
THE
FRACTAL
ARCHIVE

Rochester Institute of Technology

A Thesis submitted to the Faculty of
The College of Fine and Applied Arts
in Candidacy for the Degree of
Master of Fine Arts

Introduction to Fractal by Using Interactive Media Design

by

Wu, Diing-Wuu Vale

May 20, 1991

Approvals

Advisor: **Bob Keough**

Date: 6-10-91

Associate Advisor: **Jim Ver Hague**

Date: 6.7.91

Associate Advisor: **Mark Collien**

Date: 6/10/91

Special Assistant to the Dean
for Graduate Affairs: **Phil Bornarth**

Date: 6/13/91

Acting Dean, College of Fine and
Applied Arts: **Dr. Peter Giopulos**

Date: 6/18/91

I, Wu, Diing-Wuu , hereby grant permission to the Wallace Memorial Library of RIT, to reproduce my thesis in whole or part. Any reproduction will not be for commercial use or profit.

Date: May 30, 1991

Dedication

This book is dedicated to my mother Wong, Fon-En, father Wu, Ching-Yu, elder brother Wu, Diing-Wen, younger brother Wu, Diing-Don for their continuing support in spirit and material. In fact, I couldn't have done it without you.

Acknowledgments

Special thanks to my thesis committee:

Bob Keough--for his great help to let me keep in track

Jim Ver Hague--who gave me the great help in scripting debug, and some suggestions to my user interface design

Mark Collien--for his great knowledge of HyperCard 2.0 and HyperTalk language

And, I would like to thank the following individuals for their helps and comments:

R. Roger Remington--for his great concept of graphic design giving me some suggestions to my poster design

Heinz Klinkon--who gave me some great ideas and comments in my poster design

David Dickinson--for his great support in Fractal software and some books

Leonard Urso--for his different ideas and comments of graphic design

Keith--for his big help and patience

Preface

Like so many interactive media, the use of computers for instructional purposes would be used frequently to each field. And, this aspect of computer application, it is known as Computer-Assisted Instruction (CAI). It could be incorporated into a training lesson by using music, sound effects, graphics, and computer animation to give the users a vivid way of expression. Basically, CAI includes so many kind of forms and styles. Here, my thesis project is more educational-oriented or tutorial-oriented. For my thesis, I created a computer-Assisted Instruction for a field of Science and Art--Fractal, with the software of HyperCard 2.0, which is a powerful environment of interactive media interface.

Though the use of an interactive environment, which is more visually-oriented, the user has easy access to my fractal stack, such as history, scientist, theory, fractal type, fractal micro, slide show, cartoon and glossary,etc.

For a non-mathematics person, a professional individual or even a graphic artist, my fractal stack is a convenient resource for reference. Moreover, it can be an electronic database file kept in a library or a college or any academic institute to let different fields of users access to it.

Table of Contents

Introduction

In 1983, I was an undergraduate student studying in Physics. It was my first experience to read some articles about fractals, and the Mandelbrot set was the first fractal image I ever seen. It also gave me strong feelings of beauty which combine the fields of Science and Art. However, it sometimes confused me about how the Science field could influence Art. Now, I think I could figure out what is the reason behind it. What I thought is the reason existed in the highly developed area of modern scientific technology. The invention of computer technology made rapid progress in a large number of computations. And, by means of the computer it became an effective way to produce any kind of fractal, such as Julia sets, Mandelbrot sets, Phonex curves, Dragon curves, von Koch curves and so on.

Actually, fractals are all around us; for example, the shape of a mountain, the windings of a coast line, the shape of flash lighting, the contour of flickering fires, and the formations of clouds. It seems that fractals exist everywhere in nature. Yes, many fractals are actually familiar to us, but only recently did research go deeply enough to make important progress in the relevant field of fractals.

It is true that everything exists in nature almost with an irregular shape in its appearance. However, it is hard to describe those irregular geometric figures

by using the geometry principles of Galileo. What we learn from a single point, a straight line, a round circle, or even a triangle is that we can exactly describe regular shapes of geometric figures. But, it seems impossible to describe the shape of a cloud or mountain by using the shape of circles or ellipses. From the view point of traditional geometry, there is no way to describe the nature view of fractal. That is the reason why traditional geometry can not connect together with nature. Traditional geometry can only describe nature by using idealized mode. Thus, to some extent traditional geometry has made some contributions , but when we go deeper and irregular to find out the rule of regularity, it seems only that fractal geometry can deal with it.

The term Fractal was developed by a French mathematician, Benoit. B Mandelbrot who overcame the restraint of Galileo to discover a new rule from the chaotic shape of nature.

He pondered: The mathematical concept of a fractal characterizes objects with structures on various scales, large as well as small, and thus reflects a hierarchical principle of organization. He found an important character which is that all the fractal objects are self-similar[1]. However, he also thought that this is a principle of organizing a whole universe of structures in an unexpected way.[2]

Software

There are five softwares I used for my thesis topic. They are Icon Designer, HyperCard 2.0, MacroMind Director, Adobe Photoshop, The Beauty of Fractal Lab, and the postscript language.

The Icon Designer is a kind of icon builder software. In this software we can create icons for our own, or for some purpose of utility to fit the need of a special interface design. For example, here I created twelve special icons called Julia sets and the other one called Mandelbrot set icon to install into my thesis stack. In addition, the mandelbrot set icon also plays a central role as a logo in my thesis project. It exists on every card and gives the users a sense of clicking on this logo will directly bring you back to the very first control card under any circumstances.

HyperCard 2.0 is a powerful software to allow people to build up their own custom programs. HyperCard 2.0 is just like a warehouse of architecture which has any kind of tools and resources to let the users build up their own style of building (custom program). In addition, the new version of HyperCard 2.0 has some useful functions to access its resources. For example, I can create an animation in MacroMind Director then play it in HyperCard 2.0 and don't need to transfer the animation to Videoworks II to play it in HyperCard 2.0. So, the former one more

directly lets me accomplish my goal. The old version of HyperCard 1.0 can play a MacroMind Director animation only under some circumstances. For instance, you must ensure inside the animation score, color palette, sound, transmission can't exist when you want to transfer to Videoworks II animation. So, those conditions don't fit the need of my custom program.

MacroMind Director is a software of animation. Here, I used it to create two color cycling animations and installed them into my very first card of my thesis stack. However, I made those animations function as "attract mode", which with accompanying music (loud) and color cycling effect present a vivid screen. Here, the color cycling animation would play with a background music when the interface wasn't being accessed. Thus, in the introduction guide I put a paragraph that says: " Click on any where to turn off the current animation".

Adobe Photoshop is used to resize the images which were captured from NTSC video digitizer or flatbed scanner, and then fit them into my thesis stack. Images created by "The Beauty of Fractal Lab", were opened under Adobe Photoshop and resized. The first step used the screen capture function to copy the fractal image, then save it into the scrapbook as a PICT file format. Then I went to Adobe Photoshop, opened a new screen (the size is 512 x 512 pixels), and pasted

the fractal PICT file from the scrapbook to resize the image.

The last software I used was "The Beauty of Fractal Lab" which could let me create 2-Dimensional, 2.5-Dimensional, and 3-Dimensional fractal images. Actually, those fractal images could be used as an application for graphic design, illustration, and printmaking and so on.

When you open "The Beauty of Fractal Lab" software, it will show a Mandelbrot set. Click on a rectanglar button called Julia set and it will let you choose a c-value parameter from the Mandelbrot set for the computation of the Corresponding Julia set. At this moment, a dialog box will show up to allow the user to input some parameters. After you key in the specific number, click on the OK button. Then, a new Julia set will be drawn in a new window.

Postscript, a page description language, is a powerful and flexible language used for printing high-quality text and graphics. And it is also a device-independent page description language. I used it as a programming tool to create different kinds of Fractal Trees and snowflakes by using recursion (see Appendix H).

Hardware

The hardware involved with my thesis project are the Macintoch IIci, Sound Recorder, Flatbed Scanner, and NTSC video digitizer.

The Flatbed Scanner (or Flatbed desktop print scanner) is my first choice to grab images. This scanner provides the option of line art work (black and white), half tone, and grayscale scanning capabilities. Here I used the Flatbed Scanner to grab some grayscale images, such as the scientist and cartoon pictures, then brought it to Adobe Photoshop to resize it down to a specific size.

Another image graber is the NTSC video digitizer which allowed me to capture the images directly from magazines, books, or poster cards. In addition, the NTSC video digitizer with the color space card provided full color or grayscale capture in a high resolution output image. After saving the image file name, it was brought into Adobe Photoshop. Under the Adobe Photoshop file menu open "open" command, I then used the "drive " to find out the right image file name and resized it to a specific size.

Evaluation

The evaluation of my thesis project was based on the ten principles of Human Interface Design[3].

1. Use of metaphors
2. Direct manipulation
3. See - and - point (instead of remember - and - type)
4. Consistency
5. WYSIWYG (what you see is what you get)
6. User control
7. Feedback and dialog
8. Forgiveness
9. Perceived stability
10. Aesthetic integrity

Use of metaphors

Within my thesis stack I made use of the visual effect of zoom in and zoom out to simulate the experience of real world. For example, in the content of FRACTAL MICRO I install some rectangle buttons on the Mandelbrot set to imply a tool of zoom lens which can give the visual effect of zoom in and zoom out to simulate the real function of zoom lens. And, the audio sound is also available

for eight main navigation buttons which are HISTORY, FRACTAL TYPE, FRACTAL THEORY, FRACTAL MICRO, SCIENTIST, CARTOON, SLIDE SHOW, and GLOSSARY. Actually, the sound I recorded here is a part of Kitaro music. This music is recorded by using Sound Recorder. Then this sound file was saved as a resource file into my thesis main stack. I then used the XCMD play " sound file name" to play the music. In addition, the eight main navigation button is located at the bottom of the screen. If the user clicks on one of them, the selected rectangle button will turn to highlight and function as a real button being used.

Direct manipulation

I provided three characters for self-directed manipulation.. The first one is to pop up a dialog box to inform the user on how to make their choice in the next step. The second one is to auto highlight an icon button when the user moves the browse tool and enters into the button area. The highlight button will also turn off when the browse tool moves out of the range of this button area. The third one is to turn on a blinking button when a related button is being selected until a mouse click turns off this blinking button.

See - and - point (instead of remember - and - type)

In my thesis project I have a main navigation button called- Mandelbrot set icon button which means Quit or Return to Start and it is available on each card.

However, when a user opened a content to navigate, I provided a text field of see - and - point way to inform the user what the next step is.

Consistency

Consistency within a stack is an important factor for the user to focus on the content of my stack. Actually, I built up eight contents of cards, and they are consistent with each other. For example, the graphic look, the grouping of buttons, the placement of buttons, the layout of a card, stack structure, and visual and audio feedback. Those elements are considered as consistent design factors for my thesis stack. (please refer to Appendix D -- User Interface Screen Design)

WYSIWYG (what you see is what you get)

The feature of WYSIWYG is to let the users clearly know where they are , and to know what the relationship is to the whole stack. The way I used here is to provide a highlighted active button and a label field of text to let users know where they are. Here the label field of text acted as an enhancement of expression, so to speak, an indicator of "You - are - here".

User control

The approach taken to let users navigate my stack is to give them total control. Within the first card of my stack I designed an "attract mode", which is an

animation with music accompanying. At the same time, I put a field as an Introduction Guide to give users some information on how to navigate my stack. Finally, a last statement says "Click on any where to stop the animation". Another approach is giving users a convenient way to access a specific content by providing the eight contents buttons on each card so that the users can jump into any content under any circumstance. So, a good implementation is : give more right of control to the users.

Feedback and dialog

To address the concern of user-friendliness to the individual user, using a dialog form of communication exchange between a user and my stack is the best way. Within my first card when the animation is turned off, the user just clicks on one of the eight buttons of my thesis contents. Here, I provided a function of immediate feedback with a pop-up dialog box and a related highlighted button to inform the user what the reaction is for this implementation. For example, a user moves the browse tool into the FRACTAL TYPE button. At the same time, immediate feedback would be provided by this button which becomes highlighted, and a visual effect of a pop-up dialog box comes out to give the user brief information about clicking on the button. After the user reads over this information and clicks on the button, an immediate feedback sound would bring the user to the exact content.

Forgiveness

Don't use long text in a field , because most users don't want to read it. For the sake of preventing mistakes or exploring further then they really want to, I provided a limited short text field so that the user has the patience to read it, and kept a clear design interface through my whole stack. To this principle, I used most of my time to modify my stack by navigating around it , to find out possible errors that users could make. So, I would rather figure out what mistakes the users could make than just forgive them.

Perceived stability

There are six different areas within each card of my stack to give the users six familiar landmarks to depend upon. The first one is an icon button of the Mandelbrot set with a text field of specific content. The second one is the main text indicator (or description) area. The third one is a secondary text indicator area. The fourth one is the main navigation buttons area. The fifth is the secondary navigation buttons area. The last one is the Main active screen which is the biggest area to display different contents. Here, I used a font of Futura book for the main navigation buttons area and main text indicator. And a font of Geneva for the area of active screen, the secondary text indicator, and the secondary

navigation buttons area. However, here I also designed four different functions of button. The first one is the main control button of the Mandelbrot set icon which plays the important role of landmark. The second one is the sub control button which also plays a landmark of telling the user where they are. The third one is the sub-sub control buttons which show a sub contents of the main content. The last one is the navigation buttons which are the eight main contents to let the users navigate around my thesis stack.

Aesthetic integrity

Actually, the consideration of aesthetics to my thesis stack is a long term of struggling for me to layout the screen interface. The design principle for the graphic layout of my card is the grid design. As I have mentioned, it was a long period of struggling to refine and modify it to be a final version of stack. Though, the environment of HyperCard 2.0 is a black and white version, it still has some powerful functions of XCMD (external command) to playback a colorful animation or a color picture or even an audible communication. So, I think the HyperCard 2,0 plays an important role of human-machine interface and also a versatile multimedia.

Based on these ten general Human Interface Design principles I tried to evaluate and modify my thesis stack to be more and more usable to general users.

Because these ten principles are highly humanity-oriented, so people could navigate simply around my stack. Generally speaking, I am highly confident of my thesis project, but not just because I followed the ten general principles. There is also a balance of nature existing in the Human-Computer Interface to support me in building up my thesis stack.

Development

The purpose of my thesis stack actually is more educationally-oriented or you can say it is CAI (Computer-Assisted Instruction). Actually, its full name is The Fractal Archive prototype1.01 which could serve as an electronic database of knowledge in a college library for users to access this database. Or, it could be a resource for teaching.

The very first card of the Fractal Archive simply acts as a guider to my whole stack. Here, I played a color cycling animation on the active screen to act as an "attract-mode". This animation would play when the user hasn't accessed the stack. Before the user started to use my stack I provided an introduction guide to show the user how to operate this stack comfortably. The first thing is to click on any place to turn off the current animation, then position the browsing hand onto one of the eight rectangle buttons at the bottom of the screen. At the same time it will pop up a field of brief introduction to this specific button. Then, if the user pressed the mouse down, it would show a visual effect of dissolving, and bring the user to the exact content.

In this card I designed an icon button of the Mandelbrot set at the upper left hand corner. If the user clicks on this icon button, it means the user wants to quit or exit this stack under any card or any circumstance.

The first content is Fractal History. Here, I presented a scrolling field of fractal history in the active screen area. If a user wants to see more information, just click on the downward button at the right hand corner. There are three aspects of information being accessed; they are scientists, timeline, and brief description. At any time, if the user wants to see another contents, just click on one of the eight navigatation buttons in the active screen area. Here, the HISTORY button was highlighted to mean you are in the navigation of Fractal History, or you can click on the Mandelbrot icon to quit the navigation and back to the very card (or guider card).

The second content is Fractal Micro which means we can zoom into any location within the Mandelbrot set, and the picture always gives a different look to its appearance. Here, I provided twelve Julia icon buttons at the left hand side, and show a field of "Click on the Julia icon buttons at left hand side" to let the user know what is the next step to operate it. Meanwhile, if the user clicks on one of the Julia icon buttons, the active screen will show up a detailed color picture to this specific Julia icon button and come with a blinking radio button at a specific location. In addition, the main text indicator area also changes to a brief description of mathematics to this Julia set and a field of "Click on any place to turn off the current picture" or instead of clicking on another Julia icon button to see another

information. Actually, the best way to allow the user to navigate this content is a real-time function of navigation which means a specific Julia set picture and a specific radio button within the Mandelbrot set will show up at the same time.

Meanwhile, the user just dragged the moveable radio button to any location within the Mandelbrot set, and a relative Julia set picture will show up at the same time. I think this function is a good way to let the user more easily understand the relationship between Julia sets and the Mandelbrot set. In this content I provided a button called "see more" which can bring you to the second card of Fractal Micro. The main function of this card is to allow a user to zoom into five specific parts of the Mandelbrot set. Here, I provided a specific part of zooming effect to let the users know the character of self-similarity which also plays an important role of Fractal geometry.

The third content is Scientist. Here, I choose twelve scientists who recently have issued some reports, papers or research in mathematics of Fractals. In this content, I provided their names in the secondary navigation buttons area to allow the users to navigate through it. For example, if a user clicked on a scientist name, the active screen would show up a picture of the scientist and his personal information. Here, I gave a user guide under the secondary navigation buttons area

to let users know how to select the scientist list field. The user simply clicks on the scientist list with the mouse still held down, then moves the browsing hand until the selection the user desires is highlighted in black. Then, releasing the mouse, the image and the text relating to the user selection will appear. Here, the main navigation buttons area kept highlighted is SCIENTIST.

The fourth content is FRACTAL TYPE. In this content I provided ten different Fractal types in the secondary navigation field area, and provided a user guide to let users know the exact way to navigate through it. Actually, the exact way to navigate the content of FRACTAL TYPE, FRACTAL THEORY, and SCIENTIST is the same. But the only difference among them is the sub-secondary navigation field which exists in the content of FRACTAL TYPE. The secondary navigation field of the content of FRACTAL TYPE has the sub-navigation field to itself. For example, von Koch curves is one of the FRACTAL TYPE list. If the users select it, a sub-navigation field belonging to von Koch curves would show up. From this sub-navigation field of the von Koch curves, there exist eight different types of von Koch curves. Then, if the user selects one of the eight different types of von Koch curves, the name of the von Koch curve the user desires would be highlighted in black, and a field of text or picture related to this name of von Koch curve would show up.

If the user doesn't want to navigate any more within this content, just simply click on the main navigation buttons area to see another content or click on the Mandelbrot icon button to quit the navigation. The button of Fractal Type in the main navigation button area would be highlighted during the working of the content of FRACTAL TYPE.

The fifth content is FRACTAL THEORY. Here, I selected seven important theories about Fractals. The most important one is self-similarity. The property of self-similarity is one of the central concepts of Fractal geometry. If the users are interested about it, just make reference to the bibliography list. Actually, this content is more difficult to a user of non-mathematics background. The idea of my thesis was to build up three versions of stack to allow different user levels to use it. They are general users, mathematics-oriented users, and art/design users. However, in my eight contents of my thesis project, the content of FRACTAL TYPE, FRACTAL THEORY, and FRACTAL MICRO are suitable for mathematics-oriented users. The content of SCIENTIST, HISTORY, CARTOON, and GLOSSARY are suitable for general users. The content of SLIDE SHOW and FRACTAL MICRO are more fit to art design users. This content has the same usage as the content of SCIENTIST and FRACTAL THEORY in the main navigation buttons area would be highlighted.

The sixth content is SLIDE SHOW. There are four different types of SLIDE SHOW: 2-Dimension, 2.5-Dimension, 3-Dimension, and Fractal Landscape. Within this content I used a software called The Beauty of Fractal Lab which let me create different types of visual effects. Actually, I just played with a simple formula x*x + c. For each parameter value c, the iteration of the quadratic formula x*x + c in the plane of complex numbers shows a different result, in that for each value of c we could find out another picture of the Julia set. Here, I make use of the 3-Dimension Fractal image to combine with another image under the application of Adobe Photoshop to create a new feeling of visual experience. For example, I used NTSC video digitizer to grab a landscape image from a graphic book. Then, I saved this landscape image in PICT file format, and under the Adobe Photoshop opened this landscape image. Then, I selected a good view point of the 3-Dimension Fractal image, and cropped some useless parts. When everything is done, I used a function of screen capture to grab the 3-Dimension Fractal image and saved it as a PICT file format. After this step I reopened the landscape image under Adobe Photoshop, then open the 3-Dimension Fractal image. Then, under the environment of Adobe Photoshop I resized the landscape image and the 3-D Fractal image, after everything is done. I then copied the 3-D Fractal image to the landscape image. Here, I faced another problem, which is how to locate this 3-D Fractal image to a proper site and how to fit the color of the 3-D Fractal image to the environment color of

the landscape image. Above all, how to fit the perspective and shadow of light of 3-D Fractal image together with a landscape image is always a big problem that I should be careful about. Actually, this content could give the art and design users a new field of application.

