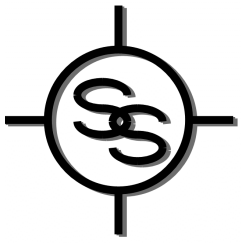
BEAM FOUR
Optical Ray Tracer

Java Edition

Release 169



STELLAR SOFTWARE
P.O. Box 10183 Berkeley CA 94709 USA
www.stellarsoftware.com
Tel (USA) 510-845-8405
Email info@stellarsoftware.com

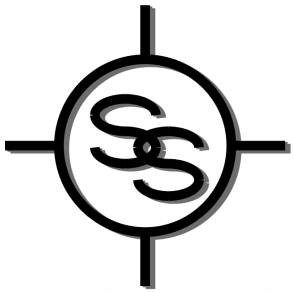
BEAM FOUR JAVA EDITION --- MENU REFERENCE

File	Edit	Run	Options	Help
New Optics	Cut	InOut	InOut	Show Error
New Rays	Copy	Layout	Layout	Special Keys
New Media	Paste	Plot2Dim	AutoAdjust	About....
Open Optics	Delete	MultiPlot	Plot2Dim	
Open Rays	Select All	Map	MultiPlot	
Open Media		Plot3Dim	Map	
Save Table		Histo1Dim	Plot3Dim	
Save Table As		MTF	Histo1Dim	
QuickPNG		Histo2Dim	Histo2Dim	
WriteCAD		AutoAdjust	Random	
WriteHisto		AutoRay	CAD	
Print/PDF		Random	Startup Files	
Quit		Demo	Factory Settings	
			Editors	
			Graphics	
			Default Rays	
			Ray Generators	
			Look and Feel	

BEAM FOUR OPTICAL RAY TRACER

JAVA EDITION

TABLE OF CONTENTS



STELLAR SOFTWARE
P.O. Box 10183
Berkeley CA 94709 USA

www.stellarsoftware.com
info@stellarsoftware.com

tel: USA-510-845-8405

Chapter 1: Introduction and What's New

Welcome to BEAM FOUR. This optical ray tracer combines features of many previous editions of optical ray tracers from Stellar Software with several new additions and improvements suggested by our worldwide users. In 2004, this Java® edition has been rewritten entirely in pure Java language and compiled to Java byte code, ready to run on a great variety of computers for which a native Java environment is freely available from Sun Microsystems (www.sun.com) and from Apple (www.apple.com)). The fact that Java supports a broad variety of computers and operating systems means that now, with BEAM FOUR Java Edition, a single optical ray tracer package can run almost anywhere.

BEAM FOUR is a table driven ray tracer. That is, an optical system is described by a table of entries. Each successive table row or record represents a successive optical surface in the optical system to be traced. This information is created, edited, and saved as an optics file with extension .OPT and quite a few examples are furnished with the BEAM FOUR software distribution. Similarly, a .RAY table is a text file listing a number of rays that you create to probe your optic. Each successive row or record represents a ray to be traced. As a user you are responsible for starting each ray (specifying its starting position and direction and in many cases a wavelength). Given this starting information, the ray tracer completes the computation of each ray as it propagates through your optical system, delivering numerical and graphical outputs in a variety of useful formats. Finally there are optional media tables (file extension .MED) that list refractive index information about glasses that your optic uses at a series of wavelengths that your ray table specifies. Each row of the table represents a glass type. Again, working examples of media tables are furnished with this software distribution.

For your convenience these data tables are plain ASCII text files. They may be freely copied, e-mailed, and incorporated into other documents. They can be loaded into popular spreadsheets such as Microsoft Excel® for further analysis, and conversely you can generate suitable .OPT, .RAY, and/or .MED files within a spreadsheet and use them in BEAM FOUR. The editor is able to read text files written on a variety of operating systems (Unix®, Solaris®, Linux®, MS-Windows®, Macintosh®) to simplify sharing data among platforms and within workgroups.

