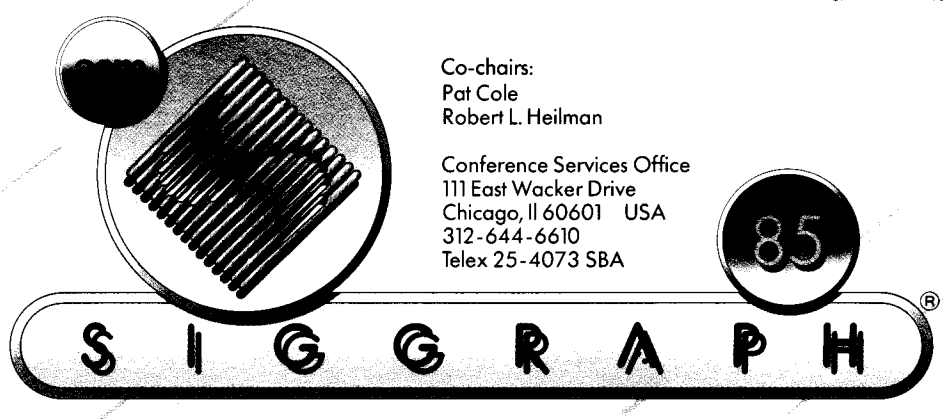


Co-chairs:
Pat Cole
Robert L. Heilman

Conference Services Office
111 East Wacker Drive
Chicago, Il 60601 USA
312-644-6610
Telex 25-4073 SBA

June 14, 1985

Mr. Woody Vasulka
Rt 6 Box 100
Santa Fe, NM 87501

Dear Mr. Vasulka:

Thank you very much for entering the SIGGRAPH '85 Art Show. There were over 3500 entries this year from around the world.

The following artwork has been accepted by the SIGGRAPH '85 jurying committee for display in the art show:

The Commission, 1984, Video

In order for us to frame and mount the work in a timely fashion, we are requesting that you send your artwork to our office by June 26. Please insure the work in transit, and let us know the insurance value of your work. Ship the pieces in either a protected flat shipper or a mailing tube. Be certain that the corners and edges are well padded. We will mount and frame the work and returned it framed and insured to you. While we are taking great care in mounting and framing the work, please be advised that due to our budget limitations we cannot provide a true museum quality or archival mounting.

Artists working in large-scale format or providing installations or videotapes for the show will receive instructions for shipping and handling of their work by phone.

We look forward to seeing you at SIGGRAPH '85, and thank you for your contribution to what we believe is a remarkable exhibition.

Sincerely,

Louise R. Etra, Chair
SIGGRAPH '85 Art Show

Rachel E. Carpenter, Administrator
SIGGRAPH '85 Art Show

4414 Piedmont Avenue, Oakland CA 94611 (415) 653-8444

SIGGRAPH'82

July 26-30, 1982
Boston, Massachusetts
Sponsored by ACM

P.O. Box 353
Derby, Connecticut 06418
203 735-9980

April 15, 1982

Dear Steina & Woody
Vasulka

Thank you for submitting your video/film work to the SIGGRAPH'82 Art Show. The response for the call for art was overwhelming. We would like to include the following video/film piece(s) in the exhibition:

1. videotape
2.
3.
4.

Works will be compiled on 3/4" video tape for continuous showing from submitted tapes. If you have a better copy of your piece, please contact us so that we can make arrangements to get it. All work will be insured by SIGGRAPH'82. Original material will be returned upon completion of the SIGGRAPH'82 Art Show edit. The SIGGRAPH'82 Art Show edit will be destroyed after the conference.

Please send us slides of stills from the accepted work for possible reproduction in the SIGGRAPH'82 Art Show catalog and/or poster. Send slides (only) to:

Louise Etra/SIGGRAPH'82 Art Show
c/o GESI
1440 San Pablo Avenue
Berkeley, CA 94702
Tel: 415/527-7700

Send videotapes and/or films to:

Copper Giloth
Real Time Design, Inc.
531 S. Plymouth Ct., #102
Chicago, IL 60605
Tel. 312/663/0584

Acceptance of your work entitles you (or one of the other artists if any) to a complimentary Siggraph'82 registration worth $175. You will receive information under separate cover.

Yours truly,

Copper Giloth

Copper Giloth
SIGGRAPH'82 Art Show Committee Chair
312/663/0584

encl:

1133 AVENUE OF THE AMERICAS
NEW YORK, N.Y. 10036
(212) 265-6300
Telex: 421686

acm

Association for Computing Machinery

SIGGRAPH—SPECIAL INTEREST GROUP ON COMPUTER GRAPHICS

Chairperson
Thomas DeFanti
Information Engineering
University of Illinois at Chicago Circle
Box 4348
Chicago, IL 60680
(312) 996-5485

Vice-Chairperson
Norman I. Badler
Computer & Info. Science
Moore School of Electrical Eng. D2
University of Pennsylvania
Philadelphia, PA 19104
(215) 243-5862

Secretary
Maxine D. Brown
ISSCO
4186 Sorrento Valley Blvd.
San Diego, CA 92121
(714) 452-0170

Treasurer
Sara A. Bly
Lawrence Livermore National Laboratory
P.O. Box 808 MS L-7
Livermore, CA 94550
(415) 422-6562

Past-Chairperson
James E. George
Mesa Graphics
411 Cheryl Ave.
Los Alamos, NM 87544
(505) 672-1998

Directors
Pat Cole
Lucasfilm, Ltd.
P.O. Box 2009
San Rafael, CA 94912
(415) 499-0239

Robert A. Ellis
Boeing Computer Services
Energy Tech. Applications Div.
565 Andover Park West M/S 9C-02
Tukwila, WA 98188
(206) 575-7048

Stephen R. Levine
Electronic Graphics Associates
841 Katrina
Livermore, CA 94550
(415) 443-3833

Editor-in-Chief
Bary W. Pollack
Datapoint Corporation
Graphics Development Center
2126 Sixth St.
Berkeley, CA 94710
(415) 848-7100

SIGGRAPH '82 Chairperson
Elaine L. Sonderegger
264 Shagbark Dr.
Derby, CT 06418
(203) 735-9980

SIGGRAPH '83 Chairpeople
Kellogg Booth
John Beatty
Computer Science Dept.
University of Waterloo
Waterloo, Ontario N2L 3G1
CANADA
(519) 886-1351

8/20/82

Dear Vasulkas, Siggraph Video Review Contributor:

~~Many thanks for the materials you left me at Siggraph'82.~~
I just want to be sure you understand the nature of the Siggraph Video Review (SVR) and I have a few other avenues of distribution to ask you about.

First, the SVR has been sent all over the world. Nearly 1000 tapes have been distributed at cost. As the enclosed flyer indicates, four hours of material have been available for a year now. The primary customers have been schools, libraries, and computer graphics companies. No abuses of the material have occurred to my knowledge, as yet. With your help, I plan to have two more hours ready soon and another two hours about January.

If you choose not to have your work considered public domain, please include a proper copyright notice on the enclosed form if your film/tape does not have one already. (Example: Copyright 1982 J.Q.Graphix.) Besides the legalistic implications, a copyright notice means people must get your explicit permission to make further copies, edits, etc. of your work from the SVR copies. If, on the other hand, you wish not to be bothered with requests, you may either skip the copyright notice or qualify it with a statement like "permission granted in advance for use by newsmedia and educational institutions," for example. I will also include a character-generated paragraph of text of your choice preceeding your entry on the SVR if you so indicate in the space labeled "Statement." In addition, I will include your name, address and/or phone number if you desire.

We have also budgeted enough money to produce a one-hour 16mm film for use by libraries, science museums and international conferences since they do not normally have access to the wonderful video projection gear we put together for the Siggraph annual conferences. Entries on 16mm film will be cut into a master; video entries will be transferred to film at Image Transform to assure quality. The film will be available on a loan-only basis until further notice. Any suggestions you might have to better this process are very welcome.

People at WGBH in Boston and MIT have expressed interest in producing a videodisk of computer graphics with slides, films and videotapes on it. Digital data will also be encoded for various experiments in random access, branching, etc. Please indicate your interest in this project on the form, if any.

(over)

APPLIED ENTERTAINMENT TECHNOLOGIES

3855 Lankershim Boulevard Universal City, California 91604

213 ~~760-3801~~ 441-3895

December 8, 1982

Messrs. Steine, Vasulka & Smith
Route 6, Box 100
Santa Fe, New Mexico 87501

Dear Messrs. Steine, Vasulka & Smith:

On behalf of the L. A. Chapter Siggraph I would like to thank you for your contribution to Siggraph West, held at the Art Center College of Design in Pasadena on October 30-31, 1982. Enclosed, for your information, is a brochure of the program.

The event was more successful than we imagined (or planned for) but it would not have been so if it weren't for the videotapes and films that were submitted for the show.

It is yours and all your co-workers' work that shined, and rightfully so. The time and energy spent in producing computer-generated imagery can sometimes go unrewarded: we hope that this event served as a token of appreciation and recognition for all those participating.

I am proud to say that the tapes and films submitted represented the most comprehensive and current accomplishments in computer graphics (analogue and digital) in the world! And, because of the registrant response, we feel that we can offer, next year, a continuation of this historic evolution to a larger audience with a greater professional support force.

The '82 Siggraph West was a West Coast inaugural event dedicated to the acknowledgement of achievement (and patience) in realizing "electronic perception".

Believe you me, the show "knocked their socks off".

Thank you so much.

Sincerely,

Edward R. Arroyo

Edward R. Arroyo

ERA:ll
Encls.

SIGGRAPH '82

July 26-30, 1982
Boston, Massachusetts
Sponsored by ACM

P.O. Box 353
Derby, Connecticut 06418
203 735-9980

August 15, 1982

HELLO!

...and THANK YOU! for participating in the Siggraph '82 Art Show. I thought you'd like to know that, thanks to the high quality of your artwork, the Art Show was a tremendous success and that we've already started planning next year's show. For those of you who were unable to attend this year's Art Show, it was extremely gratifying to see such a wonderful collection of work gathered at one location.

On behalf of the Siggraph '82 Art Show Committee, I wish to express our absolute pleasure in dealing with each of the participating artists and our hope that your interaction with us was as positive for you as it was for us. The Siggraph '83 Call for Art will be released by November 1st. We sincerely hope that you will continue your participation by submitting work for next year's show in Detroit.

All videotapes and films are currently in the process of being packaged and returned to the authoring artists. Hardcopy works were mailed two weeks ago. If you have not yet received your hardcopy work, please contact me.

For your personal use and reference, I enclose the following Art Show items:

- 10 Siggraph '82 Art Show Catalogs
- Artist's Address List
- extra copies of your piece if it was in the catalog

Under separate cover, I am also forwarding two copies of the Art Show poster.

The Siggraph '82 Art Show Slide Set is available and may be obtained by sending me a check made out to Siggraph in the amount of $20.00.

Once again, it has been a pleasure working with you and I look forward to the opportunity of working with you in the future.

Sincerely,

Copper Giloth

Copper Giloth
Siggraph '82 Art Show Chair
c/o Real Time Design, Inc.
531 S. Plymouth Ct., Suite 102
Chicago, IL 60605

PIXEL

the computer animation newsletter

Volume 2, Number 4
July, 1985

HIGHLIGHTS IN THIS ISSUE

INDUSTRY MAKEUP CHANGING RAPIDLY - ARE YOU READY?

The computer animation industry is changing rapidly - and dramatically.

A quick walk around the Trade Show Exhibit Hall at SIGGRAPH '85, with its growing number of software and systems exhibitors, gives a good indication of what is happening (see our SIGGRAPH '85 analysis on page 5). As outlined in our Annual Market Report, the computer animation industry is changing. Many new players are entering the field and the number of computer animation users is growing daily.

It is significant that these new entrants into the computer animation production arena are purchasing packaged systems. These are either complete "turn-key" systems of combined hardware and software, or one of the growing number of software packages now available which they can run on mainframes they already have or on one of the new "super-mini" types of computers. And many of them are not in the film/video production business, yet their uses of computer animation fall within our definition of this category that excludes CAD/CAM and Business Graphics.

Very few new computer animation producers are willing to invest the time and money required to develop their own software. To a large degree this is because there now is so much excellent software available. The older companies didn't have the option. Most of it wasn't even available two years ago, and what was on the market did not usually produce an image of sufficient quality for TV broadcast work. They had to develop their own.

What is also changing is the use these new systems are being put to. We use the term "computer animation" to differentiate our subject from "computer graphics", which we use to refer to a single frame or computer generated image, whereas "computer animation" is used

Published monthly by:
PIXEL - THE COMPUTER ANIMATION NEWS PEOPLE, INC.
217 George Street, Toronto, Canada M5A 2M9 (416) 367-0088
Robi Roncarelli, editor

to describe a series of sequential images or frames strung together to give an "animated" picture that moves.

The industry's capabilities are also changing. We now are talking about more than just the "animation" of images that are the result of someone's imagination or creative action. We now have to include the word "**simulation**" in describing our subject matter - using computer generated images to "simulate" scenes, events and objects from the real world around us. And because the real world never stands still, these simulated images are animated images.

These simulations of real world images form much of the computer animation outputs that are starting to be used with some regularity in the fields of medicine, education, architecture ... and entertainment. And at SIGGRAPH, more and more of the promotional works shown at vendor's booths were of a "simulation" nature than a non-realistic or fantasy production.

The implications of these changes will affect everyone involved in the computer animation/simulation area - producers, designers, hardware/software suppliers and users. Just as today's industry is very different from that of just two years ago, this change will continue, actually accelerate, over the next two years. Particularly as more non-technical people enter the industry as purchasers and users of packaged "turn-key" systems.

The result is that potential technological advantages will become less important. Instead, the major factors in determining success in the marketplace will be Creativity and Marketing Ability.

PACIFIC DATA IMAGES

Mention the name Pacific Data Images, and most people who follow computer animation usually have visions of clean, sharp, brightly colored computer animated on-air sequences produced for major American broadcasters.

Although PDI had been producing computer animation for the better part of three years before their outstanding visuals for "Entertainment Tonight" first hit the airwaves, that series is what many people first remember of their work.

Started five years ago next month by Carl Rosendahl, the company began operations with just three people - Rosendahl, Glenn Entis and Richard Chung. They have written all their own software, with Entis and Chung doing most of the early foundation work. Much has since been written on top of that, both by these two, and by a number of other people. This strong emphasis on continuing development is to a large degree the result of Rosendahl's policy of fostering personal development among his staff, a policy which also pays big dividends on a corporate level.

Operating with a relatively small, but very dedicated group of fifteen people, Rosendahl makes sure that everyone has the opportunity to persue their particular interests, using any computer free-time to produce animation or develop software. Much of their work ends-up incorporated in PDI's client productions.

A considerable amount of Pacific Data Images' work is seen outside of North America. They have done several productions for TV Globo in Brazil, as well as supplying their software for TV Globo's new computer animatiom facility, and helping them set it up. Another steady customer is Australia's Channel 9.

American customers have ranged from video games to the major TV networks. About half of their design work is done in-house, and they have a close collaboration with well known designer Harry Marks, whose name often appears on their credits, particularly on broadcast work.

PDI's new facility in Sunnyvale, California, 40 miles south of San Francisco, seems to reflect their corporate personality. It is a bright, airy, stand-alone building situated in an industrial park surrounded by green lawns and shrubs. At the building's core is the computer room housing their nine Ridge computers, one IMI 500 computer for motion design and seven Raster Technologies frame buffers. An interesting feature of their layout is three 'client rooms', each with a work station, offering facilities for client interaction with privacy and comfort.

The building is completely 'wired', which means they can route signals to any workstation in the building from any combination of computers. Output is via three video controllers they built themselves, two for 3/4" video for test shots and one for final production onto a Sony BVU 2000 1" videotape recorder. Most of their work is produced onto video at 512 x 512 lines resolution, which Rosendahl believes adequately covers most of their requirements, however, they can digitize images at up to 2000 x 2000 lines resolution and output onto film via a Celco recorder.

They can also input from video or tape and digitize these images for combination with, or inclusion into their animation.

As a client, you won't find anything too different from regular industry production procedures at PDI. Productions follow the normal steps of vector motion tests, raster test frames, approvals and final productions, and on average are completed in 6 to 8 weeks. What we do think you will find different is the way they interact with you. They are a friendly, enthusiastic and very competent group, and with only 15 people in the company, you will get a feeling of real involvement.

Although they have sold a their software to TV Globo and have enough software to put together a comprehensive package, this is not an area PDI is actively pursuing. They don't have the people available to package or market their system, or provide follow-up support, and they don't want to detract from their main business of computer animation production. However, they probably wouldn't turn down any sales that walked through the door.

Pacific Data Images have also not been too active in the areas of **advertising** and **entertainment** production. But this they do want to change, being particularly interested in doing more work for advertisers and their agencies. They will run up against quite a few competitors in vying for the advertiser's dollars, some of them sizable organizations with sales staffs almost as large as PDI's total company, so it won't be easy.

But then Pacific Data Images has excellent credentials which are exhibited in their demo tapes, and a long list of prestigeous awards which always appeals to advertising agencies. So we think we will soon be seeing some of their excellent computer animation used by major advertisers and more television commercials with PDI's name listed on their credits.

Pacific Data Images
1111 Karlstad Drive, Sunnyvale CA, USA 94089. (408)745-6755

* * *

Carl Rosendahl started Pacific Data Images because he figured it was the only way he could get into the computer animation business.

"Back when I started the company five years ago, there weren't that many people doing it and I couldn't really figure out why anyone would hire me anyway. I mean, I was a kid out of Stanford with a Batchelor's Degree in Electrical Engineering and some film making experience from my high school days down in Los Angeles. So I figured the only way I was going to get a job in this industry was to make my own."

That kind of "gutsy" approach is still very much evident at PDI today. With a staff of only fifteen, and a production load that some of their larger competitors yet aspire to, Rosendahl maintains a policy of allowing his people to have a certain amount of time between production jobs to work on their own projects - animation, developing software, any type of project or whatever else they want. "We get a lot of little 2 to 5 second experiments, but we try to encourage people to do larger projects. And really encourage people if they start a larger job to finish it, allowing a certain amount of time for them to get it together afterwards."

This almost academic approach is quite different from the policy at most other computer animation facilities where employees have considerably less flexibility and opportunity to pursue their own works. It says volumes about how Rosendahl runs his company - and it does result in a continuing stream of bright ideas that the company can tap into, and some excellent test pieces to show on their demo tape.

Another area where Rosendahl differs from many of his colleagues is in his defence of production directly onto videotape for broadcast/video use. "I don't agree that you get a cleaner look on film. You get a cleaner look by going direct to video because there is nothing that you are going to put on film that is going to get transferred to video that you can't put directly onto tape. The only reason I can say that they would do that is to get the higher resolution of film, but by the time you have transferred to tape you have lost that resolution anyway.

"If you are going direct to video you have all your colors available to you. Film has a certain color range that it can reproduce and it won't reproduce everything you have when you go to it. Nor will NTSC. On an RGB monitor you get a beautiful yellow but you will never be able to get that on NTSC. Film to tape you lose some more, because there are a lot of colors and contrasts on film

that you can't get on tape. So what you finally end up with by the time you've gone to film and back to tape is only the colors and contrasts that can be reproduced on both film and tape."

As for his future plans, Rosendahl sees continued steady growth. "We'll certainly continue what we are doing. We haven't been very visible in the advertising world and we're going to start becoming much more active there. Beyond that we'll probably go out into other areas. There's a wide variety of industries out there in addition to the ones we're into now. We are always talking to various individuals in other businesses."

Yet he adds a word of caution. "Keep in mind we're just 15 people here. We've got to be kind of busy." Not that Rosendahl, or his associates, are afraid of hard work. It's part of his philosophy, which shows up regularly in their work. "You do it because you get to try something new and different, and really have a good time with it."

SIGGRAPH '85

Tony de Peltrie, Abel Image Research, Lucasfilm's Pixar, 3-D computer animation and systems, software and more systems. Siggraph '85 at a glance.

At every graphics conference, exposition and festival, the biggest crowds are usually found at the computer animation showings. Siggraph '85 was no exception, and the two biggest attractions were the outstanding computer generated character animation displayed in "Tony de Peltrie", a 7 minute and 50 second film produced by four young men at the Universite de Montreal (see following article), and the 3-D stereo computer animation film produced for the Hitachi Pavillion at Expo '85 in Tsukuba by Digital Productions.

The Hitachi 3-D stereo film, viewed with the help of special polarized glasses, gave an excellent feeling of being immersed in the scenes, with viewers ducking as images from the screen seemed to come right out at them. A non 3-D version was also shown on the video monitors.

Other noteworthy showings were Blowin' in the Wind by Bill Reeves of Lucasfilm which simulated a grassy field on a windy day, Chasing the Rainbow by Kiri Miyagaki produced on the Links 1 system at Osaka University and Tuber's Two Step by Chris Wedge of Ohio State University. The commercial demo reels were, as always, visual and technical masterpieces, with some of the most interesting work shown resulting from various employee's own after-hours productions. Pacific Data Images gathered all theirs onto a special reel separate from their commercial show reel.

On the Exhibition Hall floor the two 'hotspots' that attracted the **BIG** crowds were Lucasfilm's Pixar booth (see our Feb/85 issue) and the Abel Image Research booth. Their displays were well presented, but it was the glory attached to their names that drew people to them. However, the real thrust of the Exhibition, and they were very much a part of it, was in the areas that confirm the direction the computer animation, graphics and simulation industry is heading - turnkey systems and software packages.

The turn-key systems and software packages displayed covered a wide range of sophistication and cost. At the low end of the spectrum is the simple paint software that runs on your personal computer and is operated through keyboard, small digitizing tablet and/or lightpen, usually requiring the addition of a graphics board to your PC that can cost from US$1,000 to US$5,000.

Next are the 2-D and 3-D systems based on an IBM PC or compatible as a central processor, with software and peripherals added as part of the package. Cubicomp, one of the leaders in this area, demonstrated a considerably upgraded system, with improved resolution, capabilities and peripherals. Its base price is US$29,500. With add-ons you can turn it into a full graphics or animation production system. Their software was actually used in the production of images and effects for the films "2010" and "Starman".

For more money you start getting into the mini-computer based systems. Prices vary widely here, from US$49,500 for the basic Images II system from Computer Graphics Lab to almost US$300,000 for a fully configured Alias Research system. In between are systems such as those from Symbolics, Artronics, Neo-Visuals and Bosch with their well established FGS-4000.

Software is available from several companies, including Wavefront Technologies, Neo-Visuals, Antics and Computer Graphics Lab, but the big news at the show was the new offering from Abel Image Research. Their software is configured (and priced) to run on a wide range of hardware, from a Single Iris workstation at US$80,000 to a Cray at about US$300,000. The first license for Abel's software package was issued well before the announcement at Siggraph to Electric Image in London, England. They started production this past May and have already turned out several jobs and some excellent work.

Computer manufacturers were out in force, all displaying software from various suppliers ported to their machines. The wide range of available computers suitable for computer animation production and the variety of software offerings to run them, give prospective purchasers considerable choice.

Included in the interesting French industry exhibit was the De Grafe computer which comes with its own integral video graphics software allowing a wide range of applications as supplied. Other computer animation systems from France were the Xcom Graph 9 and Telmat's CUBI 7.

With all the systems, software and computer configurations now available, getting into the computer animation business is becoming much easier and less expensive every day. And this trend is continuing. It is getting to the point where the equipment required to output and record your images is considerably more expensive than the equipment needed to create those images.

Producing complex images quickly still requires lots of expensive computer power, so the high-end of the business will remain fairly exclusive. But the broad range of production for video output, as indicated by the Trade Exhibition and even some of the productions shown, is becoming a very competitive market. For both equipment/systems/software suppliers as well as producers, how you well you use your equipment or market your product, is becoming the key to business success.

TONY DE PELTRIE

The highlight of the Siggraph '85 Film & Video Show was the world premiere of the short animated film Tony de Peltrie. Entirely created by computer, it not only is a technical 'tour-de-force', but unlike many other developmental computer animation projects, it is creatively entertaining also.

This is full three dimensional character animation, complete with changing facial expressions, smooth body movements and even close-ups of the character's hands and fingers as he plays his piano. Running for 7 minutes and 50 seconds, it is more than a brief systems/software test. It tells the story of an aging piano player in a piano-bar, who although once the center of popularity has now been left behind by times and changing fashions. Surrounded by his memories, he plays his last performance.

Tony de Peltrie is the creation of an independent group made up of Pierre Lachapelle, the originator of the project and its producer and co-director, Philippe Bergeron, co-director and character animator, Pierre Robidoux, co-director and special effects animator and Daniel Langlois, co-director and art director. Initiated in July, 1982, final production required about four man-years to complete and cost over US$1.5 million.

The film was produced using TAARNA, a powerful 3-D computer graphics system conceived for artists and designers developed at the Centre de Calcul de l'Universite de Montreal and programmed with DADS software. Filming in 35mm was shot at Canada's National Film Board, and financial support came from the Ministere des Affairs Culturelles du Quebec and the Canada Council. About 20 people in total worked on or assisted directly in the project.

Obviously not a cost-effective production, Tony de Peltrie is one of those landmark efforts that dot the short history of computer animation. Like Triple III's Adam the Juggler, and The Adventures of Andre and Wally B from Lucasfilm's Computer Division, just to name a couple, it shows what can be done with the medium and gives us a target to aim for in our more mundane everyday productions. But trust our inventive French Canadians to make a dramatic and entertaining production out of it.

Lachapelle, Bergeron, Robidoux & Langlois
11793 Drapeau, Montreal Nord PQ, Canada H1H 3K7. (514)737-4800 or (514)325-0524

ELECTRIC IMAGE

Our apologies to Paul Docherty and Electric Image in London. Contrary to what we wrote in our May/85 issue, they are not a division of Visions. Although they share premises, Visions has no financial or administrative interest in Electric Image, or vice-versa.

Electric Image is the first licensees of the new Abel Image Research software announced at Siggraph '85, and was just getting their system up and running when visited last May. They have been very busy since, and we'll have some samples of their excellent work to show our readers soon.

INTERACTIVE MACHINES INC

One of the less imposing computer manufacturer's booths at the Siggraph '85 Trade Forum was that of Interactive Machines Inc. No fancy furniture or colorful graphics, just an IMI 500 and an IMI 455, both up and running, and always several interested people.

Although their computers may have wider distribution in the design, engineering and aeronautical industries, it seems that just about any serious computer animation facility has at least one IMI 500. Sometimes that's all they have although they are generally used in conjunction with VAXs, Ridges, Pyramids or even Crays. One reason is that the IMI 500 is designed for a graphics application.

Another is that IMI President Ken Dozier, with a business background ranging from flight simulators and super computers to computer graphics production, has a good grasp of what's happening and where the business is going. When questioned, he is quite emphatic about it, "In an emerging marketplace the ability to innovate is the ability to survive. So that's what everyone is looking for, the ability to be able to innovate and therefore they will thrive and survive in this emerging marketplace. And that's really very important."

Having worked with many computers, including the Cray, he believes that "flexibility" in equipment choice is equally important, "It's the flexibility that you're looking for. I think that anybody that paints themself into a hardware position right now is really going to feel it."

