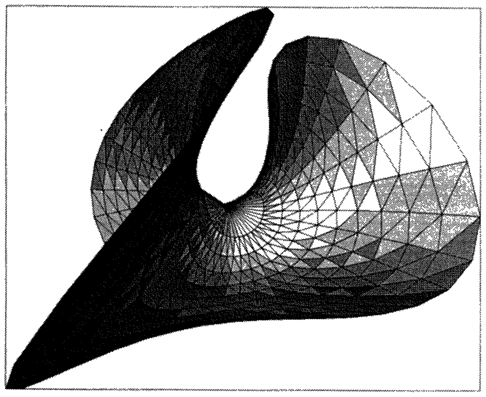
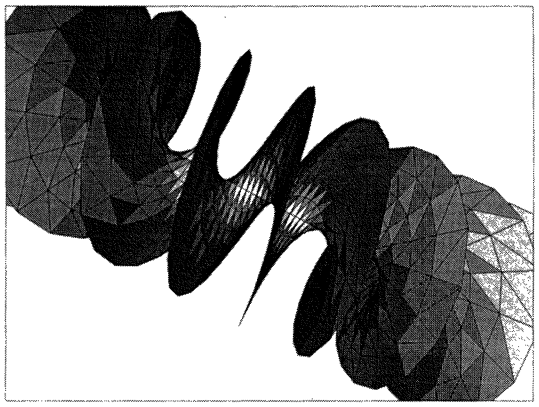
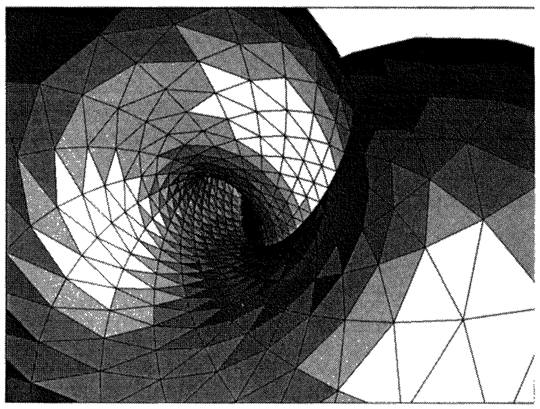
A first definition of a parametrized surface is given by a continuous map X with an (open) subset U of the plane R^2as its domain and taking values in the 3-dimensional Euclidean space R^3. Typical examples are graphs of functions $f : R^2 \to R$ such as $f : (u,v) = \exp(-(u^2+v^2))$ yielding a bell shaped (infinite) surface. Here the map X would be written as $X(u,v) = (u,v,f(u,v))$. There are examples of nondifferentiable parametrized surfaces, such as fractal landscapes, however, usually it is required that X satisfies some differentiability conditions. If additionally the surface admits a tangent plane at all points $X(u,v)$, i.e. more precisely, if the derivative of X is nonsingular at all points $(u,v) \in U$, then the surface is called a regular, parametrized surface. For an introduction to differential geometry which includes the theory of regular surfaces we recommend [Car76]. The well known torus is a good example for a regular, parametrized surface. Here

$$X(u,v) = ((r \cos u + R) \cos v, (r \cos u + R) \sin v, r \sin u),$$
$$0 < u < 2\pi, 0 < v < 2\pi.$$

The surface is obtained by rotating a circle of radius r in the xz-plane and center $(R, 0,0)$, $R > r$ around the z-axis. In this example the domain U of$X$ is the square of length 2π. In a computer program which should be capable of displaying a whole palette of surfaces it should be convenient to parametrize all surfaces over the unit square $(0,1) \times (0,1)$. This is possible for all of the examples discussed here. For the torus we would simply scale up our (u,v)-coordinates of the unit square by a factor of 2π before applying X. The next step is to impose a rectangular grid on the unit square. After choosing the numbers n_u and n_v of subdivisions on the u- and v-axis we obtain $(n_u + 1)(n_v + 1)$ equally spaced grid points