This manual is organized to help you get started using BEAM FOUR. The quick reference guide on the inner front cover shows the main menu and submenus for all the activities: loading files, performing ray traces, and specifying your preference options for the various activities. The first six chapters contain introductory material describing the capabilities of ray tracers in general and BEAM FOUR in particular. Chapters 7 and 8 describe the file system and the built in table editor, which has capabilities beyond the usual text editor to simplify working with the data tables. Chapters 9 through 11 describe how to set up the input tables for optics, rays, and refractive media. Chapters 12 through 22 describe BEAM FOUR's output functions and their uses. Chapter 23 "CAD"

describes the technical graphics file outputs available. Chapter 24 "Options" shows how to set your preferences for the many features of BEAM FOUR. Chapter 25 describes the help menu. Chapter 26 deals with spreadsheets for input to BEAM FOUR and for post-processing BEAM FOUR output. Chapters 27 and 28 illustrate program operation and describe the demo files furnished with this software. Chapter 29 offers suggestions for viewing stereoscopic images that B4JE produces. Warranty and licensing information is to be found in Chapter 30. Appendix 1 summarizes the mathematical properties of simple conic section surfaces. Appendix 2 describes rotation sequences and their uses. Appendix 3 describes Zernike polynomials, useful in characterizing perturbed optical surfaces and wavefronts. Appendix 4 describes diffraction gratings, and Appendix 5 discusses holographic optical elements.

What's New? If you are already experienced with our previous product lines you will find things very familiar. Your existing data tables can be used with only one minor change (ganged autoadjustment – see Chapter 20), giving you all the added features of this current product. For a quick start, install the software (Chapter 4), read the overview given in Chapter 2, and then try the guided tour in Chapter 27. The new features that distinguish this Java edition of BEAM FOUR from previous releases deliver some exciting new capabilities:

- Electronic distribution of programs, examples, and documentation (Chapter 4);
- Multi-platform functionality thanks to pure Java code (Chapter 3);
- Handles negative refractive index media (Chapter 9);
- Computation and optimization using wavefront error (Chapters 6, 10, 20);
- Includes arrays of lenses, mirrors, and irises (Chapter 9);
- Handles groups of mirror surfaces, each separately specified (Chapter 9);
- Offers 3-D viewing capability using red-blue spectacles (Chapter 29);
- Offers choice of asphericity vs shape for conic & higher surfaces (Chapter 9);
- MultiPlot capability: a grid of plots with stepped parameters (Chapter 15);
- Map: a new way to visualize optical performance over a field (Chapter 16);
- AutoRay helps with setup of ray start locations or directions (Chapter 21);
- Improved ganging and anti-ganging of auto adjustables (Chapter 20);
- Zernike polynomial representation of surfaces & deviations (Appendix 3);
- Built-in ray generators for four common illumination situations (Chapter 10).

Chapter 2: BEAM FOUR Capabilities

BEAM FOUR is a ray tracer that offers a quick and easy means of exploring the properties of optical systems. It is suited to a variety of tasks in optics, engineering, and science, including

- pilot design of lens, mirror, and prism systems for cameras, telescopes, copier and relay systems, magnifiers, microscopes, collimators, spectrographs etc;
- production of spot diagrams to show the expected ray distributions to be found within an optical system;
- preparation of quantitative graphs and plots that illustrate optical system performance;
- image analysis for tilt, decenter, curvature, profile of individual elements.

BEAM FOUR can handle optical systems having up to 99 successive surfaces. This sequence of surfaces can include any sequence of refractor, reflector, diffraction grating, holographic element, iris, lenslet, and phantom surfaces. (A phantom surface is one that has no effect on the light propagation but instead serves as a mathematical reporting point at which ray positions and directions can be gathered.) Your description of the optical system to be traced is created, edited, and stored in a text file whose extension is .OPT. Because every unspecified parameter has a reasonable built-in default value, these data tables can be relatively simple. One example describing a biconvex lens will give you the general idea:

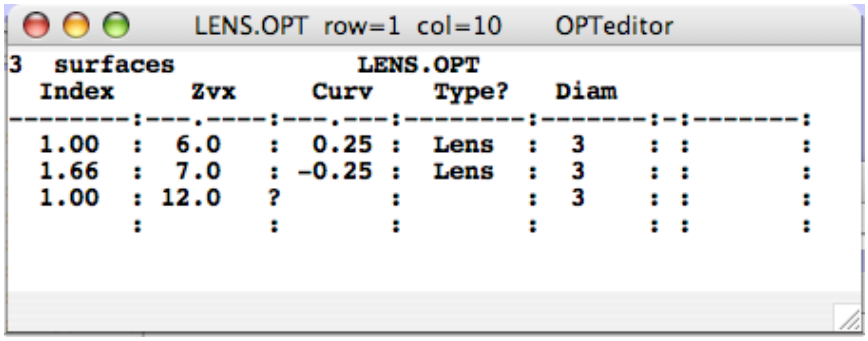


Fig. 2-1 Example of an optics file being edited. Editor features and customization are described in Chapter 8 "Editors" and Chapter 24 "Options." Optics file information is discussed in Chapter 9, "Optics".

Surfaces may be arbitrarily located in space with arbitrary orientations -- that is, a surface may be tilted, pitched, rolled, and decentered to any degree. Any surface may be assigned a definite working diameter or X and Y sizes, to impose a limited acceptance of rays. Alternatively it can be assigned no particular size, in which case it will accommodate all rays that can mathematically reach it. Although default surface shapes are flat or spherical, any surface can be assigned a conic constant parameter value to represent the conic section shapes: hyperboloids, paraboloids, ellipsoids. In addition,

polynomial, cylindrical, and toric surfaces profiles can be assigned to any surface, and combinations of all the above. Zernike polynomials 0..35 can be applied to any surface to model various forms of surface deviation. The limiting cases of these profiles are also included so that cones, axicons, straight dihedrals, toric dihedrals etc can be modelled. Irises can be circular, elliptical, square, or rectangular, and can be assigned blind centers. Arrays of lenses, mirrors, and irises can be specified. From these individual elements surfaces may be combined to represent an enormous variety of optical systems. Here's a very simple example, the biconvex lens whose table appears just above:

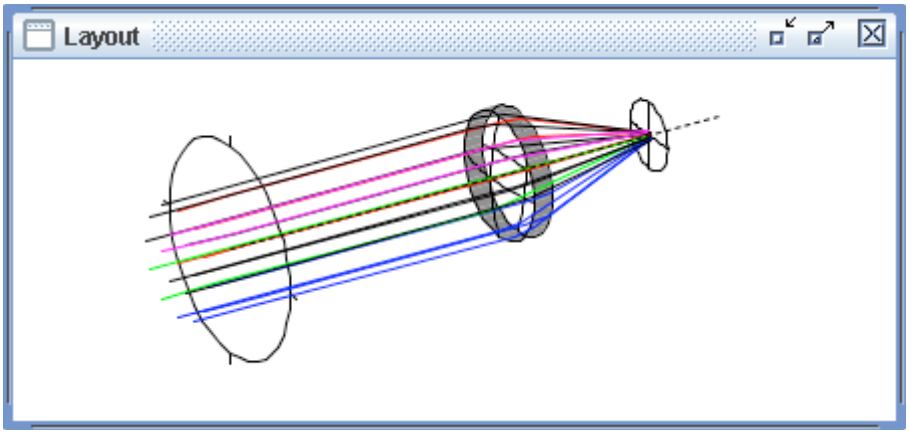


Fig. 2-2: A biconvex lens drawn using the **Run::Layout** function, including rays whose starting positions and directions are defined by the demo file LENS.RAY. Viewing pan, zoom, and orientation are all controlled by the mouse.