The seventh content is CARTOON. In this content, I installed fourteen cards to let the users navigate through it. Actually, this content is more easy to understand for general users by means of the sequence of cartoon illustration and dialog to simplify a theory of self-similarity. Here, I put an arrow button and page number on the left hand corner to let the users know where they are and where they want to go. If the users don't want to see any more pictures or want to go back to page 1 , they just click on the CARTOON button on the secondary navigation button area, and it will bring them back to page 1. During the working of this content, the button of CARTOON in the main navigation button area would be highlighted.

The eighth content is GLOSSARY. In this content, it plays just like a dictionary to let the users check the meaning of a word. For instance, first, you should check the first alphabet of this word, then click on the alphabet keyboard on the secondary navigation buttons area. Then, it would show up a list of words in the alphabet. Clicked on this list of words with the mouse still held down until the

word is highlighted in black. Then, release the mouse and the text field relating to this word will appear. Here, I just input a few words and their meanings to be a sample example.

Conclusion

During the period of the development of my thesis stack, I have faced many problems, such as the scripting problem in the HyperTalk language, the appropriateness of user interface design, the aesthetics layout to my thesis stack, and the hardware problem of connecting the NTSC video digitizer to the Macintoch IIci main system. Here, I hope that I can accomplish my ideas of refining eight contents of my thesis stack in the latest future. For instance, the HISTORY content involved some scientists, theories, or even different types of fractals. From this HISTORY content the user can directly jump onto a specific scientist, theory, or fractal type if necessary. Instead of clicking on a specific content in the main navigation button area, then go searching for a specific scientist, theory, or fractal type. Because the former one is more directly to navigate around my stack. However, I appreciated the help from my thesis committee to overcome all the obstacles to accomplish my thesis of Fractal Archive prototype 1.01.

Once upon a time, one of my friends asked me a question: How could we use the environment of HyperCard 2.0 to build up an interactive multi-media? My answer is: The environment of HyperCard 2.0 is just like a warehouse of architecture. You can choose any kind of building element to decorate the interior of a house. The interior design for a house is just like human-machine interface design. They also play a role of humanity. A good design of human-machine

interface also gives the users a comfortable environment to operate the computer. Here, I have to say: HyperCard 2.0 is just like a good structure, good interior and exterior design of building to let people dwell in it.

Finally, I have to say that I was pleased with the decision and the result of my thesis. And, I will introduce to my people the environment of HyperCard 2.0 and show how it can create such a good human-machine interface.

Bibliography

Tay Vaughan. " Using HyperCard from Home to HyperTalk™ ", Que® Corporation, Carmel, Indiana, 1988.

Addison-Wesley Publishing Company, Inc. " PostScript Language Tutorial and Cookbook " by Adobe Systems Incorporated, 1985.

Benoit B. Mandelbrot, International Business Machines, Thomas J. Watson Research Center. " The Fractal Geometry of Nature ", W. H. Freeman and Company, New York, 1983.

Roger T. Stevens. " FRACTAL programming in Turbo Pascal ", M&T Publishing, Inc. Redwood City, California, 1990.

H.-O. Peitgen, P. H. Richter. " The Beauty of Fractals ", Springer-Verlag, Berlin, Heidelberg, 1986.

Heinz-Otto Peitgen, Dietmar Saupe Editors. " The Science of Fractal Images". Michael F. Barnsley, Robert L. Devaney, B. B. Mandelbrot, Heinz-Otto Peitgen, Dietmar Saupe, Richard F. Voss with Contributions by Yuval Fisher, Michael McGuire. Springer-Verlag, New York Inc. 1988.

Diane Gayeski, David Williams, " INTERACTIVE MEDIA ", Prentice- Hall, Inc, 1985.

Parian Stewart, "LES CHRONIQUES DE ROSE POLYMATH",1985.

Apple Computer Co., 'HyperCard® Stack Design Guidelines', Addison-Wesley Publishing Company, Inc.

The FRACTAL STUFF Company, "A FRACTAL BALLET", "THE ART OF FUTURING", 1988.

A. J. Crilly, R. A. Earnshaw, H. Jones, "Fractals and Chaos", Springer-Verlag New York Inc., © 1991.

Bibliography

HyperCard 2.0, © 1990

MacroMind Director 2.01, © 1990

Adobe Photoshop™ 1.07, Adobe Systems Incorporated, © 1990

The Beauty of Fractals Lab 1.0.1, Springer-Verlag, © 1990

Postscript Language, © 1990

Aldus PageMaker 3.0, Aldus Corporation, © 1985~1988

Footnotes

1. H. -O. Peitgen, P. H. Richter, "The Beauty of Fractals", Springer-Verlag, Berlin, Heidelberg, 1986, p. vi.

2. H. -O. Peitgen, P. H. Richter, "The Beauty of Fractals", Springer-Verlag, Berlin, Heidelberg, 1986, p. vi.

3. Apple Computer Co., "HyperCard® Stack Design Guidelines", Addison-Wesley Publishing Company, Inc., p. 175~183.

WHAT ARE FRACTALS ?

The term "fractal" comes from a recently developed field of mathematics known as "Fractal Geometry". The latin root, "fractus", meaning broken or irregular, was chosen to describe this new school of thought in the field of geometry; a geometry that describes and creates an astonishing range of fragmented, irregular, wiggly forms known as Fractals.

The theory of fractal geometry was developed by IBM mathematician, Benoit Mandelbrot (pronounced Ben-wah'). In fact, the proper technical term for the larger "turtle" in figure 1 is the "Mandelbrot Set", and it is considered to be one of the most complexity, the aesthetic appeal of fractal patterns seems to cause a sense of fascination for most who view them.

Fractal shapes tend to repeat themselves in similar patterns on different scales.. Specifically, an enlargement of a small portion of the pattern will look similar the whole. This is termed "self-similarity" and its influence can be seen in the repeating pattern on this garment, which is the first in a series of computer generated patterns to be offered.

The design is a magnified, close-up view of the interior edge of the larger "fractal turtle". "Turtle" is the nick-name of the distinguishing master-pattern that continues to appear and reappear as one begins to zoom in on the edges of this larger fractal. Using increasing degress of magnification, an almost infinite range of unique and intriguing patterns can be found.

Upon closer inspection, fractals reveal many of the forms found in nature, such as rivers, trees, clouds, the flames of a fire, or galaxies. These natural shapes are better described through the chaotic structures of fractal geometry than the rectangles, spheres and cones of traditional Euclidian geometry. The random patterns generated within fractals capture the texture of reality as it is found in nature.

Appendix: A

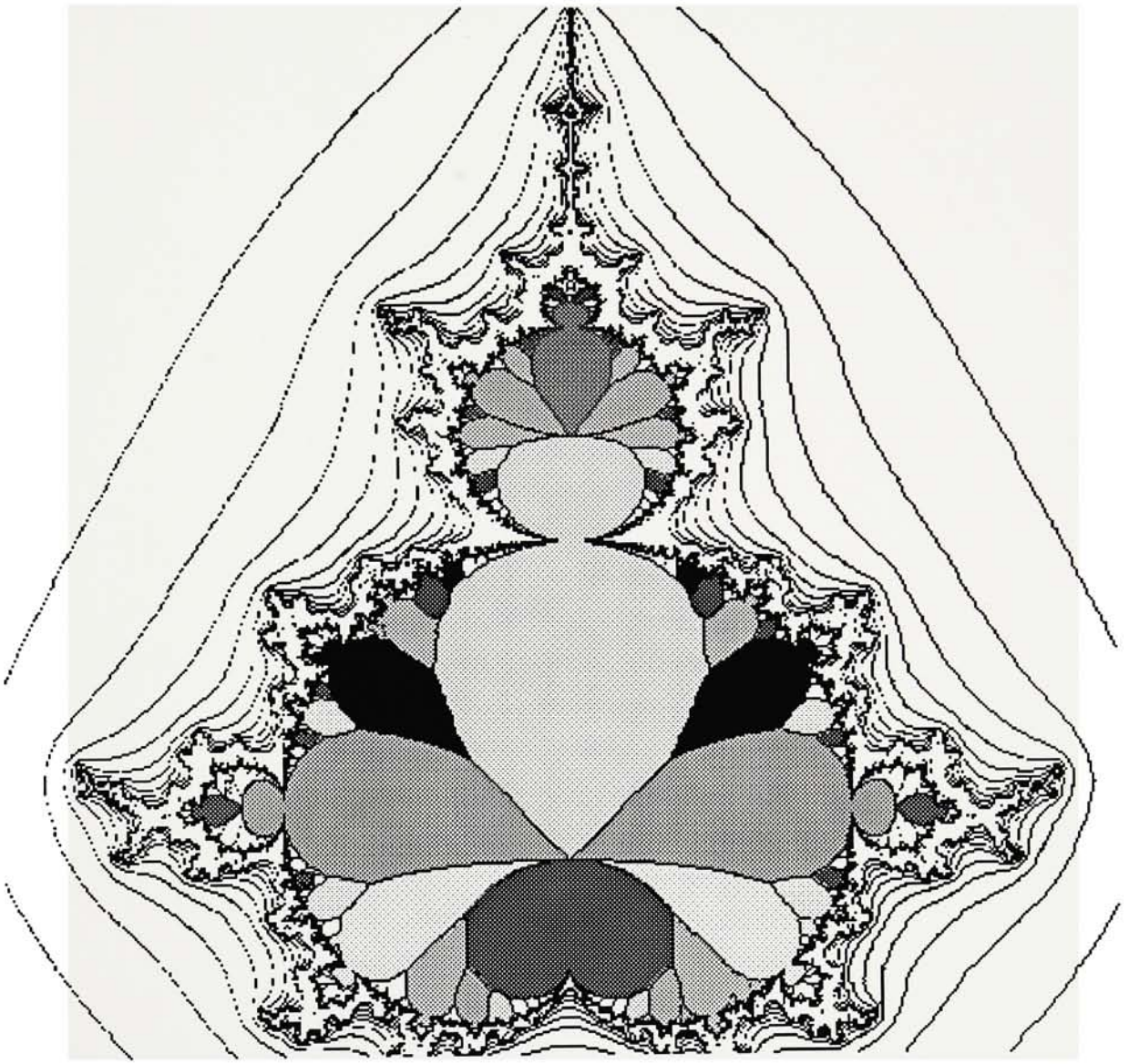


Figure 1.

Thesis scripting in HyperTalk Language

```
--This is a card script at the very beginning card.
on opencard
  hide menubar
  hide cd fld 11
  hide cd fld 18
  hide cd fld 26
  hide cd fld 35
  hide cd fld 44
  hide cd fld 52
  hide cd fld 59
  hide cd fld 68

  set hilite of btn 5 to false
  picture "fractal title-103",file,rect,false
  show window "fractal title-103" at 135,1
  picture "fractal title-105",file,rect,false
  show window "fractal title-105" at 465,1
  set hilite of button 2 to false
  set hilite of button 10 to false
  hide bg fld scientist
  hide bg fld "card id"
  set hilite of button 4 to false

  playMovie "Fractal Anim-55",Moviepreload
  playMovie "Fractal Anim-55",Movieloop,MovieNoClear,Movielocation,¬
  85,63,Movieclick
end opencard

on closecard
  close window "fractal title-105"
end closecard
```

Thesis scripting in HyperTalk Language

--This is a card script in the content of FRACTAL HISTORY.

```
on opencard
  set hilite of btn 4 to true
  picture "HISTORY TITLE",file,rect,false
  show window "HISTORY TITLE" at 465,1
end opencard

on closecard
  close window "HISTORY TITLE"
end closecard
```

--This is a card script in the content of FRACTAL MICRO.

```
on opencard
  set hilite of btn 31 to true
  picture "MICRO TITLE",file,rect,false
  show window "MICRO TITLE" at 465,1
  show cd fld datainfo

  hide cd fld datainfo1
  hide cd fld datainfo2
  hide cd fld datainfo3
  hide cd fld datainfo4
  hide cd fld datainfo5
  hide cd fld datainfo6
  hide cd fld datainfo7
  hide cd fld datainfo8
  hide cd fld datainfo9
  hide cd fld datainfo10
  hide cd fld datainfo11
  hide cd fld datainfo12
end opencard

on closecard
  close window "MICRO TITLE"
end closecard
```

Thesis scripting in HyperTalk Language

--This is a card script in the content of FRACTAL TYPE.

```
on opencard
  picture "TYPE TITLE",file,rect,false
  show window "TYPE TITLE" at 465,1
  put empty into cd fld typebox
  put empty into cd fld Peanobox

  hide cd fld "Mandelbrot Sets"
  hide cd fld "Julia Set"
  hide cd fld "Bifurcation Diagrams"
  hide cd fld "Hilbert Curves"
  hide cd fld "von Koch Curves"
  hide cd fld "Peano Curves"
  hide cd fld "Sierpinski Curves"
  hide cd fld "Dragon Curves"
  hide cd fld "Phoenix Curves"
  hide cd fld "Trees"

  repeat with x=14 to 21
    hide cd fld x
  end repeat
  hide cd fld 12

  repeat with y=23 to 65
    hide cd fld y
  end repeat

  set hilite of button 2 to true
end opencard

on closecard
  close window "TYPE TITLE"
end closecard
```

Thesis scripting in HyperTalk Language

--This is a card script in the content of FRACTAL THEORY.

```
on opencard
 set hilite of btn 11 to true
 picture "THEORY TITLE",file,rect,false
 show window "THEORY TITLE" at 465,1
 repeat with x=3 to 5
  hide cd fld x
 end repeat

 repeat with x=7 to 12
  hide cd fld x
 end repeat

 put empty into cd fld 6
end opencard

on closecard
 close window "THEORY TITLE"
end closecard
```

--This is a card script in the content of FRACTAL SCIENTIST.

```
on opencard
 set hilite of btn 5 to true
 picture "SCIENTIST TITLE",file,rect,false
 show window "SCIENTIST TITLE" at 465,1

 hide cd fld thelist
 hide cd fld output
end opencard

on closecard
 close window "SCIENTIST TITLE"
end closecard
```

Thesis scripting in HyperTalk Language

--This is a card script in the content of SLIDE SHOW.

```
on opencard
  set hilite of btn 13 to true
end opencard
```

--This is a card script in the content of CARTOON.

```
on opencard
  set hilite of btn 12 to true
  set hilite of btn 3 to true
  picture "Cartoon-1",file,rect,false
  show window "Cartoon-1" at 230,75
end opencard

on closecard
  close window "Cartoon-1"
end closecard
```

--This is a card script in the content of GLOSSARY.

```
on opencard
  hide cd fld algebra
  hide cd fld Fractals

  set hilite of btn 36 to true
  picture "GLOSSARY TITLE",file,rect,false
  show window "GLOSSARY TITLE" at 465,1
  repeat with x=3 to 28
    hide cd fld x
  end repeat
end opencard

on closecard
  close window "GLOSSARY TITLE"
end closecard
```

Thesis scripting in HyperTalk Language

--This is a script of HISTORY button.

```
on mouseenter
  hide cd fld 18
  hide cd fld 26
  hide cd fld 35
  hide cd fld 44
  hide cd fld 52
  hide cd fld 59
  hide cd fld 68
  set hilite of btn 7 to true
  repeat with count=3 to 11
    show cd fld count
  end repeat
  repeat with count=3 to 10
    hide cd fld count
  end repeat
end mouseenter

on mouseleave
  set hilite of btn 7 to false
end mouseleave

on mousedown
  play "Kitaro music-1"
  hide cd fld 11
  set hilite of button 4 to false
  visual effect dissolve fast
  go to card id 31057
end mousedown
```

Thesis scripting in HyperTalk Language

--This is a script of FRACTAL MICRO button.

```
on mouseenter
  hide cd fld 11
  hide cd fld 26
  hide cd fld 35
  hide cd fld 44
  hide cd fld 52
  hide cd fld 59
  hide cd fld 68

  set hilite of btn 3 to true
  repeat with count=12 to 18
    show cd fld count
  end repeat
  repeat with count=12 to 17
    hide cd fld count
  end repeat
end mouseenter

on mouseleave
  set hilite of btn 3 to false
end mouseleave

on mousedown
  play "Kitaro music-1"
  hide cd fld 18
  set hilite of button 4 to false
  visual effect dissolve
  go to card id 15065
end mousedown
```

Thesis scripting in HyperTalk Language

--This is a script of FRACTAL TYPE button.

```
on mouseenter
  hide cd fld 11
  hide cd fld 18
  hide cd fld 35
  hide cd fld 44
  hide cd fld 52
  hide cd fld 59
  hide cd fld 68

  set hilite of btn 12 to true
  repeat with count=19 to 26
    show cd fld count
  end repeat
  repeat with count=19 to 25
    hide cd fld count
  end repeat
end mouseenter

on mouseleave
  set hilite of btn 12 to false
end mouseleave

on mouseUp
  play "Kitaro music-1"
  hide cd fld 26
  set hilite of button 4 to false
  visual effect dissolve
  go to card id 5241
end mouseUp
```

Thesis scripting in HyperTalk Language

```
--This is a script of FRACTAL THEORY button.
on mouseenter
  hide cd fld 11
  hide cd fld 18
  hide cd fld 26
  hide cd fld 44
  hide cd fld 52
  hide cd fld 59
  hide cd fld 68

  set hilite of btn 4 to true
  repeat with count=27 to 35
    show cd fld count
  end repeat
  repeat with count=27 to 34
    hide cd fld count
  end repeat
end mouseenter

on mouseleave
  set hilite of btn 4 to false
end mouseleave

on mousedown
  play "Kitaro music-1"
  hide cd fld 35
  visual effect dissolve
  go to card id 5699
end mousedown
```

Thesis scripting in HyperTalk Language

--This is a script of SCIENTIST button.

```
on mouseenter
  hide cd fld 11
  hide cd fld 18
  hide cd fld 26
  hide cd fld 35
  hide cd fld 44
  hide cd fld 52
  hide cd fld 59

  set hilite of btn 9 to true
  repeat with count=60 to 68
    show cd fld count
  end repeat
  repeat with count=60 to 67
    hide cd fld count
  end repeat
end mouseenter

on mouseleave
  set hilite of btn 9 to false
end mouseleave

on mousedown
  play "Kitaro music-1"
  hide cd fld 68
  visual effect dissolve
  go to card id 2260
end mousedown
```

Thesis scripting in HyperTalk Language

--This is a script of SLIDE SHOW button.

```
on mouseenter
  hide cd fld 11
  hide cd fld 18
  hide cd fld 26
  hide cd fld 35
  hide cd fld 44
  hide cd fld 52
  hide cd fld 68

  set hilite of btn 6 to true
  repeat with count=53 to 59
    show cd fld count
  end repeat
  repeat with count=53 to 58
    hide cd fld count
  end repeat
end mouseenter

on mouseleave
  set hilite of btn 6 to false
end mouseleave

on mousedown
  play "Kitaro music-1"
  hide cd fld 59
  set hilite of button 4 to false
  visual effect dissolve
  go to card id 6205
end mousedown
```

Thesis scripting in HyperTalk Language

--This is a script of CARTOON button.

```
on mouseenter
  hide cd fld 11
  hide cd fld 18
  hide cd fld 26
  hide cd fld 35
  hide cd fld 44
  hide cd fld 59
  hide cd fld 68

  set hilite of btn 5 to true
  repeat with count=45 to 52
    show cd fld count
  end repeat
  repeat with count=45 to 51
    hide cd fld count
  end repeat
end mouseenter

on mouseleave
  set hilite of btn 5 to false
end mouseleave

on mousedown
  play "Kitaro music-1"
  hide cd fld 52
  set hilite of button 4 to false
  visual effect dissolve
  go to stack "Cartoon"
end mousedown
```

Thesis scripting in HyperTalk Language

--This is a script of GLOSSARY button.

```
on mouseenter
  hide cd fld 11
  hide cd fld 18
  hide cd fld 26
  hide cd fld 35
  hide cd fld 52
  hide cd fld 59
  hide cd fld 68

  set hilite of btn 10 to true
  repeat with count=36 to 44
    show cd fld count
  end repeat
  repeat with count=36 to 43
    hide cd fld count
  end repeat
end mouseenter

on mouseleave
  set hilite of btn 10 to false
end mouseleave

on mousedown
  play "Kitaro music-1"
  hide cd fld 44
  set hilite of button 4 to false
  visual effect dissolve
  go to card id 4430
end mousedown
```

Thesis scripting in HyperTalk Language

--This is a script of card field "typelist" in the content of FRACTAL TYPE

```
on mousedown
  put empty into cd fld typebox
  put empty into cd fld Peanobox

  close window "Mandelbrot Set"
  close window "Julia Set"

  close window "Bare Tree"
  close window "Tree with Foliage"
  close window "One-Sided Tree"
  close window "Bronchial System Tree"
  close window "Arterial System Tree"
  close window "90 Degree Branch Tree"
  close window "85 Degree Branch Tree"
  close window "90 D Tree and Wider Stem"

  hide cd fld "Mandelbrot Sets"
  hide cd fld "Julia Sets"
  hide cd fld "Bifurcation Diagrams"
  hide cd fld "Hilbert Curves"
  hide cd fld "von Koch Curves"
  hide cd fld "Peano Curves"
  hide cd fld "Sierpinski Curves"
  hide cd fld "Dragon Curves"
  hide cd fld "Phoenix Curves"
  hide cd fld "Trees"

  repeat with x=14 to 21
    hide cd fld x
  end repeat

  repeat with y=23 to 65
    hide cd fld y
  end repeat
```