$$(u_i, v_j), i = 0, \dots, n_u, j = 0, \dots, n_v$$

from top to bottom:
View from inside the torus with radii 1 and 3
Enneper's surface parametrized by polar coordinates
The helicoid
Mouth of the Klein bottle, self intersection near center

with

$$u_i = i / n_u, v_j = j / n_v$$

in the domain U of X and corresponding points $X(u_i, v_j)$ on the parametrized surface. A wireframe model of the surface can now easily be drawn as a family of coordinate lines, effectively mapping the grid of the unit square onto the surface. The picture can be made clearer by choosing different colors of the coordinate lines in the u- and v-direction. Also, depth-cueing enhances the 3-dimensional character of the object.

For the purpose of shading we may further subdivide the grid squares $(u_i, u_{i+1}) \times (v_j, v_{j+1})$ into two triangles which are then mapped to the triangles defined by the vertices .

$$X(u_i, v_j), X(u_i+1, v_j), X(u_{i+1}, v_{j+1})$$

and

$$X(u_i, v_j), X(u_{i+1}, v_{j+1}), X(u_i, v_{j+1}).$$

We can now define consistent normal vectors for all triangles by computing the corresponding cross products, which in conjunction with a simple shading model yield values for flat polygon shading. For the smooth Gouraud shading surface normals for each of the vertices $X(u_i, v_j)$ have to be calculated. These may either be computed as normalized cross products of the partial derivatives $X_v(u_i, v_j)$ and $X_u(u_i, v_j)$, if available or via numerical approximation of these derivatives.

The hidden surface problem is most easily resolved by the Z-buffering technique supported by the IRIS workstation. However, the simplest version of the painter's algorithm is often sufficient as our pictures here prove. In that algorithm we compute the barycenters of all triangles, then sort them according to the distance from the eye position for perspective viewing or from a plane orthogonal to the parallel projection vector. Finally the sorted polygons are rendered in the order of decreasing distance. This algorithm actually has two advantages over the more precise Z-buffering technique. It is generally faster if the number of polygons is not larger than a couple of thousands. Moreover, it is very instructive to watch the computer draw the polygons from back to front. In this way the geometry of more complicated surfaces like the Klein bottle become much clearer. To profit from advantages of both methods, of course, we can render the sorted polygons from back to front with Z-buffering in effect as well.

3. examples of parametrized surfaces

In the following we list formulas for the generation of most of the pictures accompanying this article. The first example, the torus, is already given in the previous section.

minimal surfaces

The helicoid and Enneper's surface are two particularly accessible examples of minimal surfaces, i.e. surfaces with vanishing mean curvature. The parametrization of the helicoid is

$$X(u, v) = (v \cos u, v \sin u, u)$$

and Enneper's surface is given by

$$X(u, v) = (u - \frac{u^3}{3} + uv^2, v - \frac{v^3}{3} + vu^2, u^2 - v^2)$$

The coordinate lines in our figure are the polar coordinate lines for $u = r \cos \phi$, $v = r \sin \phi$. In Enneper's surface we have $r \le 1.5$ and for larger values of r we obtain self-intersections in the xz- and the yz-plane.

the Moebius strip

The Moebius strip is obtained by taking a rectangular planar strip, twisting its ends by half a rotation and gluing them together. Its relevance stems from the fact, that it is the prototype example for a non-orientable surface, i.e. one cannot define a differentiable field of surface unit normal vectors on the whole surface. A system of coordinates for the Moebius strip is given by

$$X(u, v) = ((2 - v \sin\frac{u}{2}) \sin u, (2 - v \sin\frac{u}{2}) \cos u, v \cos\frac{u}{2})$$

with $0 < u < 2\pi$ and $-1 < v < 1$.