Dozier believes that the IMI 500 is right in step with the new industry emphasis on **simulation.** "Production graphics is hierarchical by nature. The portion that we fill is the interactive portion. We are a high resolution, real time computer simulation device. And what we have is an internal graphics processor that enables us to do an extensive amount of mathematics in real time." He is also a big promoter of the growing use of computers for scene pre-production, "We can show up to 100 objects moving dynamically in real time, which plays right into the capability of doing scene planning. Entertainment is constantly putting their emphasis on pre-production - especially in the area of special effects. They want to minimize risks."

As for IMI's future prospects, he believes that the inherent flexibility of the IMI 500 will continue to stimulate their sales. "The community out there is insatiable, and once it sees it, it tires of it and wants something new. The days of the old host computer with the program running on it and the dumb terminal are dead. Now the local intelligence design/simulation station has to have intelligence of its own. Its own disks and its own operating system. And it has to be very, very, very interactive. And then it will be strapped onto a host to provide secondary number crunching abilities for it."

We think Ken Dozier's comments on what's happening in our industry are right on target, and spotlight some of the industry changes reflected at Siggraph.

Interactive Machines Inc
733 Lakefield Road, Westlake Village CA, USA 91361-2694. (818)707-1880

BOSCH FGS-4000 UPGRADES

With lots of new vendors appearing in the **turn-key systems** market, Robert Bosch is working hard to keep their FGS-4000 computer graphics system competitive and maintain their market lead.

Two new software packages offered are their Light Source Editor, which can control and configure up to 16 independent light sources, and their Terrain Modeler for generating random land sources with fractal-like surfaces. They have also added two major enhancements to their existing software, a 'gouraud' shading model and a 'phong' shading model, and are continuing development of a High Quality Paint System.

To increase sales into industrial markets and other areas that have not traditionally used their video/broadcast products but are potentials for the FGS-4000, and are also being pursued by their competitors, Bosch has repositioned its national sales force into two separate divisions, with one dedicated to these new market areas.

The Bosch demo tape shown at Siggraph '85 is an eight-minute compilation of work produced on the FGS-4000 by users around the world.

Robert Bosch Corporation - Video Equipment Division
PO Box 31816, Salt Lake City UT, USA 84131. (801)972-8000

UPCOMING EVENTS

October 4-9:
F.I.D.A.A.M. - First International Festival of Cartoon Animation and Puppetry, Juan-Les-Pins, France. Computer animation included.
28 Rue Gioffredo, 06000 Nice, France. (93)62 02 20

October 16-18:
Computer Graphics '85,
Wembly Conference Center, London, England.
Online Conferences, Pinner Green House, Ash Hill Drive, Pinner, Middlesex, HA5 2AE, England. (01)868-4466

November 5-10:
Cambridge Animation Festival '85,
PO Box 332, London, England N6 5YH. (01)341-5015

November 28:
One Day Seminar in Computer Aided Art & Design (CAAD)
Faculty of Art & Design,
Middlesex Polytechnic, Cat Hill, Barnet, Herts EN4 8HT, England (01)440-5181

There are many events of potential interest to our readers occuring regularly all over the world. Keeping track of them is almost a full time job and to list them within these pages is impossible. Instead we are compiling an on-going list that we will regularly update and publish periodically for our readers.

NEW NETWORK GRAPHICS

The new US television networks graphics packages are going on-air.

NBC have new visuals designed by Marks Communications and produced by Pacific Data Images. Created by Marks and PDI for both the NBC network and its affiliates, the computer animation features a hugh neon sign with chase lights that forms the basis of this year's "Lets All Be There" campaign theme.

The CBS fall image campaign, also including affiliated packages, was produced by Omnious Computer Graphics at their Los Angeles, New York and Toronto facilities on both film and tape. It features multi-level 3-D images forming the campaign logo, based on the theme "We've Got The Touch". Creative direction was by Leprovost Corporation.

In Canada, Omnibus is also producing the new fall network on-air promo packages both CBC English and CBC French networks, CTV and Global.

BUSINESS REVIEW

Business is picking-up. Our reports from around the world indicate that there are many new computer animation projects either in production or about to start, and most production facilities are beginning to breath a bit easier after the uncertainties of the past few months. The next three months should continue to keep everyone busy.

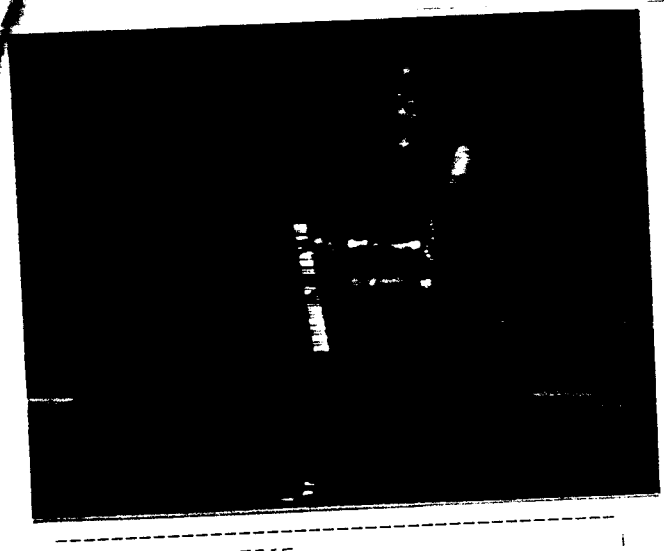
This new work is coming from many sources - advertising, TV program production, industrial films and the entertainment industry. We also predict a big increase in computer animation for demonstrating research projects of a wide nature, but many scientific and medical organizations are looking to set up their own in-house production units to serve their needs. They already have the computers and technical people who can take advantage of the new software packages coming available.

The exact impact of the new 'turn-key' systems being put into production will take a while to be felt. Many of them are going to post production and service companies who will phase them in slowly as their work loads demand. Not being dedicated single-product computer animation producers, they don't have that pressure for instant production required to generate cash flow.

Expo '85 in Tsukuba, Japan is a veritable showcase of computer animation. The real eye-boggler is the 3-D stereo OMNIMAX production. We will have an on-the-spot report of what's happening in Japan in our next issue.

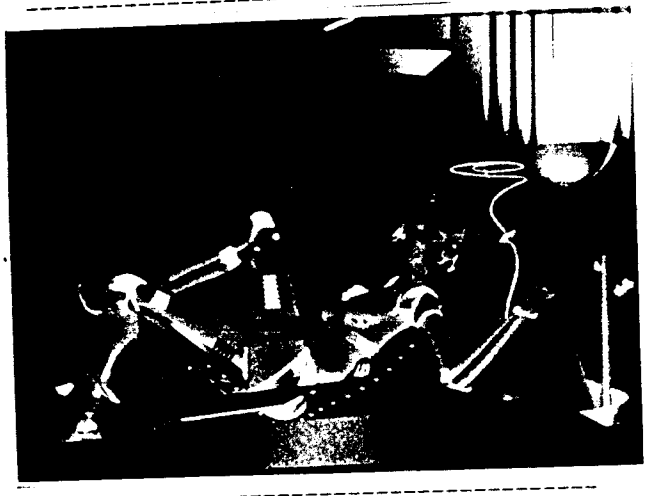
The English industry is experiencing considerable changes: people moving, new owners and capital, new equipment and systems. All this at a time when they are about to lose some of their past European clients to new facilities opening-up on the Continent. Paul Brown, our London associate is preparing a comprehensive report for us.

And the film "Explorers" has just been released. Any bets on boxoffice?

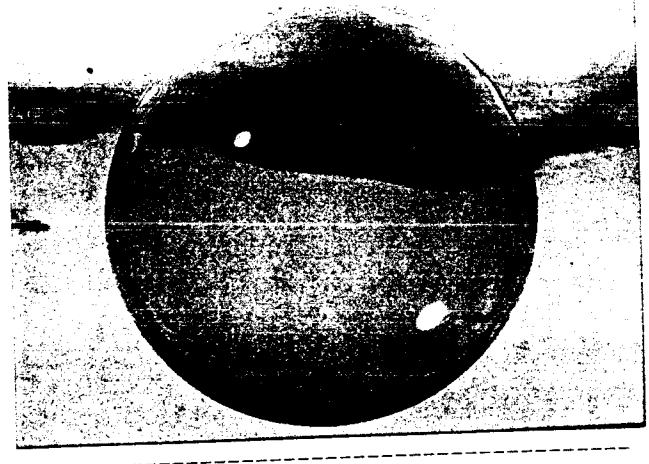


1. TONY DE PELTRIE.
The standout production at SIGGRAPH '85 | 7 minutes & 50 seconds of 3-D character animation with excellent expression.

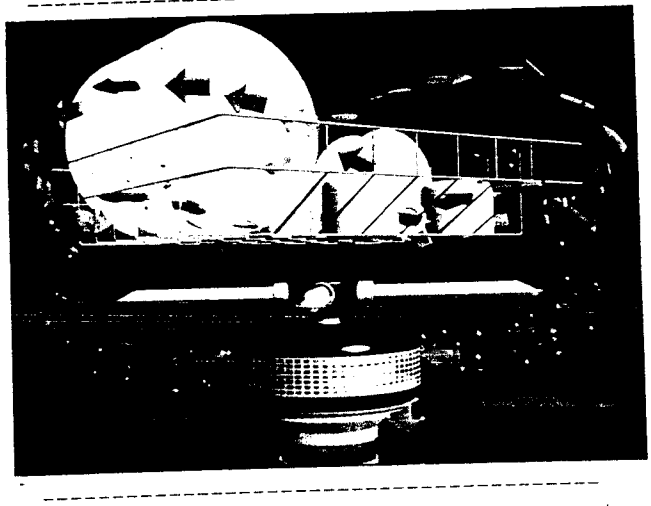
2. TONY DE PELTRIE.
Produced by Pierre Lachapelle, Philippe Bergeron, Pierre Robidoux and Daniel Langlois at Universite de Montreal.



3. ROBERT ABEL & ASSOCIATES.
"Brilliance", Sexy Robot 30 second spot for Canned Food Information Council is Abel showcase for new software package.

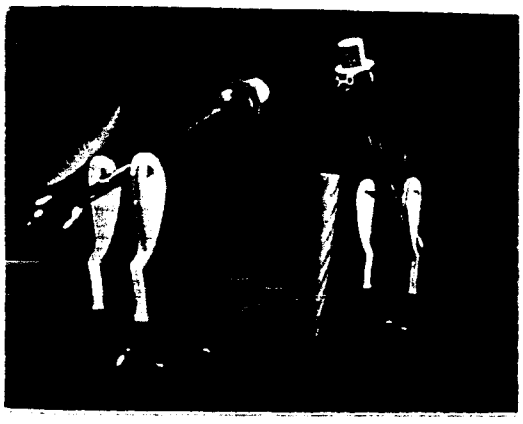


4. MAGI SYNTHAVISION.
Experimental work showing transparency from their new showreel.

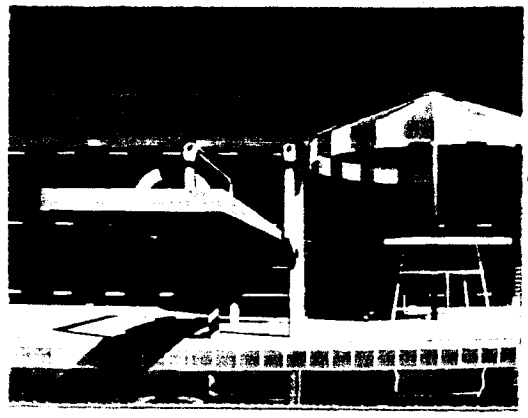


5. INTELLIGENT LIGHT.
5 second insert for Kitchen Aid spot. Produced by Intelligent Light using their new software package.

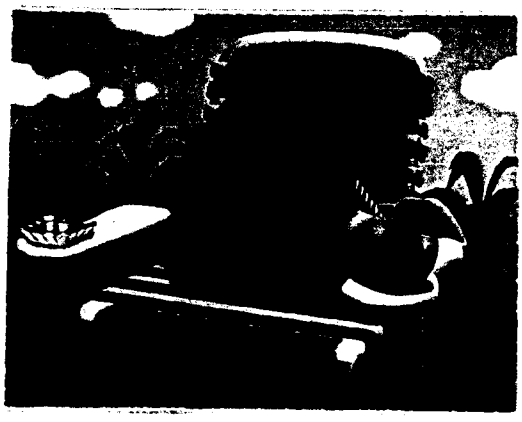
6. CHRIS WEDGE.
"Tuber's Two Step", 3-D non geometric characters in well choreographed set. Produced at Ohio State University.



1. PACIFIC DATA IMAGES.
"Happy Drinking Birds", developmental animation by Richard Cohen, who trained at Sheridan College in Toronto.



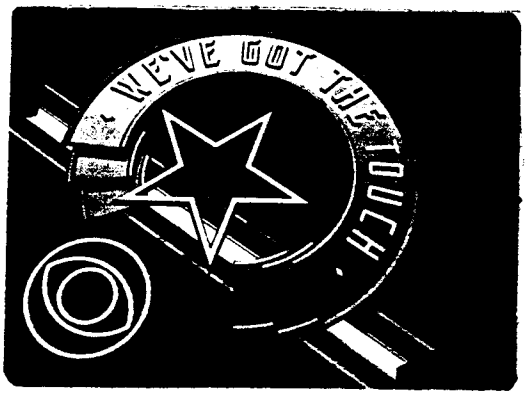
2. PACIFIC DATA IMAGES.
"Poolside", developmental animation by Thaddeus Beier. Part of PDI's after-hours personal work program.



3. PACIFIC DATA IMAGES.
"Toast on a Lounge Chair", developmental animation by Roger Gould. Work is great when you are having fun.

4. PACIFIC DATA IMAGES.
NBC's fall '85 on-air promos continue their "Be There" campaign. Design by Harry Marks. John Miller for NBC.

5. OMNIBUS COMPUTER GRAPHICS.
CBS's fall '85 on-air campaign. Design by John LeProvost. Mort Pollack VP for CBS Entertainment Division.



6. OMNIBUS COMPUTER GRAPHICS.
Special promo by Shell Brazil for major music festival, produced for MoviArt, co-ord by Zillion, design by O&M Publ.

Papers from Siggraph 76: The Third Annual Conference on Computer Graphics, Interactive Techniques, and Image Processing

Sponsored by ACM Special Interest Group on Computer Graphics
July 14–16, 1976

The organizers of the first annual conference on computer graphics and interactive techniques believed that a joint SIGGRAPH-SIGPLAN effort to identify, consolidate, and study computer graphics principles would lead to increased, productive understanding of many topics. The success of the first annual conference, held in Boulder in July 1974, followed by an equally successful second annual conference, held in Bowling Green in June 1975, gave rise to plans for the third annual conference held in Philadelphia. The topics studied in the papers for the third annual conference covered a rich spectrum of current research interest, including: programming languages and data structures; hardware; device independent graphics, animation; satellite graphics; cartography; graphics input/output devices; system software; application areas such as medical/biomedical, civil engineering and architecture; surface representations, CAD: large systems and social graphics.

The program committee for the third annual conference received 143 abstracts, 118 papers, representing almost all areas of current research, and selected 58 papers for presentation. From this program of papers, four have been chosen for publication in this issue.* All papers were reviewed by referees in the usual fashion and the four appearing here are each revised or extended revisions of the original papers, with only an abstract of these four papers appearing in the Conference Proceedings.

The paper by Blinn and Newell, an excellent extension to the Catmull algorithm, presents developments in the areas of texture simulation and lighting models, with some beautiful generated images resulting from the reported work. In addition, the authors present simulation of mirror reflections on curved surfaces, incorporating reflected light intensity as well as the intensity obtained from the texture mapping.

Two of the papers study algorithms for displaying surfaces or advanced hidden surface techniques. The paper by Clark deals with the design of systems for efficiently producing computer generated pictures and picture sequences of very complex three-dimensional environments. An important aspect of the proposed hierarchical model is that it provides a way to vary detail, yielding an incremental improvement to existing systems for producing computer pictures without modifying their visible surface algorithms. An additional improvement is suggested by a totally new recursive descent visible surface algorithm in which the computation time potentially grows linearly with the visible complexity of a scene rather than as a worse than linear function of the object-space complexity.

The paper by Levin considers an algorithm for drawing pictures of three-dimensional objects, with surface made up of patches of quadric surfaces. Methods of parameterization for each type of quadric surface intersection curve are discussed, as well as surface bounding and hidden surface removal. The principal contributions of this algorithm are that it uses all real quadric surfaces, allowing smooth intersections of approximate higher-order surfaces, and that it may be fast enough to be used in conjunction with a shading algorithm.

The problem of motion dynamics in animation is made challenging by the fact that the dynamics description must not only describe the behavior of the object but also all motion subcomponents relative to each other. The paper by Burtnyk and Wein develops a significant increase in the capability for controlling motion dynamics in key frame animation through skeleton control. This technique allows an animator to develop a complex motion sequence by animating a stick figure representation of an image, and then driving this image sequence through the same movement.

The members of the executive committee for the third annual conference were T. Johnson (General Chairman), U.W. Pooch (Technical Program Chairman and Proceedings Editor), Norman Sondak (Publicity/Publications Chairman), and Katherine Moses (Local Arrangements Chairman).

U.W. POOCH
Special Issue Editor

* The remaining papers appear in Computer Graphics (ACM SIGGRAPH newsletter), Vol. 10, No. 2 (Summer 1976). This special issue is available prepaid from: ACM, P.O. Box 12105, Church Street, New York, NY 10249; ACM and/or SIGGRAPH members $10.00, nonmembers $20.00.

Graphics and
Image Processing

Texture and Reflection in Computer Generated Images

James F. Blinn and Martin E. Newell
University of Utah

In 1974 Catmull developed a new algorithm for rendering images of bivariate surface patches. This paper describes extensions of this algorithm in the areas of texture simulation and lighting models. The parametrization of a patch defines a coordinate system which is used as a key for mapping patterns onto the surface. The intensity of the pattern at each picture element is computed as a weighted average of regions of the pattern definition function. The shape and size of this weighting function are chosen using digital signal processing theory. The patch rendering algorithm allows accurate computation of the surface normal to the patch at each picture element, permitting the simulation of mirror reflections. The amount of light coming from a given direction is modeled in a similar manner to the texture mapping and then added to the intensity obtained from the texture mapping. Several examples of images synthesized using these new techniques are included.

Key Words and Phrases: computer graphics, graphic display, shading, hidden surface removal
CR Categories: 3.41, 5.12, 5.15, 8.2

Copyright © 1976, Association for Computing Machinery, Inc. General permission to republish, but not for profit, all or part of this material is granted provided that ACM's copyright notice is given and that reference is made to the publication, to its date of issue, and to the fact that reprinting privileges were granted by permission of the Association for Computing Machinery.

A version of this paper was presented at SIGGRAPH '76: The Third Annual Conference on Computer Graphics, Interactive Techniques, and Image Processing, The Wharton School, University of Pennsylvania, July 14-16, 1976.

This work was supported in part by ARPA under Contract DAH15-73-C-0363. Author's address: Computer Science Department, University of Utah, Salt Lake City, UT 84112.

Introduction

In 1974 Edwin Catmull [2] developed an algorithm for rendering continuous tone images of objects modeled with bivariate parametric surface patches. Unlike most earlier algorithms [6, 8, 9, 10], which require that objects be approximated by collections of planar polygons, Catmull's algorithm works directly from the mathematical definition of the surface patches. The algorithm functions by recursively subdividing each patch into smaller patches until the image of each fragment covers only one picture element. At this stage, visibility and intensity calculations are performed for that picture element. Since the subdivision process will generate picture elements in a somewhat scattered fashion, the image must be built in a memory called a depth buffer or Z-buffer. This is a large, random access memory which, for each picture element, stores the intensity of the image and the depth of the surface visible at that element. As each patch fragment is generated, its depth is compared with that of the fragment currently occupying the relevant picture element. If greater, the new fragment is ignored, otherwise the picture element is updated.

This paper describes extensions of Catmull's algorithm in the areas of texture and reflection. The developments make use of digital signal processing theory and curved surface mathematics to improve image quality.

Texture Mapping

Catmull recognized the capability of his algorithm for simulating variously textured surfaces. Since the bivariate patch used is a mapping of the unit square in the parameter space, the coordinates of the square can be used as a curvilinear coordinate system for the patch. It is a simple matter for the subdivision process to keep track of the parameter limits of each patch fragment, thereby yielding the parameter values at each picture element. These parameter values may then be used as a key for mapping patterns onto the surface. As each picture element is generated, the parametric values of the patch within that picture element are used as input to a pattern definition function. The value of this function then scales the intensity of that picture element. By suitably defining the pattern function, various surface textures can be simulated.

As Catmull pointed out, simply sampling the texture pattern at the center of each picture element is not sufficient to generate the desired picture, since two adjacent picture elements in the image can correspond to two widely separated points in the patch parameter space, and hence to widely separated locations in the texture pattern. Intermediate regions, which should somehow influence the intensity pattern, would be skipped over entirely. This is a special case of a phe-

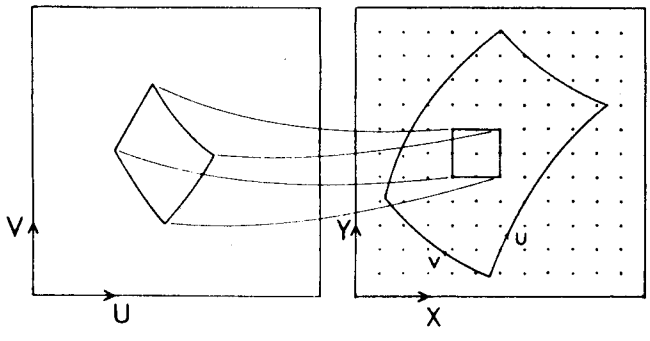
nomenon known as "aliasing" in the theory of digital signal processing. This theory [7] treats the image as a continuous signal which is sampled at intervals corresponding to the distance between picture elements. The well-known "sampling theorem" states that the sampled picture cannot represent spatial frequencies greater than 1 cycle/2 picture elements. "Aliasing" refers to the result of sampling a signal containing frequencies higher than this limit. The high spatial frequencies (as occur in fine detail or sharp edges) reappear under the alias of low spatial frequencies. This problem is most familiar as staircase edges or "jaggies." In the process of texture mapping, the aliasing can be extreme, owing to the potentially low sampling rate across the texture pattern.

To alleviate this problem we must filter out the high spatial frequency components of the image (in this case the texture pattern) before sampling. This filtering has the effect of applying a controlled blur to the pattern. This can be implemented by taking a weighted average of values in the pattern immediately surrounding the sampled point. Digital image processing theory provides a quantitative measure of the effectiveness of such weighting functions in terms of how well they attenuate high frequencies and leave low frequencies intact.

Catmull achieved the effect of filtering by maintaining an additional floating-point word for each picture element. This word contained the fraction of the picture element covered by patch fragments. For each new fragment added to the picture element, the texture pattern was sampled and the intensity was averaged proportionally to the amount of the picture element covered by the patch fragment. Examination of the spatial frequency filter effectively implemented by this technique shows that it is much better than point sampling but is not optimal.

The method discussed here does not require the extra storage and uses a better anti-aliasing filter. This filter is implemented by a weighting function originally used by Crow [3] to minimize aliasing at polygon edges ("jaggies"). It takes the form of a square pyramid with a base width of 2×2 picture elements. In the texture mapping case, the 2×2 region surrounding the given picture element is inverse mapped to the corresponding quadrilateral in the u,v parameter space (which is the same as the texture pattern space), see Figure 1. The values in the texture pattern within the quadrilateral are weighted by a pyramid distorted to fit the quadrilateral and summed.

Fig. 1. Region of texture pattern corresponding to picture element: left-hand side shows texture; right-hand side shows image.



The derivation of the quadrilateral on the texture pattern makes use of an approximation that the parametric lines within one picture element are linear and equally spaced. The X,Y position within a picture element can then be related to the u,v parameters on the patch by a simple affine transformation. This transformation is constructed from the u,v and X,Y values which are known exactly at the four corners of the patch fragment.

Given this algorithm, we now investigate the effects of various texture definition methods. We will use, as our sample object, a plain teapot constructed of 26 bicubic patches. First, the pattern may be some simple function of the u,v parametric values. A useful example of this is a simple gridwork of lines. The result is as though parametric lines of the component patches are painted on the surface, Figure 2. Note that the edges of the pattern lines show very little evidence of aliasing in the form of staircases. Second, the pattern may come from a digitized hand drawn picture, Figure 3. Third, the pattern may come from a scanned-in photograph of a real scene, as in Figure 4. Incidentally, this picture makes the individual patches very clear. This type of pattern definition enables the computer production of "anamorphic" pictures. These are pictures which are distorted in such a way that when viewed in a curved mirror the original picture is regenerated, Figure 5. The patch itself is defined so that the parametric lines are stretched in approximately the correct fashion and a real photograph is mapped onto the patch. Figure 5 should be viewed in a cylindrical mirror (e.g. a metal pen cap) with the axis perpendicular to the page. The fourth source of texture patterns shown here is Fourier synthesis. A two-dimensional frequency spectrum is specified and the inverse Fourier transform generates the texture pattern. This is a simple way of generating wavy or bumpy patterns. Certain restrictions on the form of the input spectrum must be followed to ensure that the pattern has an even distribution of intensities and is continuous across the boundaries. An example of this type of texture is shown in Figure 6. The texture patterns used here were generated before picture synthesis began and stored as 256×256 element pictures in an array in random access memory.

Reflection in Curved Surfaces

The second topic discussed in this paper concerns lighting models. Typically, visible surface algorithms

Fig. 2. Simple gridwork texture pattern: left-hand side shows texture pattern; right-hand side shows textured object.