An important restriction for BEAM FOUR work is that you must specify the optical surfaces in the same sequence that rays will arrive at them. The sequence of specified surfaces defines the sequence in which the ray segments are traced. BEAM FOUR relies on your optical description to establish its trace sequence. It is therefore ***sequential***, not a free-form tracer of the kind commonly used in illumination engineering or stray light programs. (See however Chapter 9 and groups of mirror surfaces.) The advantage is that a sequential tracer is computationally efficient. No time or machine cycles are lost in evaluating the multitude of all possible intercepts and deciding which of all possible surfaces is the immediate target for each given ray segment. Of course this approach presupposes that you have a clear idea as to the intended light path through your system, so that you can specify the sequence of surfaces to trace. In classical optical design situations this assumption is valid. However it is invalid in situations where an unknown number of reflections or scatterings occur. For those cases you will need to trace several alternative ray sequences, or adopt a non-sequential ray trace program.

Contemporary optical practice often requires that a number of refractive materials be combined to greate a highly achromatized system. BEAM FOUR supports work of this type by allowing your refractive index information to be stored in glass tables and loaded as needed for each design task. When computing a ray trace, BEAM FOUR uses this information automatically. One example is the file GLASS.MED that has eleven common refractive media at four wavelengths in the visible part of the spectrum; another is SCHOTT.MED which has 95 Schott glasses at six wavelengths. You may use these,

edit them, augment them, or generate your own glass tables for use as needed. Glass tables are completely open: you may name your glasses and your wavelengths however you wish, and then use these as lookup tables to perform ray traces.

Up to 1000 distinct ray starts can be specified as part of one illuminating beam and can be maintained as a single .RAY file. Additional files, limited only by your disk space, can of course also be employed. In this way a variety of optical illuminators can be applied to one optical system, but also a variety of optical system variants (differing .OPT files) can be used with a single .RAY file. These interactions of file data allow you to prototype and evaluate a range of optical designs and compare their performance.

Ray start information can be entered manually into a ray table, or can be transferred from a spreadsheet. Alternatively there are built-in ray start generators that can initialize your ray table for you with the three most common ray grids: a simple linear sequence for fan beams or ribbon beams, a two dimensional rectangular grid, and a two dimensional circular pupil grid.

The .RAY table serves for numerical output as well as input. Beyond just specifying where each ray is to start and how it is to be initially directed, a .RAY table can be set up to provide space for the results of the ray trace at intermediate and/or final locations within the optical system. In this way you can keep input and output data together in a compact and easily reviewed way. A nice feature of the .RAY table is its total programmability --- you can set up columns that represent just those optical data that interest you, and have it default (on input) or ignore (on output) all other data.