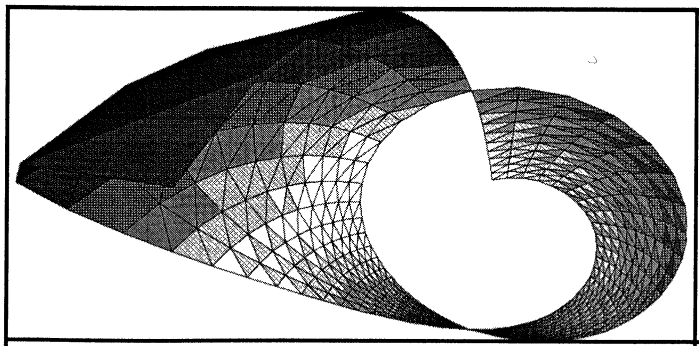
figures Klein 1a, 1b, ..., 5a, 5b

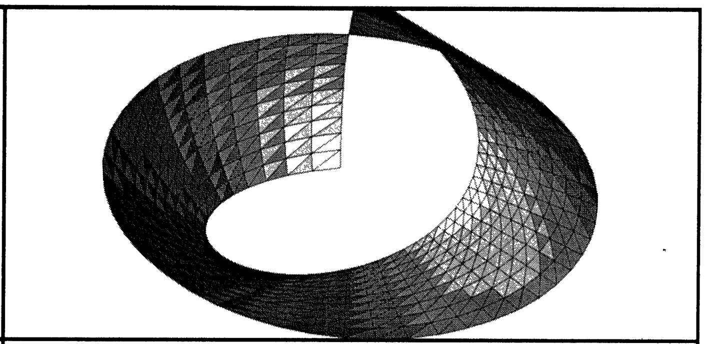
In the oppositetwo series of figures we have split the Klein bottle K into two parts : $K(u,v)$ with $v \le \pi$ and $K(u,v)$ with $v \ge \pi$. Let us denote by $K_{i,j}$ the part of the Klein bottle with coordinates $0 \le u \le 2\pi$ and $2\pi i/10 \le v \le 2\pi j/10$. Then the complete bottle is the union of $K_{0,5}$ and $K_{5,10}$. These sets are shown in figures 5a and 5b. $K_{1,1}$ is one of the Moebius strips in the Klein bottle, which we also cut into two halfs, namely $K_{0,1}$ (fig. 1a) and $K_{9,10}$ (=$K_{-1,0}$) (fig. 1b). The above sequences show how one obtains the two halfs of the figures 5a,b from the two half Moebius strips of the figures 1a,b by consecutively adding parts of the surface : $K_{0,2}$ (2a), $K_{0,3}$ (3a), $K_{0,4}$ (4a), $K_{0,5}$ (5a) and $K_{8,10}$ (2b), $K_{7,10}$ (3b), $K_{6,10}$ (4b), $K_{5,10}$ (5b). Note that a full circle, namely the coordinate line $v = 0$ (same as $v = 2\pi$) is part of all the surfaces shown.

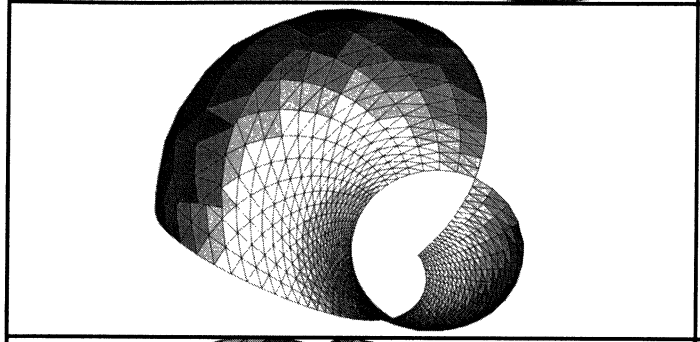
Try to follow the evolution of the boundary of the surface from figures 1a,b to 5a,b. In 1a,b the boundary is composed of the circle and three sides, and we have three vertices, one on the circle (reached twice) and two other vertices (corners). As we progress down to 5a,b these two corners both tend to the antipode of the first vertex on the circle. In the finishing stage we have that the boundary consists of the circle, which is traversed twice accounting for the self-intersection of the Klein bottle and another larger circle which meets the first one in the two antipodal vertices. To assemble the complete Klein bottle, we have to mate the two doubly traversed circles of 5a,b (due to different scalings the one in 5b seems to be larger than the one in 5a) and also the two other circles. Of course it is best to actually manipulate wireframes of both halfs to appreciate the full complexity and beauty of this classical surface.