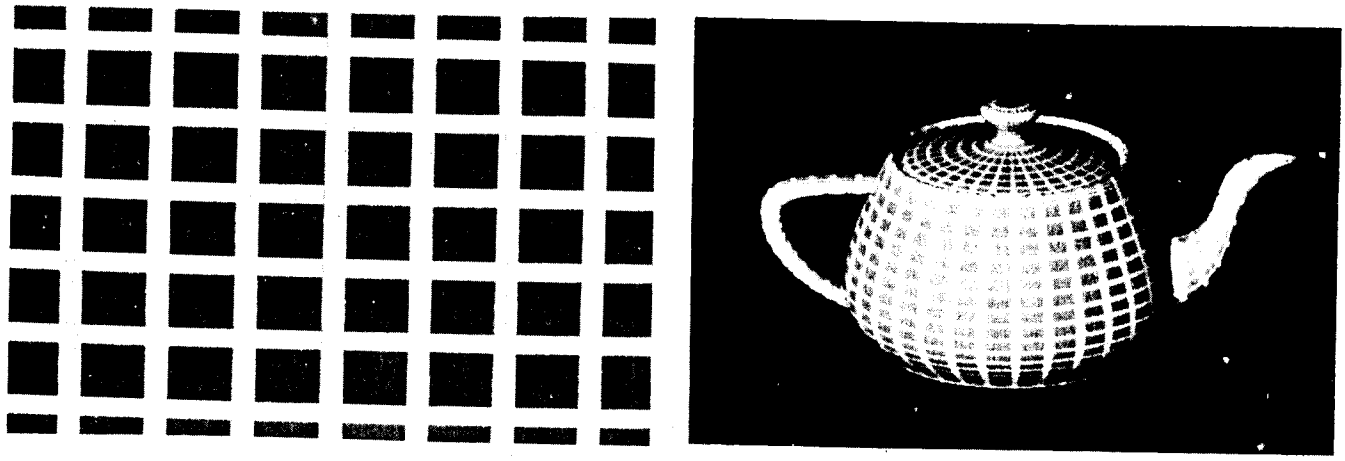
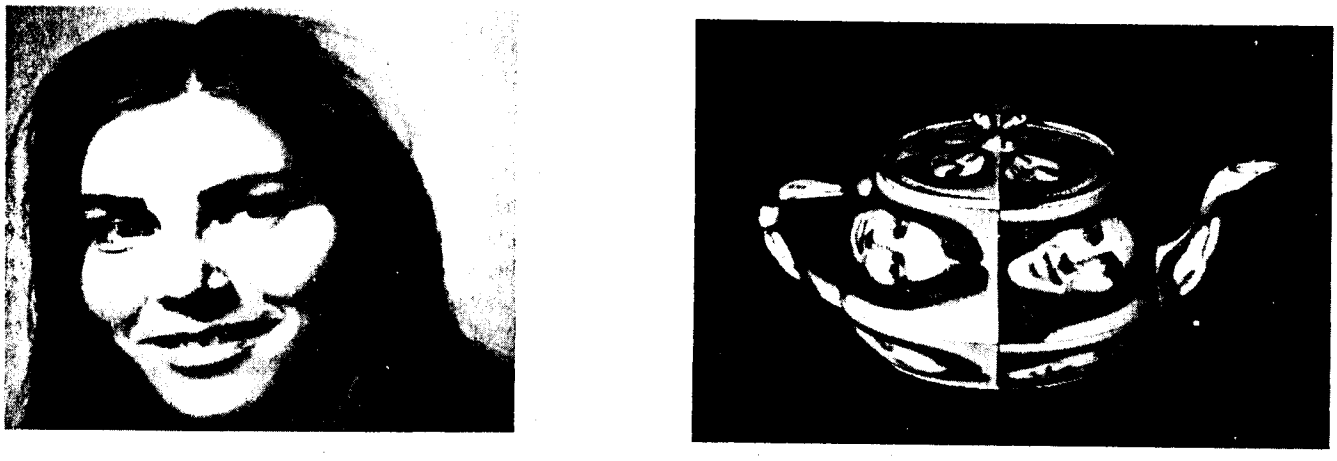


Fig. 3. Hand sketched texture pattern: left-hand side shows texture pattern; right-hand side shows textured object.

Fig. 4. Photographic texture pattern: left-hand side shows texture pattern; right-hand side shows textured object.



determine intensities within an image by using Lambert's (cosine) law: $i = s(L \cdot N)$, where i = intensity, s = surface shade, L = light direction vector, N = surface normal vector, and "$\cdot$" denotes vector inner product.

Variants on this function, such as

$i = s(L \cdot N)**n$, for $n > 1$

have been used to give the impression of shiny surfaces, but there is little physical justification for such functions, and the range of effects is limited. The modeling of more realistic lighting was first investigated by Bui-Tuong Phong [1]. His model of reflection incorporated a term which produced a highlight over portions of the surface where the normal falls midway between the light

source direction and the viewing direction. This is motivated by the fact that real surfaces tend to reflect more light in a direction which forms equal angles of incidence and reflectance with the surface normal. This can be easily implemented by simulating a virtual light source in a direction halfway between the light source and viewing directions, and raising the result to some high power to make the highlights more distinct:

$$i = s(L \cdot N) + g(L' \cdot N)^{**}n$$

where L' = virtual light source direction, g = glossiness of surface (0 to 1). Figure 7 shows an image generated using the above function with $n = 60$. These techniques work well for satin type surfaces but images of highly polished surfaces still lack realism. This is largely due to the absence of true reflections of surrounding objects and distributed light sources.

The simulation of reflections in curved surfaces requires an accurate model of the properties of the surface and access to accurate normal vectors at all points on the surface. The approximation of curved surfaces by collections of planar polygons is inadequate for this purpose, so extensions of the techniques of Gouraud [5] and Bui-Tuong Phong [1] hold little promise.

The subdivision algorithm, however, provides accurate information about surface position and can be made to give accurate surface normals at every picture element. This is the first algorithm that provides the appropriate information for the simulation of mirror reflections from curved surfaces. For each picture element, the vector from the object to the observer and the normal vector to the surface are combined to determine what part of the environment is reflected in that surface neighborhood. It can be shown that, for surface normal vector (Xn, Yn, Zn) and viewing position (1, 0, 0), the direction reflected, (Xr, Yr, Zr), is

$$Xr = 2{*}Xn{*}Zn, \quad Yr = 2{*}Yn{*}Zn, \quad Zr = 2{*}Zn{*}Zn - 1,$$

Having established the direction of the ray which is reflected to the eye, it remains to find what part of the environment generated that ray. For this, a model of the environment is needed which represents surrounding objects and light sources. Clearly, the view of the environment as seen from different points on the surface will vary. However, if it is assumed that the environment is composed of objects and light sources which are greatly distant from the object being drawn, and that occlusions of the environment by parts of the object itself are ignored, then the environment can be modeled as a two-dimensional projection surrounding the drawn object. Stated another way, the object is positioned at the center of a large sphere on the inside of which a picture of the environment has been painted. These simplifications allow the environment to be modeled as a two-dimensional intensity map indexed by the polar coordinate angles of the ray reflected

Fig. 5. Anamorphic image.

Fig. 6. Fourier synthesis of texture: top shows texture pattern; bottom, texture object.

Fig. 7. Plain teapot with highlights.

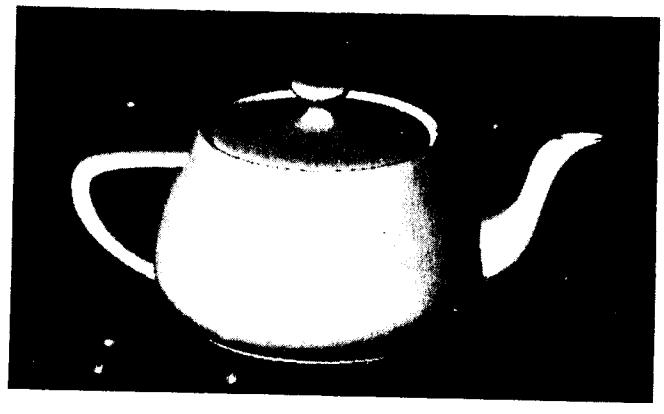
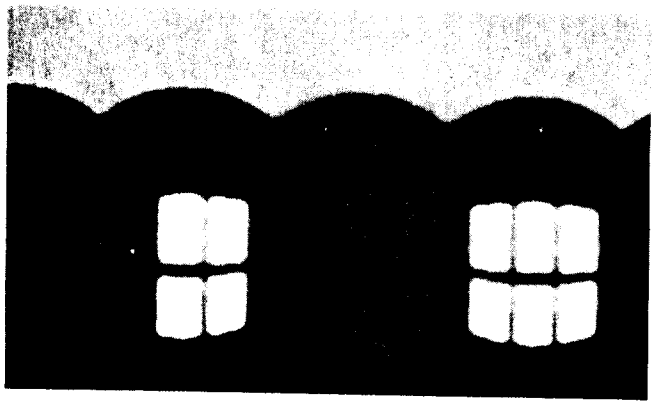
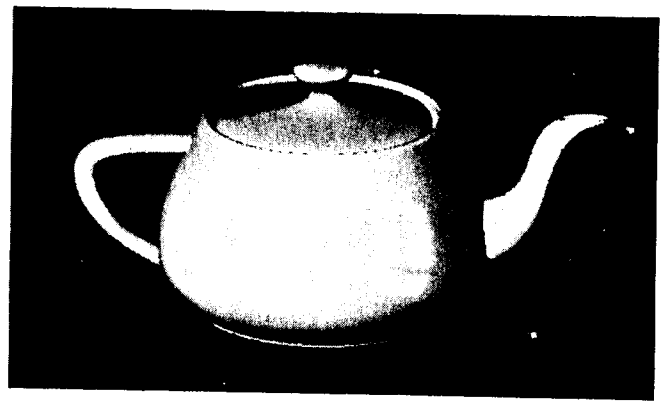


Fig. 8. Computer generated reflections: left-hand side shows environment map; right-hand side shows reflection in teapot.





(Xr, Yr, Zr). (Such maps will be shown with azimuthal angle plotted as abscissa and polar angle plotted as ordinate.)

When a reflection direction is computed, it is converted to polar coordinates and the reflected light intensity for that direction is read from the map. This is similar to the technique used to map texture onto a surface, except that the reflection direction, instead of parametric surface position, determines the coordinates in the map. Figure 8 shows an image generated using these techniques.

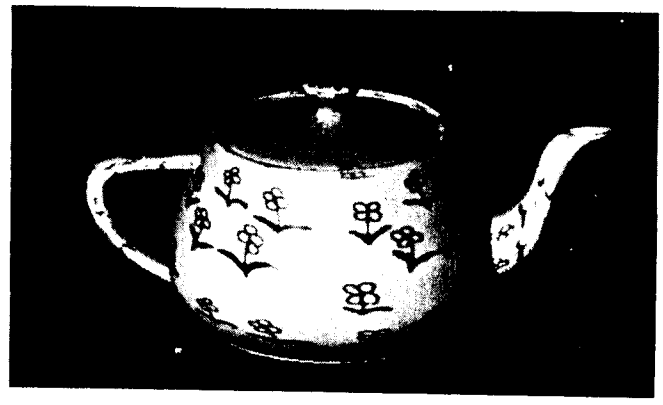
Use of the surface normal alone is tantamount to modeling the environment on an infinitely large sphere. This has the undesirable effect that the reflected intensity at all silhouette points on the object is the same, and corresponds to the intensity of the environment model diametrically opposite the eye. This deficiency can be corrected by using both the surface normal and surface position to determine what part of the environment map is reflected in a given surface fragment.

Combinations of Techniques

The techniques described above for simulating texture and reflection can be combined to produce images of objects having patterned shiny surfaces. When highlighting is combined with texture mapping, only the component from the real light source should be scaled. This models the highlight as being specularly reflected at the surface and not being affected by the pigment within it. In Figure 9 note how the highlights wash out the texture pattern underneath them. The technique for texture mapping actually keys the texture to the surface so that it moves with the object. Some other techniques, which essentially apply texture to the 2-D image, do not have this property. Note, in Figure 9, how the highlights hardly move with the teapot, whereas the texture does.

Given the texture mapping technique and the environment reflecting technique, we can combine them

Fig. 9. Textured object with highlights, two orientations.



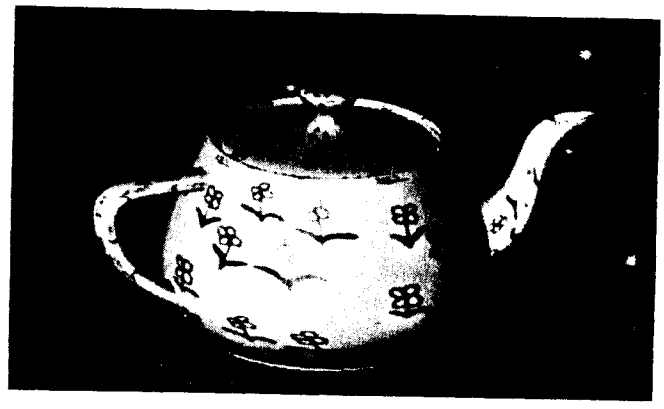
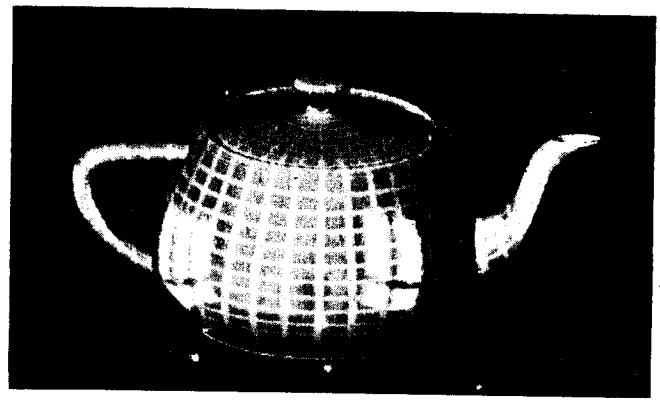


Fig. 10. Highly glazed patterned teapot.



to produce an image of a highly glazed patterned teapot, as in Figure 10.

Resource Requirements

The images shown in this paper were all generated on a PDP-11/45 computer having a 256K-byte random access frame buffer which was used as the depth buffer. The main routines were written in Fortran and the critical parts were written in assembly language. The computation time of the extended subdivision algorithm is roughly proportional to the area covered by visible objects. Images of nontextured objects of the type used in this paper take about 25 minutes. The addition of texture or reflection increases this time by about 10 percent. All images have a resolution of 512×512 picture elements.

Conclusions

By refining and extending Catmull's subdivision algorithm, images can be generated having a far higher degree of naturalness than was previously possible. These generalizations result in improved techniques for generating patterns and texture, and in the new capability for simulating reflections.

References

1. Bui-Tuong Phong. Illumination for computer generated images. *Comm. ACM 18*, 6 (June 1975), 311–317.
2. Catmull, E.A. Computer display of curved surfaces. Proc. Conf. on Comptr. Graphics, Pattern Recognition, and Data Structure, May 1975, pp. 11–17 (IEEE Cat. No. 75CH0981-1C).
3. Crow, F.C. The aliasing problem in computer-synthesized shaded images. Tech. Rep. UTEC-CSC-76-015, Dep. Comptr. Sci., U. of Utah, Salt Lake City, Utah, March 1976.
4. Forrest, A.R. On Coons and other methods for the representation of curved surfaces. *Computer Graphics and Image Processing 1* (1972), 341.
5. Gouraud, H. Computer display of curved surfaces. Tech. Rep. UTEC-CSC-71-113, Dep. Comptr. Sci., U. of Utah, Salt Lake City, Utah, June 1971.
6. Newell, M.E., Newell, R.G., and Sancha, T.L. A solution to the hidden surface problem. Proc. ACM 1972 Ann. Conf., Boston, pp. 443–450.
7. Oppenheim, A.V., and Schafer, R.W. *Digital Signal Processing*. Prentice-Hall, Englewood Cliffs, N.J., 1975, pp. 26–34.
8. Sutherland, I.E., Sproull, R.F., and Schumaker, R.A. A characterization of ten hidden-surface algorithms. *Computing Surveys 6*, 1 (March 1974), 1–55.
9. Warnock, J.E. A hidden-line algorithm for halftone picture representation. Rep. TR 4-15, Dep. Comptr. Sci., U. of Utah, Salt Lake City, Utah, 1969.
10. Watkins, G.S. A real-time visible surface algorithm. Tech. Rep. UTEC-CSC-70-101, Dep. Comptr. Sci., U. of Utah, Salt Lake City, Utah, June 1970.

Graphics and
Image Processing

Hierarchical Geometric Models for Visible Surface Algorithms

James H. Clark
University of California at Santa Cruz

The geometric structure inherent in the definition of the shapes of three-dimensional objects and environments is used not just to define their relative motion and placement, but also to assist in solving many other problems of systems for producing pictures by computer. By using an extension of traditional structure information, or a geometric hierarchy, five significant improvements to current techniques are possible. First, the range of complexity of an environment is greatly increased while the visible complexity of any given scene is kept within a fixed upper limit. Second, a meaningful way is provided to vary the amount of detail presented in a scene. Third, "clipping" becomes a very fast logarithmic search for the resolvable parts of the environment within the field of view. Fourth, frame to frame coherence and clipping define a graphical "working set," or fraction of the total structure that should be present in primary store for immediate access by the visible surface algorithm. Finally, the geometric structure suggests a recursive descent, visible surface algorithm in which the computation time potentially grows linearly with the visible complexity of the scene.

Key Words and Phrases: visible surface algorithms, hidden surface algorithms, hierarchical data structures, geometric models

CR Categories: 5.31, 8.2

Copyright © 1976, Association for Computing Machinery, Inc. General permission to republish, but not for profit, all or part of this material is granted provided that ACM's copyright notice is given and that reference is made to the publication, to its date of issue, and to the fact that reprinting privileges were granted by permission of the Association for Computing Machinery.

A version of this paper was presented at SIGGRAPH 76: The Third Annual Conference on Computer Graphics, Interactive Techniques, and Image Processing, The Wharton School, University of Pennsylvania, July 14–16, 1976.

Author's address: Information Sciences, University of California, Santa Cruz, CA 95064.

1. Introduction

1.1 Background

Early research in computer graphics was concerned with the organization and presentation of graphical information in the form of real-time line drawings on a CRT. Many of the concepts of structuring graphical information were developed by Sutherland in Sketchpad [19], and the line-drawing graphical displays that resulted from his early research remain the most widely used today. With the development of integrated circuit technology, research interests shifted to producing very realistic, shaded, color pictures of the visible parts of complex three-dimensional objects. Because of the desire to utilize television technology, the algorithms for producing these pictures generated output for a raster CRT. The pioneering works in this area were by Schumacker et al. [18] and Wylie et al. [23].

Computer produced pictures now provide one of the most direct and useful ways of communicating with the computer. The ability to produce shaded pictures that illustrate mathematical functions and physical properties of mathematical models is of incontestable value in both research and education. With the development of computer controlled simulators, a real-time computer displayed environment is now used to train pilots of aircraft [11, 16], spacecraft [9] and ocean vessels [2]. Other significant uses of computer pictures include computer aided design [4], modeling of chemical structures [22], and computer animation [7, 12]. With this increased value of computer generated pictures, comes an increasing need to devise efficient algorithms that improve the realism and enhance the descriptive power of these pictures.

1.2 Motivation for New Research

The underlying motivation for new research on computer produced pictures is to either enhance the realism of the pictures or improve the performance of the algorithms that generate them. Most recent research has addressed a combination of these issues.

There are three basic approaches to improving picture quality. The first is to devise clever ways to add information value to a scene without significantly increasing the total amount of information in the database for the scene, for example, without increasing the number of polygons used in representing the objects. Approaches of this type usually make subtle changes to the visible surface and shading algorithms that result in greatly improved pictures. Examples are the improvements to shading algorithms devised by H. Gouraud [10] and Bui-Tuong Phong [15].

The second approach is to employ more refined mathematical models for the objects being rendered and to devise algorithms that can find the visible surfaces using these models. The goal of these methods is to model smooth surfaces with surface patches, such as Coons patches [5] or B-splines [4, 17], rather than with clusters of polygons, and still not increase the size of the database. Catmull's [3] ingenious algorithm is an example of this approach. The benefit of these methods is that an arbitrarily refined description of the model is present, thus allowing much better renditions of contours and shading. The disadvantage is that because of nonlinear mathematics, the algorithms are less efficient than polygon-based algorithms.

The third approach is to increase the information in the database and employ more structured methods for handling the increased information. The motivation for this approach is that the information value of a scene grows in proportion to the amount of information in the database for the scene. Newell's [13] algorithm is an example of this approach.

The structured approach appears to be the most promising of these approaches since it potentially improves both picture quality and algorithm performance. However, there are several problems associated with this approach. First, increased complexity of a scene, or increased information in the database, has less value as the resolution limits of the display are approached. It makes no sense to use 500 polygons in describing an object if it covers only 20 raster units of the display. How do we select only that portion of the data base that has meaning in the context of the resolution of the viewing device? Second, how do we accommodate the increased storage requirements of this additional information? We might, for example, wish to model a human body to the extent that a closeup view of the eye shows the patterns of the iris, yet such a fine description of the entire body will indeed require large amounts of store. Third, how much information must be presented to convey the information content of the scene? In other words, we would like to present the minimal information needed to convey the meaning of what is being viewed. For example, when we view the human body mentioned above from a very large distance, we might need to present only "specks" for the eyes, or perhaps just a "block" for the head, totally eliminating the eyes from consideration. The amount of information "needed" can be the subject of psychological debate, but it is clear that even coarse decisions will yield more manageable scenes than attempting to use all of the available information.

These issues have not previously been addressed in a unified way. The research described here represents an attempt to solve these and related problems.

2. Summary of Existing Algorithms

Visible surface algorithms may be categorized according to whether they employ polygons, parametric surface patches, or procedures as the underlying method of modeling the surfaces they render. The most thoroughly studied types of algorithms use polygons. However, because of the shortcomings of representing

smooth surfaces with faceted clusters of polygons, some research interest has recently been devoted to parametric surface algorithms, which allow higher degrees of continuity than just positional continuity. The algorithms for these different modeling methods will be discussed separately.

2.1 Polygon-Based Algorithms

A highly informative survey of existing polygon-based visible surface algorithms has been written by Sutherland et al. [20]. As they point out, a convenient way to classify these algorithms is according to the order in which they sort the image space polygons that are potentially visible in a scene. The basic difference between the major algorithms is in whether they sort in depth (from the viewpoint) before the vertical-horizontal sort, or vice versa.

Depth-first sort. The most significant algorithms to use this sorting order are due to Schumacker et al. [18] and Newell et al. [14]. Schumacker utilizes this order along with a polygon clustering concept to achieve a coherence from one frame to the next, while Newell utilizes it to render translucent images. By first computing a priority ordering of polygons according to their image space distance from the screen, they are able to establish which polygon *segments* on a given scan line have visibility priority.

Newell uses this information to write those segments with a lesser priority into a scan-line buffer before writing in those with a greater priority. Thus greater priority segments which are from translucent polygons only modify the intensity values in the buffer rather than completely overwriting them. While there is clearly a considerable overhead in writing into the buffer segments that might eventually be obscured, some beautiful pictures have resulted from this work.

Schumacker's goal is to produce real-time picture sequences. Rather than writing the polygon segment information for a scan-line into a buffer according to its priority, a set of priority-ordered hardware registers are simultaneously loaded with the priority-ordered segment information. Then as the scan line is displayed, the register information is counted down and a combinational-logic network selects the appropriate highest priority register according to its lateral displacement on the screen. This approach requires a separate set of registers for each polygon segment that intersects the scan line. Nonetheless, it represents the first real-time solution to the visible surface problem [9].

There are two very significant features to Schumacker's work. First, he makes use of a priori knowledge of the database to compute fixed priorities for clusters of polygons. If the polygons in a group of polygons are not subject to changes in relative placement, they form a *cluster* and may be assigned fixed priorities which work no matter from where the cluster is viewed. Thus part of the priority ordering is fixed with the environment and need not be recomputed each frame. Second, he shows that if the environment is restricted so that the clusters are linearly separable, an intercluster priority can be established that does not change unless the viewpoint crosses one of the separating planes; hence, the priority ordering remains fixed from one frame to the next unless one of the planes is crossed.

This work by Schumacker and coworkers represents the only visible surface algorithm to make use of both structured information (clustering) and frame to frame coherence (relatively constant intercluster priority). These very important concepts will be discussed in more detail later.

Depth-last sort. The algorithms that use this sorting order have been devised by Watkins [21], Bouknight [1], and Wylie et al. [23]. They are referred to as scan-line algorithms and differ only in their use of various image-space coherence properties. All three first perform a vertical bucket (radix) sort of polygon edges according to their uppermost vertices. Then for each scan line, the various polygon segments on that scan line are sorted according to their horizontal displacements. The depth sort is deferred until last under the assumption that the initial two sorts will decrease the number of depth comparisons needed to determine final visibility.

Of the three approaches, Watkins' is the most economical because of its uses of scan-line coherence and a logarithmic depth search. The assumption of scan-line coherence is that in going from one scan line to the next, the number of changed polygon segments is small; hence the horizontal sort may be optimized to take advantage of this. Watkins' is the only other algorithm besides Schumacker's that has been implemented in hardware.

2.2 Parametric Surface Algorithms

Modeling smooth surfaces with collections of polygons leads to problems both in shading the surface and in rendering the contour edges. While there have been a number of very clever improvements to the quality of such pictures without significantly increasing the amount of information used, notably those of Gouraud [10], Phong [15], and Crow [6], the most direct approach is to employ a more refined model, such as parametric surface patches. Such patches can be used to define the surface using no more, and usually even less, information than is required with polygons. Yet they can join together with tangent or even higher continuity, thus eliminating the above problems. The difficulty with this method is that the mathematics is no longer linear; to explicitly solve for such things as the curve of intersection of two bi-cubic patches or of a patch and a clipping plane are very difficult problems.

Catmull [3] solves such problems, but not explicitly. Rather, he does so by employing the discrete character of the image space, a recursive algorithm, and what he calls a Z-buffer. For each patch in the

environment his algorithm asks: does the patch extend over more than a single raster unit? If the answer is yes, the patch is subdivided (by a very fast algorithm for bi-cubic patches) into four patches and the same question is *recursively* asked of these patches. When the answer finally is no, an attempt is made to write the intensity and depth coordinates for the resulting "patch" into a buffer for the raster unit, or *pixel*, in question. The attempt fails if the pixel buffer already has in it a depth coordinate nearer to the observer (with some minor modifications to allow for translucent patches).

A very significant feature of Catmull's algorithm is that, despite the more complex mathematics, it will actually work faster than polygon-based algorithms if the object being rendered occupies a very small area of the screen. Because of the recursive structure of the algorithm, it will "structure" the surface no more finely than the resolution of the display dictates, whereas current polygon-based algorithms keep the same structural description, i.e. the same number of polygons, no matter how much of the screen area is occupied. This notion of structuring will be extended to include polygon-based algorithms in the next section.

2.3 Procedurally Modeled Objects

Newell [13] has recently employed procedural modeling to solve the visible surface problem for complex scenes. According to this approach, objects are modeled using procedures which "know how" to render themselves in terms of their own primitives, which might include activations of other object procedures; such knowledge includes rendering only their visible parts. This is a very general way to represent objects.

Although the underlying philosophy of this approach is very general, in the actual implementation Newell user polygons as the basic primitives for the objects. The object procedures are activated according to a priority ordering so that more distant objects are activated first. Each procedure renders the object it represents by activating the Watkins process, the results of which are written into a frame buffer. The net result is therefore a "hybrid" Watkins/Newell priority algorithm.

The significant point about this algorithm is not the procedural modeling but that it represents another example of structuring to simplify the total sorting problem, namely that the geometric primitives of one object need be compared with those of another only when the objects overlap.

3. Hierarchical Approach

It was indicated in the previous section that, aside from uses of image-space coherence to reduce the amount of sorting required, the most fruitful gains in visible surface algorithm research have resulted from structuring the environments being rendered. However, the structures employed take a diverse variety of forms, from Catmull's implicit structuring of surface patches to Newell's procedural objects. What is needed is a single, unified, structural approach that embodies all of the ideas from these algorithms. Before presenting one such approach it is instructive to consider two ways in which structure has been utilized to prepare objects for visible surface processing.