Thesis scripting in HyperTalk Language

```
repeat until the mouse is up
  get item 2 of the mouseloc - top of me + textHeight of me
  put trunc (.45 + it / the textHeight of me) into linenumber
  select line linenumber of me
end repeat

  select line linenumber of cd fld typelist
  get line linenumber of cd fld typelist


  put it into cd fld typebox
  put it into lineholder
  show cd fld lineholder
  set hilite of btn typebutton to true

end mousedown
```

Thesis scripting in HyperTalk Language

--This is a script of card field "Trees" in the content of FRACTAL TYPE

```
on mousedown
  close window "Bare Tree"
  close window "Tree with Foliage"
  close window "One-Sided Tree"
  close window "Bronchial System Tree"
  close window "Arterial System Tree"
  close window "90 Degree Branch Tree"
  close window "85 Degree Branch Tree"
  close window "90 D Tree and Wider Stem"

  hide cd fld "Real Trees"
  hide cd fld "Mathematical Trees"
  hide cd fld "Bare Tree"
  hide cd fld "Tree with Foliage"
  hide cd fld "One-Sided Tree"
  hide cd fld "Bronchial System Tree"
  hide cd fld "Arterial System Tree"
  hide cd fld "90 Degree Branch Tree"
  hide cd fld "85 Degree Branch Tree"
  hide cd fld "90 D Tree and Wider Stem"

  put empty into cd fld Peanobox
  repeat until the mouse is up
    get item 2 of the mouseloc - top of me + textHeight of me
    put trunc (.35 + it / the textHeight of me) into linenumber
    select line linenumber of me
  end repeat

  select line linenumber of cd fld "Trees"
  get line linenumber of cd fld "Trees"
  put it into cd fld Peanobox
  put it into lineholder
  show cd fld lineholder
```

```
  if it is found then
  picture it,file,rect,false
  show window it at 399,230
  else
  answer "Sorry, No picture here!"
  end if
end mousedown
```

Thesis scripting in HyperTalk Language

```
--This is a script of card field "theorylist" in the content
--of FRACTAL THEORY
on mousedown
  global show
  put empty into cd fld 6
  repeat with x=4 to 5
    hide cd fld x
  end repeat

  repeat with w=7 to 12
    hide cd fld w
  end repeat

  repeat until the mouse is up
    get item 2 of the mouseloc - top of me + textHeight of me
    put trunc (.45 + it / the textHeight of me) into linenumber
    select line linenumber of me
  end repeat

  select line linenumber of cd fld theorylist
  get line linenumber of cd fld theorylist
  put it into cd fld 6
  put it into lineholder
  show cd fld lineholder
  set hilite of btn theorybutton to true
end mousedown
```

Thesis scripting in HyperTalk Language

--This is a script of card field "scientistbox" in the content of
--SCIENTIST

```
on mousedown
  global theshow
  close window "Richard F. Voss"
  close window "Dietmar Saupe"
  close window "Michael McGuire"
  close window "Yuval Fisher"
  close window "Robert L. Devaney"
  close window "Michael F. Barnsley"
  close window "Peter H. Richter"
  close window "Herbert W. Franke"
  close window "Heinz-Otto Peitgen"
  close window "Gert Eilenberger"
  close window "B.B. Mandelbrot"
  close window "Adrien Douady"

  repeat until the mouse is up
    get item 2 of the mouseloc - top of me + textHeight of me
    put trunc (.45 + it / the textHeight of me) into linenumber
    select line linenumber of me
  end repeat

  select line linenumber of cd fld scientistbox
  get line linenumber of cd fld scientistbox
  put it into theshow
  go to cd theshow
end mousedown
```

Thesis scripting in HyperTalk Language

--This is a card script in the content of CARTOON.

```
on opencard
  set hilite of btn 12 to true
  set hilite of btn 3 to true
  picture "Cartoon-1",file,rect,false
  show window "Cartoon-1" at 230,75
end opencard

on closecard
  close window "Cartoon-1"
end closecard
```

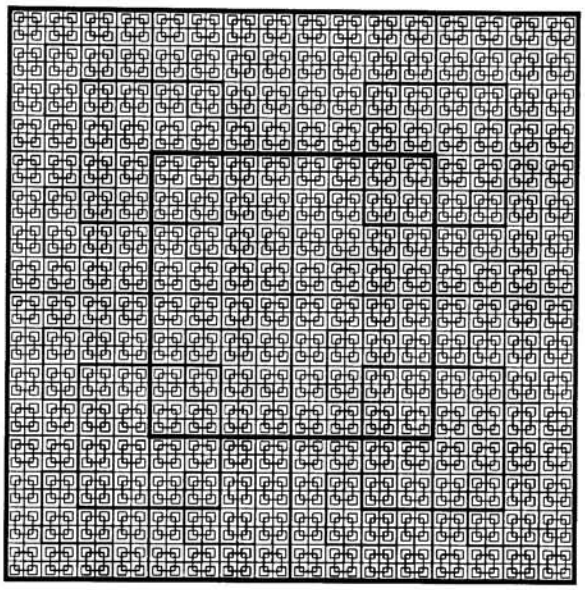
--This is a script of card button "F" in the content of GLOSSARY

```
on mouseenter
  set hilite of btn f to true
end mouseenter

on mouseleave
  set hilite of btn f to false
end mouseleave

on mouseDown
  set hilite of btn f to true
  show cd fld f
  repeat with x=3 to 7
    hide cd fld x
  end repeat
  repeat with x=9 to 28
    hide cd fld x
  end repeat
  set hilite of btn f to false
end mouseDown
```

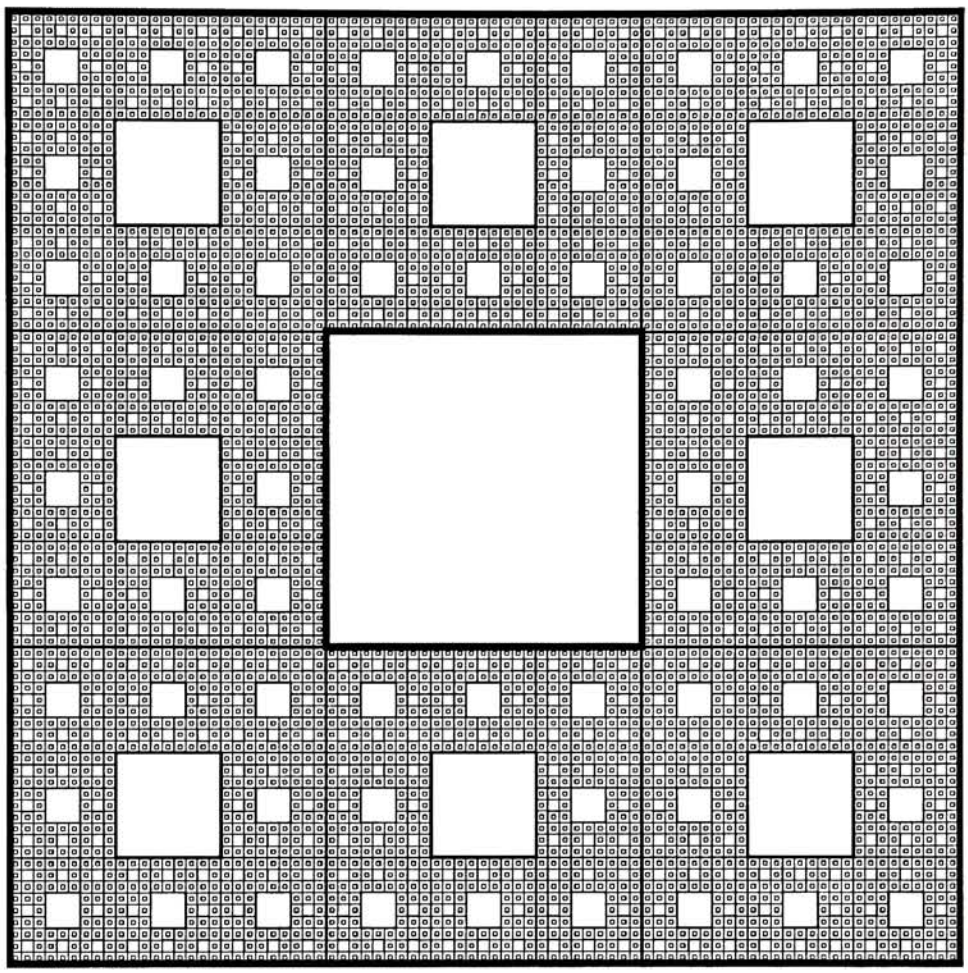
Appendix: C

Scripting in Postscript Language

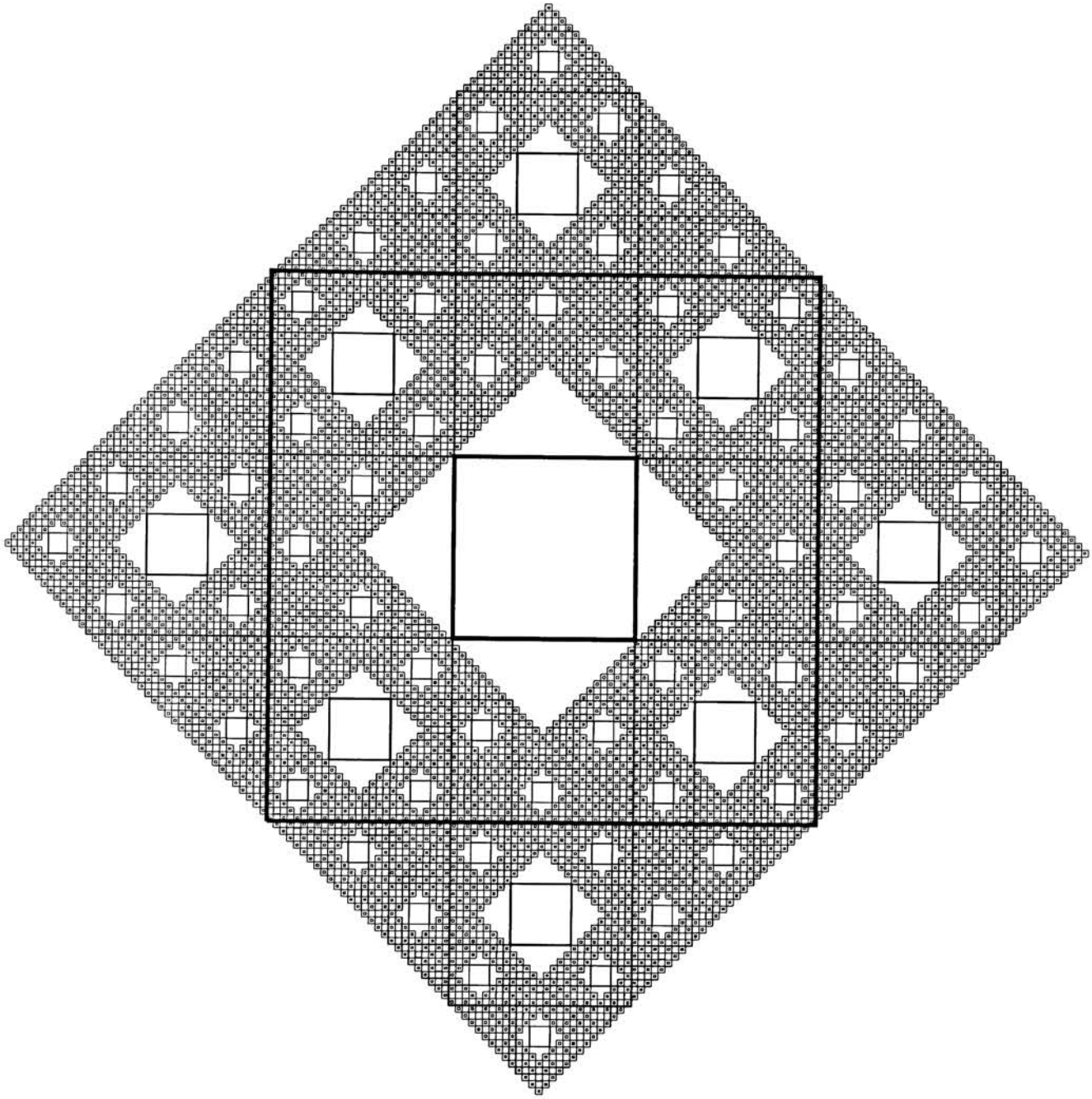
```
% Recursion Program-1

% Define Variables

 /depth 0 def
 /maxdepth 5 def

% Define Procedures

 2 setlinecap
 .5 setlinewidth
 /DrawPath
   {newpath 0 0 moveto 72 0 lineto
   72 72 lineto 0 72 lineto
   0 0 lineto
   18 18 moveto 54 18 lineto
   54 54 lineto 18 54 lineto
   closepath stroke} def

 /Shape
    {gsave
       /depth depth 1 add def
       DrawPath
       depth maxdepth le
           { .5 .5 scale
             Shape
             72 0 translate Shape
             0 72 translate Shape
             -72 0 translate Shape
            } if
        /depth depth 1 sub def
     grestore
     } def

% Main Program

  200 300 translate
  3 3 scale
  Shape
  showpage
```



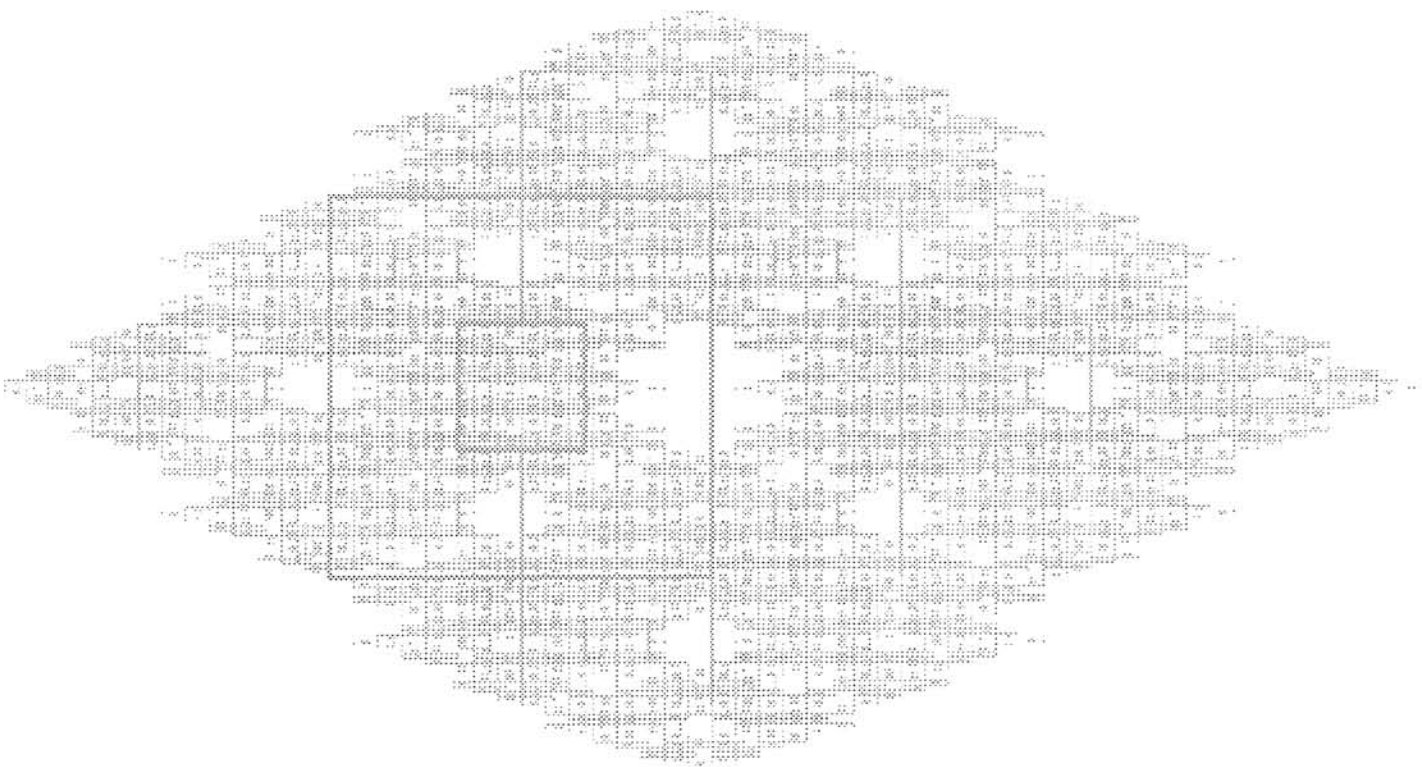
```
% Recursion Program-2

% Define Variables

/depth 0 def
/maxdepth 4 def

% Define Procedures

2 setlinecap
.5 setlinewidth
/DrawPath
  {newpath
  0 0 moveto 72 0 lineto
  72 72 lineto 0 72 lineto
  0 0 lineto

  24 24 moveto 48 24 lineto
  48 48 lineto 24 48 lineto
  closepath
  stroke} def

/Shape
   {gsave
      /depth depth 1 add def
      DrawPath
      depth maxdepth le
          { .333 .333 scale
            Shape
            72 0 translate Shape
            72 0 translate Shape
            0 72 translate Shape
            0 72 translate Shape
            -72 0 translate Shape
            -72 0 translate Shape
            0 -72 translate Shape
           } if
       /depth depth 1 sub def
    grestore
    } def

% Main Program

  200 300 translate
  5 5 scale
  Shape
  showpage
```



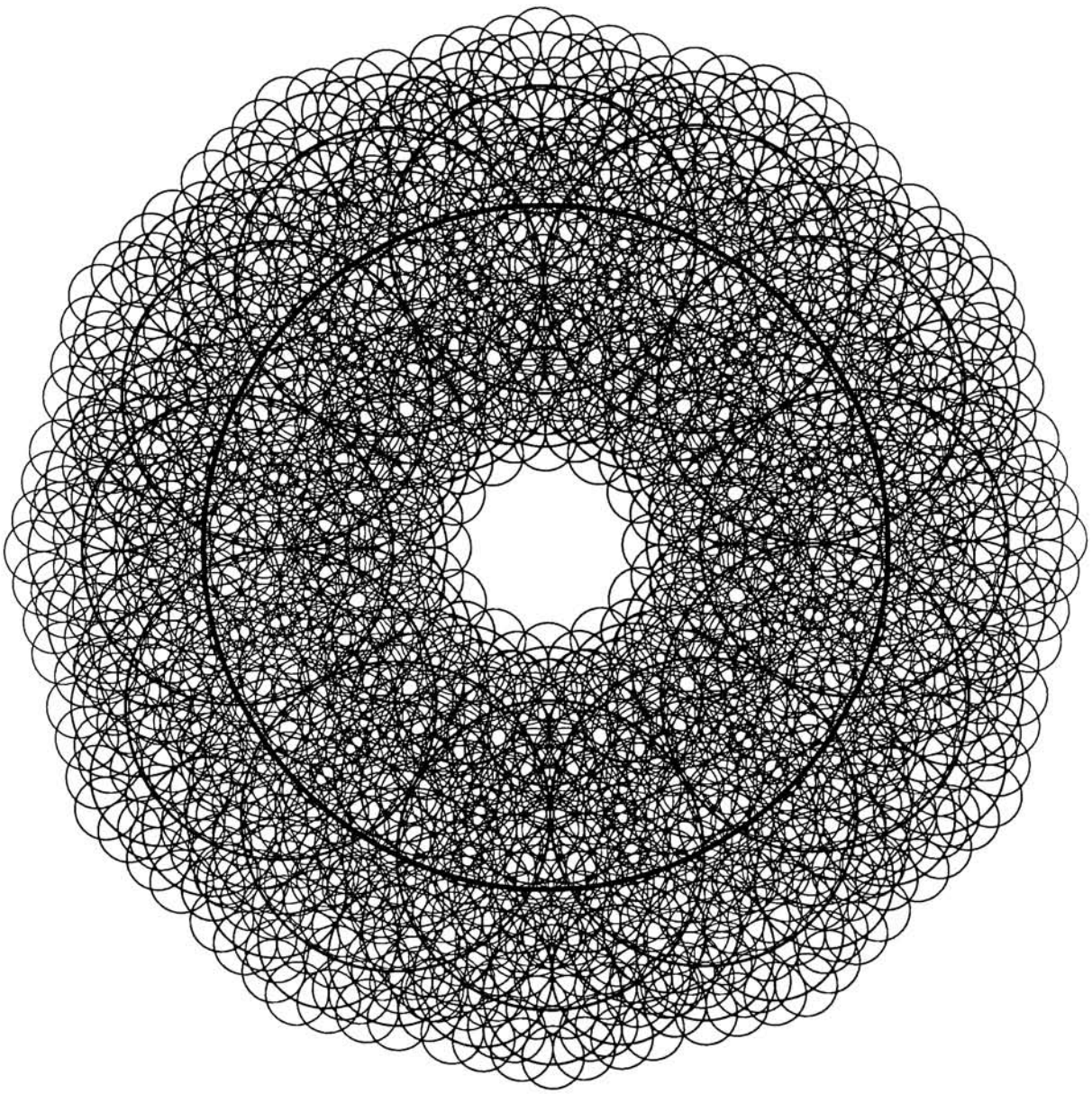
```
% Recursion Program-3

% Define Variables

 /depth 0 def
 /maxdepth 4 def

% Define Procedures

 .5 setlinewidth
 /DrawPath
   {newpath
   0 0 moveto 72 0 lineto
   72 72 lineto 0 72 lineto
   0 0 lineto

   24 24 moveto 48 24 lineto
   48 48 lineto 24 48 lineto
   closepath
   stroke} def

 /Shape
    {gsave
       /depth depth 1 add def
       DrawPath
       depth maxdepth le
           { .333 .333 scale
             Shape
             72 -72 translate Shape
             72 72 translate Shape
             72 72 translate Shape
             -72 72 translate Shape
             -72 72 translate Shape
             -72 -72 translate Shape
             -72 -72 translate Shape
            } if
        /depth depth 1 sub def
     grestore
     } def

% Main Program

  150 300 translate
  3.5 3.5  scale
  Shape
  showpage
```