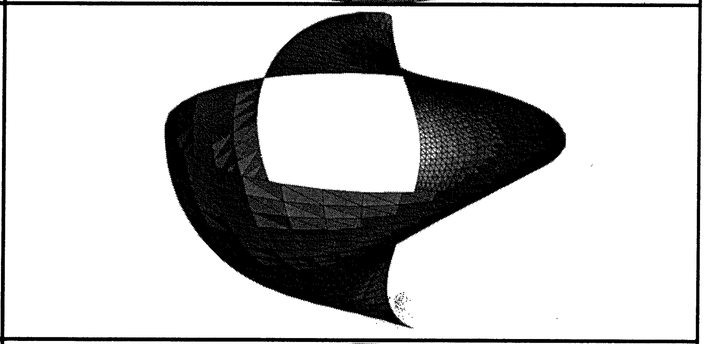
Klein bottle figures
Left row: Figures 1a-5a

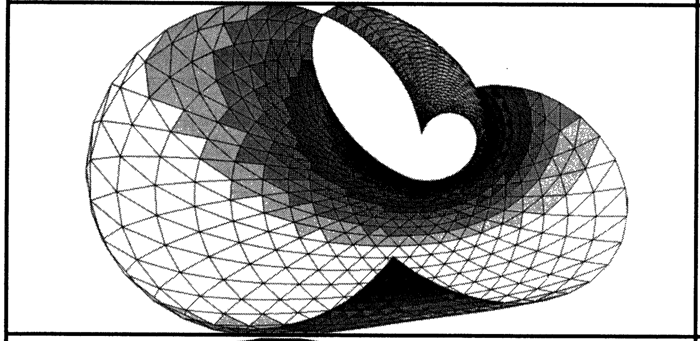
Right row: Figures 1b-5b

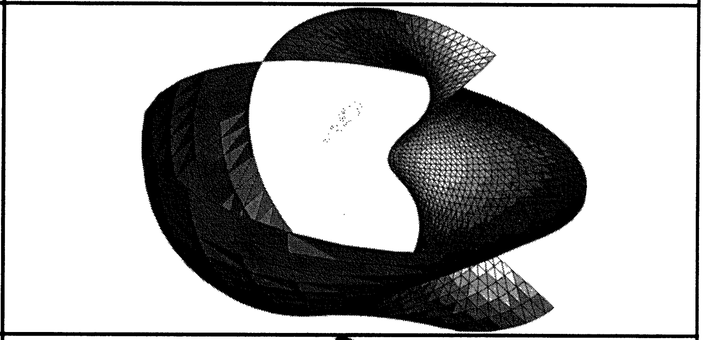


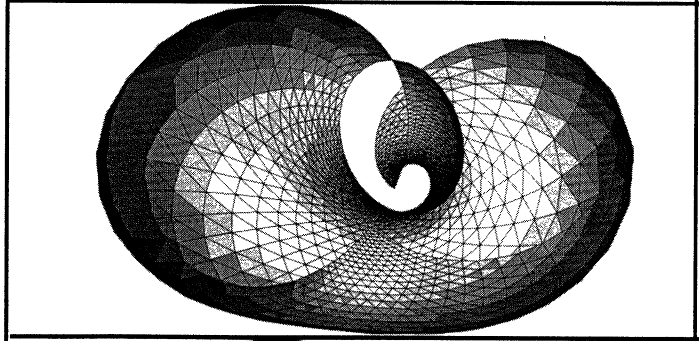


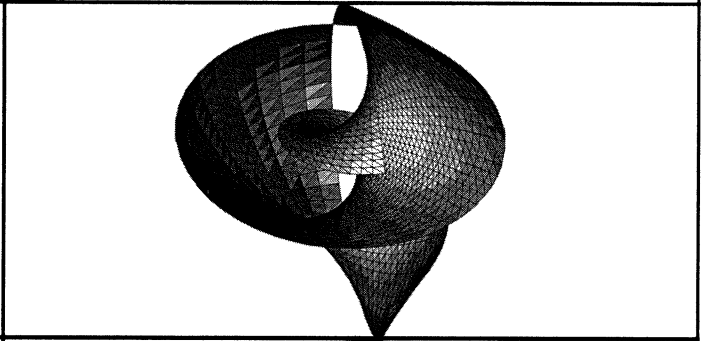


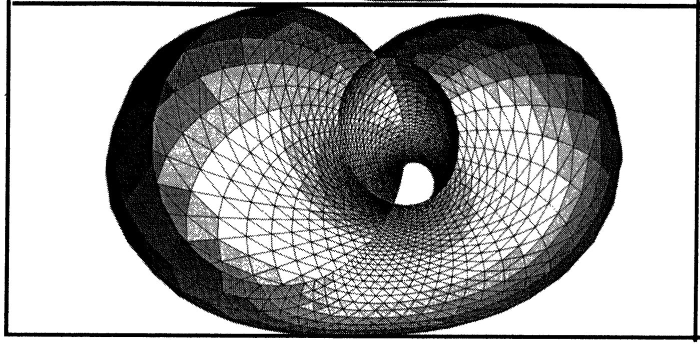


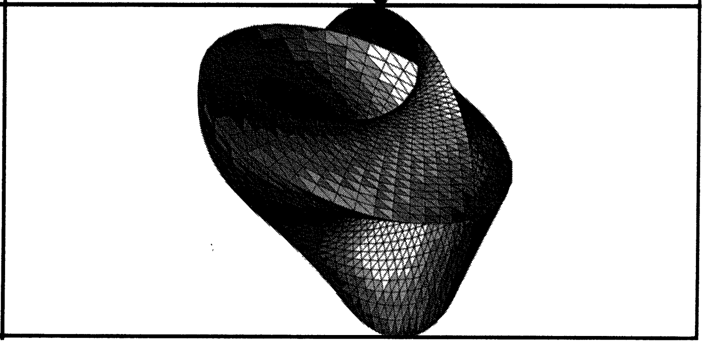




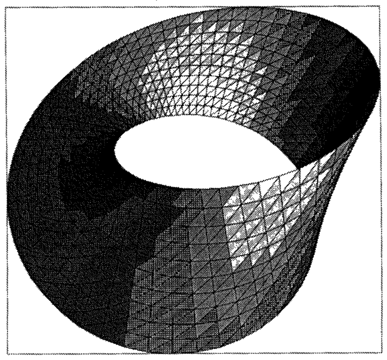


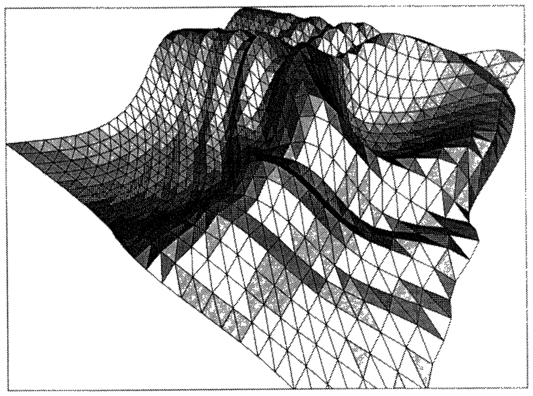


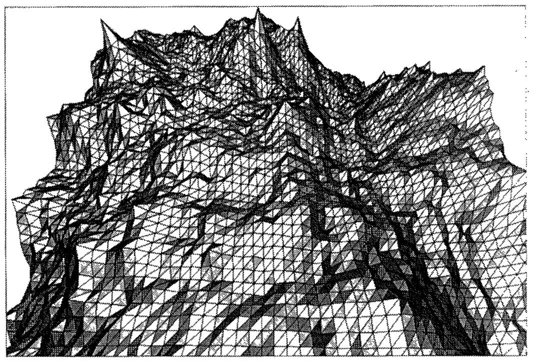


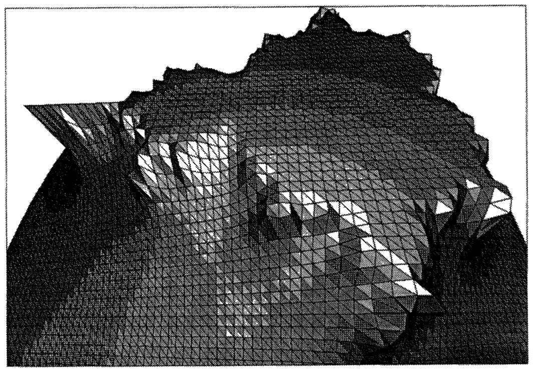


from top to bottom:
Moebius strip
Summation of two "cowboy hats"
Fractal surface generated by midpoint displacement
Potential of the full Mandelbrot set









the Klein bottle

The Klein bottle is an abstract regular surface obtained from the usual torus in 3-space by identifying a point (x, y, z) on the torus with its antipode $(-x, -y, -z)$. One can represent this abstract surface as regular parametrized surfaces in 4-space. In order to project the surface into 3-space it is convenient to use the following 4-dimensional version:

$$K(u, v) = (\cos u \cos v\,,\ \sin u \cos v\,,\ \cos\frac{u}{2}\sin v\,,\ \sin\frac{u}{2}\sin v)$$

where u and v range from 0 to 2π. Here the identification of antipodal points is established by