3.1 Existing Uses of Structure

Defining relative placement. The benefits of a position or motion structure have been realized for some time. Sutherland used such concepts in two dimensions in Sketchpad, and a number of graphics hardware companies incorporate transformation hardware in their display devices to accommodate structural descriptions. Most of the visible surface algorithms presented used a position or motion structure to describe positions and orientations of objects relative to each other. However, all but the few mentioned in Section 2 disregard the structure at the visible surface algorithm level. That is, all polygons of the objects are transformed into a common screen coordinate system in which the visible surface algorithm works.

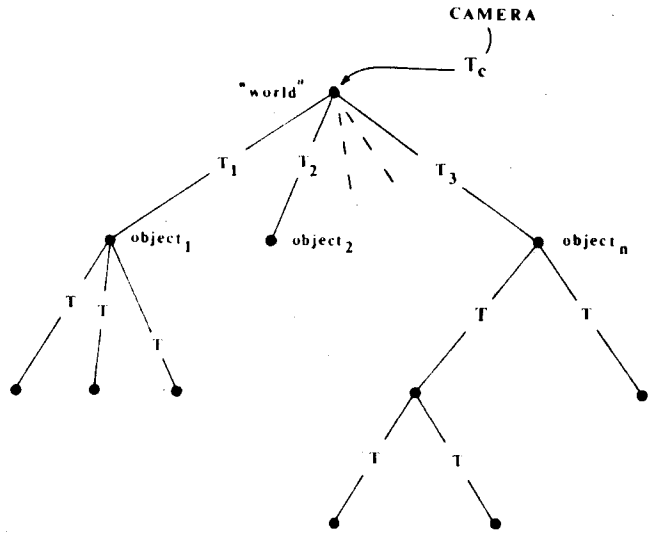
An example of such a structure is shown in Figure 1. Each node in the hierarchy represents a set of geometric primitives (e.g. polygons) defining the node and the arc leading to the node represents a transformation defining the orientation and placement of the node relative to its "parent." Because each node has its own unique transformation defining it, it may represent one of many "instances" of the same primitive description, or data set. This is a very convenient and general way to define and place objects.

Decreasing clipping time. When simulating a camera in a computer-generated environment, some parts of the environment must be "clipped" to the field of view of the simulated camera. This can be done either by transforming all of the geometric primitives of each object into the camera, or screen, coordinate system and clipping each of them separately or by first clipping some bounding volume of the object to see if it intersects the boundaries of the field of view. If it does not, then the parts of the object lie either totally within or totally outside of the field of view and thus need not be separately clipped. This utilization of the above mentioned position hierarchy is implicitly assumed, although this author does not know if the authors of the various algorithms actually made such use of it.

3.2 New Uses of Structure

Varying environment detail. By choosing to represent an object with a certain amount of detail, one fixes the minimum distance from which the object may be displayed with a realistic rendering. For example, a dodecahedron looks like a sphere from a sufficiently large distance and thus can be used to model it so long

Fig. 1. The traditional motion structure used to position objects relative to the "world" and subobjects relative to objects. Each arc in the graph represents a transformation.



as it is viewed from that or a greater distance. However, if it must ever be viewed more closely, it will look like a dodecahedron. One solution to this is simply to define it with the most detail that will ever be necessary. However, then it might have far more detail than is needed to represent it at large distances, and in a complex environment with many such objects, there would be too many polygons (or other geometric primitives) for the visible surface algorithms to efficiently handle.

As mentioned in Section 2, the solution to this problem has been to define objects relatively coarsely and employ clever algorithms that smooth appropriate contours or improve shading to make the object look more realistic at close observation. The difficulty with these approaches is that at best the range of viewing depth is only slightly improved, and the problem of too much detail at large distances usually remains. Although these approaches have yielded results of unquestionable value, it seems evident that multiple levels of description must be used to adequately represent complex environments.[1]

How does one represent these multiple levels of description? A solution is to define "objects" in a hierarchy like that of Figure 2. The entire environment is itself an "object" and is represented as a rooted tree. ("Object" is a generic term for the things represented by nodes of the tree. This generic term will be used for the remainder of this paper.) There are two types of arcs in the tree, those that represent transformations as before and those that represent pointers to more detailed structure (the identity transformation). Each nonterminal node represents a "sufficient" description of the "object" if it covers no more than some small area of the display; the arcs leading from the node point to more detailed "objects" which collectively define a more detailed version of the original object if its description is insufficient because it covers a larger area of the screen. The terminal nodes of the tree represent either polygons or surface patches (or other primitives) according to whether they are primitive elements of a faceted or a smooth object.

As an example of such a description, consider a model of the human body. When viewed at a very large distance, for example when the body covers only 3 or 4 display raster units, it is sufficient to model the body with a single rectangular polyhedron with appropriate color. Therefore the uppermost node, or "object," for this body represents this simple description. If the body is viewed from a closer distance—for example, if its topmost node's description covers 16 raster units—then this topmost description is no longer sufficient, and the next level of more refined description is needed. At this next level the body is now perhaps described as a collection of rectangular polyhedra appropriately attached to each other, for example using one polyhedron for each of the arms and legs, the head and the torso. Then so long as each of these "objects" covers only a few raster units of the display, their description is "sufficient." When the viewing distance decreases such that any of them covers a critical maximum area of the display, its more detailed subobjects are used to replace its description. This process is carried out to whatever maximum level of detail will be needed. For example, a terminal level of description of the fingertip might be several surface patches (which could be implicitly structured even more finely using Catmull's algorithm).

The body described is just one "object" of an environment, or larger hierarchy. There might be many such bodies, or other objects. The significant point, however, is that in a complex environment, the amount of information presented about the various objects in the environment varies according to the fraction of the field of view occupied by these objects.

It is worth noting again that Catmull's algorithm, described in the previous section, implicitly built such a structure. His algorithm used this structure in such a way that, despite the more complex mathematics of surface patches, it outperforms polygon-based algorithms if the surface occupies a small area of the screen. Thus it seems that such a structure should lead to improvements in polygon-based algorithms as well.

Clipping: a truncated logarithmic search. The choice of this structural representation poses another problem. How does one select only that portion of a potentially very large hierarchy that is meaningful in the context of the viewpoint and the resolution of the viewing device? In other words, clipping in a broader sense must mean selecting not only that part of the environment within the field of view (the usual meaning) but also just the *resolvable* part. This implies finding the visible nodes of

[1] Actually, Evans and Sutherland made use of a three-level description of the New York skyline in its Maritime simulation, but in an ad hoc way [2].

the tree, as shown in Figure 2. The contour shown in the figure represents a possible set of objects that are within the field of view and are both not too large and not too small for the screen areas they occupy.

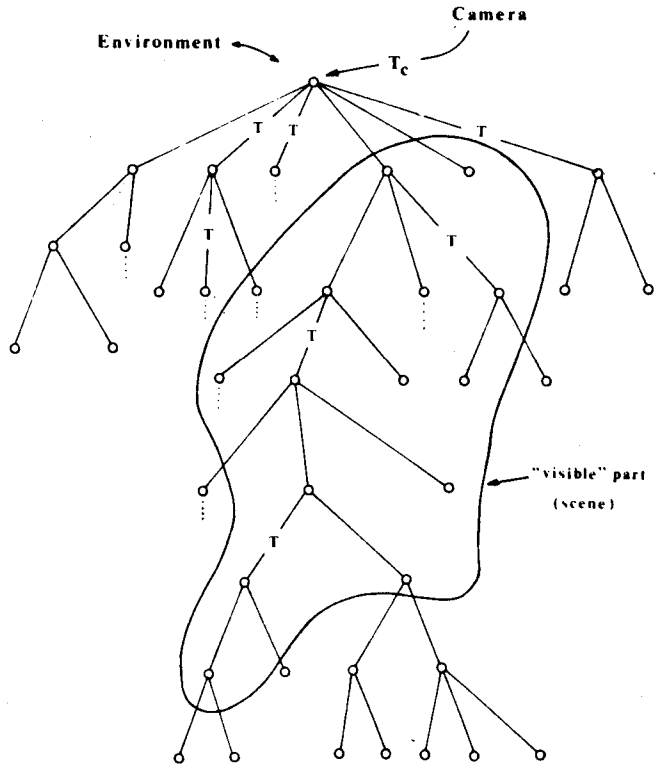
In order to efficiently perform this clipping operation some minimal description of object sizes must be available. For example, a bounding rectangular box or a bounding sphere would be sufficient information to test whether an object is totally within or totally outside of the field of view. The minimum necessary information is the center and radius of a bounding sphere.

This general structure therefore suggests a very fast clipping algorithm which recursively descends the tree, transforming (if necessary) this minimal information into perspective viewing coordinates and testing both the area occupied by the bounding sphere and its intersection with the boundaries of the field of view. The criterion for descending a level is the area test, while the criterion for inclusion rejection is the field of view boundary test. Only after either the area test terminates the descent or the terminal level of representation is reached is it necessary to actually transform and possibly clip the polygons or surface patches represented by the node. Clipping therefore resembles a logarithmic search that is truncated by the area (resolvability) test.

This relatively simple mechanism for varying the detail in a scene suggests several other interesting possibilities. Since the center of attention of a scene is often its geometric center, one might effectively render the scene with a center-weighting of detail. In other words, the maximum area an object is allowed to cover before splitting it into its subobjects becomes larger towards the periphery of the field of view. This is somewhat analogous to the center-weighted metering systems of some cameras. Likewise, since moving objects are less resolved by both the human eye (because of saccadic suppression) and a camera (because of blurring), one can render them with an amount of detail that varies inversely with their speeds. Indeed, an entire scene might be rendered with less detail if the camera is moving. Thus "clipping" can be extended to include these concepts as well.

Graphical working set. Since the problems addressed by this model are those associated with producing pictures and picture sequences of very complex environments, the excessive storage needed for the geometric description of these environments must somehow be accommodated. Denning's "working set" model for program behavior provides a useful analogy [8]. According to this model, a computer program that makes excessive demands on immediate-access store is structured or segmented, and its storage demands are managed in such a way that only those segments most recently in use are actually kept in immediate-access store. The remaining potentially large number of segments are kept on a slower, secondary store, such as a disk. The "working set" is that set of segments

Fig. 2. A very deep hierarchy that structures the environment much more than the traditional motion structure. Arcs in this graph represent either transformations or pointers to more refined definitions of the node. The visible part contour represents a possible result of clipping.



available for immediate access, and is usually defined by a time average of past program reference patterns. Reference to an unavailable segment causes that segment to become part of the working set, and segments not accessed after some period of time are deleted from the working set.

This working set model coupled with the broader sense of clipping mentioned above suggests a suitable way to accomplish a particular type of frame coherence. The working set in this context is that set of objects in the hierarchy that are "near" to the field of view, inside it, or "near" to the resolution of the image space. Only if an object is a member of this set is its description kept in immediate-access store. The set membership will change slowly since the differences between one scene and the next are usually small. Those cases in which the differences are large due to fast camera (or object) motion are easily accommodated by rendering the scene (or object) with less detail, as mentioned above. Moreover, the minimal description of node size needed for clipping suffices as the graphical analog of the segment table used in the computer program context. That is, this minimal clipping description must always be available in immediate-access store to facilitate determining the working set. This working set model therefore seems particularly well suited to the graphics context.

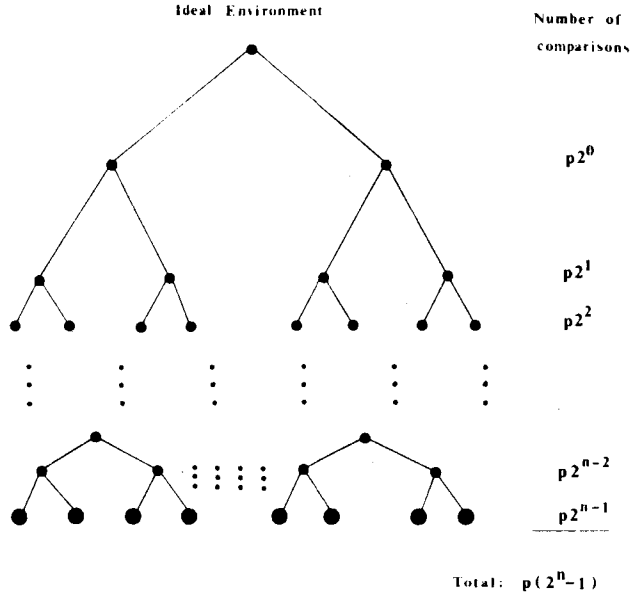
Improving existing algorithms. There are two ways in which a geometric hierarchy should lead to improvements in existing algorithms. The first is by reducing the number of comparisons needed to sort objects and the second is by eliminating from potential consideration an entire portion of the environment because an object obscures it.

Since sorting is the central problem of visible surface algorithms, the performance of these algorithms improves with improved sorting methods. Indeed, many of the fast visible surface algorithms that have been discussed have resulted from clever utilization of image-space coherences, such as scan-line coherence, to improve sorting speeds. In the present hierarchical framework, the geometric proximity of the subobjects of an object provides an object-space coherence that can also be utilized to decrease sorting time.

For example, consider an ideal case of a binary tree as shown in Figure 3. Each node of the tree has associated with it a bounding volume, but since this ideal tree is the result of clipping, only the terminal nodes actually represent geometric primitives, e.g. polygons or patches. Assuming that there are n levels in the tree, not counting the root node, there are $m = 2^n$ terminal nodes.

If the structure is ignored, then the fastest possible sort of these terminal nodes is accomplished with proportional to $m \log_2 m$ comparisons using a quicksort. However, if the structure is utilized and if the bounding volumes of siblings do not overlap, which is admittedly an optimum arrangement, then the number of required operations is $p2^0$ for the first level, $p2^1$ for the second level, $p2^2$ for the third, etc., where p is a proportionality factor. Summing the number of operations performed at all levels yields $p\sum_{i=0}^{n-1} 2^i = p(2^n - 1)$, or roughly pm. In other words, by using the structure, in the optimum situation of no overlap, the sorting time grows linearly rather than as $m \log_2 m$.

Fig. 3. An ideal binary hierarchy in which none of the terminal nodes overlap. The first $p2^0$ comparison sorts all objects into two classes, the second $p2^1$ comparisons sort them into 4 classes, etc. Summing all comparisons from all levels yields $p(2^n - 1)$ comparisons.



Of course, this analysis holds only for a binary hierarchy in which none of the siblings' bounding volumes overlap, which is an idealized situation. A binary hierarchy might not be appropriate, and any complex environment will no doubt have some overlap, although presumably not a very large amount. However, the point here is that sorting methods which utilize the geometric structure can yield a considerable performance improvement over those which do not, even under less than ideal conditions.

The other improvement provided by a deeply structured geometric hierarchy is that of eliminating a potentially large part of the structure from consideration because an object obscures it. Such an improvement requires defining for each object (in the generic sense) both a simple occluded volume, Δ, such that if Δ is obscured then the entire object is obscured, and a simple occluding volume, δ, such that if δ obscures something then that thing is sure to be obscured by the object. Clearly, Δ exists for all objects, whereas δ might not exist for some objects, such as an open-ended cylinder or a transparent object. Δ can be just the bounding sphere used in clipping, but δ is in general additional information that must be kept for each object.

Recursive descent, visible surface algorithm. The above considerations suggest a totally new recursive-descent visible surface algorithm in which at each level all objects are sorted according to their bounding volumes. If any of the bounding volumes overlap both laterally and vertically then the occlusion test potentially allows one (or more) of the objects, and hence all of its descendents, to be totally eliminated from consideration.

Using the ordering thus obtained, the same sorting and occlusion tests are recursively applied to each of the descendants of these objects; in those cases where two or more objects' bounding volumes overlap in all three dimensions, indicating potential intersections, the descendents of these objects are treated as if they have the same parent nodes at the next level of recursion. Of course, recursion terminates when a terminal node is reached, and the net result of descending the tree is a very rapid sort of the primitives represented by these terminal nodes. Under ideal conditions, the computation time of this algorithm grows linearly with the *visible* complexity of the scene.

Since both this algorithm and the clipping algorithm described above recursively descend a tree structure, it seems natural to combine them. Doing so not only potentially eliminates area tests on occluded objects but also potentially decreases the size of the working set. If all processing is performed by a single pro-

cessor, such as a general purpose computer, then the algorithms are probably most conveniently integrated into a single algorithm. However, if multiple processors are available, whether special purpose hardware or general purpose computers, then the algorithms might be left separate or combined according to whether parallelism is achieved by pipelining or otherwise.

Building structured databases. Obtaining a good graphical database is a very time consuming and difficult part of computer picture research. Databases obtained by careful measurement of real objects, by "building" objects from collections of simple mathematical objects, or by sculpturing surfaces in three dimensions [4] are at least as valuable as the visible surface algorithms that render them.

At first glance it appears that the structural framework multiplies the dimensions of this problem since multiple descriptions of the same object must be defined. However, in the case of carefully measured real objects, the multiple descriptions can be produced by judicious "bottom-up" pruning of existing definitions of the objects in their most detailed form. Therefore use can be made of all objects that have already been defined.

Those existing objects modeled with surface patches also present no problem. The coarser, high-level descriptions of these objects can be obtained by replacing the patches themselves with polygons and proceeding with the "bottom-up" pruning mentioned above to obtain even coarser descriptions. The finer, low-level descriptions of the objects can be obtained by "top-down" splitting of the surface patches, as in the Catmull algorithm. This can be done either at display time or beforehand in building the database; the difference is the traditional time/space tradeoff.

4. Conclusions

All of the recent major advances in computer picture research have resulted from either explicitly or implicitly incorporating structure information in the geometric modeling techniques. This research represents an attempt to encompass all of these advances in a more general structural framework as a unified approach to solving a number of the important problems of systems for producing computer pictures.

The proposed hierarchical models potentially solve a number of these problems. They provide a meaningful way to vary the amount of detail in a scene both according to the screen area occupied by the objects in the scene and according to the speed with which an object or the camera is moving. They also extend the total range of definition of the object space and suggest convenient ways to rapidly access objects by utilizing a graphical working set to accomplish frame coherence.

An important aspect of the hierarchical models is that by providing a way to vary detail they can yield an incremental improvement to existing systems for producing computer pictures without modifying their visible surface algorithms. Another incremental improvement is then possible by incorporating the structure in the sorting phases of existing algorithms. A final improvement is suggested by a totally new recursive descent visible surface algorithm in which the computation time potentially grows linearly with the visible complexity of a scene rather than as a worse than linear function of the object-space complexity.

References

1. Bouknight, W.J. A procedure for generation of three-dimensional half-toned computer graphics representations. *Comm. ACM, 13*, 9 (Sept. 1970), 527.
2. Computer Aided Operations and Research Facility, U.S. Maritime Service Simulator (principal contractor Philco-Ford, visible-surface processor by Evans and Sutherland Comptr. Corp.).
3. Catmull, E. A subdivision algorithm for computer display of curved surfaces. Tech. Rep. UTEC-CSc-74-133, U. of Utah, Salt Lake City, Utah, Dec. 1974.
4. Clark, J.H. 3-D design of free-form B-spline surfaces. UTEC-CSc-74-120, Ph.D. Th., U. of Utah, Salt Lake City, Utah, (abridged version Designing surfaces in 3-D. *Comm. ACM 19*, 8 (Aug. 1976), 464–470.)
5. Coons, S.A. Surfaces for computer-aided design of space forms. Project MAC TR-41., M.I.T., Cambridge, Mass., June 1967.
6. Crow, F.C., and Bui-Tuong Phong. Improved Rendition of Polygonal Models of Curved Surfaces. Proc. Second USA-Japan Comptr. Conf., Aug. 1975, p. 475.
7. Csuri, C. Computer animation, Computer Graphics *9*, 1 (1975), 92–101 (Issue of Proc. Second Ann. Conf. Comptr. Graphics and Interactive Techniques).
8. Denning, P.J. The working set model for program behavior. *Comm. ACM, 11*, 5 (May 1968), 323–333.
9. Electonic scene generator expansion system. Final Rep., NASA Contract NAS 9-11065, Defense Electronic Div., General Electric Corp., Syracuse, N.Y., Dec. 1971.
10. Gouraud, H. Computer display of curved surfaces. *IEEE Trans. Computers C-20* (June 1971), 623.
11. Nasa-Ames Short Take-off and Landing Simulator (built by Evans and Sutherland Comptr. Corp.).
12. New York Inst. Tech., Comptr. Animation Dep.
13. Newell, M. The utilization of procedure models in digital image synthesis. Ph.D. Th., Comptr. Sci., U. of Utah, Salt Lake City, Utah, 1975.
14. Newell, M.E., Newell, R.G., and Sancha, T.L. A new solution to the hidden-surface problem. Proc. ACM 1972 Ann. Conf., pp. 443–448.
15. Bui-Tuong Phong. Illumination for computer generated pictures. *Comm. ACM 18*, 6 (June 1975), 311–317.
16. Rediflow Flight Simulation, Ltd., NOVOVIEW Visual Systems (video system provided by E&S Comptr. Corp.).
17. Riesenfeld, R.E. Applications of B-spline approximation to geometric problems of computer aided design. Ph.D. Th., Syracuse U., Syracuse, N.Y., 1972.
18. Schumacker, R.A., Brand, B., Gilliland, M., and Sharp, W. Study for applying computer-generated images to visual simulations. AFHRL-TR-69-74, US Air Force Human Resources Lab., Washington, D.C., Sept. 1969.
19. Sutherland, I.E. Sketchpad: a man-machine graphical communication system. TR 296, M.I.T Lincoln Labs, M.I.T., Cambridge, Mass., Jan. 1963.
20. Sutherland, I.E., Sproull, R.F., and Schumacker, R.A. A characterization of ten hidden-surface algorithms. Computing Surveys, *6*, 1 (March 1974), 1–55.
21. Watkins, G.S. A real-time visible-surface algorithm. UTECH-CSc-70-101, Ph.D. Th., Comptr. Sci. Dep., U. of Utah, Salt Lake City, Utah, June, 1970.
22. Wipke, T., et al. *Computer Representation and Manipulation of Chemical Information*. Wylie Interscience, New York, 1974.
23. Wylie, C., Romney, R.S., Evans, D.C., and Erdahl, A. Halftone perspective drawings by computer. Proc. AFIPS 1967 FJCC, Vol. 31, AFIPS Press, Montvale, N.J., pp. 49–58.

Graphics and
Image Processing

A Parametric Algorithm for Drawing Pictures of Solid Objects Composed of Quadric Surfaces

Joshua Levin
New York University

An algorithm for drawing pictures of three-dimensional objects, with surfaces made up of patches of quadric surfaces, is described. The emphasis of this algorithm is on calculating the intersections of quadric surfaces. A parameterization scheme is used. Each quadric surface intersection curve (QSIC) is represented as a set of coefficients and parameter limits. Each value of the parameter represents at most two points, and these may easily be distinguished. This scheme can find the coordinates of points of even quartic (fourth-order) intersection curves, using equations of no more than second order. Methods of parameterization for each type of QSIC are discussed, as well as surface bounding and hidden surface removal.

Key Words and Phrases: computer graphics, hidden-surface removal, quadric surface intersection curves

CR Categories: 3.41, 5.12, 5.13, 5.19, 8.2

Copyright © 1976, Association for Computing Machinery, Inc. General permission to republish, but not for profit, all or part of this material is granted provided that ACM's copyright notice is given and that reference is made to the publication, to its date of issue, and to the fact that reprinting privileges were granted by permission of the Association for Computing Machinery.

A version of this paper was presented at SIGGRAPH 76: The Third Annual Conference on Computer Graphics, Interactive Techniques, and Image Processing, The Wharton School, University of Pennsylvania, July 14–16, 1976.

The work reported here is discussed in detail in a technical report [4] and was supported in part by the Air Force Office of Scientific Research, Directorate of Mathematical and Information Sciences, under Grant AFOSR 75-2755.

Author's present address: Department of Electrical and Systems Engineering, Renssalaer Polytechnic Institute, Troy, NY 12181.

1. Introduction

This paper presents an algorithm for computer generation of orthographic and perspective drawings of three-dimensional objects whose surfaces are made up of segments of quadric surfaces, also known as quadric patches.

There are now many algorithms for generating pictures of solid objects with surfaces consisting of planar polygons. A polygon is a segment of a first-order surface (plane) bounded by other first-order surfaces. The Braid algorithm [1] extends this to allow cylinders and parts of cylinders. Cylinders are quadric surfaces.

This paper is partially based on the Woon algorithm [11, 12], which processes second-order (quadric) surfaces bounded by second-order surfaces. Quadric surfaces include such familiar objects as spheres, cylinders, cones, and hyperbolic paraboloids. They are not too complex mathematically.

An important advantage of using quadric patches is that an object whose surface is approximated by many small polygons may be better modeled by a few larger quadric patches. Another advantage is that shaded pictures may be made with better control over such effects as mach banding and specular reflection [9].

Previous quadric patch algorithms, such as those by Mahl [6] and Weiss [10], as well as the Woon algorithm, run into difficulty in solving for the intersection of two quadric surfaces. This is a fourth-order problem, and there is no easy method for finding roots of a fourth-order equation, let alone discovering if there are any real roots.

The algorithm presented here reduces these fourth-order problems to second-order problems. The quadratic equation formula, familiar from high school mathematics, provides a rather simple method to solve these equations.

In addition, this algorithm uses a parametric representation. Every point on the intersection of two quadric surfaces is determined uniquely by a numerical parameter and, in many but not all cases, by a Boolean sign parameter. Since these fourth-order quadric surface intersection curves (QSICs) can be represented parametrically, the computer memory need only store about a score of numbers to represent a QSIC, rather than storing the coordinates of dozens of points for each QSIC.

2. Theory—Quadratic Form

2.1 Discriminant

The quadratic surface, as represented in eq. (1) below:

$$q(x, y, z) = q_1x^2 + q_2y^2 + q_3z^2 + q_4xy + q_5yz + q_6zx + q_7x + q_8y + q_9z + q_0 = 0 \quad (1)$$

is a quadratic form [3, 8, 11] which may be represented in vector matrix form, as follows.

Suppose the location of a point in 3-space is represented as the following vector: $\mathbf{x} = (x\ y\ z\ 1)$. Equation (1) may be represented as

$$q(x, y, z) = q(\mathbf{x}) = (x\ y\ z\ 1) \begin{pmatrix} q_1 & \frac{1}{2}q_4 & \frac{1}{2}q_6 & \frac{1}{2}q_7 \\ \frac{1}{2}q_4 & q_2 & \frac{1}{2}q_5 & \frac{1}{2}q_8 \\ \frac{1}{2}q_6 & \frac{1}{2}q_5 & q_3 & \frac{1}{2}q_9 \\ \frac{1}{2}q_7 & \frac{1}{2}q_8 & \frac{1}{2}q_9 & q_0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \quad (2)$$

or, for short,

$$q(\mathbf{x}) = \mathbf{x}Q\mathbf{x}^T \quad (3)$$

where Q is the 4×4 matrix in eq. (2). Q is called the discriminant of the quadric surface. Note that Q is symmetric, and that, for any real nonzero scalar β, that βQ is equivalent to Q, since they describe the same surface. In this paper, the same symbol may be indiscriminantly used for both a quadric surface and its discriminant.