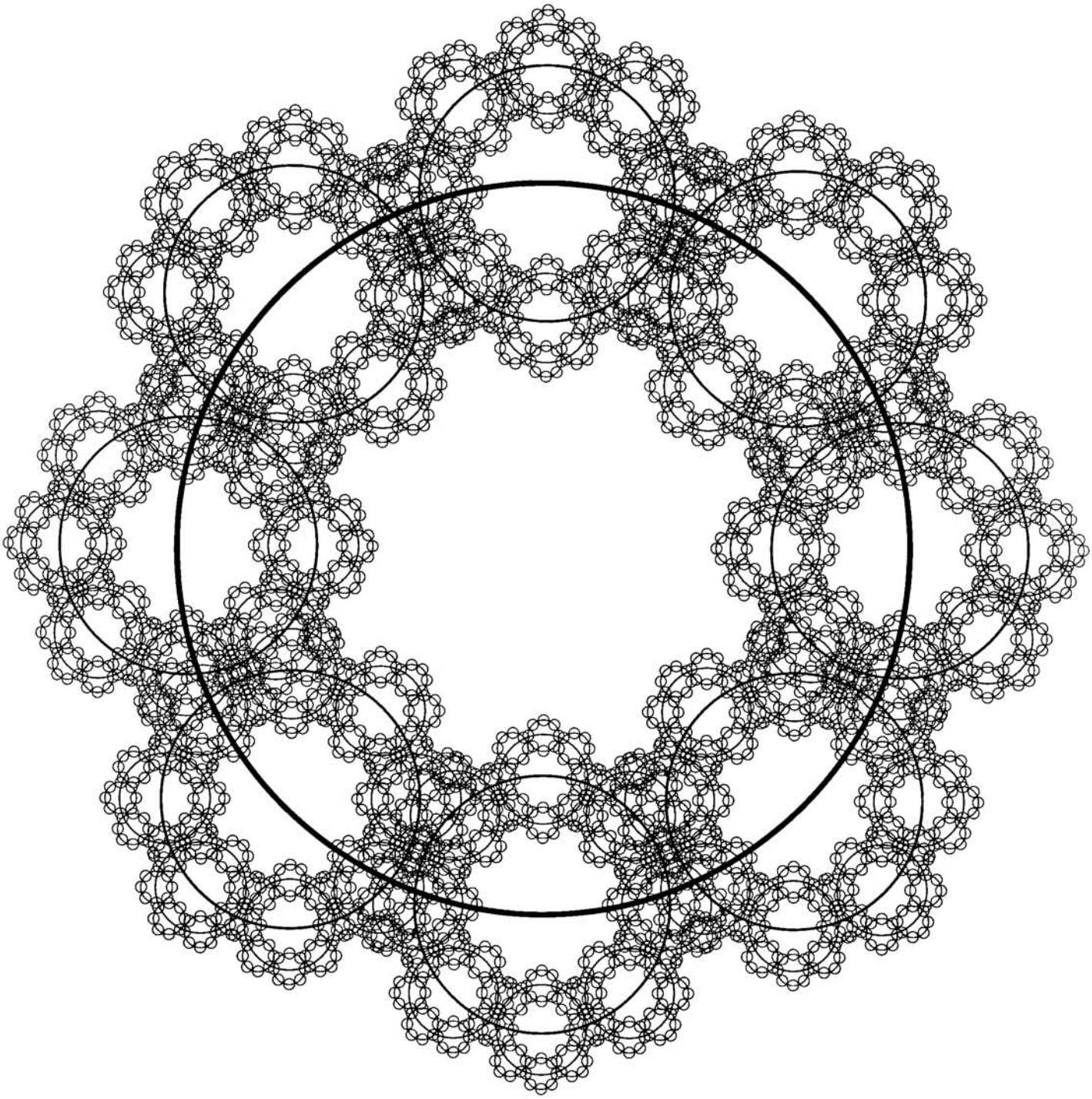
```
% Recursion Program-4

% Define Variables

 /depth 0 def
 /maxdepth 4 def

% Define Procedures

 .5 setlinewidth
 /DrawPath
   {newpath
   0 0 moveto 72 0 lineto
   72 72 lineto 0 72 lineto
   0 0 lineto

   24 24 moveto 48 24 lineto
   48 48 lineto 24 48 lineto
   closepath
   .85 setgray
   stroke} def

 /Shape
    {gsave
       /depth depth 1 add def
       DrawPath
       depth maxdepth le
           { .5 .333 scale
             Shape
             72 -72 translate Shape
             72 72 translate Shape
             72 72 translate Shape
             -72 72 translate Shape
             -72 72 translate Shape
             -72 -72 translate Shape
             -72 -72 translate Shape
            } if
         /depth depth 1 sub def
     grestore
     } def

% Main Program

  150 300 translate
  2 2  scale
  Shape
  showpage
```



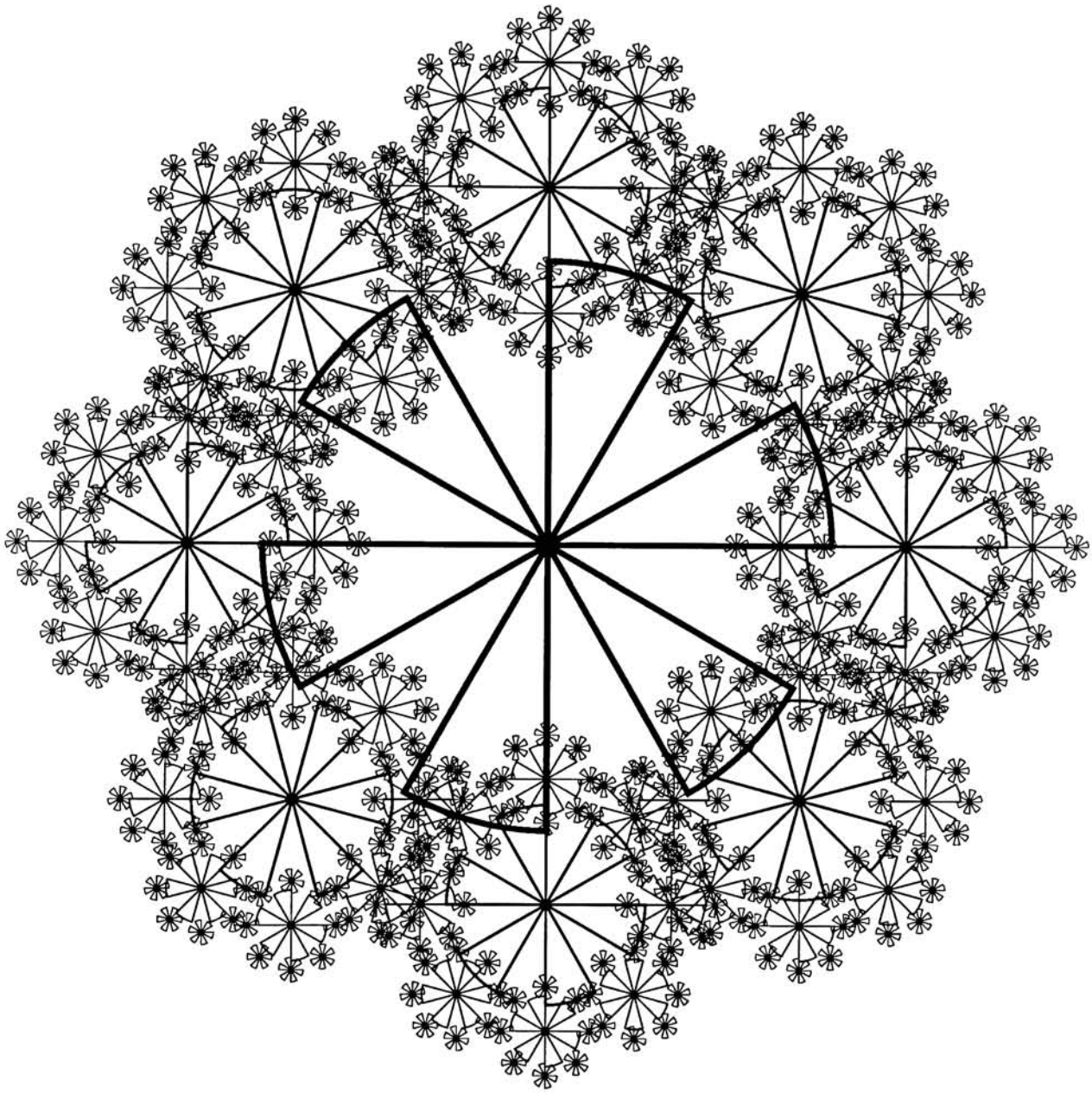
```
% Recursion Program-5

% Define Variables

 /depth 0 def
 /maxdepth 3 def

% Define Procedures

 /DrawPath
     {0 0 72 0 360 arc} def

 /Shape
     {gsave
        /depth depth 1 add def
        DrawPath stroke
        depth maxdepth le
          {.45 .45 scale
            12{30 rotate
            144 0 translate
            Shape
            -144 0 translate
              } repeat
           }if
         /depth depth 1 sub def
      grestore
     } def

% Main Program

 300 400 translate
 30 rotate
 2 2 scale
 Shape
 showpage
```



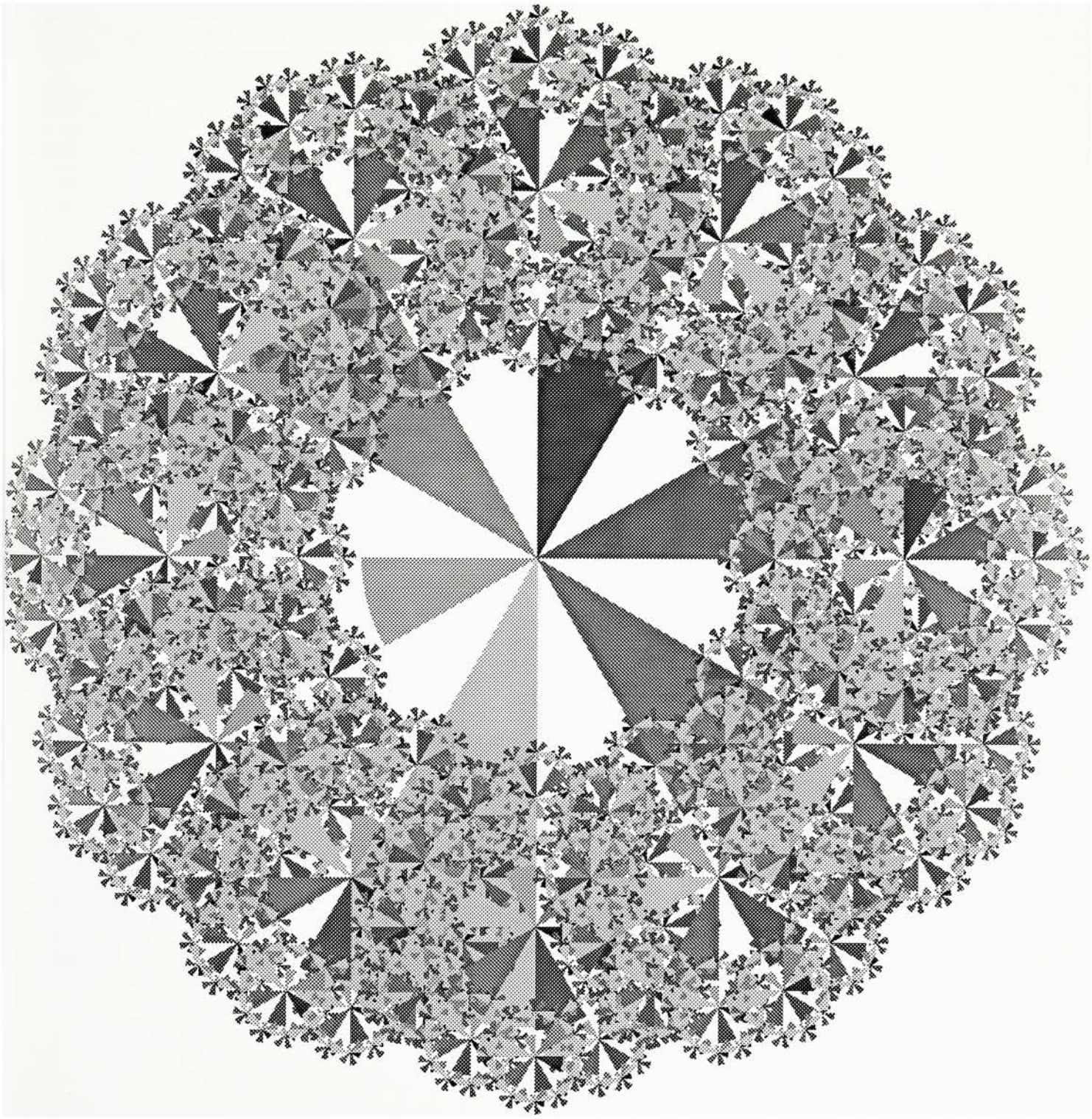
```
% Recursion Program-7

% Define Variables

 /depth 0 def
 /maxdepth 4 def

% Define Procedures

 /DrawPath
     {0 0 72 0 360 arc} def

 /Shape
     {gsave
        /depth depth 1 add def
        DrawPath stroke
        depth maxdepth le
          {.35 .35 scale
            8{45 rotate
            200 0 translate
            Shape
            -200 0 translate
              } repeat
           }if
         /depth depth 1 sub def
      grestore
     } def

% Main Program

 300 400 translate
 2.5 2.5 scale
 Shape
 showpage
```



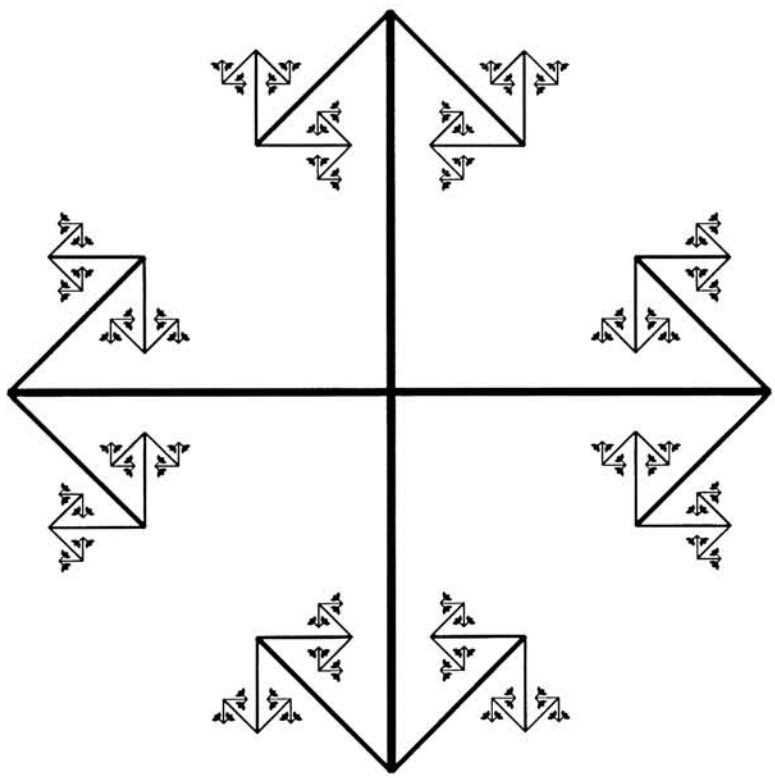
```
% Recursion Program-8

% Define Variables

 /depth 0 def
 /maxdepth 3 def

% Define Procedures

 /DrawPath
     {newpath 0 0 moveto 0 0 50 0 30 arc
      closepath stroke
              0 0 moveto 0 0 50 60 90 arc
      closepath stroke
              0 0 moveto 0 0 50 120 150 arc
      closepath stroke
              0 0 moveto 0 0 50 180 210 arc
      closepath stroke
              0 0 moveto 0 0 50 240 270 arc
      closepath stroke
              0 0 moveto 0 0 50 300 330 arc
      closepath stroke} def

 /Shape
     {gsave
        /depth depth 1 add def
        DrawPath stroke
        depth maxdepth le
          {.35 .35 scale
            8{45 rotate
            180 0 translate
            Shape
            -180 0 translate
              } repeat
           }if
         /depth depth 1 sub def
      grestore
     } def

% Main Program

 300 400 translate
 2.5 2.5 scale
 Shape
 showpage
```



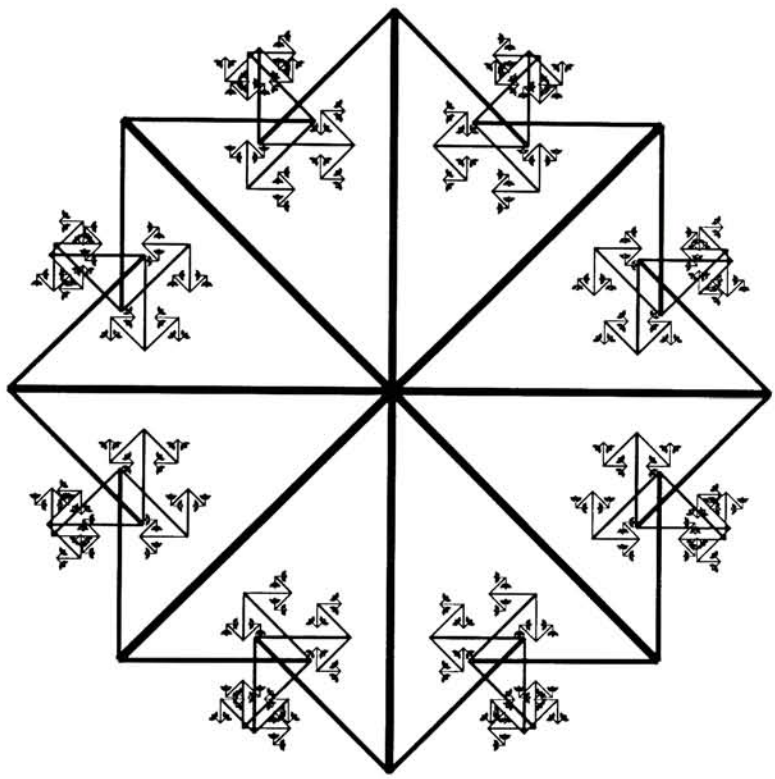
```
% Recursion Program-9

% Define Variables

 /depth 0 def
 /maxdepth 3 def

% Define Procedures

 /DrawPath
     {newpath 0 0 moveto 0 0 50 0 30 arc
      closepath  .5 setgray fill
              0 0 moveto 0 0 40 60 90 arc
      closepath .3 setgray fill
              0 0 moveto 0 0 60 120 150 arc
      closepath .7 setgray fill
              0 0 moveto 0 0 30 180 210 arc
      closepath  .8 setgray fill
              0 0 moveto 0 0 80 240 270 arc
      closepath  .9 setgray fill
              0 0 moveto 0 0 50 300 330 arc
      closepath  .6 setgray fill} def

 /Shape
     {gsave
        /depth depth 1 add def
        DrawPath stroke
        depth maxdepth le
          {.35 .35 scale
            12{30 rotate
            180 0 translate
            Shape
            -180 0 translate
              } repeat
           }if
         /depth depth 1 sub def
      grestore
     } def

% Main Program

 300 400 translate
 3 3 scale
 Shape
 showpage
```