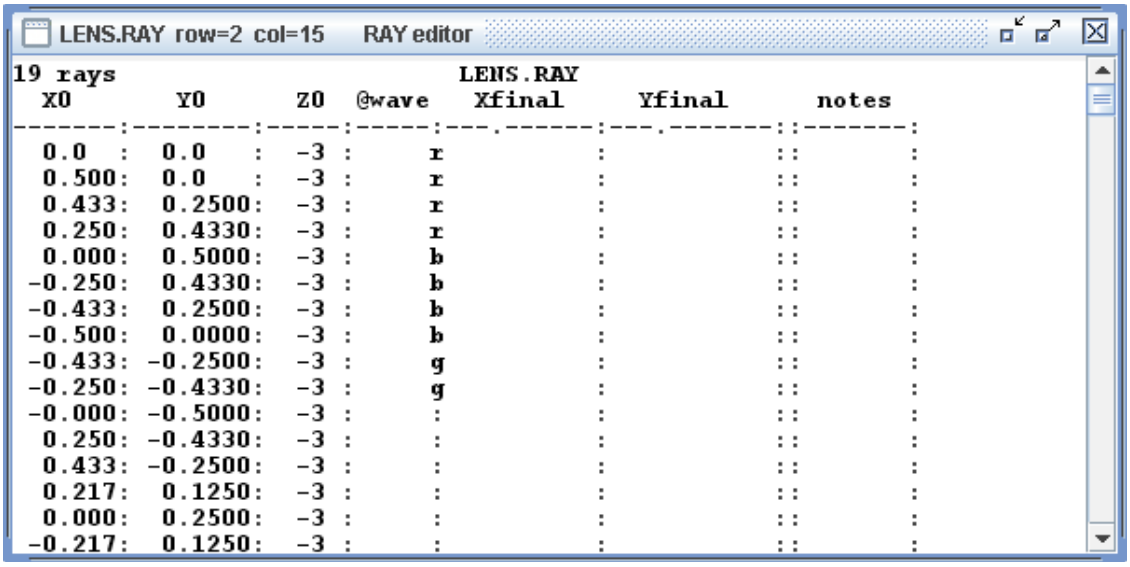


Fig. 2-3 Example of a portion of a .RAY table being edited. User input data define the ray start positions and directions and wavelengths. Other columns can be assigned to display output data. RAY tables are often much bigger than .OPT files because a .RAY table has as many ray data records as there are ray starts, while the .OPT table has only as many records as there optical surfaces. To get the numerical results of a trace, click **Run::InOut**.

The BEAM FOUR ray trace algorithm is fully three dimensional. It is not limited to paraxial ray groups, or to meridional or sagittal ray fans. Ray redirection accuracy is maintained from normal through extreme grazing angles, and makes full use of Java's double precision numerical accuracy, typically 14 decimal digits.

BEAM FOUR includes a text table editor to use in creating and revising the program's data tables (.OPT, .RAY, and .MED). These tables are plain ASCII text files that let you share the information between various platforms, and also share them via email and use them with spreadsheet or database manager software.

BEAM FOUR includes a Monte Carlo random ray generator for probing optical systems. Each graphic display window has an associated **Run::Random** menu item that begins the process of generating and tracing random rays through your optic. These rays span the extent of the overall ray beam outlined by the most extreme ray positions and directions supplied in your current .RAY table. Using Run::Random you can populate the various output products (layouts, histograms, spot diagrams, etc) with thousands or millions of rays to better understand the properties of the images formed by your optical system.

BEAM FOUR includes a Levenberg-Marquardt nonlinear iterative least squares optimizer. It allows tagged optical and/or ray start parameters to be adjusted for least mean square discrepancy between the computed final ray states and your specified ray goal values. This function is invoked with **Run::AutoAdjust**.

BEAM FOUR uses a graphical user interface (GUI) with pop-up menus, mouse-selected items, keyboard shortcuts, dialog windows, and a multiple-document interface wherein the various input and output data windows are organized in a single application window equipped with a main menubar. The underlying Java engine has the ability to mimic the salient aspects of popular operating system user interfaces so that your user experience remains nearly seamless.

BEAM FOUR output graphics appear on-screen and can be zoomed, panned, and annotated using the mouse and keyboard. The 3-D graphics can also be twirled, i.e. rotated to show various features. Of course the graphics can be captured by using your computer's screen grabber to put copies of the graphics into other documents. Beyond this, each graphic output can be regenerated and exported in any of six graphics formats (.PS, .PLT, .DXF, .PNG, .GIF, and .JPG) so that you may include them into an even wider range of documents and technical drawings. Moreover, whichever window is in front (text or graphics) can be saved as a bitmap file of type .PNG by simply clicking the QuickPNG item on the File menu. This lets you easily document your work as a series of graphic bitmap window shots that are WYSIWYG: they respect size and cropping.

One very handy diagnostic is the general-purpose **Run::Layout** function. When you first set up your optic and its illuminating beam, Layout can show you where the rays go --- or don't go! After making some adjustments to the rays --- perhaps aiming them differently, or perhaps enlarging an iris within the optic --- you may wish to zoom in on some portion of the graphic for a closer look at ray-surface and ray-iris relationships. You may also

need to change your view position angle. Views are conveniently controlled by the mouse: the wheel manages zoom, the left button drag manages pan, and the right button drag manages vertical and horizontal twirl. For repeatability of views, the Options::Layout function lets you specify your preferred view and save it for future use.