This author often prefers to use an alternate form:

$$Q = \begin{pmatrix} A & D & F & G \\ D & B & E & H \\ F & E & C & J \\ G & H & J & K \end{pmatrix},$$

such that

$$q(x, y, z) = Ax^2 + By^2 + Cz^2 + 2Dxy + 2Eyz + 2Fzx + 2Gx + 2Hy + 2Jz + K = 0.$$

The upper-left 3×3 principal submatrix of the discriminant contains all the second-order terms. This will be called the "subdiscriminant":

$$Q_u = \begin{pmatrix} q_1 & \frac{1}{2}q_4 & \frac{1}{2}q_6 \\ \frac{1}{2}q_4 & q_2 & \frac{1}{2}q_5 \\ \frac{1}{2}q_6 & \frac{1}{2}q_5 & q_3 \end{pmatrix} = \begin{pmatrix} A & D & F \\ D & B & E \\ F & E & C \end{pmatrix}.$$

The subdiscriminant will always be represented with the subscript u.

The rank of the discriminant and the subdiscriminant are helpful in classifying quadric surfaces. These are invariant under the transformations outlined below. In fact, the rank of the discriminant is invariant under any nonsingular transformation, and the rank of the subdiscriminant is invariant under any transformation in which the upper-left 3×3 submatrix is nonsingular.

2.2 Transformations of the Discriminant

In this paper, there will usually be two coordinate systems. In setting up the object, the *untransformed* space will be the *uvw* space; and the *transformed* space will be the *xyz* space. The transformation matrices will be F and $\mathfrak{F}$, each being the inverse of the other.

First, let us consider the transformation: $\mathbf{x} = \mathbf{u}F$,

Table I. Guide to Classification of Quadric Surfaces. (The imaginary part is in parentheses.)

d	m	s	Conditions	Surface
			Singular planar	
1	0	0		*invalid*
1	1	1		coincident planes
2	0	0		single plane
2	1	1	$D_2 > 0$	*invalid* (imaginary parallel planes)
2	1	1	$D_2 < 0$	two parallel planes
2	2	0	$T_2 < 0$	two intersecting planes
2	2	2	$T_2 > 0$	*line* (two intersecting imaginary planes)
			Singular nonplanar	
3	1	1		parabolic cylinder
3	2	0	$T_2 < 0$	hyperbolic cylinder
3	2	2	$T_2 > 0$; $T_1D_3 < 0$	elliptic cylinder
3	2	2	$T_2 > 0$; $T_1D_3 > 0$	*invalid* (imaginary cylinder)
3	3	1	α	cone
3	3	3	β	*point* (imaginary cone)

$\lvert Q_u\rvert$	$\lvert Q\rvert$	s	Conditions	Surface
			Nonsingular	
0	+	0	$(T_2 < 0)$	hyperbolic paraboloid
0	−	2	$(T_2 > 0)$	elliptic paraboloid
±	+	1	α	hyperboloid of one sheet
±	−	1	α	hyperboloid of two sheets
±	+	3	β	*invalid* (imaginary ellipsoid)
±	−	3	β	ellipsoid

$d = 4$; $m = 2$ if $det(Q_u) = 0$, $m = 3$ if $det(Q_u) \neq 0$
0 denotes zero, ± denotes nonzero.
+ denotes positive, − denotes negative;

α: $T_2 > 0$; $det(Q_u) \times T_1 \leq 0$ or $T_2 \leq 0$.
β: $T_2 > 0$; $det(Q_u) \times T_1 > 0$.

$$Q = \begin{pmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{21} & q_{22} & q_{23} & q_{24} \\ q_{31} & q_{32} & q_{33} & q_{34} \\ q_{41} & q_{42} & q_{43} & q_{44} \end{pmatrix} = \begin{pmatrix} A & D & F & G \\ D & B & E & H \\ F & E & C & J \\ G & H & J & K \end{pmatrix}, \quad Q_u = \begin{pmatrix} A & D & F \\ D & B & E \\ F & E & C \end{pmatrix}$$

$d = rank(Q)$; $m = rank(Q_u)$; $s = abs(signature(Q_u))$,

$$T_1 = \sum_{i=1}^{3} q_{ii} = A + B + C,$$

$$T_2 = \sum_{i=1}^{2} \sum_{j=i+1}^{3} \begin{vmatrix} q_{ii} & q_{ij} \\ q_{ji} & q_{jj} \end{vmatrix} = AB + AC + BC - D^2 - E^2 - F^2,$$

$$D_2 = \sum_{i=1}^{3} \sum_{j=i+1}^{4} \begin{vmatrix} q_{ii} & q_{ij} \\ q_{ji} & q_{jj} \end{vmatrix} = T_2 + T_1K - G^2 - H^2 - J^2,$$

$$D_3 = \sum_{i=1}^{2} \sum_{j=i+1}^{3} \sum_{k=j+1}^{4} \begin{vmatrix} q_{ii} & q_{ij} & q_{ik} \\ q_{ji} & q_{jj} & q_{jk} \\ q_{ki} & q_{kj} & q_{kk} \end{vmatrix} = ABC + ACK + ABK + BCK + 2(DEF + FGJ + DGH + EHJ) - D^2(C + K) - E^2(A + K) - F^2(B + K) - G^2(B + C) - H^2(A + C) - J^2(A + B).$$

where $\mathbf{x} = (x\ y\ z\ 1)$, $\mathbf{u} = (u\ v\ w\ 1)$, and

$$F = \begin{pmatrix} f_{ux} & f_{uy} & f_{uz} & 0 \\ f_{vx} & f_{vy} & f_{vz} & 0 \\ f_{wx} & f_{wy} & f_{wz} & 0 \\ \delta_x & \delta_y & \delta_z & 1 \end{pmatrix}.$$

F is a congruence transformation. The subdiscriminant F_u is an orthogonal transformation. If F_u is orthonormal, then the determinants of both F_u and F are unity.

Assume that $\mathfrak{F} = F^{-1}$. $\mathfrak{F}_u$ is the transpose of F_u if they are orthonormal, but the bottom row of $\mathfrak{F}$ bears a more complicated relationship. The lower-right element of $\mathfrak{F}$ (the constant term) is unity.

Suppose the equation for the quadric surface P is

$$\mathbf{u}P\mathbf{u}^T = 0 \quad \text{in untransformed space, and} \qquad (4a)$$
$$\mathbf{x}\mathcal{P}\mathbf{x}^T = 0 \quad \text{in transformed space.} \qquad (4b)$$

Using the transformation $\mathbf{x} = \mathbf{u}F$ on eq. (4b), we have $\mathbf{u}F\mathcal{P}F^T\mathbf{u}^T$. Comparing this with eq. (4a), we have $P = F\mathcal{P}F^T$.

By pre- and post-multiplication, we have:

$$\mathfrak{F}P\mathfrak{F}^T = \mathfrak{F}F\mathcal{P}F^T\mathfrak{F}^T = I\mathcal{P}I = \mathcal{P}, \text{ or}$$
$$\mathcal{P} = \mathfrak{F}P\mathfrak{F}^T.$$

It is possible to break down the transformation into two parts: *rotation* and *translation*. They may be applied repeatedly in any order.

The rotation matrix is

$$R = \begin{pmatrix} f_{ux} & f_{uy} & f_{uz} & 0 \\ f_{vx} & f_{vy} & f_{vz} & 0 \\ f_{wx} & f_{wy} & f_{wz} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

The subdiscriminant is affected only by rotation. Rotation does not affect the constant term, in the lower-right corner of the discriminant.

The translation matrix is

$$T = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \delta_x & \delta_y & \delta_z & 1 \end{pmatrix}.$$

Translation does not affect the subdiscriminant. The constant term is affected only by translation.

A transformation can be formed by a rotation followed by a translation ($F = RT$), or by a translation followed by a rotation ($F = TR$). T and R do not usually commute, just as rotation and translation do not usually commute.

2.3 Canonical Form

It is possible to put the discriminant of a surface into a canonical form, in which the axes of the coordinate system are the axes of the object, with nonzero diagonal elements of the discriminant being given precedence. Basically, this is done by diagonalizing the sub-discriminant, and then eliminating, by congruence transformations, as many of the other elements as possible. For more details, see [4].

Table II. Guide to Classification of Conic Sections.

d	m	s	Conditions	Curve
			Singular	
1	0	0		*invalid*
1	1	1		coincident lines
2	0	0		single line
2	1	1	$D_2 > 0$	*invalid* (imaginary parallel lines)
2	1	1	$D_2 < 0$	two parallel lines
2	2	0	$\lvert Q_u \rvert < 0$	two intersecting lines
2	2	2	$\lvert Q_u \rvert > 0$	*point* (intersecting imaginary lines)
			Nonsingular	
3	1	1		parabola
3	2	0	$\lvert Q_u \rvert < 0$	hyperbola
3	2	2	$\lvert Q_u \rvert > 0, \lvert Q \rvert T_1 < 0$	ellipse
3	2	2	$\lvert Q_u \rvert > 0, \lvert Q \rvert T_1 > 0$	*invalid* (imaginary ellipse)

$$Q = \begin{pmatrix} q_{11} & q_{12} & q_{13} \\ q_{21} & q_{22} & q_{23} \\ q_{31} & q_{32} & q_{33} \end{pmatrix} = \begin{pmatrix} A & D & G \\ D & B & H \\ G & H & K \end{pmatrix}, \quad Q_u = \begin{pmatrix} A & D \\ D & B \end{pmatrix}$$

$d = rank(Q)$, $m = rank(Q_u)$, $s = abs(signature(Q_u))$

$T_1 = q_{11} + q_{22} = A + B$

$$D_2 = \sum_{i=1}^{2} \sum_{j=i+1}^{3} \begin{vmatrix} q_{ii} & q_{ij} \\ q_{ji} & q_{jj} \end{vmatrix} = AB + BK + AK - D^2 - G^2 - H^2.$$

2.4 Discriminant Form for Conic Sections

The conic sections are quadric curves, being also of the quadratic form. The discriminants of a conic are 3×3 matrices, and the subdiscriminants are the 2×2 upper-left submatrices of the discriminants. Other than the reduction in the number of dimensions, the conic sections are completely analogous to quadric surfaces, except that the transformation matrices are slightly simpler:

$$F = \begin{pmatrix} f_{ux} & f_{uy} & 0 \\ f_{vx} & f_{vy} & 0 \\ \delta_x & \delta_y & 1 \end{pmatrix} = \begin{pmatrix} K & S & 0 \\ -S & K & 0 \\ \delta_x & \delta_y & 1 \end{pmatrix}$$

where $K^2 + S^2 = 1$.

2.5 Classification of Quadric Surfaces and Conic Sections

Tables I and II are guides to classification of quadric surfaces and conic sections, respectively. These are adapted, in part, from a chart in [3, p. 230].

One of the columns in these tables is the absolute value of a *signature*. The signature of a matrix is the number of positive eigenvalues minus the number of negative eigenvalues.

3. Classification of Quadric Surface Intersections

When one has two quadric surfaces, the first two questions that arise are the following:

1. Do they intersect?
2. If so, what is the nature of their quadric surface intersection curve (QSIC)?

3.1 Pencil of Two Quadric Surfaces

Suppose we have two quadric surfaces, with discriminants P and Q. In matrix form, the equations for the two surfaces are $\mathbf{x}P\mathbf{x}^T = 0$ and $\mathbf{x}Q\mathbf{x}^T = 0$. The equation, $\mathbf{x}(Q - \alpha P)\mathbf{x}^T = 0$, represents, for all real values of α (finite or infinite), a surface on the "pencil" of P and Q.

For $\alpha = 0$, we have the surface Q. For $\alpha = \pm\infty$, we have the surface P.[1]

If surfaces P and Q intersect, then their intersection (QSIC) is the "base curve" of the pencil, and it lies in all the surfaces of the pencil. This "base curve" is not to be confused with the "base curve" of a parameterization surface, which will be used extensively later on.

If the two surfaces do not intersect, then none of the real surfaces of the pencil intersect. In addition, the pencil contains some imaginary surfaces, and these may be among those listed as *invalid* in Table I.

The general form for a member of the pencil of P and Q is given by $R(\alpha) = Q - \alpha P$.

3.2 Classification of Pencils

One may classify pencils by the classification of the simplest surface in the pencil. If the pencil has a base curve, then the base curve (a QSIC) has the same classification as its pencil.

If, for some α, $R(\alpha)$ has rank *one* or *two*, then $R(\alpha)$ represents either a plane or pair of planes. This pencil is called "planar." Failing this, if, for some α, $R(\alpha)$ has rank *three*, the pencil is "nonplanar singular". If $R(\alpha)$ is never singular, then it always has rank *four*, and the pencil is *nonsingular*.

One can tell if two surfaces do not intersect if one of the following occurs: (1) for some value of α, $R(\alpha)$ is *invalid*, or (2) either P or Q does not intersect some $R(\alpha)$.

3.3 Determining if a Surface Intersection is Planar or Singular

If, for some value of α, $R(\alpha)$ has rank of two or less, then the pencil is planar. As Woon has pointed out [11, pp. 34–36], this happens when two conditions are met for the same value of α: (1) $Det(R(\alpha)) = 0$, and (2) the sum of the 3×3 principal minors of $R(\alpha)$ vanishes. This may be expressed as: $D_3(R(\alpha)) = 0$. If condition (1) is met but condition (2) is not, then the surface is nonplanar singular.

3.4 Nonsingular Pencils

If $R(\alpha)$ is never singular, then there is an $R(\alpha)$ which is a hyperbolic paraboloid. See the Appendix for proof that the intersection between any two quadric surfaces lies in a ruled quadric surface. The crux of this algorithm is that, in calculating points on a QSIC, one uses the parameter to select a line in the ruled surface, and then solves for the intersection of the line and another quadric surface. This is at most a second-order problem; the technique is outlined in Section 4.3.

4. Theory of Parameterization

In this section, methods of parameterization are discussed. The parameterization is done in a *uvw* coordinate system. A congruence transformation is then used to transform point coordinates into the *xyz* system. The parameter is called t.

4.1 Parabola

Suppose we have a parabola of the form: $Au^2 + 2Hv = 0$. Taking $m = -A/2H$, we have: $v = mu^2$. The parameter is t. Therefore we can say

$$u = t, \quad v = mt^2.$$

4.2 Ellipse

Suppose we have an ellipse of form $Au_2 + Bv_2 + K = 0$. Taking $r_u = (-K/A)^{\frac{1}{2}}$ and $r_v = (-K/B)^{\frac{1}{2}}$ as being the semi-axes, we have $u^2/r_u^2 + v^2/r_v^2 - 1 = 0$, where we take

$$u = \frac{2t}{1 + t^2} r_u, \quad v = \frac{1 - t^2}{1 + t^2} r_v.$$

(This is a variation of a "well-known" parameterization mentioned in [1]).

This form is to be used only for $-1 \leq t \leq +1$. This gives only positive values of v. Section 4.4 below gives a more complete method.

This parameterization is well behaved for values of t within the range $[-1, +1]$. That is, if $ds = (du^2 + dv^2)^{\frac{1}{2}}$, ds/dt does not vary too much if r_u and r_v are of the same order of magnitude.

4.3 Hyperbola

Suppose we have a hyperbola of the form $Au^2 + Bv^2 + K = 0$. Taking $r_u = (K/A)^{\frac{1}{2}}$ and $r_v = (-K/B)^{\frac{1}{2}}$, we have

$$u = \frac{2t}{1 - t^2} r_u, \quad v = \frac{1 + t^2}{1 - t^2} r_v,$$

for $-1 < t < +1$. A hyperbola has two disjoint parts. This parameterization gives only one of them, in which $v > 0$.

In contrast to the parameterization of the ellipse, this form is not well behaved. At values of t approaching ±1, small changes in t result in large changes in

[1] Since βP represents the same surface as P for any real nonzero scalar β; if $\beta = 1/\alpha$, $\lim_{\alpha\to\infty}(1/\alpha)(Q - \alpha P) = \lim_{\alpha\to\infty}(Q/\alpha) - P = -P$, which is equivalent to P.

u and v. However, at these places, the hyperbola is very close to its asymptotes, and is practically a straight line.

4.4 More Complete Forms

For central conics (ellipses and hyperbolas), the following modifications may be used:

Take as the parameter t', which takes values from $-2.$ to $+2.$, inclusive, for ellipses. (For hyperbolas, the values -2, 0, and $+2$ are excluded.)

For $t' \geq 0$, we have $t = t' - 1$ and $\sigma = +1$;
For $t' < 0$, we have $t = t' + 1$ and $\sigma = -1$.
For ellipses, we have

$$u = \sigma \frac{2t}{1 + t^2} r_u, \quad v = \sigma \frac{1 - t^2}{1 + t^2} r_v.$$

For hyperbolas, we have

$$u = \sigma \frac{2t}{1 - t^2} r_u, \quad v = \sigma \frac{1 + t^2}{1 - t^2} r_v.$$

Since, for an ellipse, u and v should be periodic functions, we can set up a parameter t'', which is normalized to t' by adding or subtracting multiples of four, such that t' is in the range $[-2., +2.]$.

4.5 Nonplanar Intersections

The parameterization of planar intersections is described above. For nonplanar intersection, the curve lies in a quadric surface which has a "base curve" which is either a line or conic section. For a cylinder, this base curve is a cross-section of the quadric surface, in a plane perpendicular to the main axis. There is at least one set of straight lines, one line passing through each point of the base curve, such that each line lies wholly in the quadric surface, and every point on the quadric surface lies on one of these lines.

One uses parameterization to select a point on the base curve, and its corresponding line. By solving a quadratic (second-order) equation, one can then find the intersection of the line with any other quadric surface. In this manner, all the points of the QSIC(s) may be found.

5. Information Needed by the Algorithm

Special note. The following algorithm has not been implemented, and is presented only as a guide to future implementation.

The algorithm needs the following input information in some form.

(1) *For each object*, it needs to know the dimensions of the "object box," which is a cube or rectangular solid in which the object is contained; and the resolution and vector lengths which will be used.

(2) *Surface equations* must be known for each surface. These may be presented either as ten coefficients (in the form of eq. (1)), or presented as follows: (a) the type of surface (ellipsoid, hyperboloid, cylinder, etc.), (b) the lengths of radii, semi-axes, etc., (c) the orientation and displacement from the origin. An interactive input may be accepted in which the user may specify a surface, and then manipulate and distort it to suit.

(3) *Bounds* must be specified for each quadric patch in the same manner used in the Woon algorithm. A "quadric patch" is defined as the locus of points on the surface satisfying a certain Boolean condition, which is specified for each surface. Each "bound" consists of another quadric surface, and a polarity (+ or −). If the polarity is positive (+), the bound is satisfied (true) only for points in the *exterior* of the bounding surface. If the polarity is negative (−), the bound is satisfied only for points in the *interior* of the bounding surface. Each "set of bounds" consists of one or more bounds. The set of bounds is satisfied iff all bounds in the set are satisfied. Each patch has one or more sets of bounds. The patch is the locus of all points on the surface satisfying at least one set of bounds.

(4) *Surface intersections* must also be specified. For each one, the user must tell the algorithm the two intersecting patches, the multiplicity of the intersection (how many disjoint parts it has), and if it is to be a "smooth" or a "sharp" intersection.

A "sharp" intersection abruptly separates two surfaces, so that there is always an edge which may be seen, if viewed from a proper angle. It is appropriately displayed in the drawings. "Smooth" intersections are used when several quadric patches are used to approximate a single higher-order surface. These are not included in drawings, unless they occur along limbs. Often, the first derivatives will be continuous across a smooth intersection. If one wished to approximate a torus (donut) by using patches of ellipsoids, hyperboloids of one sheet, and cones, smooth intersections would be used.

A patch is usually bounded at surface intersections. One usually specifies intersecting surfaces as bounds for each other. However, if there is a smooth intersection with a continuous first derivative, then it might be difficult to tell if a point is on one side of a boundary or the other. Therefore the program should automatically compute, for every smooth intersection, another surface from the pencil of the two intersecting surfaces. This surface will act as an auxiliary bounding surface. It should meet both intersecting surfaces at a high angle.

Surface intersections are usually also surface bounds, but other surfaces may be needed to completely describe the bounds of a patch. If two surfaces intersect in two distinct circles in parallel planes, an additional auxiliary surface may be needed to distinguish the two. For a more detailed discussion, see [11, pp. 9–13].

6. Processing a Surface Intersection

The following is a brief description of the processing needed for each quadric surface intersection curve (QSIC). Please refer to [4] for more details.

6.1 Classification of QSICs

First, one should compute all the real roots of the equation: $det(R_u(\alpha)) = det(Q_u - \alpha P_u) = 0$. If, for any such α, $R(\alpha)$ is *invalid*, then there is no intersection. If, for any such α, $R(\alpha)$ is singular, then we may have a planar intersection. Otherwise, if one of the $R(\alpha)$ is a hyperbolic paraboloid, we use that $R(\alpha)$. If not, there must be some value of α for which $R(\alpha)$ is singular, and therefore the surface $R(\alpha)$ is a cylinder. In any case, we choose the "simplest" $R(\alpha)$, which is the one closest to the top of the following list:

Planar	*line* (imaginary intersecting planes)
	single plane
	coincident planes
	parallel planes
	intersecting planes
Nonsingular	hyperbolic paraboloid
Nonplanar singular	parabolic cylinder
	elliptic cylinder
	hyperbolic cylinder

The $R(\alpha)$ that is selected is the "parameterization surface". Also, one must select some *other* member of the pencil (usually P or Q) to act as the "other" surface in the parameterization scheme.

6.2 Handling of Planar QSICs

If the parameterization surface is a *line*, then the QSIC consists of the line, or one or two points on the line. Include only those parts within bounds for both intersecting surfaces should be included.

If the parameterization surface is a single plane, the QSIC, if it exists, is a conic section. The parameterization appropriate to that conic section should be used. In the xyz coordinate system, this may be represented as

$$x = (a_x t^2 + b_x t + c_x)/\delta + d_x, \tag{5a}$$
$$y = (a_y t^2 + b_y t + c_y)/\delta + d_y, \tag{5b}$$
$$z = (a_z t^2 + b_z t + c_z)/\delta + d_z. \tag{5c}$$

Using this form means that the transformation from uvw space to xyz space need be done only once, to set up the coefficients of eqs. (5).

If the parameterization surface is a pair of planes, it is factored into two separate planes, each handled as above.

6.3 Handling of Nonplanar QSICs

For a cylinder, the "base curve" of a nonplanar parameterization surface is the cross-section of that surface in a plane perpendicular to that surface's axis. For a hyperbolic paraboloid, this is one of the straight lines which lie on the paraboloid.

For a given parameter, one finds the coordinates of the corresponding point on the base curve. These are called x_0, y_0, and z_0.

One then finds the equation for the line in the parameterization surface including point (x_0, y_0, z_0). This is of the form

$$x = x_0 + \gamma_x s, \tag{6a}$$
$$y = y_0 + \gamma_y s, \tag{6b}$$
$$z = z_0 + \gamma_z s, \tag{6c}$$

where s is the secondary parameter.

One then substitutes the values of x, y, and z found in eqs. (6) into the equation for the other surface. The result is a second-order equation of the form

$$as^2 + bs + c = 0. \tag{7}$$

The "quadratic discriminant" of this equation is

$$\mathfrak{D} = b^2 - 4ac. \tag{8}$$

There are at most two solutions to this equation. One is used when the parameter is increasing, and the other when it is decreasing. This way, each point on the QSIC is represented by a single numerical parameter, plus a Boolean "sign" parameter.

For a cylinder, we get the γ's (direction cosines) as follows: $\gamma_x = e_x$, $\gamma_y = e_y$, and $\gamma_z = e_z$. The coefficients are computed as $(e_x\ e_y\ e_z) = (0\ 0\ 1)\mathfrak{F}_u$.

6.4 Parameterization of QSICs Lying in Hyperbolic Paraboloids

This is a particularly easy form to handle. First, a combination of congruence and scaling transformations is used to put the hyperbolic paraboloid in the form

$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

This is equivalent to the equation: $uv = w$. The "base curve" is the u-axis, and the regulae, or straight lines lying wholly in the paraboloid, are parallel to the v-axis. This is expressed as $u = t$, $v = s$, and $w = st$.

The x, y, and z coordinates are expressed as

$$x = c_x t + d_x + e_x s + f_x st,$$
$$y = c_y t + d_y + e_y s + f_y st,$$
$$z = c_z t + d_z + e_z s + f_z st,$$

with

$$\begin{pmatrix} c_x & c_y & c_z & 0 \\ e_x & e_y & e_z & 0 \\ f_x & f_y & f_z & 0 \\ d_x & d_y & d_z & 1 \end{pmatrix} = \mathfrak{F},$$

where $\mathfrak{F}$ is the transformation matrix from the uvw system to the xyz system.

6.5 Parameter Limits

For a planar QSIC, there is a two-member parameter limit string of the form $\langle t_1, t_2 \rangle$. The "basic limits" are $\langle -2., +2. \rangle$ for an ellipse. For a paraboloid or hyperboloid, they are determined by the limits of the object box. The "bounded limits" are of the same form, but only include those portions of the QSIC within bounds for both intersecting surfaces. This may be done by tracing through the QSIC, and finding the places where the "bounding state" changes. Interval halving may be used to refine these values and find the bounded limits. There may be more than one such pair of limits for a QSIC. The number of such pairs is the QSIC's "multiplicity."

For nonplanar QSICs, the limit string has three elements, in the form $\langle t_1, t_2, t_3 \rangle$. Each adjacent pair, $\langle t_1, t_2 \rangle$ and $\langle t_2, t_3 \rangle$, represents a parameter trace in one direction. If $t_i < t_{i+1}$, then the quadratic equation (7) is solved with a positive radical, as $s = (-b + \mathfrak{D}^{\frac{1}{2}})/2a$. If $t_i > t_{i+1}$, we use a negative radical, $s = (-b - \mathfrak{D}^{\frac{1}{2}})/2a$. If $t_i = t_{i+1}$, no tracing is done. ($\mathfrak{D}$ is the discriminant of eq. (7), as in eq. (8).) Using this form, we may completely trace a nonplanar QSIC, each time obtaining a unique point on the curve.

For "basic limits," we select only those ranges in which the lines corresponding to points on the base curve lie partly in the object box, and the value of $\mathfrak{D}$ in eq. (8) is non-negative.

For "bounded limits," we take those ranges of the parameter satisfying the bounds of both intersecting surfaces. When tracing a QSIC, we use only the values of the parameter within the bounded limits.

7. Limbs

Since quadric surfaces are generally not flat, it is likely that, from a particular viewpoint, a quadric surface may "fold in back of itself." The locus of points where this happens is the limb. This terminology is due to Comba [2]. A *virtual edge* is a segment of a limb.

More precisely, the limb is the locus of points on a surface where the normal to the surface is perpendicular to the line of sight. These points must satisfy both of the following conditions:

$$q(x, y, z) = 0 \quad \text{and} \quad p = \mathbf{s} \cdot \mathbf{grad}\ q = 0,$$

where $\mathbf{s}$ is the line of sight vector from the object point to the viewpoint; and $\mathbf{grad}\ q$ is the surface normal, which may be expressed as the following column vector:

$$\mathbf{grad}\ q = \begin{pmatrix} 2q_1x + q_4y + q_6z + q_7 \\ 2q_2y + q_4x + q_5z + q_8 \\ 2q_3z + q_6x + q_5y + q_9 \end{pmatrix}.$$

For a more detailed discussion of limbs for orthographic and perspective projections, see [4].