$$K(u, 0) = K(u, 2\pi) \text{ and } K(0, v) = K(2\pi, 2\pi - v).$$

This implies that the image of the strip

$$0 \le u \le 2\pi \text{ and } \pi - \varepsilon < v < \pi + \varepsilon$$

under K is a Moebius strip, and therefore the Klein bottle cannot be oriented. Actually there is another Moebius strip given by the parameters

$$0 \le u \le 2\pi \text{ and } -\varepsilon < v < \varepsilon.$$

Moreover, since

$$K(u, 0) = K(u + \pi, \pi) = K(u, 2\pi)$$

we have that the coordinate lines $v = 0$, $v = \pi$ and $v = 2\pi$ are all the same, namely the unit circle in the xy-plane and this is also the common center line of the two Moebius strips. Therefore it seems to be natural to to cut away the two Moebius strips as in some of our pictures in order to get a better understanding of the 3-dimensional geometry of the Klein bottle.

We have seen that our particular parametrization K does not yield an embedding in 4-space, since self-intersections occur. However, we have the advantage, that $K(u, v)$ is always a point in the hypersphere S^3, i.e. if (x, y, z, w) is a point on the surface, then $x^2 + y^2 + z^2 + w^2 = 1$. Therefore we can use stereographic projection P to identify a point (x, y, z, w) in S^3 with