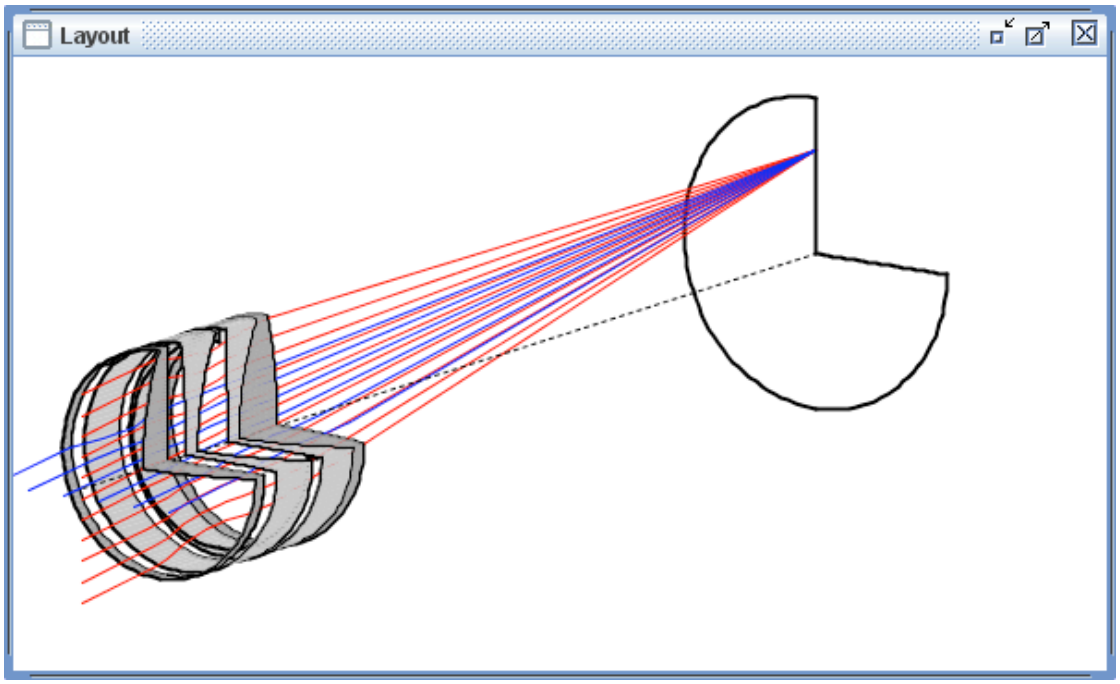


Figure 2-4 Oblique view of a four-element Tessar lens, produced using Run::Layout. The graphic view can be zoomed in or out using the mouse wheel or function keys F7 and F8. The vertical magnification can be zoomed with F5 and F6 and with the shift key + mouse wheel. The graphic view can be shifted up-down-left-right by dragging with the left mouse button, and it can be rotated left-right and up-down by dragging with the right mouse button held down. Customization features for the Layout function are at Options::Layout.

Another kind of display customization is ray color. Each ray can be assigned a color for display purposes, in addition to a wavelength for tracing in refraction or diffraction gratings. The display color for each ray shows up in layouts and in 2-D and 3-D plots. For polychromatic traces you will most likely choose each ray display color based on each ray's wavelength. In other cases you may choose each ray display color on other ray properties: for example incoming ray direction or entrance pupil zone. Rays that are not assigned any color are drawn using the default color: white if the background is black, or black if the background is white. Background can be specified in the Options dialog for each type of graphic. A stereoscopic display format is built in for use in Layout and Plot3D screen graphics provided that you have red+blue viewing spectacles.

Chapter 3: About Java

This edition of BEAM FOUR is a Java application. Java® was developed by Sun Microsystems and others as a means for creating and distributing software that runs on a wide variety of computers and operating systems, yet which offers a complete set of graphical user interface (GUI) capabilities that today's users prefer. Moreover, Java software acquires a look-and-feel characteristic of the user's computer and operating system. What BEAM FOUR needs to run is a computer that is equipped with the Java Runtime Environment "JRE" -- a specific collection of code libraries and entry points that allow a Java application to use your computer's resources and GUI tools.