8. Final Processing

This algorithm is primarily designed to handle QSICs, with hidden curve elimination a secondary consideration.

Final processing is a combination of hidden curve determination and drawing of the curves. These may be done simultaneously. A "brute force" method is used to determine hidden curves. Each point is tested against every other patch which, as determined by an envelope test, might possibly hide the point. Each point may be displayed as soon as it is determined that it is not hidden.

If a shaded picture is desired, one may set up "display parameter limits," much like the other parameter limits. For each scan line, one determines the intersections of the scan line and the projections of the QSICs. For each point in between these intersection points, one may compute the line of sight, the normal to the quadric patch, and the distance from the patch to the viewpoint. These may then be used to compute shading. It may be possible to express all these as a single function, thus reducing computation time. Pictures of transparent objects may also be generated. For more details on shading and transparency, see [9] and [7], respectively.

9. Importance of Algorithm

The chief benefits of this algorithm are:

(a) It is a complete quadric surface algorithm, allowing use of all real quadric surfaces.
(b) It allows use of "smooth" intersections to approximate higher-order surfaces.
(c) It may be fast enough to use in conjunction with a shading algorithm.

Since this algorithm distinguishes limbs and sharp and smooth intersections, each may be appropriately handled by the shading algorithm. This may include explicit handling of the Mach Band effect, specular reflection, and transparency.

Appendix

THEOREM. *The intersection of two quadric surfaces lies in a plane, pair of planes, hyperbolic or parabolic cylinder, or hyperbolic paraboloid.*

Definitions:

A *para* is a quadric surface with a singular subdiscriminant. An *elliptic para* is an elliptic cylinder or an elliptic paraboloid. A *nonelliptic para* is, by exclusion, a plane, a pair of planes, a hyperbolic or parabolic cylinder, or a hyperbolic paraboloid.

A *rotation* transformation is a transformation on the discriminant matrix representing a rotation. It does not affect the constant (zero-order, or lower-right)

element. A *translation* transformation is a transformation on the discriminant matrix representing a rectilinear translation. It does not affect the elements of the subdiscriminant. A scaling transformation is the distortion of the axes by multiplying each by a linear factor, and is represented by a nonsingular diagonal matrix. It does not affect the *quality* of a surface, in that the surface remains of the same type (an ellipsoid remains an ellipsoid), but it does distort its distances. A *valid* transformation is a combination of one or more of the above. Notice that multiplying each element of the discriminant by the same nonzero constant does not affect the surface at all.

LEMMA 1. *The intersection of two arbitrary quadric surfaces lies in a para.*

Case 1. Either one of the surfaces is a *para*. The hypothesis is satisfied.

Case 2. Let us call the surfaces P and Q. The equation, $det(R_u(\alpha)) = det(Q_u - \alpha P_u) = 0$, may be written as:

$$-det(P_u)\alpha^3 + K_2\alpha^3 - K_1\alpha + det(Q_u) = 0, \qquad (1)$$

with K_2 and K_1 being the sums of the determinants of combinations of columns of P_u and Q_u. Because neither P nor Q is a *para*, $det(P_u) \neq 0$ and $det(Q_u) \neq 0$. Therefore, eq. (1) is definitely of third order, so there must be at least one real root in α. For this α, $R(\alpha)$ is, by definition, a *para*.

LEMMA 2. *Roots of the equation,*

$$det(Q_u - \alpha P_u) = 0, \qquad (2)$$

are unaffected by any valid transformation applied to both P and Q.

PROOF. Suppose we have the valid transformation S and its upper-left 3×3 orthonormal submatrix S_u, both of which are nonsingular. Transforming both P and Q by S gives us

$$\begin{aligned} det(S_uQS_u^{-1} - \alpha S_uPS_u^{-1}) &= det(S_u(Q - \alpha P)S_u^{-1}) \\ &= det(S_u)det(Q - \alpha P) \\ &\quad \cdot det(S_u^{-1}) \\ &= det(Q - \alpha P) = 0, \end{aligned}$$

which obviously has the same roots as eq. (2).

LEMMA 3. *The subdiscriminant of an elliptic para and another arbitary quadric surface may be represented as*

$$P_u = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad Q_u = \begin{pmatrix} A & D & 0 \\ D & B & E \\ 0 & E & C \end{pmatrix}$$

by using valid transformations.

PROOF. Suppose we have an elliptic *para* P and an arbitrary quadric surface Q. Rotate the *para* so its subdiscriminant is in diagonal form:

$$P_u'' = \begin{pmatrix} S & 0 & 0 \\ 0 & T & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad ST > 0; \quad Q_u'' = \begin{pmatrix} A & D & F \\ D & B & E \\ F & E & C \end{pmatrix}.$$

If $S, T < 0$, multiply S and T by -1.

Now, if we transform P_u'' and Q_u'' by the following scaling transformation:

$$\begin{pmatrix} S^{-\frac{1}{2}} & 0 & 0 \\ 0 & T^{-\frac{1}{2}} & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

we get

$$P_u' = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad Q_u' = \begin{pmatrix} A & D & F \\ D & B & E \\ F & E & C \end{pmatrix}.$$

Then apply the rotation transformation

$$\begin{pmatrix} L & -M & 0 \\ M & L & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

with $L^2 + M^2 = 1$ and $L/M = E/F$, to get

$$P_u = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad Q_u = \begin{pmatrix} A & D & 0 \\ D & B & E \\ 0 & E & C \end{pmatrix}.$$

COROLLARY 1. *The intersection of an elliptic para (P) and an arbitrary quadric surface (Q) with $C = 0$, lies in a nonelliptic para.*

PROOF. First, take

$$\begin{aligned} R_u = Q_u - AP_u &= \begin{pmatrix} A & D & 0 \\ D & B & E \\ 0 & E & 0 \end{pmatrix} - \begin{pmatrix} A & 0 & 0 \\ 0 & A & 0 \\ 0 & 0 & 0 \end{pmatrix} \\ &= \begin{pmatrix} 0 & D & 0 \\ D & B-A & E \\ 0 & E & 0 \end{pmatrix}, \end{aligned}$$

which means that $T_2(R) = -D^2 - E^2 \leq 0$.

Case 1. If $D = E = 0$, then $T_2(R) = 0$, and

$$R_u = \begin{pmatrix} 0 & 0 & 0 \\ 0 & B-A & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad m \leq 1.$$

Therefore R is a plane, a pair of coincident or parallel planes, or a parabolic cylinder, all nonelliptic *paras*.

Case 2. If $D \neq 0$ or $E \neq 0$, then $T_2(R) < 0$. R cannot be an elliptic *para*, so it must be a nonelliptic *para*.

COROLLARY 2. *An intersection between an elliptic para and another quadric surface, with $C \neq 0$, can be represented as*

$$P_u = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad Q_u = \begin{pmatrix} A & D & 0 \\ D & B & E \\ 0 & E & 1 \end{pmatrix},$$

if one simply divides every element of Q by C.

Now, with $R(\alpha) = Q - \alpha P$, we can get the equation

$$\begin{aligned} det(R(\alpha)) &= -0\alpha^3 + \alpha^2 - (B - E^2 + A)\alpha \\ &\quad + (AB - AE^2 - D^2) = 0. \end{aligned}$$

Taking $\beta = B - E^2$, we have

$$det(R(\alpha)) = \alpha^2 - (A + \beta)\alpha + (A\beta - D^2) = 0. \quad (3)$$

Note that the quadratic discriminant of this equation is $\mathfrak{D} = (A - \beta)^2 + 4D^2 \geq 0$, so eq. (3) has at least one real root.

LEMMA 4. *The intersection of two elliptic paras lies in a nonelliptic para.*

PROOF. By Lemma 3, we can take

$$P_u = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad Q_u = \begin{pmatrix} A & D & 0 \\ D & B & E \\ 0 & E & C \end{pmatrix}.$$

If $C = 0$, then the hypothesis is satisfied by Corollary 1. Therefore, by Corollary 2, we can take $C = 1$. For Q to be an elliptic *para*, we must have

$$det(Q_u) = AB - AE^2 - D^2 = 0, \text{ and} \quad (4)$$
$$T_2(Q) = AB + A + B - D^2 - E^2 > 0. \quad (5)$$

Taking $\beta = B - E^2$, we have

$$det(Q_u) = A\beta - D^2 = 0, \quad \text{or}$$
$$A\beta = D_2 \geq 0 \quad \text{(condition a)}$$
$$T_2(Q) = AB + A + B - A\beta - E^2$$
$$= A(1 + E^2) + \beta > 0 \quad \text{(condition b)}$$

Now, find α such that $R(\alpha) = Q - \alpha P$ is a *para*. This α can be found by taking eq. (3) with $A\beta - D^2 = 0$. The two roots are $\alpha = 0, A + \beta$. Rejecting the first root, we have $\alpha = A + \beta$, or

$$R(A + \beta) = \begin{pmatrix} -\beta & (A\beta)^{\frac{1}{2}} & 0 \\ (A\beta)^{\frac{1}{2}} & E^2 - A & E \\ 0 & E & 1 \end{pmatrix}$$

For $R(A + \beta)$,

$$T_2 = -[A + \beta(1 + E^2)]. \quad (6)$$

Assume R is an elliptic *para*, then $T_2(R) > 0$, and therefore

$$A + \beta(1 + E^2) < 0. \quad \text{(condition c)}$$

It is obvious that conditions a, b, and c are contradictory. Therefore R cannot be an elliptic *para*, and so it must be a nonelliptic *para*.

THEOREM. *The intersection of two quadric surfaces lies in a nonelliptic para, these being a plane, pair of planes, hyperbolic or parabolic cylinder, or hyperbolic paraboloid.*

PROOF. Assume that there are two quadric surfaces, designated Q_1 and Q_2. If either is a nonelliptic *para*, then the hypothesis is satisfied. If both are elliptic *paras* then the hypothesis is satisfied by Lemma 4.

Otherwise, at least one of the surfaces must be a non-*para*. Call this Q. By Lemma 1, there must be a *para* in the pencil of Q_1 and Q_2. Call this P. If P is a nonelliptic *para*, the hypothesis is satisfied. Otherwise, P is an elliptic *para*.

By Lemma 3, P_u and Q_u may be expressed as

$$P_u = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad Q_u = \begin{pmatrix} A & D & 0 \\ D & B & E \\ 0 & E & C \end{pmatrix}.$$

If $C = 0$, the intersection is contained in a nonelliptic *para* by Corollary 1. By Corollary 2, if $C \neq 0$, then we can set $C = 1$, and

$$det(R_u(\alpha)) = \alpha^2 - (A + \beta)\alpha + (A\beta - D^2) = 0, \text{ or} \quad (7)$$
$$\alpha = \tfrac{1}{2}[A + \beta \pm ((A - \beta)^2 + 4D^2)^{\frac{1}{2}}]. \quad (8)$$

Case 1. $A = \beta = B - E^2$; $D = 0$; thus $\alpha = A$.

$$Q_u = \begin{pmatrix} A & 0 & 0 \\ 0 & A + E^2 & E \\ 0 & E & 1 \end{pmatrix}, \quad R_u(A) = \begin{pmatrix} 0 & 0 & 0 \\ 0 & E^2 & E \\ 0 & E & 1 \end{pmatrix}.$$

$R(A)$ is either a plane, a pair of planes, or a parabolic cylinder, all of these being nonelliptic *para*'s.

Case 2. One of the roots in α is zero. This implies that Q is a *para*. If Q is a nonelliptic *para*, the hypothesis is satisfied. If Q is an elliptic *para*, then it is satisfied by Lemma 4.

Case 3. $A \neq \beta$ or $D \neq 0$. Here, the quadratic discriminant of eq. (7), (which is $\mathfrak{D} = (A - \beta)^2 + 4D^2$) is positive. Therefore these equations have two distinct roots. If either $R(\alpha)$ is nonelliptic, the hypothesis is satisfied. If either is elliptic, the other must be nonelliptic by Lemma 4, thereby satisfying the hypothesis.

Acknowledgments. The author wishes to thank Herbert Freeman for introducing me to this problem and for his guidance, and Peter Woon, for his personal encouragement.

References

1. Braid, I.C. The synthesis of solids bounded by many faces. *Comm. ACM 18*, 4 (April 1975), 209–216.
2. Comba, P.G. A procedure for detecting intersections of three-dimensional objects. *J. ACM 15*, 3 (July 1968), 354–366.
3. Dresden, A. *Solid Analytical Geometry and Determinants.* Dover, New York, 1964.
4. Levin, J.Z. A parametric algorithm for drawing pictures of solid objects bounded by quadric surfaces. Tech. Rep. CRL-46, School of Eng., Rensselaer Polytechnic Inst., Troy, N.Y., 1976.
5. Loutrel, P. A solution to the hidden-line problem for computer-drawn polyhedra. *IEEE Trans. Computers C-19*, 3 (March 1970), 205–213.
6. Mahl, R. Visible surface algorithm for quadric patches. *IEEE Trans. Computers C-21*, (Jan. 1972), 1–4.
7. Metelli, F. The perception of transparency. *Scientific American 230*, 4 (April 1974), 90–98.
8. Newman, W.M., and Sproull, R.F. *Principles of Interactive Computer Graphics.* McGraw-Hill, New York, 1973, Appendix II, pp. 467–480.
9. Bui-Tuong Phong. Illumination for computer generated pictures. *Comm. ACM 18*, 6 (June 1975), 311–317.
10. Weiss, R.A. BE VISION, a package of IBM 7090 FORTRAN programs to draw orthographic views of combinations of plane and quadric surfaces. *J. ACM 13*, 2 (April 1966), 194–204.
11. Woon, P.Y. A computer procedure for generating visible-line drawings of solids bounded by quadric surfaces. Tech. Rep. 403–15, Dep. Electr. Eng., School of Eng. and Sci., New York U., New York, Nov. 1970.
12. Woon, P.Y., and Freeman, H. A procedure for generating visible-line projections of solids bounded by quadric surfaces. Information Processing 71, Vol. 2, North-Holland Pub. Co., Amsterdam, 1971, pp. 1120–1125.

Graphics and
Image Processing

Interactive Skeleton Techniques for Enhancing Motion Dynamics in Key Frame Animation

N. Burtnyk and M. Wein
National Research Council of Canada

A significant increase in the capability for controlling motion dynamics in key frame animation is achieved through skeleton control. This technique allows an animator to develop a complex motion sequence by animating a stick figure representation of an image. This control sequence is then used to drive an image sequence through the same movement. The simplicity of the stick figure image encourages a high level of interaction during the design stage. Its compatibility with the basic key frame animation technique permits skeleton control to be applied selectively to only those components of a composite image sequence that require enhancement.

Key Words and Phrases: interactive graphics, computer generated animation, key frame animation, interactive skeleton, skeleton control, stick figure animation

CR Categories: 3.41, 3.49, 4.9, 8.2

Copyright © 1976, Association for Computing Machinery, Inc. General permission to republish, but not for profit, all or part of this material is granted provided that ACM's copyright notice is given and that reference is made to the publication, to its date of issue, and to the fact that reprinting privileges were granted by permission of the Association for Computing Machinery.

A version of this paper was presented at SIGGRAPH '76: The Third Annual Conference on Computer Graphics, Interactive Techniques, and Image Processing, The Wharton School, University of Pennsylvania, July 14–16, 1976.

Author's address: National Research Council of Canada, Division of Electrical Engineering, Ottawa, Canada K1A OR8.

[1] Cel has been derived from celluloids, the material on which drawings are prepared in conventional cel animation. Component images that move separately are usually drawn on separate cels and stacked into a cel sandwich for filming.

Introduction

Previous work has demonstrated that key frame animation techniques constitute a successful approach to animation of free-form images [1–3]. Using this technique, the artist draws key images at selected intervals in an animation sequence and the playback program computes the in-between images by interpolation. Interpolation between related key images allows the animation of change of shape or distortion. It permits a direct and intuitive method for specifying the action, whereas mathematically defined distortion requires trial and error experimentation. One strength of key frame animation techniques is the analogy to conventional hand animation techniques, simplifying the transition when a classically trained animator adapts to using computers.

Figure 1 illustrates a typical image sequence generated using key frame animation. The first and last images were drawn by the artist, while the six intermediate images were selected from the 240 frames in the actual film sequence.

The animation package is implemented on a minicomputer based interactive graphics system. This package supports the four major phases of a production: (1) the drawing phase, (2) the assembly of drawings into key frame sequences, (3) the preview/modification phase, and (4) the final processing and recording of the sequences on film.

The drawing phase is carried out in two stages. The first stage is off-line at the drawing board. Analysis of the action depicted by the story board establishes key positions from which drawings are prepared. The second stage involves tracing these drawings on a graphic tablet at the display console. During this stage, the order in which strokes are traced to describe an image is important. Since the interpolation process is based on stroke to stroke mapping, this ordering of strokes between related images controls the form of the intermediate image.

The second phase consists of the interactive assembly of individual drawings or cels[1] into key frames, including a specification of the interpolation law for each cel and a key to key time interval. Concatenated key frames form a sequence. This process is repeated for all concurrent sequences that make up a composite sequence.

During the preview phase, playback of any individual sequence or concurrent sequences on the interactive display permits an assessment of the resulting animation. Modification involves returning to the interactive assembly phase to edit the sequences. In practice, direct playback assures only that the form of the interpolated images can be assessed, since it is difficult to achieve playback at the cine rate with complex images. Proper assessment of motion and timing requires further conversion to a raster format which maintains display at the cine rate independent

of the image content.

This technique of animating free-form images has been directed mainly towards drawn images and hence two-dimensional. The image material, of course, attempts to represent a 3-D space much as in conventional cel animation. The basic capability includes a simplified solution to the problem of hidden surfaces by treating the image as a hierarchy of parallel planes. The simplification lies in the fact that the animator-specified order of planes establishes the order of visibility computation, thus eliminating any programmed sorting of data by depth. The composite playback facility produces separately a composite line image with hidden lines removed and a composite surface sequence.

Consideration of Motion Dynamics

The greatest shortcoming in key frame animation results from incomplete control of motion dynamics, both in complexity and in smoothness or continuity. It is relatively simple to have good control over the dynamics in time. The amount of change from one frame to the next is determined by a weighting factor which is a single-valued function of time. Thus one can easily compute, or store precomputed, various functions representing different "tapers." However, the same value of weighting function is applied to an entire picture component. There is no "spatial weighting."

The shortcomings manifest themselves in the following ways: (a) the motion of each point in the image is along a straight line and the relative change from one frame to the next is the same for all points belonging to one picture element, and (b) there is a discontinuity at key frames in both the amount of frame to frame change and in the direction of apparent motion. Therefore it is difficult to synthesize smooth continuous motion spanning several key positions. There is a dilemma in that smoothness is achieved by having as few key images as possible (and therefore widely spaced in time), while close control requires many closely spaced keys. In addition, a large number of closely spaced drawings negates much of the economic advantage of using computers.

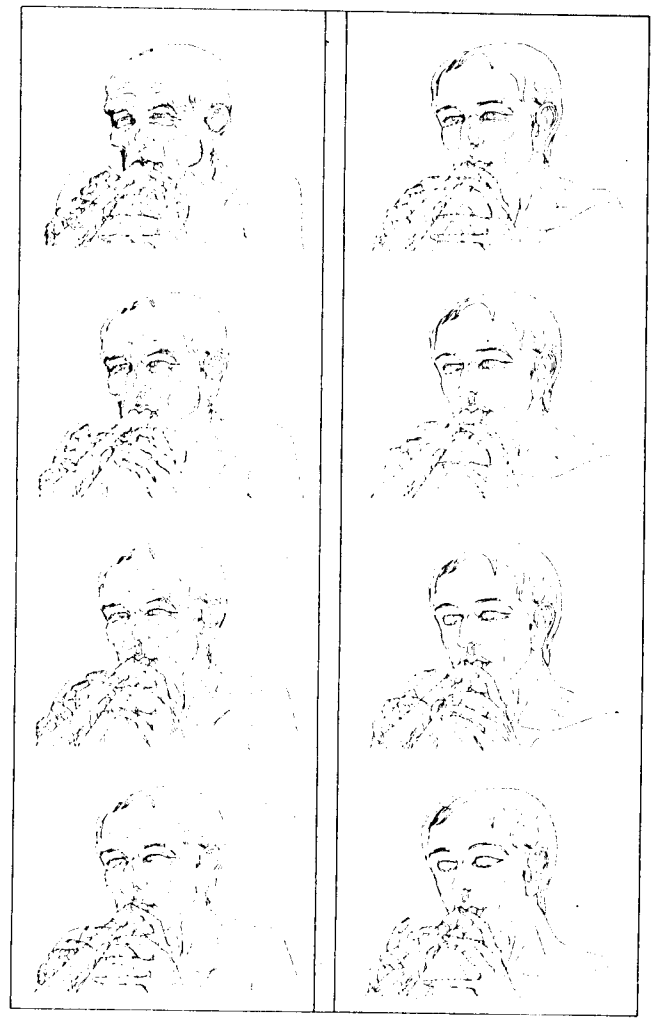
Various techniques have been examined for overcoming these problems. One technique provides an ability to include a rotational component as part of the image change, which in effect superimposes rotation on the interpolation process [2]. This permits some variation in the spatial dynamics but its application is limited and discontinuities at key frames remain.

Synthesis of complex motion could be achieved by using additional intermediate keys, but preparation of additional drawings by the animator is uneconomical. Skeleton techniques were developed to derive variations of existing key drawings to be used as intermediate keys [2]. This involves representation of a drawing by a simple skeleton and then extracting a distorted form of the image by modifying only the skeleton. Even when such additional keys are used, discontinuity in motion is difficult to avoid.

Another technique involves the use of smooth drawn paths to control the interpolation process. Motion along a path, as a method distinct from key frame animation, has been used extensively in computer animation [4–7]. A single path to control interpolation between key images offers a limited solution somewhat equivalent to the use of rotation—it tends to be satisfactory only when the distortion of the image is minimal. For a distorting image, different portions must follow entirely different paths. This immediately leads to a problem if several paths are to be drawn for different portions of the image. It is difficult to establish points of simultaneity on several paths such that one could easily perceive the shape of the image at any instant.

An examination of the methods used in conventional animation has led to a solution to this problem.

Fig. 1. Selected frames from a key frame animation sequence. The first and last images are drawn, the intermediate images are interpolated. Multilayer visibility is included in computing the composite image. From "Visage," a film by Peter Foldes



To visualize a complex movement, the animator often sketches stick figure representations at equal-time intervals between key positions. He may use smooth curves through related skeletal points as a further guide. This set of stick figures achieves both objectives: the frame to frame spacing conveys the rate of movement and the shape of each skeleton represents the shape of the object at that instant. Thus the problem reduces to animating a stick figure representation of the image which will in turn impart the movement to the actual image sequence.

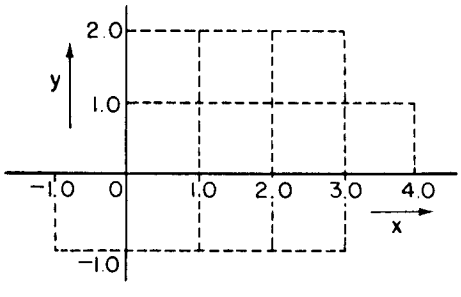
The system described in this paper incorporates the use of skeletons into the key frame technique to provide overall control in the playback process. As in the basic key frame animation system, the process of producing a sequence involves two steps. The first is the interactive stage at which the animator prepares the key images and establishes the stick figure representations at as many intermediate positions as desired. The intermediate skeletons define intermediate control keys. During playback the program selects those image components that are skeleton driven and applies the necessary deformation.

There are two significant aspects of the skeleton driven technique. First, the skeletons are simple images composed of only a few points, so that it is possible to provide a high level of interaction. The second aspect of this technique is its compatibility with basic key frame animation. Skeleton sequences are prepared only where necessary and the playback system identifies those image components that are skeleton driven and those that are not.

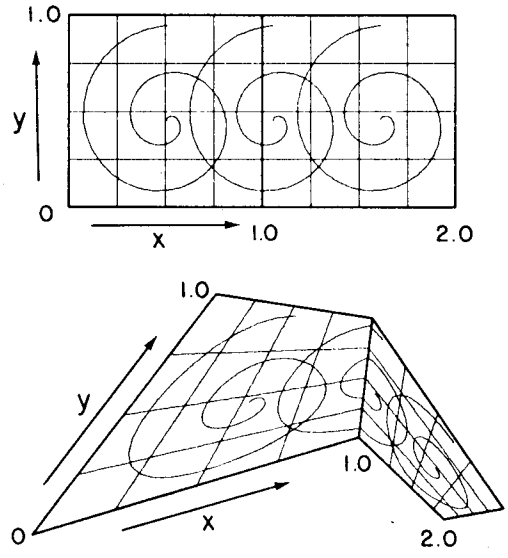
It should be noted that the concept of skeletons used in the context of this paper differs from that used by Blum [8]. Blum's skeletons are used for image representation in a compressed form and are derived automatically from the coordinate data. Our skeleton representation of an image provides a definition of some coordinate space within which the image, described in relative coordinates, is distributed.

Skeleton Coordinate System

The nature of the coordinate space that is used to define relative skeleton coordinates may be thought of as a network of polygons that form a mesh (Figure 2).

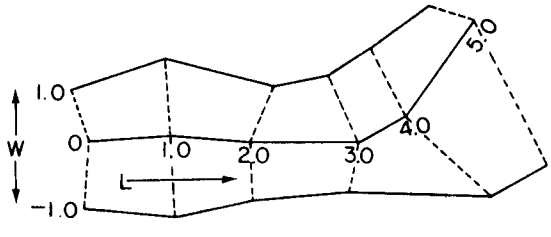


Each polygon has a relative coordinate range of 0 to 1.0 along each axis. Now the nodes in this mesh may be displaced relative to one another to change its geometry. However, because the relative coordinate system within each polygon is based on its geometry, coordinate values remain continuous across common edges between adjacent polygons. Thus any image whose coordinates are defined within this system will take on the overall distortion exhibited by the coordinate space (Figure 3).



Geometric transformation of contours from one coordinate space to another is, of course, well known in conformal mapping.

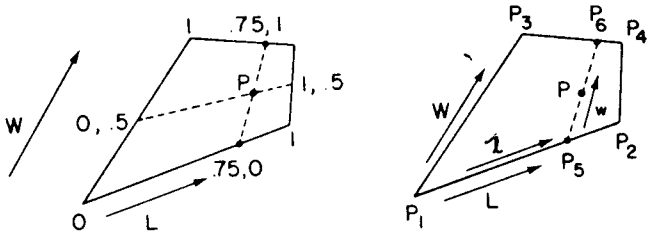
The notion of skeleton control implies a central core of connected "bones" with a surrounding image distribution. In order to restrict the transverse distance away from the core over which skeleton control will be active, delimiting boundaries must be specified. Consequently, the practical form of skeleton coordinate space spans two units in width, but may extend in length as desired (Figure 4).



