Helix

Helix is a graphics demonstration program by Christopher Tate

This documentation describes version 1.1 of Helix.

Printing This Document

This document uses the Helvetica and Courier fonts. When printing on a LaserWriter, select “Font Substitution” for best results. The documentation was prepared using Microsoft Word, and optimized for the LaserWriter. I apologize for the size of this document; several people expressed the opinion that being thorough was more desirable that being concise.

The Freeware Stuff

Helix was written in 1988-89 by Christopher Tate, using Turbo Pascal 1.1 at the Pennsylvania State University. In late 1989 and early 1990 is was rewritten in THINK C 3.02, then 4.0. It may be used and distributed freely, but may not be sold or placed under any organization's anthology copyright without the author's express written permission. Also, you may not alter this program or its documentation in any way without the express written permission of the author. In short, you may pass copies of this program along to someone else at will, or to upload it for general distribution, but you may not under any circumstances charge money for doing so, nor may you restrict the rights of the recipients to distribute this program under the terms stated here. You must include a copy of this documentation with any copy of the program that you distribute.

What Helix Does

Helix generates its pictures by independently traversing two Lissajous figures. The program keeps track of the current position along each figure, and draws lines connecting the two figures each time it takes a step along their paths. This process generates an intricate ribbon-like pattern which eventually returns to its starting point. The panels in the "About Helix..." dialog give a brief description of the effect of the program's various parameters on the finished picture. The algorithm looks roughly like this (in Pascal):

```
x1 := Radius1 * cos(Angle * Factor1);
y1 := Radius2 * sin(Angle * Factor2);
x2 := Radius2 * cos(Angle * Factor3);
y2 := Radius1 * sin(Angle * Factor4);
LineTo(x1, y1);
LineTo(x2, y2);
Angle := Angle + Increment;
```

Helix can print its pictures to an ImageWriter or a LaserWriter, and theoretically to other hardcopy devices which use the standard printing interface. The program automatically scales the screen image to be nearly full-page size when printed, regardless of the actual Radius settings. In addition, Helix prints using the best available device resolution. This results in exceptional line definition and clarity.

Helix also includes the capability to save pictures, either in its own file format or as EPSF files. Images saved normally take up only one block of disk space each, since they are only 22 bytes long.

Helix has been tested on at least one Mac Plus, Mac SE, Mac II, and Mac SE/30. The printing code has been tested on a LaserWriter IINT, an ImageWriter II, and an ImageWriter LQ. When Helix was ported to THINK C 4.0, the printing code was modified slightly. The practical upshot is that Helix must be run under System 4.1 or later.**Update - Version 1.1**

Here are the changes since version 1.0:

New Features:

Helix now keeps track of intermediate results in its calculations, so that if you move something on top of the window, it does not have to recalculate any lines already drawn. The practical upshot of this is that screen updates are significantly faster. However, some people expressed to me the opinion that watching it draw at the original (slower) speed, and so consequenly there is a new item in the File menu, called “Fast Refresh.” Unchecking this item causes Helix to slow down its refresh rate, so that you can watch the picture redraw in slow motion (especially on a Mac Plus...).

Also, Helix now offers the option to save the image in an EPSF (Encapsulated PostScript Format) file. For the uninitiated, this is a standard file format consisting of the picture itself, saved in PICT format, coupled with the actual PostScript code to draw the picture. When incorporated into a program that can handle EPSF files (most page-layout applications, for example), this allows the application to use the PostScript instead of the PICT when printing to a LaserWriter or similar PostScript output device. You can get all sorts of effects using this technique, such as stretching the images, or mixing them with text, and still retain the file line definition, because of the use of direct PostScript in the printing process. The option for this is found under the File menu.

The calculation code was also tweaked a bit to make it a touch faster, and the library organization was changed to reduce the size of the finished product. (Yes, it was *sharply* reduced. However, when I added the EPSF file support, I found myself needing to use THINK C’s printf() function with floating-point arguments. This means that I had to include the full ANSI library into Helix, and get a *lot* of unnecessary stuff pulled in with it....)

The MultiFinder partition has been enlarged to 80K, to give Helix room to retain its intermediate calculation results.

Bug Fixes:

You may no longer specify a zero angle increment or factor. This used to crash the machine.

Helix now filters out floating-point numbers entered in any of the dialogs, and ignores any digits after the decimal place. All numbers are required to be integers, so Helix ignores anything after a decimal point.

A cosmetic bug which caused the Tab key to cycle through the Factor entry fields in reverse order (bottom-to-top) has been fixed

How to Use Helix

Here's where I tell you about how the parameters determine what the pictures look like.

Radii:
These are two values, Radius 1 and Radius 2, which determine (approximately) the relative sizes of the two Lissajous figures. On a Macintosh SE, 150 is the maximum radius for which the picture will fit in the display window. I'm not sure what the maximum radius is on a Macintosh II; my guess is around 220.