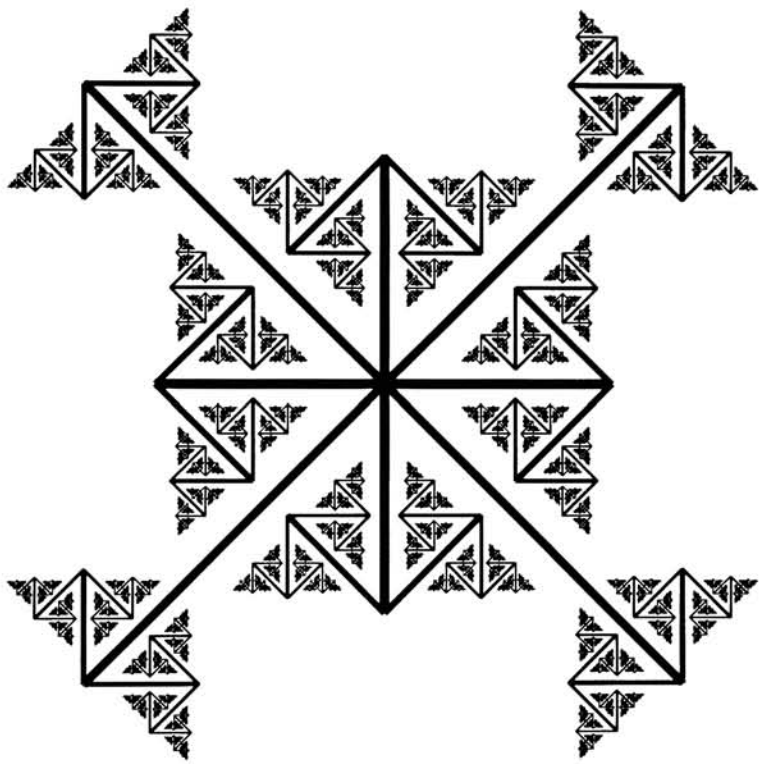
```
%------Variables & Procedures-------
% Fractal-1
/depth 0 def
/maxdepth 10 def
/down{/depth depth 1 add def} def
/up{/depth depth 1 sub def} def

/Doline
{0 288 rlineto currentpoint
 stroke translate 0 0 moveto} def

 /FractArrow-1
 {gsave .5 .5 scale
 6 setlinewidth
 down Doline
 depth maxdepth le
 {135 rotate FractArrow-1
  -270 rotate FractArrow-1}
  if up grestore} def

 /FractArrow-2
 {gsave .5 .5 scale
 6 setlinewidth
 down Doline
 depth maxdepth le
 {135 rotate FractArrow-2
  -270 rotate FractArrow-2}
  if up grestore} def

 /FractArrow-3
 {gsave .5 .5 scale
 6 setlinewidth
 down Doline
 depth maxdepth le
 {135 rotate FractArrow-3
  -270 rotate FractArrow-3}
  if up grestore} def

 /FractArrow-4
 {gsave .5 .5 scale
 6 setlinewidth
 down Doline
 depth maxdepth le
 {135 rotate FractArrow-4
  -270 rotate FractArrow-4}
  if up grestore} def

  % Main Program
  300 400 moveto
  FractArrow-1
  90 rotate
  FractArrow-2
  90 rotate
  FractArrow-3
  90 rotate
  FractArrow-4
  stroke
  showpage
```



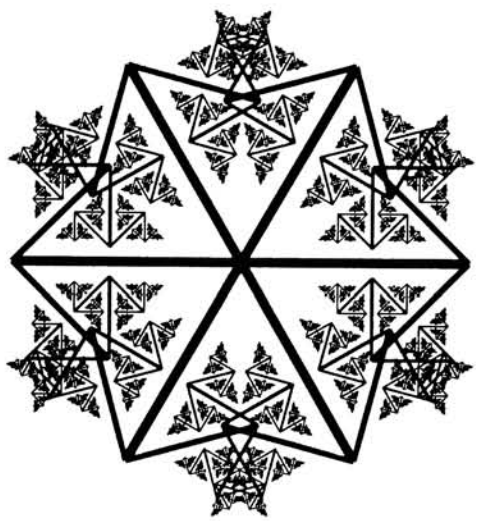
```
%------Variables & Procedures-------
% Fractal-2
/depth 0 def
/maxdepth 10 def
/down{/depth depth 1 add def} def
/up{/depth depth 1 sub def} def

/Doline
{0 288 rlineto currentpoint
 stroke translate 0 0 moveto} def

 /FractArrow-1
 {gsave .5 .5 scale
 6 setlinewidth
 down Doline
 depth maxdepth le
 {135 rotate FractArrow-1
  -270 rotate FractArrow-1}
  if up grestore} def

  % Main Program
  350 400 moveto
  FractArrow-1
  7{45 rotate FractArrow-1}repeat
  stroke
  showpage
```



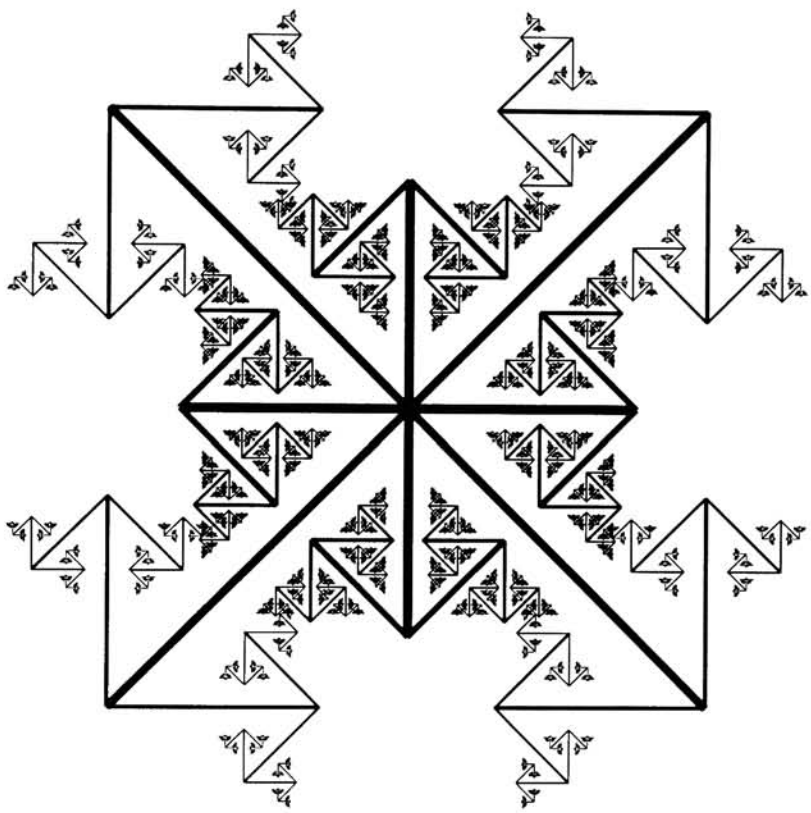
```
%------Variables & Procedures-------
% Fractal-3
/depth 0 def
/maxdepth 10 def
/down{/depth depth 1 add def} def
/up{/depth depth 1 sub def} def

/Doline-1
{0 144 rlineto currentpoint
 stroke translate 0 0 moveto} def

 /FractArrow-1
 {gsave .6 .6 scale
  6 setlinewidth
  down Doline-1
  depth maxdepth le
 {135 rotate FractArrow-1
  -270 rotate FractArrow-1}
  if up grestore} def

  /Doline-2
  {0 320 rlineto currentpoint
   stroke translate 0 0 moveto} def

 /FractArrow-2
 {gsave .5 .5 scale
  6 setlinewidth
  down Doline-2
  depth maxdepth le
 {135 rotate FractArrow-1
  -270 rotate FractArrow-1}
  if up grestore} def

  % Main Program
  300 400 moveto
  FractArrow-1
  3{90 rotate FractArrow-1}repeat
  45 rotate
  FractArrow-2
  3{90 rotate FractArrow-2}repeat
  stroke
  showpage
```



```
%------Variables & Procedures-------
% Fractal-4
/depth 0 def
/maxdepth 10 def
/down{/depth depth 1 add def} def
/up{/depth depth 1 sub def} def


/Doline-1
{0 144 rlineto currentpoint
 stroke translate 0 0 moveto} def

 /FractArrow-1
 {gsave .6 .6 scale
  6 setlinewidth
  down Doline-1
  depth maxdepth le
 {135 rotate FractArrow-1
  -270 rotate FractArrow-1}
  if up grestore} def

  % Main Program
  300 400 moveto
  30 rotate
  FractArrow-1
  5{60 rotate FractArrow-1}repeat
  stroke
  showpage
```



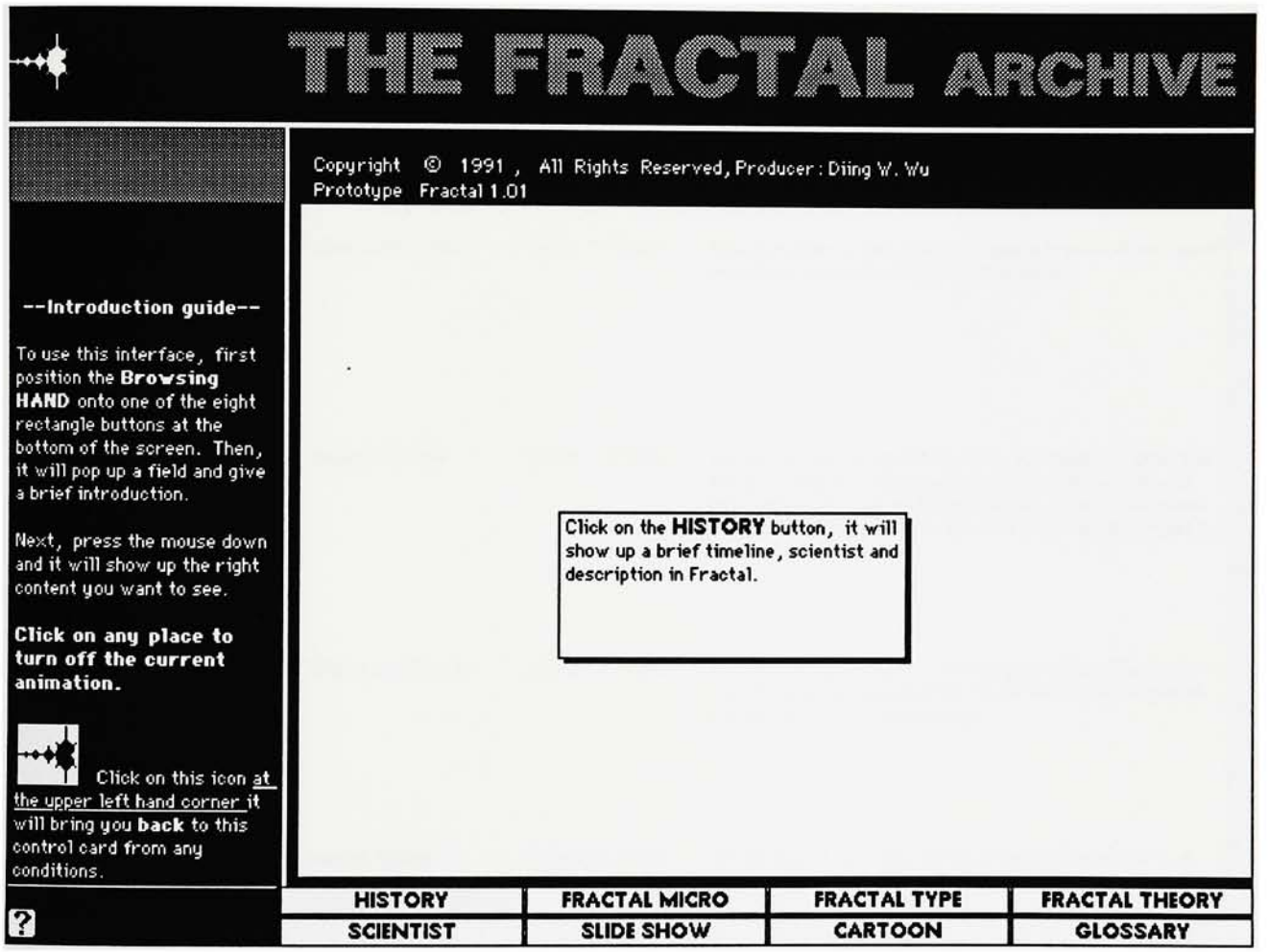
```
%------Variables & Procedures-------
% Fractal-5
/depth 0 def
/maxdepth 10 def
/down{/depth depth 1 add def} def
/up{/depth depth 1 sub def} def

/Doline-1
{0 144 rlineto currentpoint
 stroke translate 0 0 moveto} def

 /FractArrow-1
 {gsave .6 .6 scale
  6 setlinewidth
  down Doline-1
  depth maxdepth le
 {135 rotate FractArrow-1
  -270 rotate FractArrow-1}
  if up grestore} def

  /Doline-2
  {0 320 rlineto currentpoint
   stroke translate 0 0 moveto} def

 /FractArrow-2
 {gsave .5 .5 scale
  6 setlinewidth
  down Doline-2
  depth maxdepth le
 {135 rotate FractArrow-2
  -270 rotate FractArrow-2}
  if up grestore} def

  % Main Program
  300 400 moveto
  FractArrow-1
  3{90 rotate FractArrow-1}repeat
  45 rotate
  FractArrow-2
  3{90 rotate FractArrow-2}repeat
  stroke
  showpage
```