For convenience, the central core always represents the L axis, which is also the $W = 0$ coordinate reference; the delimiting boundary which is specified first is the positive or $W = 1.0$ boundary, the other is the $W = -1.0$ boundary. The L coordinate range starts at $L = 0$

and is incremented by one for each node on the central core. If desired, the L coordinate space may be separated at any coordinate boundary by providing a redefinition of that coordinate boundary before continuing the coordinate space. Of course, the related image will not normally continue through such a separation. In addition, ambiguities can occur if separated coordinate spaces overlap. In general, it is preferable to treat these as separate skeletons so that no restrictions are imposed. On the other hand, any given image need not fall entirely within the coordinate space of a skeleton. Those points which lie outside the skeleton space will remain unaffected by the distortion of the skeleton coordinate space.

Relative coordinates, denoted by $(\boldsymbol{l}, \boldsymbol{w})$, may be defined as the fractional distance along each axis which is occupied by a line passing through the point while intersecting the two opposing edges of the polygon at this fractional distance. In Figure 5(a), the coordinates of point P are (0.75, 0.5) by this definition. In order to minimize the computation involved in coordinate conversion, the simpler definition of Figure 5(b) is used.



(a) (b)

Given the absolute vertex coordinates of the polygon and an image point P, the fractional distance of points P_5 and P_6 along lines P_1P_2 and P_3P_4 is expressed by $\boldsymbol{l}$, giving

$$x_5 = \boldsymbol{l}(x_2 - x_1) + x_1, \quad y_5 = \boldsymbol{l}(y_2 - y_1) + y_1,$$

$$x_6 = \boldsymbol{l}(x_4 - x_3) + x_3, \quad y_6 = \boldsymbol{l}(y_4 - y_3) + y_3,$$

These expressions are substituted into $(x - x_6)/(y - y_6) = (x_5 - x)/(y_5 - y)$, the equation of the line P_5P_6 passing through point P, giving

$$[x - x_3 - \boldsymbol{l}(x_4 - x_3)]/[y - y_3 - \boldsymbol{l}(y_4 - y_3)] = [x - x_1 - \boldsymbol{l}(x_2 - x_1)]/[y - y_1 - \boldsymbol{l}(y_2 - y_1)].$$

This reduces to

$$(BH - DF)\boldsymbol{l}^2 + (CF - DE - AH - BG)\boldsymbol{l} + (AG - CE) = 0$$

where

$$A = x - x_1, \quad E = x - x_3,$$
$$B = x_2 - x_1, \quad F = x_4 - x_3,$$
$$C = y - y_1, \quad G = y - y_3,$$
$$D = y_2 - y_1, \quad H = y_4 - y_3,$$

The desired root for $\boldsymbol{l}$ has a value between 0 and 1, the other root will be negative or greater than 1. The $\boldsymbol{w}$-coordinate, expressing the fractional distance of point P along line P_5P_6, is given by

$$\boldsymbol{w} = (x - x_5)/(x_6 - x_5) = (A - B\boldsymbol{l})/[A - E - (B - F)\boldsymbol{l}].$$

To convert relative coordinates back to display coordinates, the $\boldsymbol{l}$-coordinate is applied to the vertex coordinates to determine the coordinates of P_5 and P_6 from which P is found.

The effect of skeleton control is to take any specified area of the display plane and distort it into another area of the display plane as if it were made up of rubber sheet patches. In that sense, it is similar to the distorted raster scan technique used in Caesar [9], and equivalent to the mapping of images into curved surfaces described by Catmull [10]. While the skeleton coordinate space is distorting, however, the relative coordinates of the image itself may be undergoing a change. Relative coordinates may be treated in the same way as absolute coordinates, as if the reference coordinate space was always uniform and orthogonal, its particular shape being important only for display purposes (Figure 6). Therefore the key frame interpolation process may still be carried out even if key images are represented in relative coordinates. It is this compatibility with key frame animation that makes the skeleton control technique so powerful and attractive. No other practical techniques have been developed that offer a comparable degree of image control in computer generated animation.

Implementation within Key Frame Animation

The benefits that may be derived in practice from skeleton control are closely related to the method of implementation. While there is little doubt that any capability for enriching motion is useful, it is equally clear that the compulsory use of skeleton control for all parts of an animation sequence would be a great hindrance. The full advantage in the use of this technique is realized only if the animator can apply it selectively. In fact, it is most attractive if it can be used to improve motion dynamics of sequences which were previously created. This is the form in which it has been implemented in our system.

Component sequences are first assembled in the usual form and displayed as a composite image sequence for previewing. If, after assessment, the animator wants to modify or improve parts of it, he does so by adding skeleton control to those components only. This is accomplished by attaching a reference skeleton, which he has drawn, to each image which will be controlled. These image components, which are referenced to skeletons, are converted to relative coordinates and tagged during assembly of the sequence.

Now the composite sequence is regenerated in this modified form. During playback, all coordinate data pass through the interpolation process, but those identified as being relative must be mapped to a particular skeleton reference for display. A skeleton reference defines a display space in absolute coordinates. These skeleton coordinate references for each frame are provided by assembling stick figure control sequences which are also played back as part of the composite sequence.

The design of the skeleton control sequence itself is developed interactively in a separate package. Stick figure representations of key images provide the starting point (Figure 7(a)). Two such skeletons are used to define a start and end frame. When the INBETWEEN display mode is active, intermediate frames are interpolated and presented on the screen as a superposition of many frames (Figure 7(b)). For convenience, the delimiting boundaries about the skeleton core are eliminated from display to prevent excessive clutter. Any frame may now be selected and modified using tablet interaction. In this mode, the modified coordinates of the selected frame are stored as a control frame within the sequence. The interpolated intermediate frames adjust accordingly in response to tablet interaction (Figure 7(c)). Additional frames may be modified in a similar manner (Figure 7(d)).

Frame to frame change is easily related to the spacing between stick figures. This interaction continues, giving the animator control over motion dynamics down to the frame level as in conventional animation, if desired. The user-modified control frames are preserved, whereas all other intermediate stick figures are recomputed when needed. Display of the final sequence of control frames is shown in Figure 7(e). The skeleton boundaries have been adjusted where required through similar tablet interaction to maintain their desired form. Alternatively skeleton keys which have been drawn in these desired positions may be brought in from the picture library and assembled as control frames in the same way.

In practice, the number of control frames that are used to generate a motion sequence will be kept to a minimum. Because of this, simple linear interpolation between specified control frames may not adequately reproduce a smooth continuous movement. This result is illustrated in Figure 7(f), where the dynamics of the movement suffer from excessive discontinuities in rate at two of the control frames. Additional intermediate frames have been interpolated and plotted for clarity.

This deficiency is removed if a smoothing function is applied during computation of intermediate frames (Figure 7(g)). This process maintains continuity of movement of corresponding points through successive control frames. The parametric method of curve fitting is adapted from the work of Akima [11]. Not only does it produce a smooth path for each point, but progression

Fig. 6. (a), (c) two drawn images in absolute coordinates; (b), (d) same images with reference skeletons; (e), (f) the relative coordinates presented on an orthogonal coordinate system.

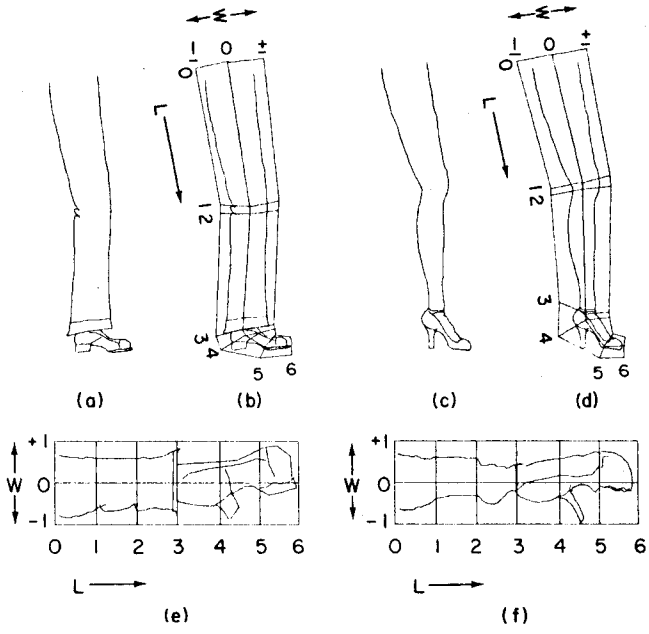
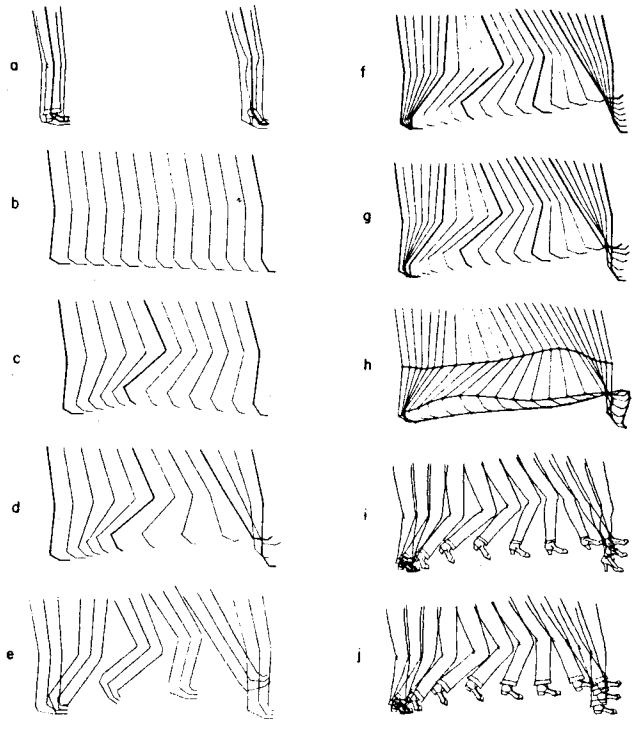


Fig. 7. Development of a skeleton control sequence: (a) start and end frames; (b) interpolated inbetweens presented for interaction; (c), (d) frames 5 and 9 modified in turn by the animator (control frames); (e) final sequence of control frames, shown with boundaries; (f) linearly interpolated inbetweens; (g), (h) with curve smoothing; (i), (j) images driven by the motion sequence.



along each path is tapered to accommodate changes in rate in a complex movement. Successive positions of several points have been joined in Figure 7(h) to emphasize the effect. This process has all the characteristics of drawing smooth paths to control interpolation between images without any of its limitations.

Now the skeleton control information is complete for driving the original image sequence through the desired movement (Figure 7(i)). It may equally well be applied to drive any other compatible image sequence through the same motion. In Figure 7(j), although the image sequence itself specifies a HOLD during that interval, the skeleton sequence drives it through the same motion cycle. The relationship between a start/end frame skeleton pair and the number of key images in that interval is arbitrary. It should also be clear that the interpolation rate between key images is independent of the progression of its skeleton through a movement. Because of this, the smoothing process may span as many key images as required to complete a continuous movement.

Various program features assist the animator in developing a skeleton control sequence. Direct viewing of skeleton movement is obtained by requesting the ANIMATE display mode at any time. This causes the sequence of intermediate frames from the start to the end frame to be continuously cycled at the cine rate instead of being presented as a static ensemble. Companion skeletons that have been developed for several cels may be previewed together in this mode as well as during interaction. If the INBETWEEN display presentation contains confusing frame to frame overlap (e.g. walking on the spot), a positional offset may be introduced into successive control frames to remove the ambiguity. Similarly any number of intermediate frames may be skipped to simplify the overall display during interactive modification.

Proposed Extensions to the System

Because final processing of skeleton control sequences is performed in the composite playback program, the same camera control commands that apply pan, tilt, and zoom to an image sequence can be applied to the control sequence. With an extension of the system, more complex processing functions could be applied to the control skeleton. This approach may be useful for superimposing complex forms of movement control on the skeleton. It may also significantly reduce the processing time for rotation since only the skeleton coordinate references need to be rotated.

Another useful extension deals with the capability for creating a library of common movements. If sequences of control frames are saved in a normalized form, they can be retrieved and superimposed on any particular skeleton as a starting point for developing variations of that movement. Since the details of the image itself are not contained in the skeleton, this necessitates only that the form of this skeleton match the standard normalized form (i.e. it consists of the same connection of bones). All the physical characteristics of the particular skeleton being used (such as relative length and width of each section) will be retained while only the stored motion characteristics will be transferred. Although this capability has not been implemented in the present system, it does indeed offer an important potential reduction in animation production costs.

References

1. Burtnyk, N., and Wein, M. Computer generated key frame animation. *J. Soc. Motion Picture and Television Engineers 80*, 3 (1971), 149–153.
2. Burtnyk, N., and Wein, M. Towards a computer animating production tool. Proc. Eurocomp Conf., Brunel U., 1974, Online Pub. Co., 172–185.
3. Burtnyk, N., and Wein, M. Computer animation of free form images. Computer Graphics *9*, 1 (1975), 78–80 (Issue of Proc. Second Ann. Conf. Computer Graphics and Interactive Techniques).
4. Baecker, R.M. Picture-driven animation. Proc. AFIPS 1969 SJCC, Vol. 34, AFIPS Press, Montvale, N.J., pp. 273–288.
5. Csuri, C. Real-time animation. Proc. Ninth Ann. Meeting UAIDE (Users of Automatic Inform. Display Equipment), Miami, 1970, pp. 289–305.
6. Burtnyk, N., et al. Computer graphics and film animation. *Canadian J. Operational Res. and Inform. Processing (INFOR) 9*, 1 (1971), 1–11.
7. Burtnyk, N., and Wein, M. A computer animation system for the animator. Proc. Tenth Ann. Meeting UAIDE (Users of Automatic Inform. Display Equipment), Los Angeles, 1971, pp. 3.5–3.24.
8. Blum, H. A transformation for extracting new descriptors of shape. In *Models of Speech and Visual Form*, MIT Press, Cambridge, Mass., 1967, pp. 362–380.
9. Honey, F.J. Computer animated episodes by single axis rotations—CAESAR. Proc. Tenth Ann. Meeting UAIDE (Users of Automatic Inform. Display Equipment), Los Angeles, 1971, pp. 3–210 to 3–226.
10. Catmull, E. Computer display of curved surfaces. Proc. Conf. Computer Graphics, Pattern Recognition and Data Structure, May 1975, pp. 11–17 (IEEE Cat. No. 75CH0981-IC).
11. Akima, H. A new method of interpolation and smooth curve fitting based on local procedures. *J. ACM 17*, 4 (1970), 589–602.

Artificial Evolution for Computer Graphics

Karl Sims

Thinking Machines Corporation
245 First Street, Cambridge, MA 02142

1 ABSTRACT

This paper describes how evolutionary techniques of variation and selection can be used to create complex simulated structures, textures, and motions for use in computer graphics and animation. Interactive selection, based on visual perception of procedurally generated results, allows the user to direct simulated evolutions in preferred directions. Several examples using these methods have been implemented and are described. 3D plant structures are grown using fixed sets of genetic parameters. Images, solid textures, and animations are created using mutating symbolic lisp expressions. Genotypes consisting of symbolic expressions are presented as an attempt to surpass the limitations of fixed-length genotypes with predefined expression rules. It is proposed that artificial evolution has potential as a powerful tool for achieving flexible complexity with a minimum of user input and knowledge of details.

2 INTRODUCTION

Procedural models are increasingly employed in computer graphics to create scenes and animations having high degrees of complexity. A price paid for this complexity is that the user often loses the ability to maintain sufficient control over the results. Procedural models can also have limitations because the details of the procedure must be conceived, understood, and designed by a human. The techniques presented here contribute towards solutions to these problems by enabling "evolution" of procedural models using interactive "perceptual selection." Although they do not give complete control over every detail of the results, they do permit the creation of a large variety of complex entities which are still user directed, and the user is not required to understand the underlying creation process involved.

Many years ago Charles Darwin proposed the theory that all species came about via the process of evolution [2]. Evolution is now considered not only powerful enough to bring about biological entities as complex as humans and consciousness, but also useful in simulation to create algorithms and structures of higher levels of complexity than could easily be built by design. Genetic algorithms have shown to be a useful method of searching large spaces using simulated systems of variation and selection [5, 6, 7, 23]. In *The Blind Watchmaker*, Dawkins has demonstrated the power of Darwinism with a simulated evolution of 2D branching structures made from a set of genetic parameters. The user selects the "biomorphs" that survive and reproduce to create each new generation [3, 4]. Latham and Todd have applied these concepts to help generate computer sculptures made with constructive solid geometry techniques [9, 28].

Variations on these techniques are used here with the emphasis on the potential of creating forms, textures, and motions that are useful in the production of computer graphics and animation, and also on the potential of using representations that are not bounded by a fixed space of possible results.

2.1 Evolution

Both biological and simulated evolutions involve the basic concepts of genotype and phenotype, and the processes of expression, selection, and reproduction with variation.

The *genotype* is the genetic information that codes for the creation of an individual. In biological systems, genotypes are normally composed of DNA. In simulated evolutions there are many possible representations of genotypes, such as strings of binary digits, sets of procedural parameters, or symbolic expressions. The *phenotype* is the individual itself, or the form that results from the developmental rules and the genotype. *Expression* is the process by which the phenotype is generated from the genotype. For example, expression can be a biological developmental process that reads and executes the information from DNA strands, or a set of procedural rules that utilize a set of genetic parameters to create a simulated structure. Usually, there is a significant amplification of information between the genotype and phenotype.

Selection is the process by which the fitness of phenotypes is determined. The likelihood of survival and the number of new offspring an individual generates is proportional to its fitness measure. *Fitness* is simply the ability of an organism to survive and reproduce. In simulation, it can be calculated by an explicitly defined fitness evaluation function, or it can be provided by a human observer as it is in this work.

Reproduction is the process by which new genotypes are generated from an existing genotype or genotypes. For evolution to progress there must be *variation* or mutations in new genotypes with some frequency. Mutations are usually probabilistic as opposed to deterministic. Note that selection is, in general, non-random and is performed on phenotypes; variation is usually random and is performed on the corresponding genotypes [See figure 1].

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

©1991
ACM-0-89791-436-8/91/007/0319 $00.75

phenotype can be saved for further manipulation. Solid polygonal branches can be generated with connected cylinders and cone shapes, and leaves can be generated by connecting sets of peripheral nodes with polygonal surfaces. Shading parameters, color, and bump textures can be assigned to make bark and leaf surfaces. These additional properties could also be selected and adjusted using artificial evolution, but due to the longer computation times involved to test samples, these parameters were adjusted by hand. In some cases, leaf shapes were evolved independently and then explicitly added to the tip segments of other evolved plant structures. A forest of plant structures created using these methods is shown in figure 3.

3.2 Mutating Parameter Sets

For artificial evolution of parameter sets to occur, they must be reproduced with some probability of mutation. There are many possible methods for mutating parameter sets. The technique used here involves normalizing each parameter for a genetic value between .0 and 1.0, and then copying each genetic value or gene, g_i, from the parent to the child with a certain probability of mutation, m. A mutation is achieved by adding a random amount, $\pm d$, to the gene. So, a new genotype, G', is created using each gene, g_i, of a parent genotype, G, as follows:

For each g_i
  If $rand(.0, 1.0) < m$
    then $g_i' = g_i + rand(-d, d)$
      clamp or wrap g_i' to legal bounds.
    else $g_i' = g_i$

The normalized values are scaled, offset, and optionally squared to give the parameter values actually used. This allows the mutation distances, $\pm d$, to be proportional to the scale of the range of valid parameter values. Squaring or raising some values to even higher powers can be useful because it causes more sensitivity in the lower region of the range of parameter values. The mutation rate and amount are easily adjusted, but are commonly useful at much higher values than in natural systems ($m = 0.2, d = 0.4$). The random value between $-d$ and d might preferably be found using a Gaussian distribution instead of this simple linear distribution, giving smaller mutations more likelihood than larger ones.

3.3 Mating Parameter Sets

When two parameter sets are found that both create structures with different successful features, it is sometimes desirable to combine these features into a single structure. This can be accomplished by mating them. Reproducing two parameter sets with sexual combination can be performed in many ways. Four possible methods are listed below with some of their resulting effects:

1. *Crossovers* can be performed by sequentially copying genes from one parent, but with some frequency the source genotype is switched to the other parent. This causes adjacent genes to be more likely to stick together than genes at opposite ends of the sequence. Each pair of genes has a *linkage* probability depending on their distance from each other.

2. Each gene can be independently copied from one parent or the other with equal probability. If the parent genes each correspond to a point in N-dimensional genetic space, then the genes of the possible children using this method correspond to the 2^N corners of the N-dimensional rectangular solid connecting the two parent points. This method is the most commonly used in this work and is demonstrated in figure 2. Two parent plant structures are shown in the upper left boxes, and the remaining forms are their children.

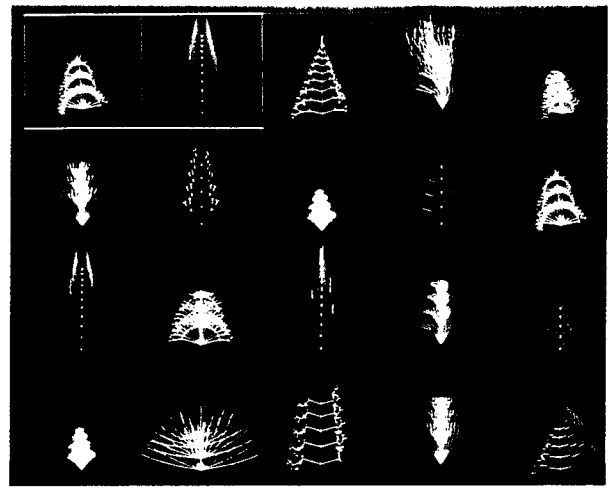


Figure 2: Mating plant structures.

Figure 3: Forest of "evolved" plants.

3. Each gene can receive a random percentage, p, of one parent's genes, and a $1 - p$ percentage of the other parent's genes. If the percentage is the same for each gene, linear *interpolation* between the parent genotypes results, and the children will fall randomly on the line between the N-dimensional points of the parents. If evenly spaced samples along this line were generated, a *genetic dissolve* could be made that would cause a smooth transition between the parent phenotypes if the changing parameters had continuous effects on the phenotypes. This is an example of utilizing the underlying genetic representation for specific manipulation of the results. Interpolation could also be performed with three parents to create children that fall on a triangular region of a plane in the N-dimensional genetic space.

4. Finally, each new gene can receive a random value between the two parent values of that gene. This is like the interpolation scheme above, except each gene is independently interpolated be-

tween the parent genes. This method results in possible children anywhere within the N-dimensional rectangular solid connecting the parent points.

Mutating and mating parameter sets allow a user to explore and combine samples in a given parameter space. In the next section, methods are presented that allow mutations to add new parameters and extend the space, instead of simply adjusting existing parameter values.

4 SYMBOLIC EXPRESSIONS AS GENOTYPES

A limitation of genotypes consisting of a fixed number of parameters and fixed expression rules as described above is that there are solid boundaries on the set of possible phenotypes. There is no possibility for the evolution of a new developmental rule or a new parameter. There is no way for the genetic space to be extended beyond its original definition – the N-dimensional genetic space will remain only N-dimensional.

To surpass this limitation, it is desirable to include procedural information in the genotype instead of just parameter data, and the procedural and data elements of the genotype should not be restricted to a specific structure or size.

Symbolic lisp expressions are used as genotypes in an attempt to meet these needs. A set of lisp functions and a set of argument generators are used to create arbitrary expressions which can be mutated, evolved, and evaluated to generate phenotypes. Some mutations can create larger expressions with new parameters and extend the space of possible phenotypes, while others just adjust existing parts of the expression. Details of this process are best described by the examples below.

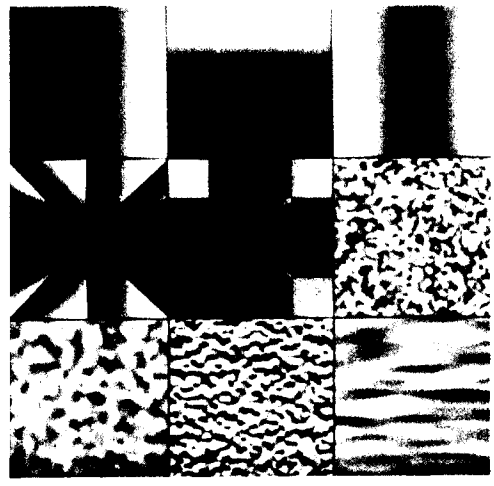


Figure 4: Simple expression examples.

(reading left to right, top to bottom)
a. X
b. Y
c. *(abs X)*
d. *(mod X (abs Y))*
e. *(and X Y)*
f. *(bw-noise .2 2)*
g. *(color-noise .1 2)*
h. *(grad-direction (bw-noise .15 2) .0 .0)*
i. *(warped-color-noise (* X .2) Y .1 2)*

4.1 Evolving Images

The second example of artificial evolution involves the generation of textures by mutating symbolic expressions. Equations that calculate a color for each pixel coordinate (x, y) are evolved using a *function set* containing some standard common lisp functions [26], vector transformations, procedural noise generators, and image processing operations:

> +, −, *, /, *mod, round, min, max, abs, expt, log, and, or, xor, sin, cos, atan, if, dissolve, hsv-to-rgb, vector, transform-vector, bw-noise, color-noise, warped-bw-noise, warped-color-noise, blur, band-pass, grad-mag, grad-dir, bump, ifs, warped-ifs, warp-abs, warp-rel, warp-by-grad.*

Each function takes a specified number of arguments and calculates and returns an image of scalar (b/w) or vector (color) values.

Noise generators can create solid 2D scalar and vector noise at various frequencies with random seeds passed as arguments so specific patterns can be preserved between generations [figure 4f, and 4i]. The warped versions of functions take (U, V) coordinates as arguments instead of using global (X, Y) pixel coordinates, allowing the result to be distorted by an arbitrary inverse mapping function [figure 4i]. Boolean operations (*and*, *or*, and *xor*) operate on each bit of floating-point numbers and can cause fractal-like grid patterns [figure 4e]. Versions of *sin* and *cos* which normalize their results between .0 and 1.0 instead of -1.0 and 1.0 can be useful. Some functions such as blurs, convolutions, and those that use gradients also use neighboring pixel values to calculate their result [figure 4h]. *Band-pass* convolutions can be performed using a difference of Gaussians filter which can enhance edges. Iterative function systems (*ifs*) can generate fractal patterns and shapes.

Details of the specific implementations of these functions are not given here because they are not as important as the methods used for combining them into longer expressions. Many other functions would be interesting to include in this *function set*, but these have provided for a fairly wide variety of resulting images.

Simple random expressions are generated by choosing a function at random from the *function set* above, and then generating as many random arguments as that function requires. Arguments to these functions can be either scalars or vectors, and either constant values or images of values. Random arguments can be generated from the following forms:

- A random scalar value such as *.4*
- A random 3-element vector such as *#(.42 .23 .69)*
- A variable such as the X or Y pixel coordinates.
- Another lisp expression which returns a b/w or color image.

Most of the functions have been adapted to either coerce the arguments into the required types, or perform differently according to the argument types given to them. Arguments to certain functions can optionally be restricted to some subset of the available types. For the most part these functions receive and return images, and can be considered as image processing operations. Expressions are

simply evaluated to produce images. Figure 4 shows examples of some simple expressions and their resulting images.