When Helix is launched, the Radius 1 is set to the largest multiple of 50 which will fit inside the display window. Radius 2 is initialized to be 50 less than Radius 1, for a slight "ribbon" effect.

Important note: the Radii are not strictly speaking the relative sizes of the Lissajous figures. In actuality, each Radius defines the x-radius of one figure and the y-radius of the other. This was done to promote a "twisted ribbon" look in the pictures.

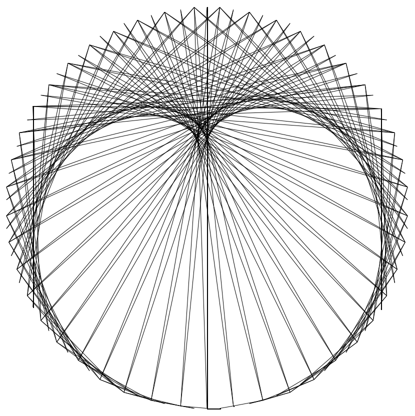
Angle Increment:
This value is the common angle increment (given in degrees) used in traversing both Lissajous figures. In general, small angle increments (less than 5) give a dense, smoothly traced single pass through the picture. Large values such as 61 or 137 give a multi-pass structure to the picture, as well as a finer crisscross pattern.

Factors:

These numbers determine the basic shape of each Lissajous figure. They form two ordered pairs of factors; each pair describes one figure. Factors 1 and 2 are the first figure; Factors 3 and 4 are the second. Setting both factors in a pair to be equal specifies a circle. Setting Factor 1 = 1 and Factor 2 = 2 specifies the "figure eight" of the default pattern. The important thing here is the ratio of the two factors. If instead you set Factor 1 = 2 and Factor 2 = 4, you will get the same figure, traversed twice as fast. In short, these numbers are traversal factors: if you double one of them, you'll traverse that axis twice as fast.

A good way to learn what the various Lissajous figures look like is to set both of the Radii to be equal, set the Angle Increment to be a small number such as 3, and set corresponding Factors to be equal. That is, Factor 1 = Factor 3 and Factor 2 = Factor 4. This will generate thin tracings of the pure Lissajous figures. The small Angle Increment makes the figures smooth. If you use a large value, the jumps along the figures will cause interesting crosshatches to evolve.

The following pictures give an example of the effects of varying angle increments, but they can also serve as a simple example of the designation of Lissajous figures in Helix. They were generated with Helix, then saved as EPSF files (which include a PICT representation of the image). They were then imported into Ready,Set,Go (the only program I have access to which could import the PICT's!), copied to the clipboard as PICT's, and pasted into the document. These graphics will print best on the LaserWriter.



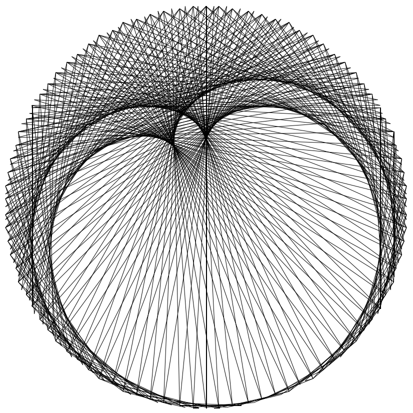
This picture was generated with these parameters:

Radius 1 = Radius 2 = 75 This makes the figure small and aligns the two Lissajous figures ("Ribbon" width of zero).

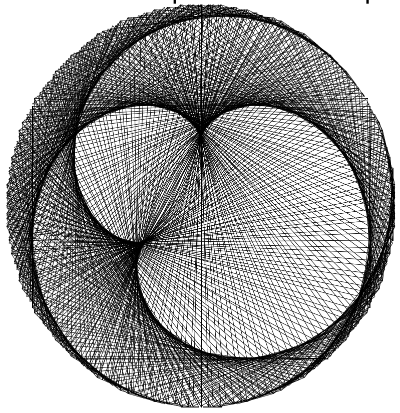
Factor 1 = Factor 2 = 1 Both factors equal defines the first Lissajous figure to be a circle.

Factor 3 = Factor 4 = 2 This makes the second Lissajous figure a circle, too, but it is traversed twice as fast as the first figure. The upshot of this is that lines are drawn fully across the picture, since the second Lissajous figure has completed one full circuit when the first is only half finished.

Angle Increment = 4 Four degrees is a fairly small increment, and it is a factor of 360. So, the picture is fairly simple.



This picture was created using the same parameters as the previous picture, except that the angle increment is 14 degrees. This makes the picture denser, especially since 14 is not a factor of 360. This causes a multiple-pass picture development, with a somewhat more complex crosshatch pattern emerging.