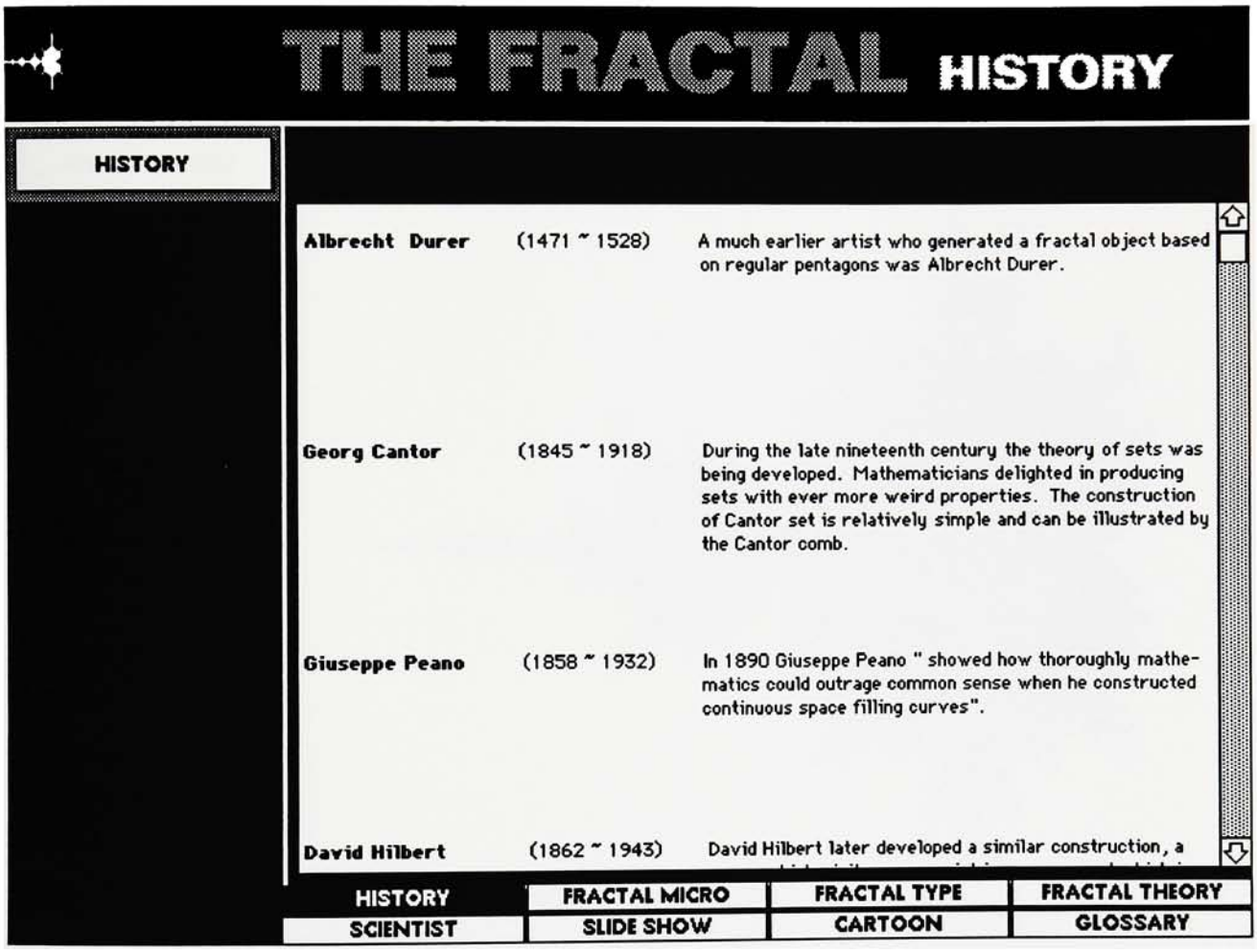
Appendix: D

User Interface Screen Design

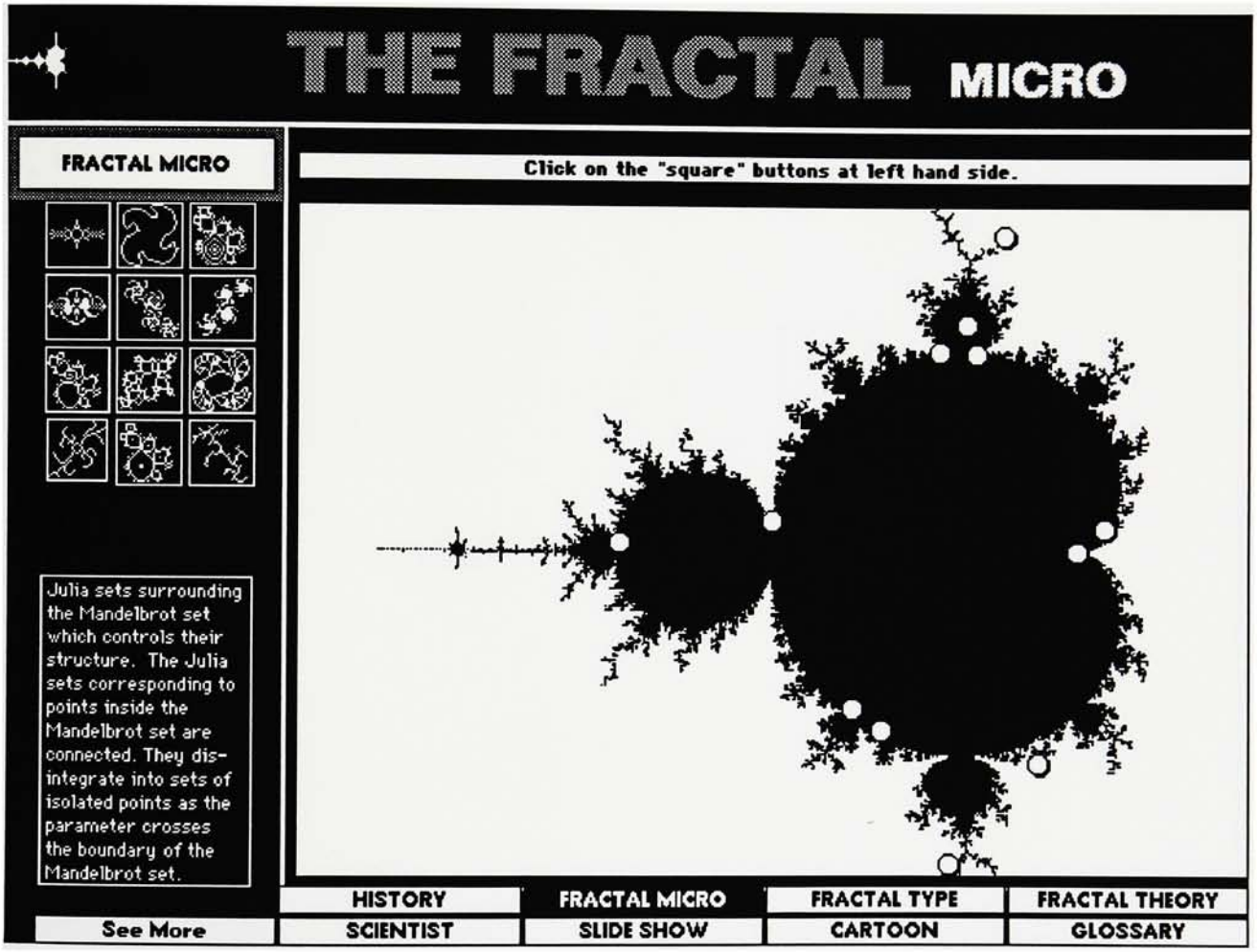
User Interface Screen Design



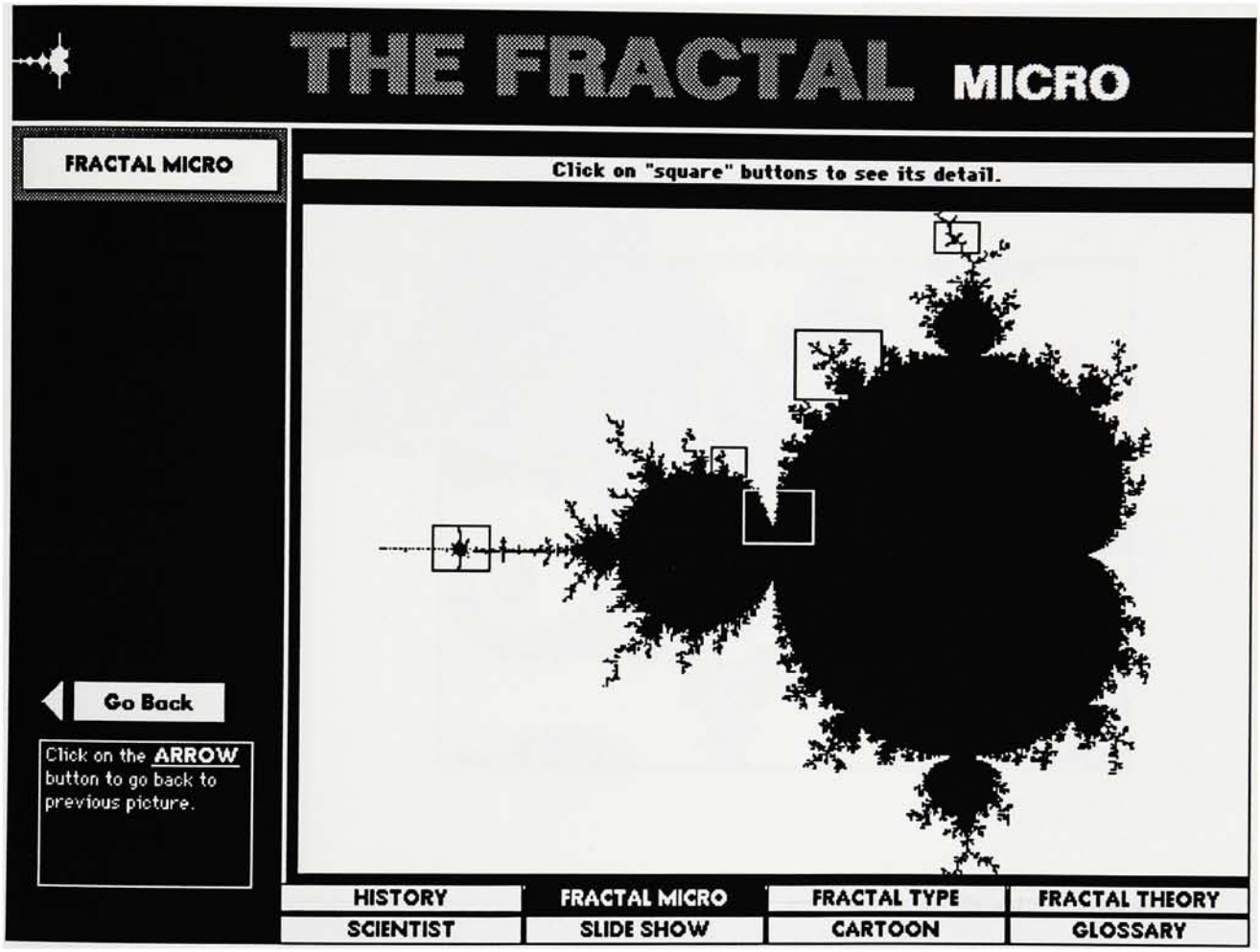
User Interface Screen Design



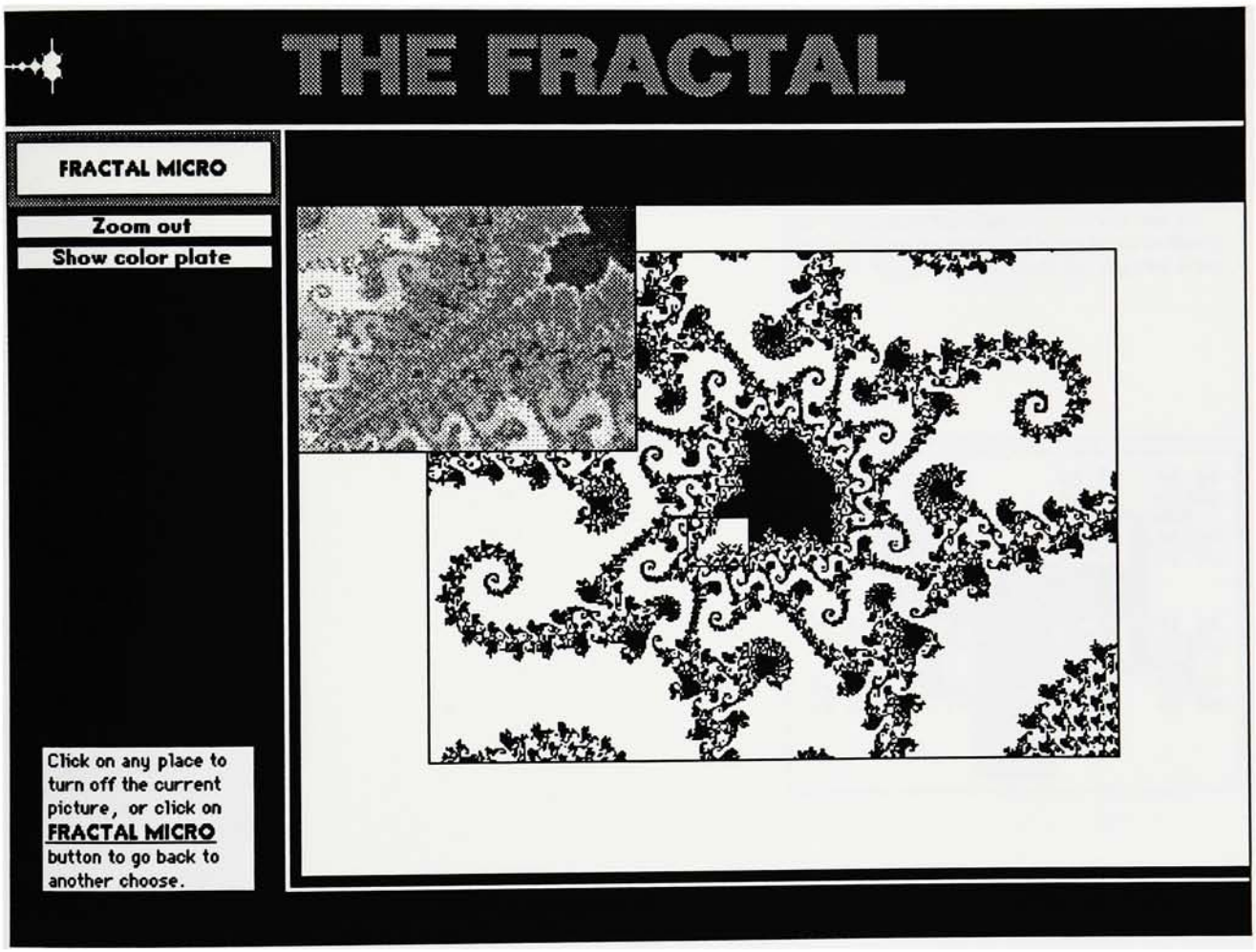
User Interface Screen Design



User Interface Screen Design



User Interface Screen Design



User Interface Screen Design

THE FRACTAL TYPE

FRACTAL TYPE

- Mandelbrot Sets
- Julia Sets
- Bifurcation Diagrams
- Hilbert Curves
- von Koch Curves
- Peano Curves
- Sierpinski Curves
- Dragon Curves
- Phoenix Curves
- Trees

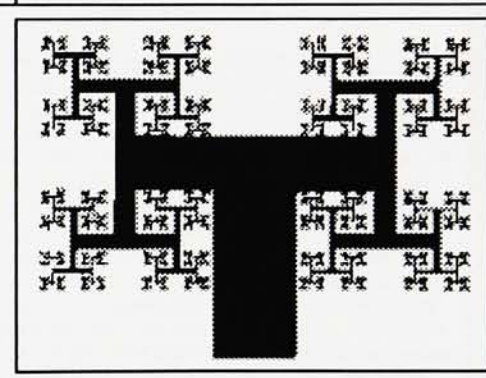
Click on **FRACTAL TYPE** list window (with the mouse still held down) move the **HAND** until the selection you desire is highlighted in black. Then, release the mouse, and the image relating to your selection will appear.

Trees

- Real Trees
- Mathematical Trees
- Bare Tree
- Tree with Foliage
- One-Sided Tree
- Bronchial System Tree
- Arterial System Tree
- 90 Degree Branch Tree
- 85 Degree Branch Tree
- 90 D Tree and Wider Stem

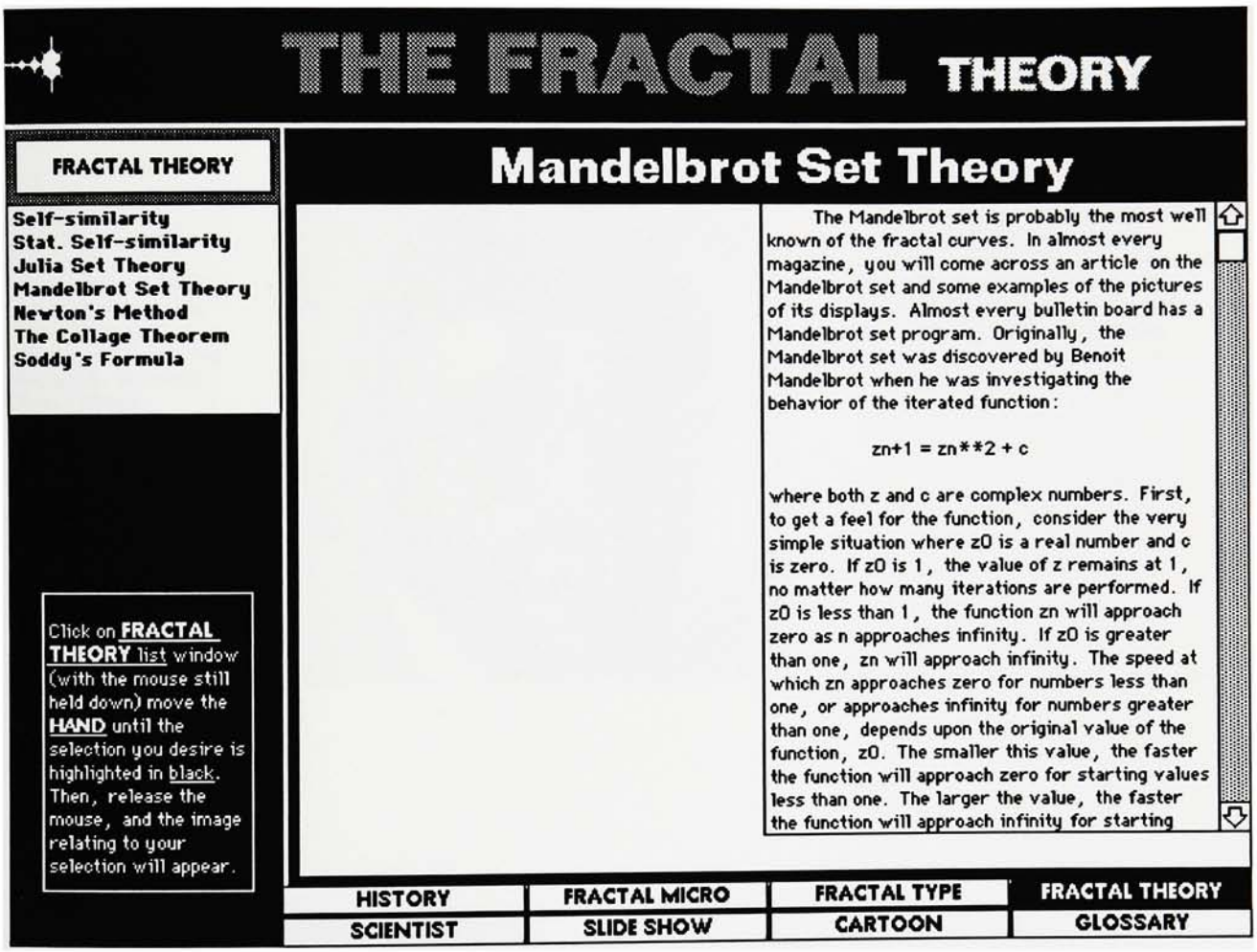
90 D Tree and Wider Stem

The figure show the interesting curves that are obtained when the branching angle is set to ninety degrees. They don't look much like any real trees.