Artificial evolution of these expressions is performed by first generating and displaying a population of simple random expressions in a grid for interactive selection. The expressions of images selected by the user are reproduced with mutations for each new generation such that more and more complex expressions and more perceptually successful images can evolve. Some images evolved with this process are shown in figures 9 to 13.

4.2 Mutating Symbolic Expressions

Symbolic expressions must be reproduced with mutations for evolution of them to occur. There are several properties of symbolic expression mutation that are desirable. Expressions should often be only slightly modified, but sometimes significantly adjusted in structure and size. Large random changes in genotype usually result in large jumps in phenotype which are less likely to be improvements, but are necessary for extending the expression to more complex forms.

A recursive mutation scheme is used to mutate expressions. Lisp expressions are traversed as tree structures and each node is in turn subject to possible mutations. Each type of mutation occurs at different frequencies depending on the type of node:

1. Any node can mutate into a new random expression. This allows for large changes, and usually results in a fairly significant alteration of the phenotype.
2. If the node is a scalar value, it can be adjusted by the addition of some random amount.
3. If the node is a vector, it can be adjusted by adding random amounts to each element.
4. If the node is a function, it can mutate into a different function. For example *(abs X)* might become *(cos X)*. If this mutation occurs, the arguments of the function are also adjusted if necessary to the correct number and types.
5. An expression can become the argument to a new random function. Other arguments are generated at random if necessary. For example X might become *(* X .3)*.
6. An argument to a function can jump out and become the new value for that node. For example *(* X .3)* might become X. This is the inverse of the previous type of mutation.
7. Finally, a node can become a copy of another node from the parent expression. For example *(+ (abs X) (* Y .6))* might become *(+ (abs (* Y .6)) (* Y .6))*. This causes effects similar to those caused by mating an expression with itself. It allows for sub-expressions to duplicate themselves within the overall expression.

Other types of mutations could certainly be implemented, but these are sufficient for a reasonable balance of slight modifications and potential for changes in complexity.

It is preferable to adjust the mutation frequencies such that a decrease in complexity is slightly more probable than an increase. This prevents the expressions from drifting towards large and slow forms without necessarily improving the results. They should still easily evolve towards larger sizes, but a larger size should be due to selection of improvements instead of random mutations with no effect.

The relative frequencies for each type of mutation above can be adjusted and experimented with. The overall mutation frequency is scaled inversely in proportion to the length of the parent expression. This decreases the probability of mutation at each node when the parent expression is large so that some stability of the phenotypes is maintained.

The evaluation of expressions and display of the resulting images can require significant calculation times as expressions increase in complexity. To keep image evolution at interactive speeds, estimates of compute speeds are calculated for each expression by summing pre-computed runtime averages for each function. Slow expressions are eliminated before ever being displayed to the user. New offspring with random mutations are generated and tested until fast enough expressions result. In this way automatic selection is combined with interactive selection. If necessary, this technique could also be performed to keep memory usage to a minimum.

4.3 Mating Symbolic Expressions

Symbolic expressions can be reproduced with sexual combinations to allow sub-expressions from separately evolved individuals to be mixed into a single individual. Two methods for mating symbolic expressions are described.

The first method requires the two parents to be somewhat similar in structure. The nodes in the expression trees of both parents are simultaneously traversed and copied to make the new expression. When a difference is encountered between the parents, one of the two versions is copied with equal probability. For example, the following two parents can be mated to generate four different expressions, two of which are equal to the parents, and two of which have some portions from each parent:

parent1: *(* (abs X) (mod X Y))*
parent2: *(* (/ Y X) (* X -.7))*

child1: *(* (abs X) (mod X Y))*
child2: *(* (abs X) (* X -.7))*
child3: *(* (/ Y X) (mod X Y))*
child4: *(* (/ Y X) (* X -.7))*

This method is often useful for combining similar expressions that each have some desired property. It usually generates offspring without very large variations from the parents. Two expressions with different root nodes will not form any new combinations. This might be compared to the inability of two different species to mate and create viable offspring.

The second method for mating expressions combines the parents in a less constrained way. A node in the expression tree of one parent is chosen at random and replaced by a node chosen at random from the other parent. This *crossing over* technique allows any part of the structure of one parent to be inserted into any part of the other parent and permits parts of even dissimilar expressions to be combined. With this method, the parent expressions above can generate 61 different child expressions – many more than the 4 of the first method.

4.4 Evolving Volume Textures

A third variable, Z, is added to the list of available arguments to enable functions to be evolved that calculate colors for each point in (X, Y, Z) space. The *function set* shown in section 4.1 is adjusted for better results: 2D functions that require neighboring pixel values such as convolutions and warps are removed, and 3D solid noise generating functions are added.

These expressions are more difficult to visualize because they encompass all of 3D space. They are evaluated on the surfaces

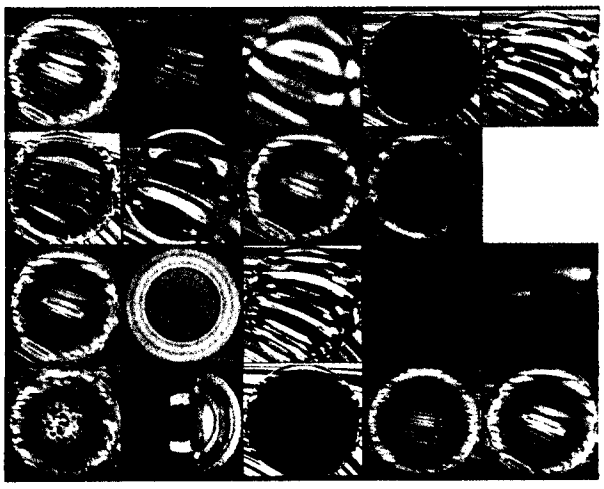


Figure 5: Parent with 19 random mutations.

Figure 6: Marble and wooden tori.

of spheres and planes for fast previewing and selection as shown in figure 5. Evolved volume expressions can then be incorporated into procedural shading functions to texture arbitrary objects. This process allows complex volume textures such as those described in [18] and [19] to be evolved without requiring specific equations to be understood and carefully adjusted by hand. Figure 6 was generated by evolving three volume texture expressions and then evaluating them at the surfaces positions of three objects during the rendering process.

4.5 Evolving Animations

Several extensions to the image evolution system described above can be used to evolve moving images. Five methods for incorporating a temporal dimension in symbolic expressions are proposed:

1. Another input variable, $Time$, can be added to the list of available arguments. Expressions can be evolved that are functions of X, Y, and $Time$ such that different images are produced as the value of $Time$ is smoothly animated. More computation is required to generate, display and select samples because a sequence of images must be calculated. An alternate method of display involves displaying various slices of the $(X, Y, Time)$ space (although operations requiring neighboring pixel values might not receive the correct information if the values of $Time$ vary between them).

2. *Genetic cross dissolves* can be performed between two expressions of similar structure. Interpolation between two expressions is performed by matching the expressions where they are identical and interpolating between the results where they are different. Results of differing expression branches are first calculated and dissolved, and then used by the remaining parts of the expression. If the two expressions have different root nodes, a conventional image dissolve will result. If only parts within their structures are different, interesting motions can occur. This technique utilizes the existing genetic representation of evolved still images to generate in-betweens for a smooth transition from one to another. It is an example of the usefulness of the alternate level of control given by the underlying genetic information. A series of frames from a genetic cross dissolve are shown in figure 7.

3. An input image can be added to the list of available arguments to make functions of X, Y, and $Image$. The input image can then be animated and processed by evaluating the expression multiple times for values of $Image$ corresponding to frames of another source of animation such as hand drawn or traditional 3D computer graphics. This is effectively a technique for evolving complex image processing and warping functions that compute new images from given input images. Figure 8 was created in this way with an input image of a human face.

4. The images that use the pixel coordinates (X, Y) to determine the colors at each pixel can be animated by altering the mappings of X and Y before the expression is evaluated. Simple zooming and panning can be performed as well as 3D perspective transformations and arbitrary patterns of distortion.

5. Evolved expressions can be adjusted and experimented with by hand. If parameters in expressions are smoothly interpolated to new values, the corresponding image will change in potentially interesting ways. For example, solid noise can be made to change frequency, colors can be dissolved into new shades, and angles can be rotated. This is another example of utilizing the underlying genetic information to manipulate images. A small change in the expression can result in a powerful alteration of the resulting image.

Finally, the techniques above can be used together in various combinations to make an even wider range of possibilities for evolving animations.

Figure 7: Frames from a "genetic cross dissolve."

Figure 8: Fire of Faces.

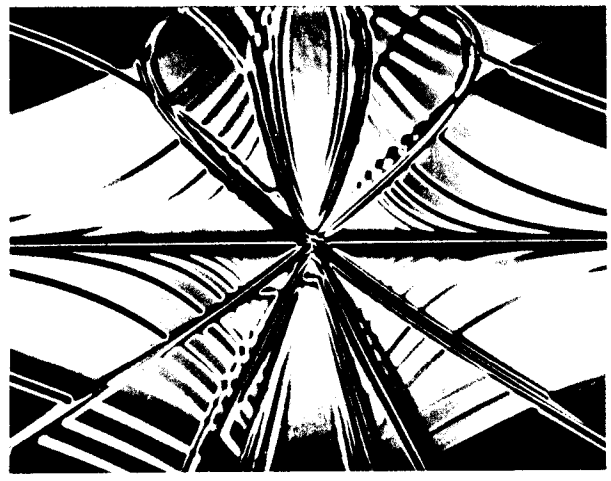


Figure 9.

5 RESULTS

Evolution of 3D plant structures, images, solid textures, and animations have been implemented on the Connection Machine (tm) system CM-2, a data parallel supercomputer [10, 27]. The parallel implementation details will not be discussed in this paper, but each application is reasonably suited for highly parallel representation and computation. Lisp expression mutations and combinations are performed on a *front-end* computer and the Connection Machine system is used to evaluate the expression for all pixels in parallel using *Starlisp* and display the resulting image.

3D Plant structures have been evolved and used in the animated short *Panspermia* [24]. A frame from this sequence is shown in figure 3 which contains a variety of species created using these techniques. An interactive system for quickly growing, displaying, and selecting sample structures allows a wide range of plant shapes to be efficiently created by artificial evolution. Populations of samples can be displayed for selection in wire frame in a grid format as shown in figure 2, or displayed as separate higher-resolution images which can be interactively flipped through by scrolling with a mouse. Typically between 5 and 20 generations are necessary for acceptable structures to emerge.

Images, volume textures, and various animations have been created using mutating symbolic expressions. These sometimes require more generations to evolve complex expressions that give interesting images - often at least 10 to 40 generations. Again, an interactive tool for quickly displaying grids of sample images to be selected amongst makes the evolution process reasonably efficient. [See figure 5.] The number of possible symbolic expressions of acceptable length is extremely large, and a wide variety of textures and patterns can occur. Completely unexpected kinds of images have emerged. Figure 9 was created from the following evolved expression:

(round (log (+ y (color-grad (round (+ (abs (round (log (+ y (color-grad (round (+ y (log (invert y) 15.5)) x) 3.1 1.86 #(0.95 0.7 0.59) 1.35)) 0.19) x)) (log (invert y) 15.5)) x) 3.1 1.9 #(0.95 0.7 0.35) 1.35)) 0.19) x)

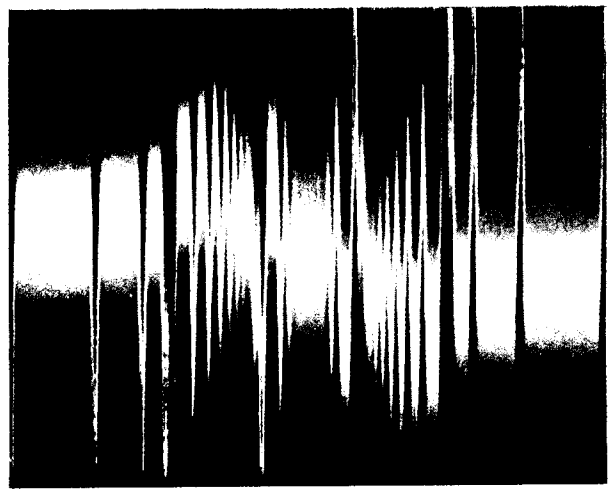


Figure 10.

Figure 11.

Figure 13 was created from this expression:

(sin (+ (- (grad-direction (blur (if (hsv-to-rgb (warped-color-noise #(0.57 0.73 0.92) (/ 1.85 (warped-color-noise x y 0.02 3.08)) 0.11 2.4)) #(0.54 0.73 0.59) #(1.06 0.82 0.06)) 3.1) 1.46 5.9) (hsv-to-rgb (warped-color-noise y (/ 4.5 (warped-color-noise y (/ x y) 2.4 2.4)) 0.02 2.4))) x))

Note that expressions only five or six lines long can generate images of fair complexity. Equations such as these can be evolved from scratch in timescales of only several minutes - probably much faster than they could be designed.

Figures 10, 11, and 12 were also created from expressions of similar lengths. Fortunately, analysis of expressions is not required when using these methods to create them. Users usually stop attempting to understand why each expression generates each image. However, for those interested, expressions for other figures are listed in the appendix.

Two different approaches of user selection behavior are possible. The user can have a goal in mind and select samples that are closer to that goal until it is hopefully reached. Alternatively, the user can follow the more interesting samples as they occur without attempting to reach any specific goal.

The results of these various types of evolved expressions can be saved in the very concise form of the final genotypic expression itself. This facilitates keeping large libraries of evolved forms which can then be used to contribute to further evolutions by mating them with other forms or further evolving them in new directions.

6 FUTURE WORK

Artificial evolution has many other possible applications for computer graphics and animation. Procedures that use various other forms of solid noise could be explored, such as those that create objects, create density functions, or warp objects [20, 15]. Procedures could be evolved that generate motion from a set of rules (possibly cellular automata, or particle systems), or that control distributions and characteristics of 2D objects such as lines, solid shapes, or brush strokes. Algorithms that use procedural construction rules to create 3D objects from polygons, or functions that generate, manipulate, and combine geometric primitives could also be explored.

These techniques might also make valuable tools in domains beyond computer simulations. New possibilities for shapes and textures could be explored for use in product design or the fashion industry.

Several variations on the methods for artificial evolution described above might make interesting experiments. Mutation frequencies could be included in the genotype itself so that they also can be mutated. This might allow for the evolution of evolvability [4]. Frequencies from the most successful evolutions could be kept as the defaults.

It might be interesting to attempt to automatically evolve a symbolic expression that could generate a simple specific goal image. An image differencing function could be used to calculate a *fitness* based on how close a test image was to the goal, and an expression could be searched for by automatic selection. Then, interactive selection could be used to evolve further images starting with that expression.

Large amounts of information of all the human selection choices of many evolutions could be saved and analyzed. A difficult challenge would be to create a system that could generalize and "understand" what makes an image visually successful, and even generate other images that meet these learned criteria.

Combinations of random variations and non-random variations using learned information might be helpful. If a user picks phenotypes in a certain direction from the parent, mutations for the next generation might have a tendency to continue in that same direction, causing evolution to have "momentum."

Also, combinations of evolution and the ability to apply specific adjustments to the genotype might allow more user control over evolved results. Automatic "genetic engineering" could permit a user to request an evolved image to be more blue, or a texture more grainy.

7 CONCLUSION

Artificial evolution has been demonstrated to be a potentially powerful tool for the creation of procedurally generated structures, textures, and motions. Reproduction with random variations and survival of the visually interesting can lead to useful results. Representations for genotypes which are not limited to fixed spaces and can grow in complexity have shown to be worthwhile.

Evolution is a method for creating and exploring complexity that does not require human understanding of the specific process involved. This process of artificial evolution could be considered as a system for helping the user with creative explorations, or it might be considered as a system which attempts to "learn" about human aesthetics from the user. In either case, it allows the user and computer to interactively work together in a new way to produce results that neither could easily produce alone.

An important limiting factor in the usefulness of artificial evolution is that samples need to be generated quickly enough such that it is advantageous for the user to choose from random samples instead of carefully adjusting new samples by hand. The computer needs to generate and display samples fast enough to keep the user interested while selecting amongst them. As computation becomes more powerful and available, artificial evolution will hopefully become advantageous in more and more domains.

8 Acknowledgments

Thanks to Lew Tucker, Jim Salem, Gary Oberbrunner, Matt Fitzgibbon, Dave Sheppard, and Zotje Maes for help and support. Thanks to Peter Schröder for being a helpful and successful user of these tools. Thanks to Luis Ortiz and Katy Smith for help with document preparation. And thanks to Danny Hillis, Larry Yaeger, and Richard Dawkins for discussions and inspiration.

9 APPENDIX

Figure 5, Parent expression:

(warped-color-noise (warped-bw-noise (dissolve x 2.53 y) z 0.09 12.0) (invert z) 0.05 -2.06)

Figure 6, Marble torus:

(dissolve (cos (and 0.25 #(0.43 0.73 0.74))) (log (+ (warped-bw-noise (min z 11.1) (log (rotate-vector (+ (warped-bw-noise (cos x) (dissolve (cos (and 0.25 #(0.43 0.73 0.74))) (log (+ (warped-bw-noise (max (min z 8.26) (/ -0.5 #(0.82 0.39 0.19))) (log (+ (warped-bw-noise (cos x) z -0.04 0.89) #(0.82 0.39 0.19)) #(0.15 0.34 0.50)) -0.04

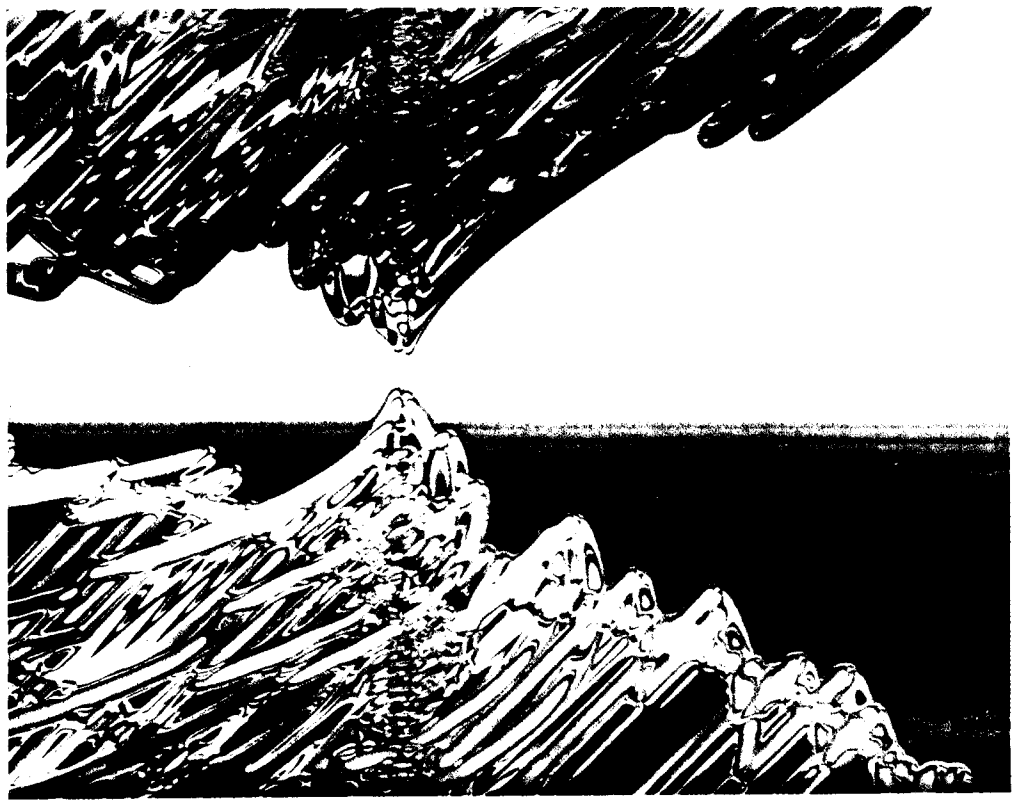


Figure 12.

Figure 13.

-3.0) y) #(0.15 0.34 0.50)) y) -0.04 -3.0) x) z y) #(0.15 0.34 0.5)) -0.02 -1.79) -0.4) #(-0.09 0.34 0.55)) -0.7)

Figure 7, Cross dissolve:

(hsv-to-rgb (bump (hsv-to-rgb (ifs 2.29 0.003 (dissolve 1.77 3.67 time) 2.6 0.1 (dissolve 5.2 3.2 time) -31.0 (dissolve 23.9 -7.4 time) (dissolve 1.13 9.5 time) (dissolve 4.8 0.16 time) 20.7 4.05 (dissolve 0.48 0.46 time) (dissolve 2.94 -0.68 time) (dissolve 0.42 0.54 time) (dissolve 0.09 0.54 time))) (atan 2.25 (dissolve 0.1 0.11 time) 0.15) (dissolve 4.09 8.23 time) (dissolve #(0.41 0.36 0.08) #(0.68 0.22 0.31) time) #(0.36 0.31 0.91) (dissolve 6.2 4.3 time) (dissolve 0.16 0.40 time) (dissolve 2.08 0.23 time)))

Figure 8, Fire of Faces:

(+ (min 10.8 (warp-rel image image (bump image x 9.6 #(0.57 0.02 0.15) #(0.52 0.03 0.38) 3.21 2.49 10.8))) (dissolve #(0.81 0.4 0.16) x (dissolve y #(0.88 0.99 0.66) image)))

Figure 10:

(rotate-vector (log (+ y (color-grad (round (+ (abs (round (log #(0.01 0.67 0.86) 0.19) x)) (hsv-to-rgb (bump (if x 10.7 y) #(0.94 0.01 0.4) 0.78 #(0.18 0.28 0.58) #(0.4 0.92 0.58) 10.6 0.23 0.91))) x) 3.1 1.93 #(0.95 0.7 0.35) 3.03)) -0.03) x #(0.76 0.08 0.24))

Figure 11 is unfortunately "extinct" because it was created before the genome saving utility was complete.

Figure 12:

(cos (round (atan (log (invert y) (+ (bump (+ (round x y) y) #(0.46 0.82 0.65) 0.02 #(0.1 0.06 0.1) #(0.99 0.06 0.41) 1.47 8.7 3.7) (color-grad (round (+ y y) (log (invert x) (+ (invert y) (round (+ y x) (bump (warped-ifs (round y y) y 0.08 0.06 7.4 1.65 6.1 0.54 3.1 0.26 0.73 15.8 5.7 8.9 0.49 7.2 15.6 0.98) #(0.46 0.82 0.65) 0.02 #(0.1 0.06 0.1) #(0.99 0.06 0.41) 0.83 8.7 2.6))))) 3.1 6.8 #(0.95 0.7 0.59) 0.57))) #(0.17 0.08 0.75) 0.37) (vector y 0.09 (cos (round y y)))))

References

[1] Aono, M., and Kunii, T. L., "Botanical Tree Image Generation," *IEEE Computer Graphics and Applications*, Vol.4, No.5, May 1982.

[2] Darwin, Charles, *The Origin of Species*, New American Library, Mentor paperback, 1859.

[3] Dawkins, Richard, *The Blind Watchmaker*, Harlow Logman, 1986.

[4] Dawkins, Richard, "The Evolution of Evolvability," *Artificial Life Proceedings*, 1987, pp.201-220.

[5] Goldberg, D. E., *Genetic Algorithms in Search, Optimization, and Machine Learning*, 1989, Addison-Wesley Publishing Co.

[6] Grenfenstette, J. J., *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*, Hillsdale, New Jersey, Lawrence Erlbaum Associates, 1985.

[7] Grenfenstette, J. J., *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, 1987, (Hillsdale, New Jersey: Lawrence Erlbaum Associates.)

[8] Haase, K., "Automated Discovery," *Machine Learning: Principles and Techniques*, by Richard Forsyth, Chapman & Hall 1989, pp.127-155.

[9] Haggerty, M., "Evolution by Esthetics, an Interview with W. Latham and S. Todd," *IEEE Computer Graphics*, Vol.11, No.2, March 1991, pp.5-9.

[10] Hillis, W. D., "The Connection Machine," *Scientific American*, Vol. 255, No. 6, June 1987.

[11] Holland, J. H., *Adaptation in Natural and Artificial Systems*, Ann Arbor, MI: University of Michigan Press, 1975.

[12] Koza, J. R. "Genetic Programming: A Paradigm for Genetically Breeding Populations of Computer Programs to Solve Problems," Stanford University Computer Science Department Technical Report STAN-CS-90-1314, June 1990.

[13] Koza, J. R. "Evolution and Co-Evolution of Computer Programs to Control Independently Acting Agents," *Conference on Simulation of Adaptive Behavior* (SAB-90) Paris, Sept.24-28, 1990.

[14] Lenat, D. B. and Brown,J.S. "Why AM and EURISKO appear to work," *Artificial intelligence*, Vol.23, 1984, pp.269-294.

[15] Lewis, J. P., "Algorithms for Solid Noise Synthesis," *Computer Graphics*, Vol.23, No.3, July 1989, pp.263-270.

[16] Oppenheimer, P. "Real time design and animation of fractal plants and trees." *Computer Graphics*, Vol.20, No.4, 1986, pp.55-64.

[17] Oppenheimer, P. "The Artificial Menagerie" *Artificial Life Proceedings*, 1987, pp.251-274.

[18] Peachy, D., "Solid Texturing of Complex Surfaces," *Computer Graphics* Vol.19, No.3, July 1985, pp.279-286.

[19] Perlin, K., "An Image Synthesizer," *Computer Graphics*, Vol.19, No.3, July 1985, pp.287-296.

[20] Perlin, K., "Hypertexture," *Computer Graphics*, Vol.23, No.3, July 1989, pp.253-262.

[21] Prusinkiewicz, P., Lindenmayer, A., and Hanan, J., "Developmental Models of Herbaceous Plants for Computer Imagery Purposes," *Computer Graphics*, Vol.22 No.4, 1988, pp.141-150.

[22] Reffye, P., Edelin, C., Francon, J., Jaeger, M., Puech, C. "Plant Models Faithful to Botanical Structure and Development," *Computer Graphics* Vol.22, No.4, 1988, pp.151-158.

[23] Schaffer, J. D., "Proceedings of the Third international Conference on Genetic Algorithms," June 1989, Morgan Kaufmann Publishers, Inc.

[24] Sims, K., *Panspermia*, Siggraph Video Review 1990.

[25] Smith, A. R., "Plants, Fractals, and Formal Languages," *Computer Graphics*, Vol.18, No.3, July 1984, pp.1-10.

[26] Steele, G., *Common Lisp, The Language*, Digital Press, 1984.

[27] Thinking Machines Corporation, *Connection Machine Model CM-2 Technical Summary*, technical report, May 1989.

[28] Todd, S. J. P., and Latham, W., "Mutator, a Subjective Human Interface for Evolution of Computer Sculptures," IBM United Kingdom Scientific Centre Report 248, 1991.

[29] Viennot, X., Eyrolles, G., Janey, N., and Arques, D., "Combinatorial Analysis of Ramified Patterns and Computer Imagery of Trees," *Computer Graphics*, Vol.23, No.3, July 1989, pp.31-40.