This will work for all points in S^3 except for the north pole (0, 0, 0, 1)

$$P(x, y, z, w) = (\frac{x}{1-w}, \frac{y}{1-w}, \frac{z}{1-w}) \in R^3.$$

of S^3 which would be mapped to ∞. Unfortunately it is the case that the north pole is a point of the Klein bottle (set $u = \pi$, $v = \pi/2$), and, thus, stereographic projection would lead to an unbounded surface in 3-space. As a solution to this problem we can rotate the Klein bottle in the hypersphere S^3 such that a new point that is not in the surface is rotated into the north pole. E.g. $(1/\sqrt{3}, 0, 1/\sqrt{3}, 1/\sqrt{3})$ is a good choice for such a point. If T denotes this rotation, then the final unit square parametrization of the Klein bottle in 3-space is

$$X(u, v) = P(T(K(2\pi u, 2\pi v))).$$

cowboy hats

As a modified version of the popular "cowboy hat" we have used the graph of the function

$$f(u, v) = \sum_{i=1}^{N} a_i e^{d_i r_i^2} \cos (f_i \; r_i^2)$$

with

$$r_i^{\,2} = (u - u_i)^2 + (v - v_j)^2$$

where

N = number of "hats" to be displayed,
(u_i , v_j) = center of the i-th "hat",
r_i = squared distance of (u, v) from center (u_i , v_j),
d_i = damping factor (> 0),
f_i = frequency factor,
a_i = amplitude.