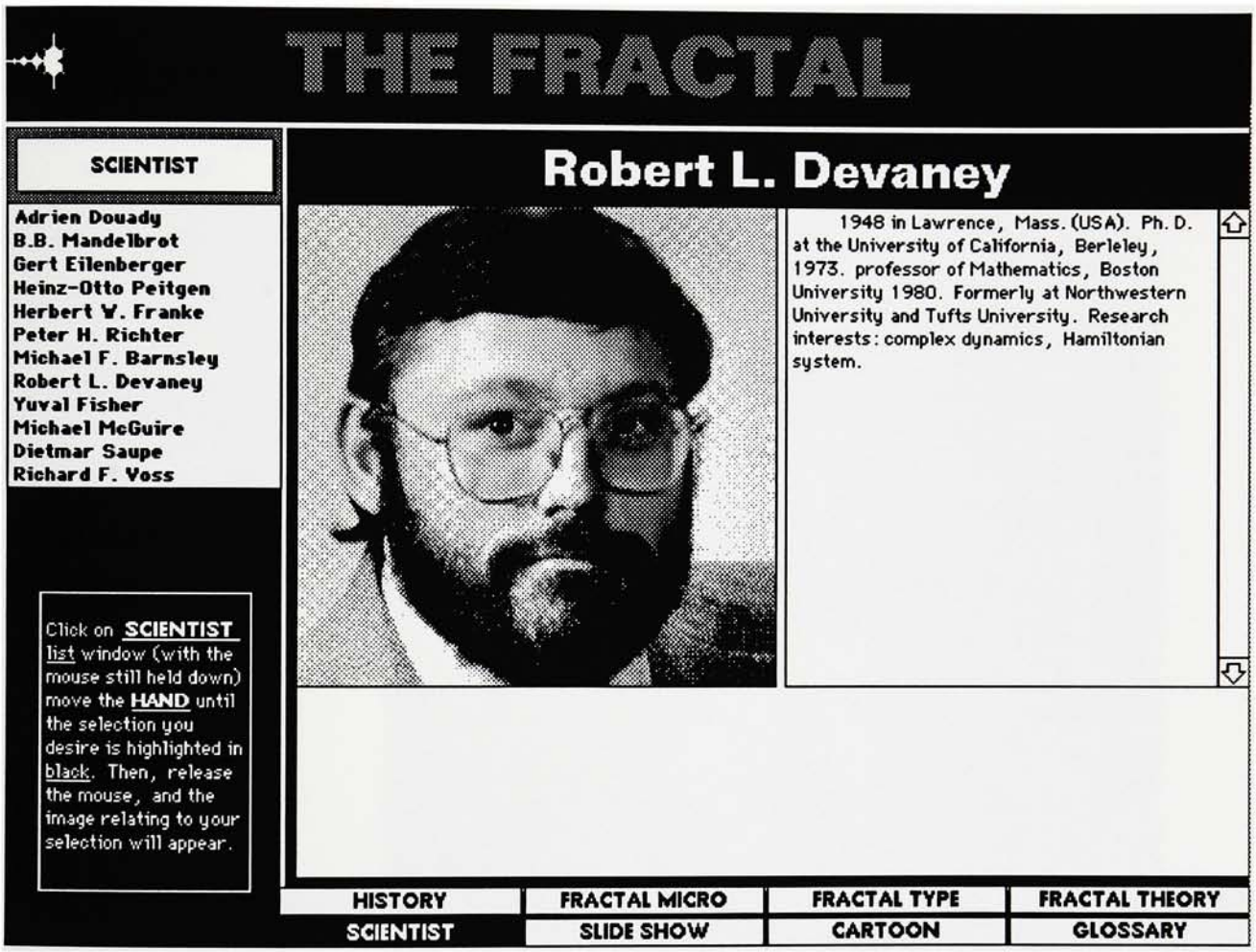


HISTORY	FRACTAL MICRO	FRACTAL TYPE	FRACTAL THEORY
SCIENTIST	SLIDE SHOW	CARTOON	GLOSSARY

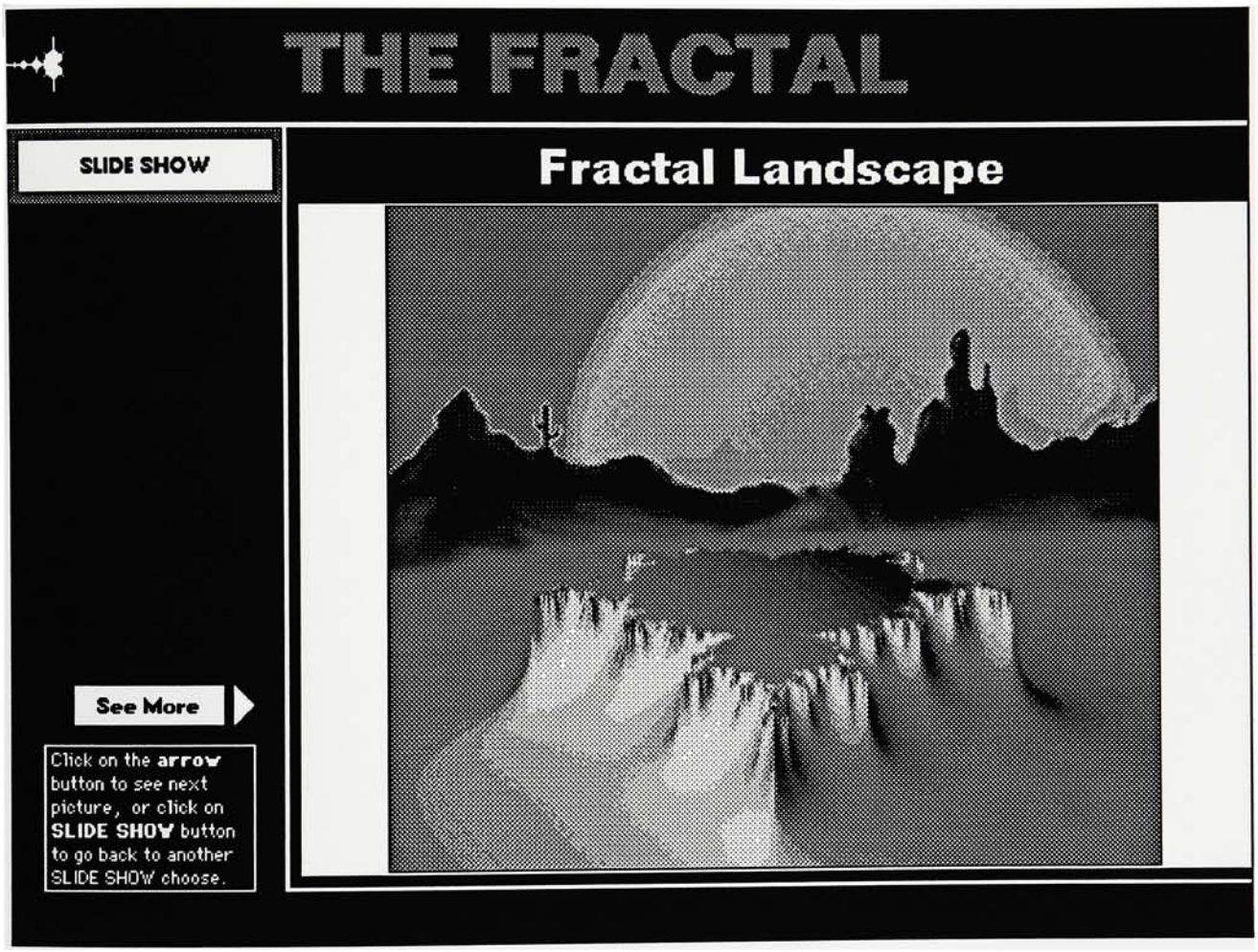
User Interface Screen Design



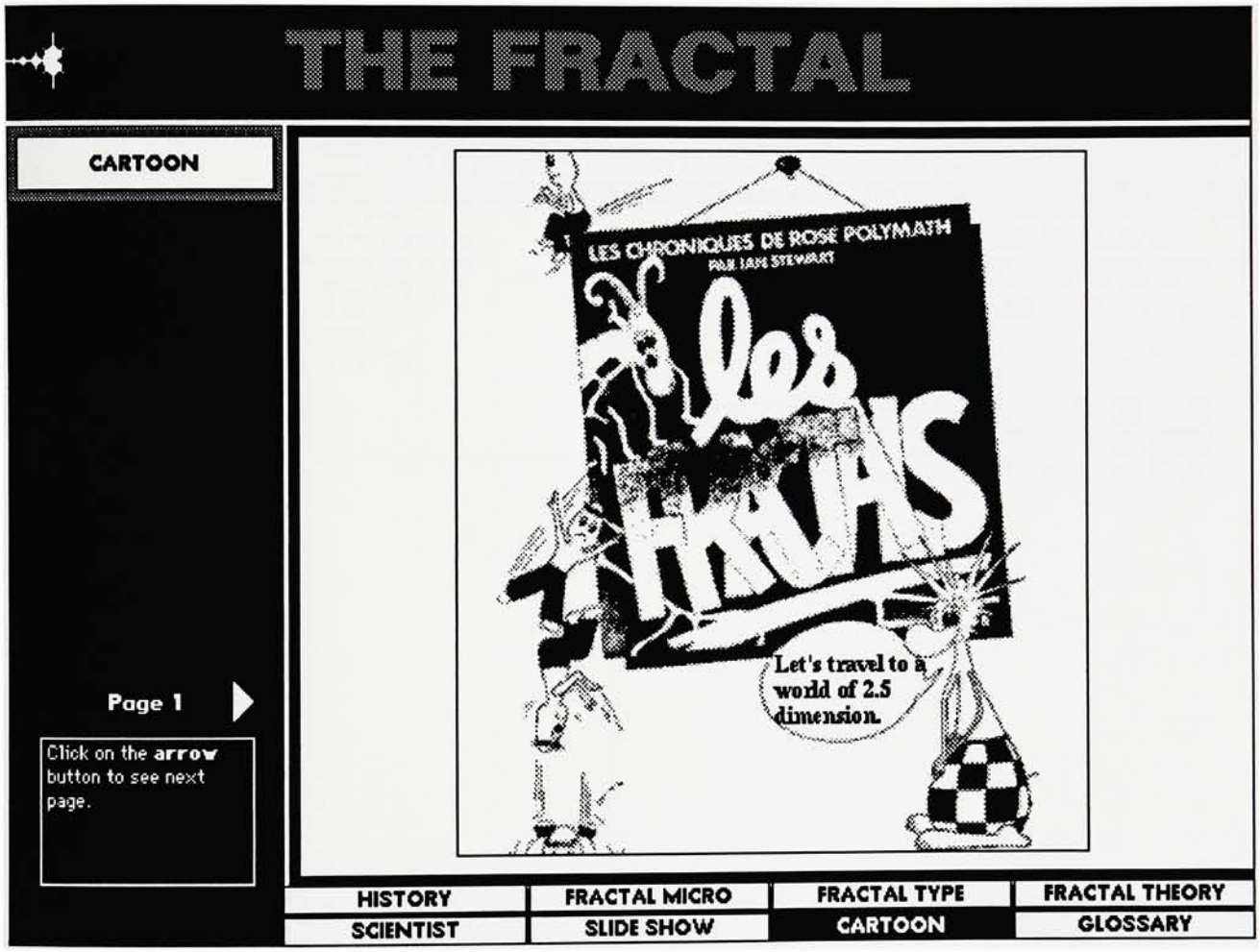
User Interface Screen Design



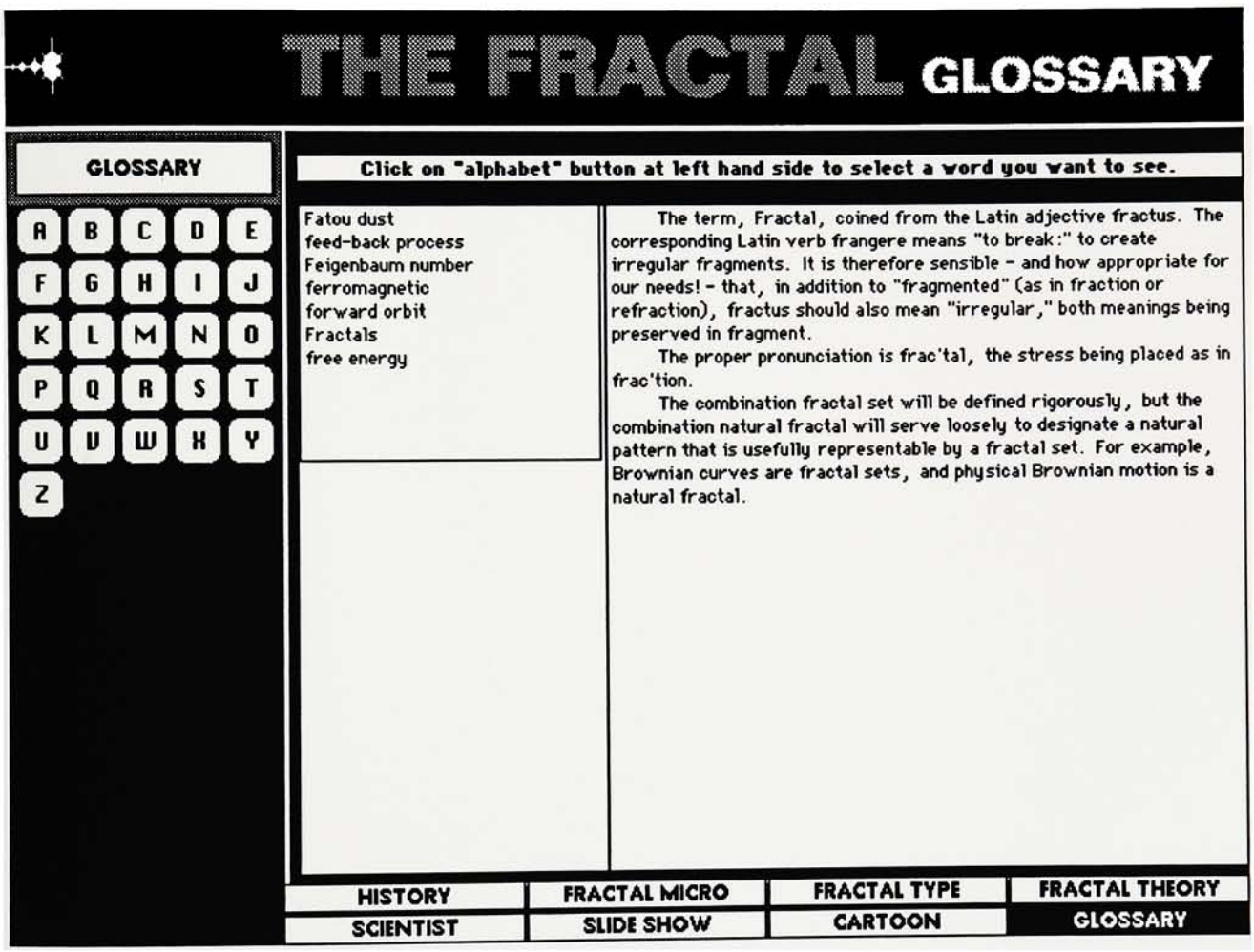
User Interface Screen Design



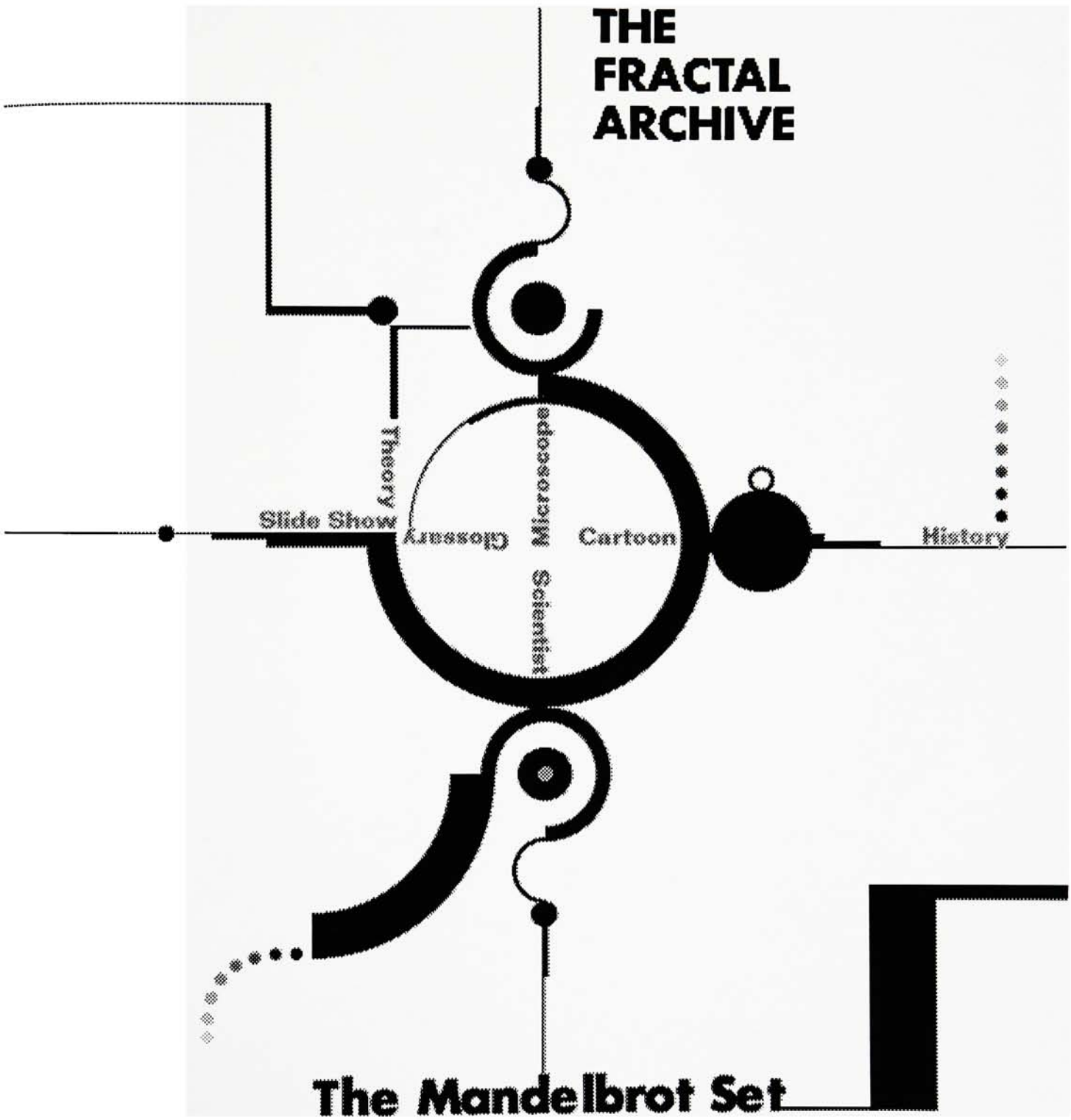
User Interface Screen Design

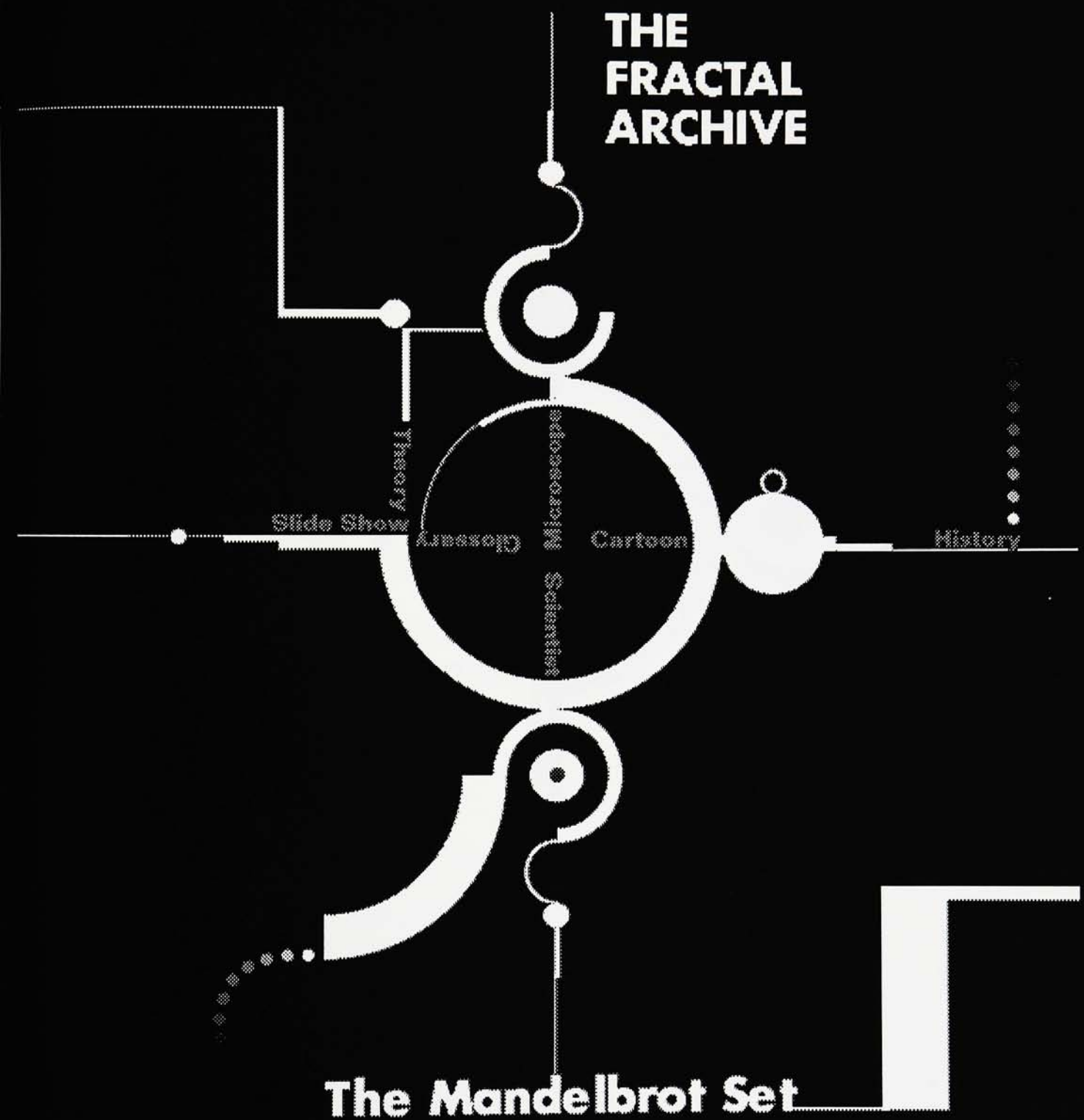


User Interface Screen Design

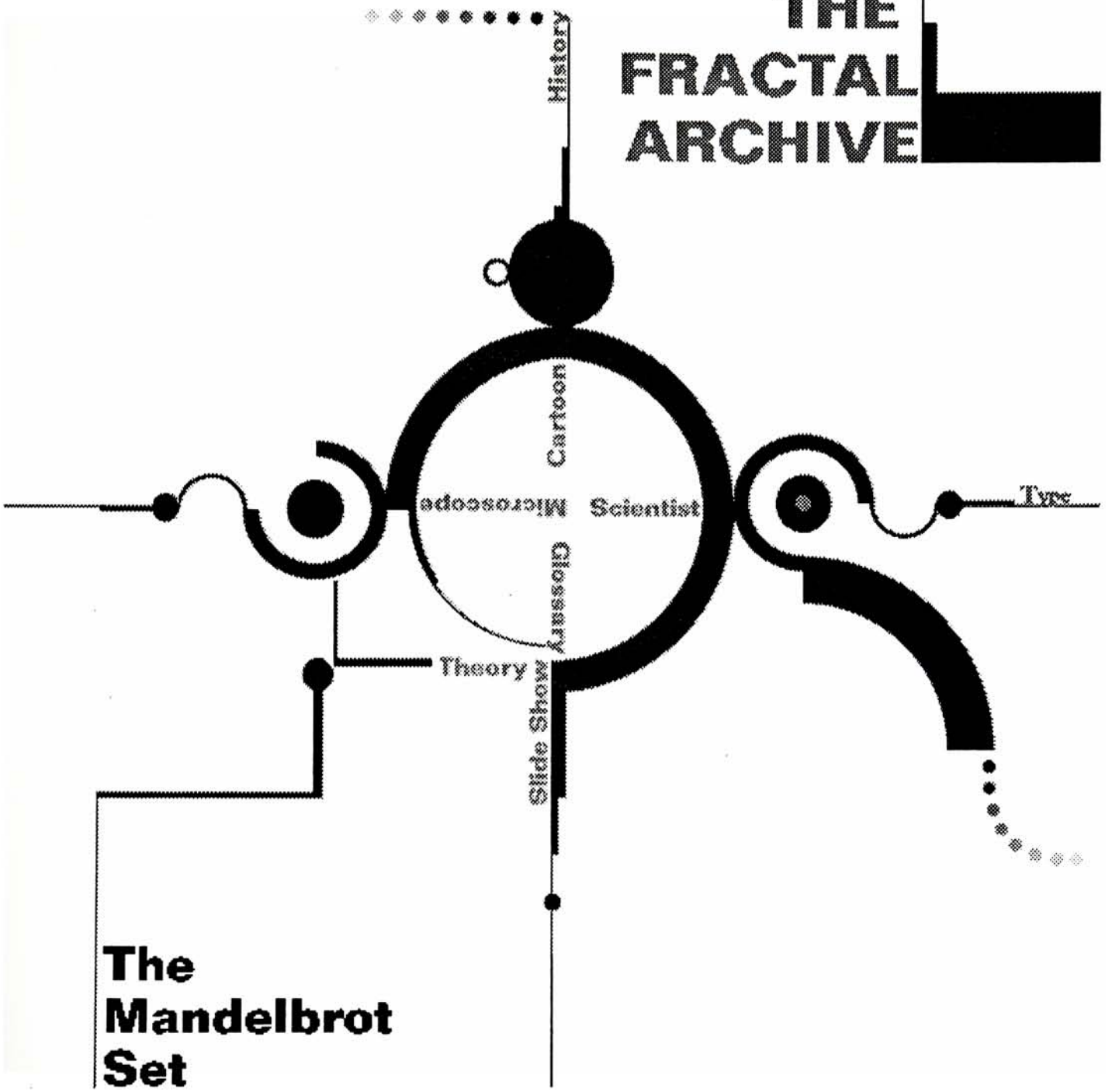


Poster Design for The Fractal Archive

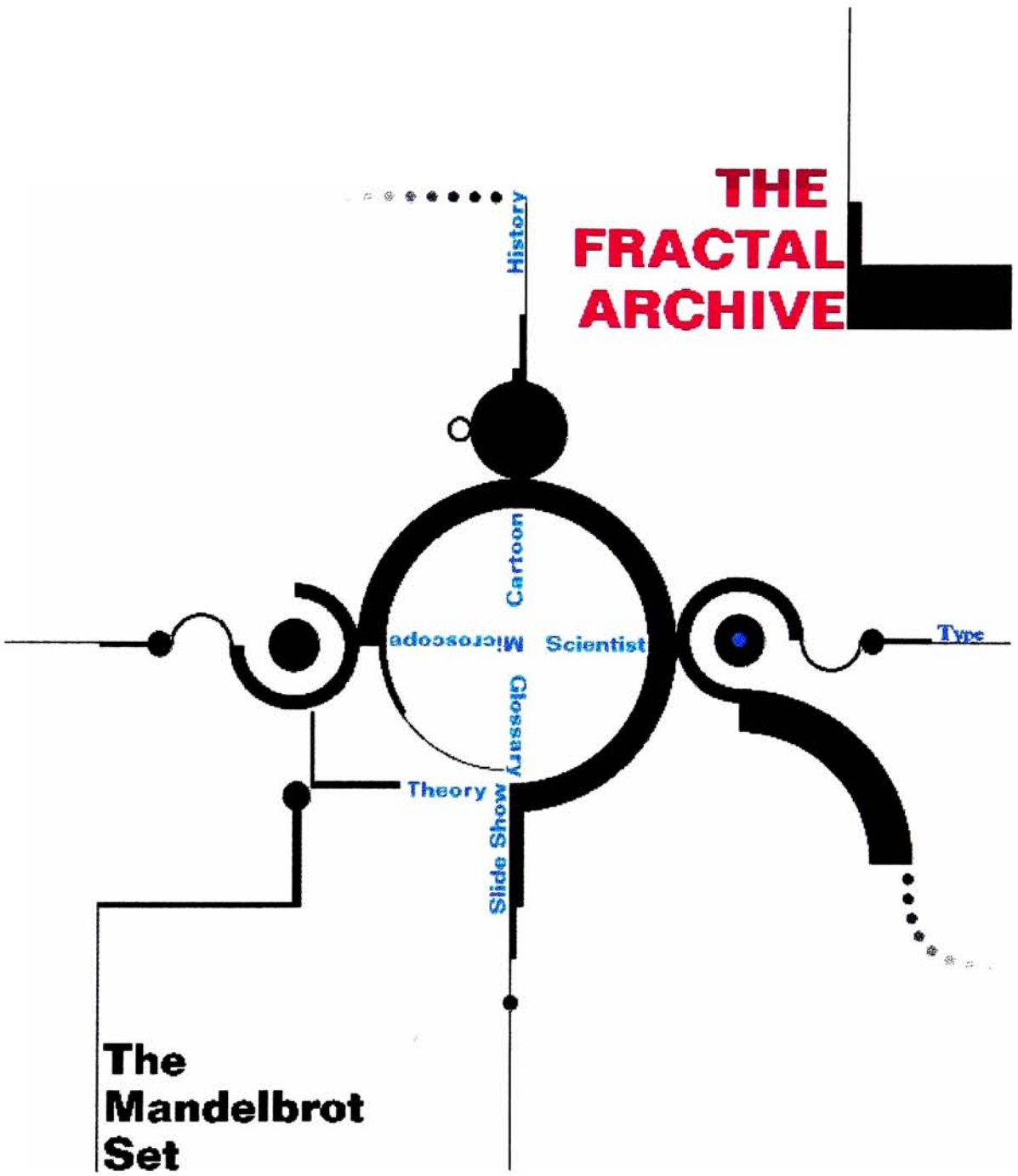

THE
FRACTAL
ARCHIVE
Theory
Slide Show
Glossary
Microscopic
Cartoon
History
Scientist
The Mandelbrot Set