This final variation (which is a sample parameter file supplied with Helix, called "Mandelbrotish") uses an angle increment of 61 degrees. 61 is prime, so many passes were required to bring both Lissajous figures back to their starting points at the same time. Unfortunately the picture is unclear because of the high line density (it looks much better on screen or printed from Helix). The high angle increment also produces a separation of the "cusp" into two widely separated images. Notice that the separation between the two new cusps is about 61 degrees — get the idea?

Watching how these and other pictures develop on the screen is a useful way to become accustomed to the manner in which the parameters affect the final picture. For this reason several parameter files are supplied in the Stuffit version of this program which is available for downloading.

One interesting thing, which I discovered when implementing the new “Fast Refresh” option, is that Helix will never require more than 360 pairs of lines to complete a picture.

Helix Menus

Here's a description of what the various menu selections do:

Apple Menu:

About Helix	Calls up a three-pane description sequence of the Helix program, which includes a brief explanation of the program's algorithm and the author's E-mail addresses.

File Menu:

Clear Screen	Erases the screen and redraws the current picture. This is useful if (for example) you really can't stand watching it fill in the gaps after a portion of the image gets erased

	by a DA or some such.
Invert Screen	Toggles between black-on-white drawing and white-on-black. Bear in mind, though, that Helix will only print black-on-white.
Fast Refresh	Toggles between fast and slow screen redrawing. By default, Helix maintains its intermediate calculations, so that redrawing the figure (at least as far as it has already been calculated) will be *much* faster. When this option is turned off, each point is recalculated during the screen refresh. This is in case you like watching the figures' construction, especially on a very fast machine.
Save Parameters	Allows the user to save the parameters for the current picture under the filename of his or her choice.
Load Parameters	Allows the user to load picture parameters from a previously saved file. Note that Helix has its own file format and file type.
Save as EPSF	Allows the user to save the image as an Encapsulated PostScript Format file.
Page Setup	The standard Page Setup dialog for printer initialization.
Print	Draw the picture on the selected printer. The image will be printed at the best possible printer resolution that still maintains a 1-to-1 aspect ratio.
Quit	Goodbye for now....

Parameters Menu

Radii	Calls up a dialog to allow the user to change the two Radii values.
Angle Increment	This dialog allows the user to specify the base angle increment discussed above, in degrees.
Factors	This dialog lists the current Factors and allows the user to specify new ones.

All of these values are required to be integers!

Contact Information

As you can see (here I borrow another line from Dave Platt), I'm not making any of money off of Helix. It's essentially been an exercise in Macintosh programming, for the purpose of getting used to all that machine-specific stuff. It's also been a lot of fun. I admit, it's not the tour-de-force of non-commercial software that MandelZot is (you might have guessed by now that I'm extremely fond of MandelZot...). But I like it, and some of my friends like it, so here it is. If you like it too, just drop me a note saying so. I’ll be glad to hear from you. If you have suggestions, drop me a note. If you find a bug, by all means tell me so I can iron it out. I'll be upgrading the versions available on public file servers whenever I have a new one to release.

I can be reached at the following electronic mail addresses:

Internet:	cxt105@psuvm.psu.edu fixer@faxcsl.dcrt.nih.gov
BITNET:	CXT105@PSUVM

The PSUVM addresses are at the Pennsylvania State University; there's no telling how permanent that account or login ID may be. Future versions of Helix will have corrected address information. For now, that is my preferred email address during the school season, but *not* during the summer and Christmas vacations. The faxcsl.dcrt.nih.gov account should be quite stable; you should be able to reach me there at any time without much trouble. It may take a little longer for me to get back to you, however.

Acknowledgments

There have been a lot of people offering helpful suggestions and tips during Helix's somewhat sporadic development. Special thanks to Sonja and John for general enthusiasm; without them this would probably never have become freeware. Thanks also to Dave Platt (indirectly and much to his surprise) for his gracious permission to look at the source code for MandelZot. Without that source, I would never have found out how to get the Mac to send PostScript to the LaserWriter. Thanks to him, also, for telling me what was wrong with that approach, and for pointing out which Tech Notes I should consult.

I am also indebted to the Pennsylvania State University, who allowed me the use of their microcomputer facilities, and encouraged me to release Helix as freeware.

Evolution of Helix

Nonpublic versions of Helix were released to a small alpha- and beta-test population. These versions may surface here and there with spurious version numbers. These were not strictly speaking public releases. The first freeware version was designated 1.0.

1.1 In addition to the stuff listed under **Update - Version 1.1**, this release of Helix uses a radically reworked code organization, optimizing for size as well as speed. People familiar with THINK C 4.0 may be interested to know that Helix does not use any of the ANSI libraries, although it *does* use a custom-built version of the sprintf() function. This is why it’s still only 21K long! This version was also posted to the Info-Mac file server and comp.binaries.mac (see below).

1.0.1 Added an 'mstr' resource, to improve MultiFinder friendliness. Limited distribution version.

1.0 First freeware release, posted to the Info-Mac file server at sumex-aim.stanford.edu and the Usenet group comp.binaries.mac.