the potential of the Mandelbrot set

The Mandelbrot set is a prototype model for the transition from order into chaos. The set in its usual definition is embedded in the complex plane. The key idea is that each point $c \in \mathbf{C}$ of the plane determines a dynamical system R_c whose characteristic properties can be derived by checking whether c is in the Mandelbrot set or not. In this line of thought the Mandelbrot set M may be viewed as a complicated road map of dynamical systems (see [Pei86]). Its formal definition is

$$M = \{ c \in \mathbf{C} \mid \lim_{k \to \infty} R_c^k (c) \neq \infty \}$$

where

$$R_c (z) = z^2 + c$$

$$R_c^k (z) = R_c (R_c (\ldots R_c (z))), \; (k\text{-times}).$$

If the limit in the above definition does not exist, then we also take $c \in M$.

If one supposed the Mandelbrot set being metallic and charged with electricity, then it would induce an electrostatic field in its neighborhood. A probe introduced near the Mandelbrot set then would be subject to a certain force due to the potential. The lines of points exhibiting an equal amount of this attracting force are the equipotential lines, and these are basically the lines of equal color seen in the many color pictures of the Mandelbrot set.

In the standard algorithm for the computation of a picture an integer $k(c)$ which later will serve as an index for a color look up table is computed for a point $c \in \mathbf{C}$ via

```
k = 0;
while ( $| R_c^k (c) | < K$ and $k < k_{\max}$ )
      k = k + 1;
```

Here $k_{\max}$ is the maximal number of iterations we allow for each point (pixel) and K should be a large number such as 1000 (try also $K = 2.25$ and note the difference). The function $k(c)$ is only piecewise constant and does not yet yield a smooth parametrized surface. To achieve a smooth equivalent of $k(c)$ we may define the potential function $H(c)$ as

$$H(c) = \lim_{k \to \infty} 2^{-k} \log |R_c^k (c)|.$$

This limit converges very rapidly once $| R_c^k (c) |$ is large. Thus it suffices to iterate only until $k = k(c)$. For points c very close to the Mandelbrot set we need many iterations ($k(c)$ is large) and there fore the corresponding potential is $H(c)$ is very small. For points $c \in M$ we have $H(c) = 0$. The parametrized surface defined as the graph of H then is given by $X(u, v) = (u, v, H(u+iv))$. Similar potentials can be defined for the basin of attraction of ∞ for a fixed parameter $c \in \mathbf{C}$ in $R(z) = z^2 + c$, see [Pei86] and our figures.

references

[Car76] M. do Carmo : "Introduction to differential geometry", Prentice-Hall, 1976

[Hof85] D. Hoffman, W. Meeks : "A complete imbedded minimal surface in R^3 with genus one and three ends", Journal of Differential Geometry 21 (109-127), 1985

[Koc87] H. Kocak, D. Laidlaw : "Computer graphics and the geometry of S^3", The Mathematical Intelligencer 9,1 (8-10), 1987

[Lor64] E. N. Lorenz : "The problem of deducing the climate from the governing equations", Tellus 16 (1-11), 1964 .

[Man80] B. Mandelbrot : "Fractal aspects of the iteration of $z \to \lambda z (1 - z)$ for complex λ, z ", Annals of the NY Academy of Science 357 (249-259), 1980

[Man86] B. Mandelbrot : "Fractals and the rebirth of iteration theory", in [Pei86] (151-160), 1986

[Pei86] H. O. Peitgen, P. Richter : "The beauty of fractals", Springer-Verlag, Berlin (1986)

[Pet85] I. Peterson : "Three bites in a doughnut", Science News 127 (168-169), March 1985