THE
FRACTAL
ARCHIVE
Theory
Slide Show
Anaesop
Mirrorworld
Cartoon
History
The Mandelbrot Set

THE FRACTAL ARCHIVE

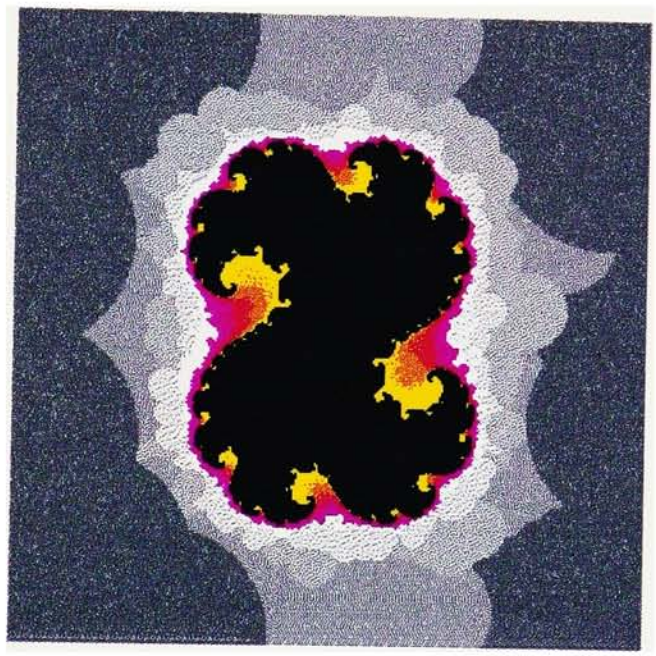


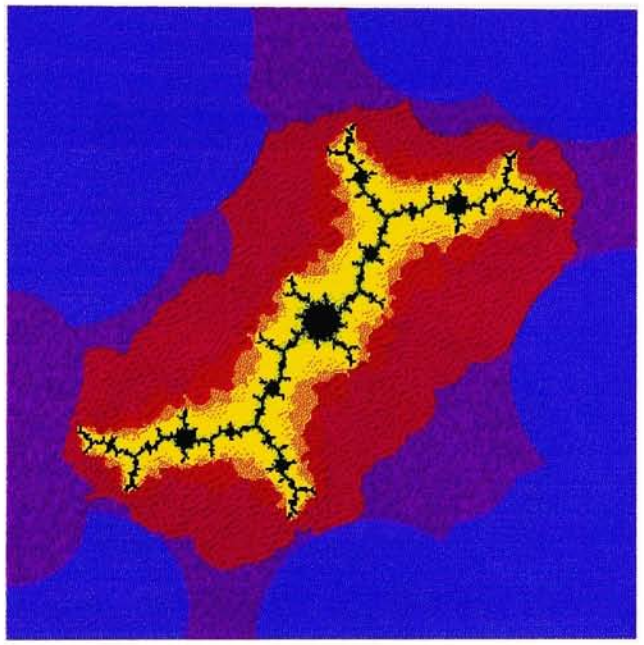
The Mandelbrot Set

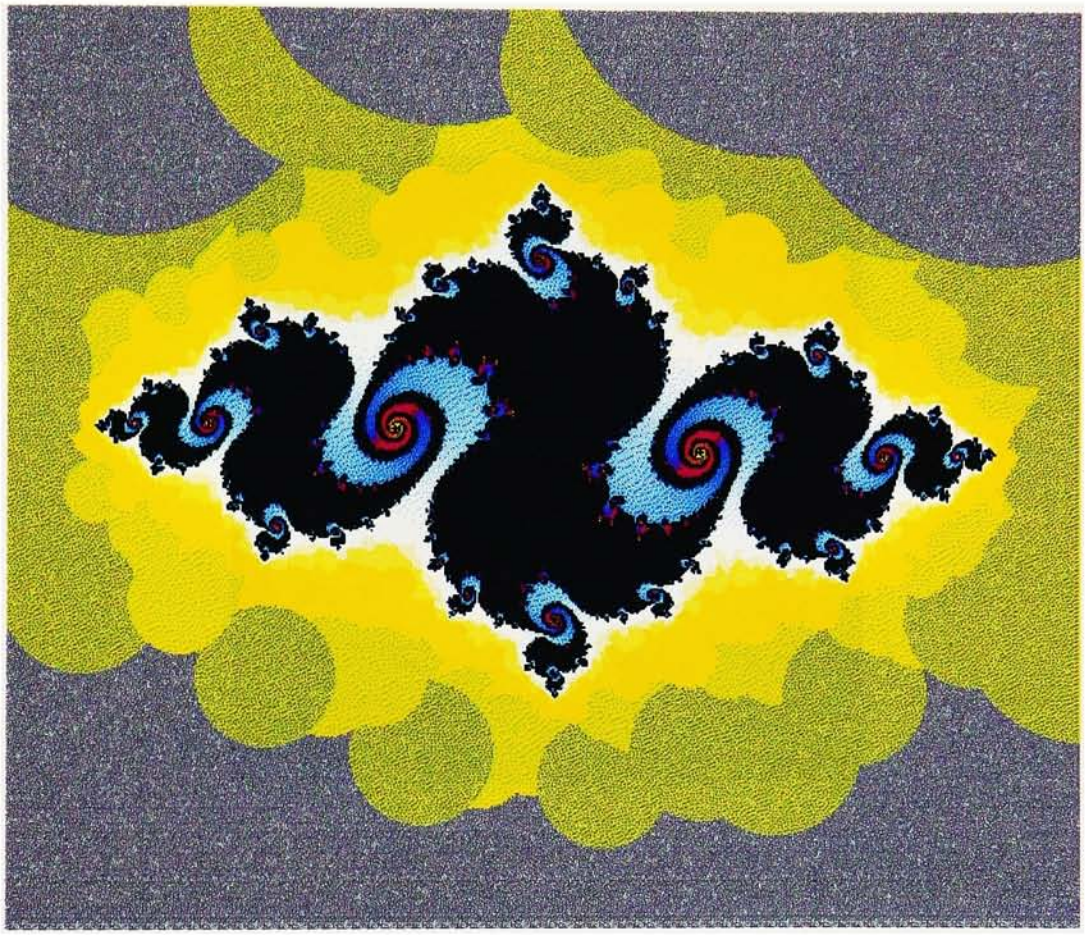

THE
FRACTAL
ARCHIVE
History
Cartoon
Microscope
Scientist
Type
Glossary
Theory
Slide Show
The
Mandelbrot
Set

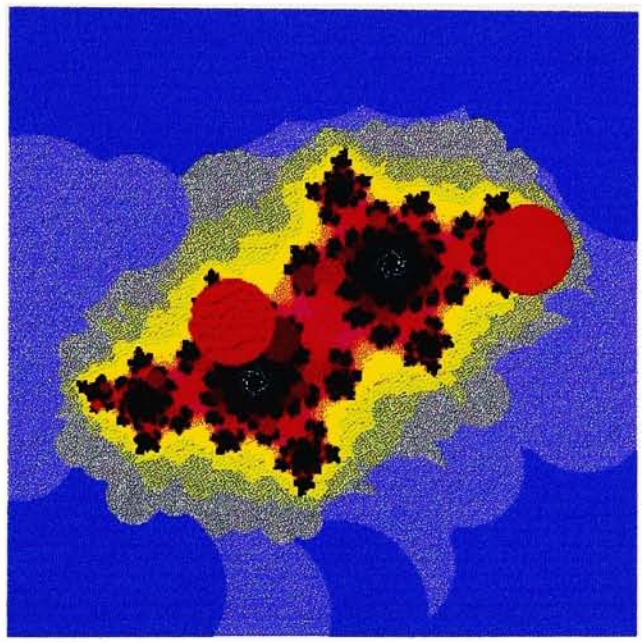
Appendix: F

The Color Plate for The Julia Sets

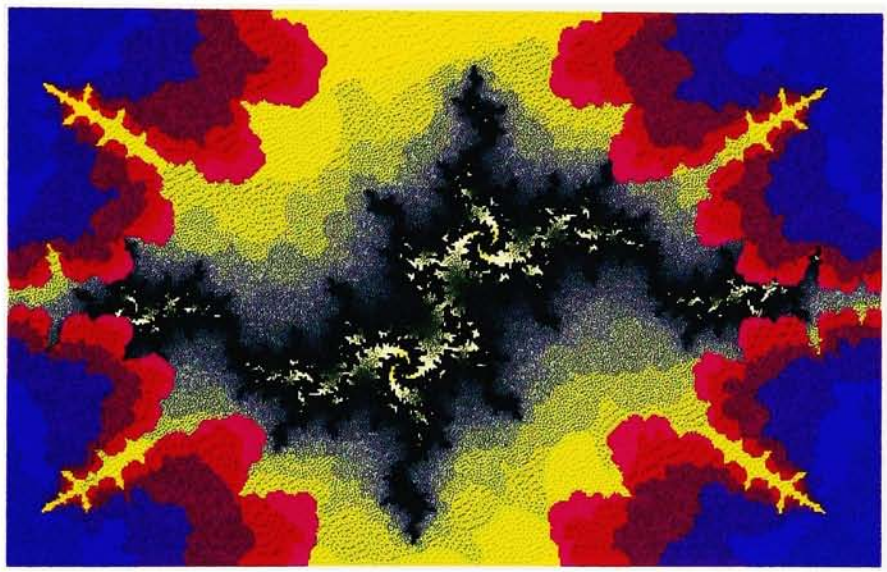








The Other Applications to The Fractal Archive



Appendix: H

Human Interface Design

CHAPTER 9

Human Interface Design: Ten General Principles

USERS EXPECT EVERY MACINTOSH APPLICATION TO BE USER-CENTERED, SIMPLE, and easy to learn. Your stack is no exception. This chapter briefly outlines the ten general design principles presented in the book *Human Interface Guidelines: The Apple Desktop Interface*, published by Addison-Wesley.

About users

The Human Interface Design Principles are based on some assumptions about people. A good interface allows people to accomplish tasks. Tasks will vary, but people share some common characteristics.

People are instinctively curious; they want to learn, and they learn best by active self-directed exploration of their environment. People strive to master their environment; they like to have a sense of control over what they are doing, to see and understand the results of their own actions. People are skilled at manipulating symbolic representations; they love to communicate in verbal, visual, and gestural languages. Finally, people are most productive and effective when the environment in which they work and play is enjoyable and challenging.

General design principles

This section describes the ten fundamental Human Interface Design Principles and discusses how each applies to designing stacks. Briefly, these principles involve

- use of metaphors
- direct manipulation
- see-and-point (instead of remember-and-type)
- consistency
- WYSIWYG (what you see is what you get)
- user control
- feedback and dialog
- forgiveness
- perceived stability
- aesthetic integrity

Metaphors from the real world

- Use concrete metaphors and make them plain, so that users have a set of expectations to apply to computer environments.
- Whenever appropriate, use audio and visual effects that support the metaphor.

People have more experience with the real world than they do with computers. To take advantage of their experience, use metaphors in your stacks that correspond to the everyday world.

HyperCard is already based on a real-world metaphor, the "card." People are familiar with using cards to organize information. The card metaphor allows users to make some important assumptions about how HyperCard works: users assume that cards can be grouped together into "stacks," that they can have both text and pictures on them, and that they can be changed or updated.

If you decide to use a new metaphor in your stack, think about how the new metaphor will affect users' expectations. For instance, a book metaphor would imply that information is presented in a linear format, that travel is limited to "forward," "backward," and "turn-to-a-given-page," and that it's possible to see all pages by simply going forward until the end.

Before you select a metaphor for your stack, make sure the content of the stack lends itself to the metaphor. Real-world metaphors tend to help users understand how to use a stack, but it's better to have no metaphor at all than to force your content into an inappropriate one.

Direct manipulation

- Users want to feel that they are in charge of the computer's activitites.
- Tell users their options by providing visible choices, ways to make their choices, and feedback acknowledging their choices.

This principle is based on the assumption that people learn best by active, self-directed exploration. People expect their physical actions to have physical results, and they want their tools to provide feedback. This feedback can be provided visually, audibly, or both.

Highlight topics of interest. Show the user what options are available. If an option is normally available, but not in a specific case, convey that information by providing a "grayed-out" version of it. If grave consequences will follow from choosing an option, warn the user before any damage is done. If a particular command is being carried out, provide visual clues. If the command can't be carried out, tell the users why it can't be carried out. Also tell them what they can do instead.

See-and-point (instead of remember-and-type)

- Users select actions from alternatives presented on the screen.
- Users rely on recognition, not recall; they shouldn't have to remember anything the computer already knows.
- Most programmers have no trouble working with interfaces that require memorization. The average user is not a programmer.

Stacks are visually and spatially oriented. The way everything appears—text, graphics, buttons, options—should be consistent and well thought out. Users should be able to anticipate what will happen when they interact with your stack by choosing objects, activities, and options.

Don't force users to remember the possible destinations and ways of getting around your stack; keep those options present on the screen, and make their use clear. Most stacks will have two kinds of see-and-point navigation options on the screen: those that are available at all times, such as Help, Return to Start, or Quit HyperCard, and those that are card specific.

There can be advantages—such as speed—to the "remember-and-type" approach. If you decide to offer keystroke alternatives, offer them in addition to, not in place of, the on-screen methods. Users who are new to your stack or who are looking for potential actions in a confused moment, must always be able to find a desired option on the screen.

Just as the average user is not a programmer, the average user is not a HyperCard power user. Don't rely on the user's knowledge of keyboard shortcuts to navigate. In fact, don't rely on the user's knowledge of stacks or HyperCard at all. Set up an environment, teach the user about it, and provide see-and-point ways to use and navigate through it.

istency

Effective applications are both consistent within themselves and consistent with one another.

Consistency within a stack is essential. The look, the usage, and the stack behavior should be the same throughout. The way the user does things should always be consistent within a stack. For example, your stack should have a consistent design for these elements:

- graphic look
- grouping of buttons
- placement of buttons
- visual and audio feedback
- card layout
- background for cards with similar functions
- stack structure

Consistency in these elements makes it easier for the user to focus on the content of the stack.

If you plan to use any of the standard elements of the Apple Desktop Interface in your stack (such as menus, dialog boxes, and so forth) follow the guidelines presented in *Human Interface Guidelines: The Apple Desktop Interface*.

YSIWYG (what you see what you get)

- There should be no secrets from the user, no abstract commands that only promise future results.
- There should be no significant difference between what the user sees on the screen and what eventually gets printed.

The WYSIWYG principle has special significance in stack modeling and navigation. The layout of your stack should not, except in special cases, be a secret to your user. Part of "What you see is what you get" is letting the users know what they're seeing, and how it relates to the whole stack.

If you provide a representation of your stack, such as a stack map, table of contents, or menu, that representation should contain an accurate and complete model. Nothing frustrates a user more than finding a part of the stack that's not on the stack map, or discovering that the stack's true structure isn't anything like what the menu implied. Make coherent models and communicate them. Let the users know where they are in relation to the whole. Provide a map, but also provide "You-are-here" indicators, or names for the individual screens.

User control

- The user, not the computer, initiates and controls all actions.

People learn best when they're actively engaged. Too often, the computer acts and the user merely reacts. Or, the computer "takes care" of the user, offering only those alternatives that are judged "good" for the user or "protect" the user from detailed deliberations.

This protective approach may seem appealing, but it puts the computer, not the user, in the driver role. In most cases, it's better to let the user try risky things. You can provide warnings, but let the action proceed if the user confirms that this action is indeed desired. This approach protects the beginner but allows the user to remain in control.

Get your user doing something quickly. Good stacks are interactive. Many stacks begin with an "attract mode," where the screen is alive with inviting animation, rich graphics, and the words "Click to begin."

Let the user choose what happens next, both in using the stack and in navigating around it. This is especially important when offering long animation or sound sequences.

Suppose you wanted your stack to provide a slide show with accompanying music. A frustrating implementation, giving the user no control, would start the slide show and music the instant the stack opened, and run for several (possibly loud) minutes until done. An implementation that gives the user more control might open on a screen that indicates the length of the slide show, asks the user to set the volume level or turn off sound, provides a button called "Start slide show" and displays an unobtrusive sentence, saying "Click any time to interrupt."

Feedback and dialog

- Keep the user informed.
- Provide immediate feedback.
- Make user activities simple at any moment, though they may be complex taken together.

To be in charge, the user must be informed. When, for example, the user initiates an operation, your stack should provide immediate feedback to confirm that the operation is being carried out, and (eventually) that it's finished.

Immediate feedback can be provided by buttons that become highlighted, click, beep, or display a visual effect. For time-consuming operations, feedback can be provided by temporarily changing the cursor into a watch or beach ball or by displaying a message that explains the reason for the delay.

If an operation can't be completed, tell the user why it can't be completed. This communication should be brief, direct, and expressed in the user's vocabulary, not the stack designer's or the programmer's.

Forgiveness

- Users make mistakes; forgive them.
- The user's actions are generally reversible—let the users know about any that aren't.
- Users get lost in stacks; help them find their way.

Most users don't like to read manuals. They would rather figure out how something works by exploration, with lots of action and lots of feedback.

As a result, users sometimes make mistakes or explore further than they really wanted to. Forgiveness means letting users do anything reasonable, letting them know they won't break anything, always warning them when they're entering risky territory, then allowing them either to back away gracefully or plunge ahead, knowing the consequences.

When options are presented clearly, with appropriate and timely feedback, alert messages should be infrequent. If the user receives a barrage of alert messages, gets lost frequently, or can't figure out how to use the stack, something is wrong with the stack's design.

Perceived stability

- Users feel comfortable in a computer environment that remains understandable and familiar rather than changing randomly.

People use computers because computers are versatile and fast. Computers can calculate, revise, display, and record information far faster than people can. If users are to cope with the complexity a computer handles so easily, they need some stable reference points.

These stable reference points are established by how your stack looks, how it acts, and how it feels. You are setting up an implicit contract with your user about the rules of this particular environment, and those rules should be clear and communicated.

Most important, your stack should provide conceptual stability. Give your user a consistent model for how to perceive the stack's function and structure. Note the emphasis on "perceived"; a user may *perceive* your stack to have a single-frame, tree, or network structure, even though in *fact* all stacks are linear sequences of cards, with different navigational control structures superimposed. Provide a clear, finite set of options, and tell the user what they are.

Your stacks should also provide visual stability. Provide a constant overall look and graphic design for your stack. Design the card layout to be constant for similar cards and visually related for all cards in the stack. Place your buttons in reliable and functionally grouped locations. Use a consistent button design; If you're using the same button on several cards, don't represent the button by an icon on one card and a text label on another.

The illusion of stability is what's important. The environment can and should change as users interact with it, but should give users a number of familiar landmarks to rely upon.

Aesthetic integrity

- Visually confusing or unattractive displays detract from the effectiveness of human–computer interactions.
- Different "things" should look different on the screen.
- Messes are acceptable only if the user makes them—stacks aren't allowed this freedom.

In traditional computer applications, the visual appearance of the screen has been a low priority and consequently somewhat arbitrary. In contrast, HyperCard stacks *depend* upon the visual appearance of the screen. As much as possible, commands, features, parameters, choices, navigational options, and data should appear as graphic objects on the screen.

People deserve and appreciate attractive surroundings. Consistent visual and audible communication is very powerful in delivering complex messages and opportunities simply, subtly, and directly.

Summary

These ten general design principles form a powerful basis for designing and evaluating your stacks. These principles provide general guidance. Most people don't have extensive backgrounds in user interface design; following these ten principles is a simple way to make your stacks more usable. A single principle, such as that of user control, can guide many decisions, from giving users buttons with which to control their navigation to giving them volume controls with which to turn sound up, down, or off.

If you plan to use elements from the standard Macintosh desktop interface, get the book *Human Interface Guidelines: The Apple Desktop Interface*, published by Addison-Wesley. In addition to discussing these design principles, this book specifies in detail how elements such as a Macintosh window, dialog box, or pull-down menu should act.