Java is almost certainly already installed on your computer if you have a web browser. To verify your Java installation, visit www.stellarsoftware.com or any website that has a Java tester. BEAM FOUR requires Java version 1.5.0 or higher to properly deliver full service. If your Java version is older than 1.5.0, visit http://www.java.com or your computer manufacturer's website to download a more recent Java edition.

An alternative way to discover which Java edition you are running -- with no need to use a browser or Web access -- is to open a command window to your operating system and simply type

```
java -version
```

and press the Enter or Return key. The response tells you the installed version of Java.

BEAM FOUR's **Help::About** function tells you which edition of BEAM FOUR you are running, and which version of Java your host is running it on.

The BEAM FOUR software is distributed as a Java archive file of type **.jar** which is accessed by opening the distribution CDROM or download **.zip** file. It is recognized by your Java installation as an executable package. The download **.zip** file also contains this owner's manual in **.pdf** format, and a number of example data files. See Chapter 4 below for instructions on installation and operation.

Chapter 4: Installation

ELECTRONIC DISTRIBUTION:
The electronic distribution consists of a single file **BEAM4.zip** which contains the program, this manual as **.pdf**, and the example files. Unpack the files to put the software to work. Here's how:

- Create a new folder or directory. Name it "RayTracing" or some such.
- Drag the downloaded **.zip** file from your downloads area into this new directory.
- Extract everything from the compressed **.zip** file:
    - Mac OS-X users: double-click the **.zip** to reveal its contents, then drag each file out of the **.zip** into "Ray Tracing."
    - ...or use your usual unzip utility.
- For Windows® or Mac OS X®, double click the **.jar** icon to run it.
- Linux users (Ubuntu, Zorin, etc): set the executable bit flag before proceeding.
- When BEAM FOUR starts, drag and drop files into its workspace to open them, or use **File::Open.**

Should it be necessary, de-installation is even simpler: just haul the entire directory to the trash. Your disk space will be recycled by your computer's operating system after you empty the trash.

Chapter 5: About Ray Tracing

A ray trace is a systematic computation of the progress of a ray of light through an optical system. A good ray tracer will not only do the computations correctly but will also offer a variety of diagnostics that help you understand and visualize the overall distribution of rays. Originally carried out by hand calculation, ray tracing was historically one of the very first numerical activities to be adapted to the computer when the early mainframe machines became available in the 1950s. In 1986, Stellar Software's BEAM family of products was introduced, initially for the early PC and later also for Macintosh machines and their operating systems. Now, with this Java edition of BEAM FOUR, ray tracing can be conducted efficiently on a wide variety of computers and operating systems.

To conduct a ray trace you need:

- a description of the optical system to be traced;
- a description of how it is to be illuminated, i.e. the positions and directions that the light rays have as they are launched towards the system.

If the incident light rays have more than a single wavelength, you are conducting a polychromatic trace. If your optic uses refractive materials, you will also need:

- a lookup table giving the refractive indices of the optical glasses at each wavelength of interest.

In BEAM FOUR each kind of ***input*** data is prepared using a text table editor and saved in a text file. The contents of the .OPT, .RAY, and .MED tables are detailed in Chapters 9, 10, and 11. Examples of these tables are presented here in this manual and as distribution files provided for your use.

BEAM FOUR ***output*** can be obtained in the form of tables of numbers and graphical output. The tables can be computed during each ray trace and be updated with the specific results that you specify in the data column headers. The graphical output is called up with any of several Run menu choices: layouts, spot diagrams, and ray histogram data are shown.

Powerful though it is, BEAM FOUR is a ray tracer and as such is limited to the optical ***ray approximation*** in its analysis of optical systems. Mathematically speaking, a ray is a purely geometric entity described by a point in space where the ray originates and by a direction in which the ray propagates. A ray has no lateral extent, no angular extent, and no polarization. It contains neither electric nor magnetic fields, nor any spectral intensity distribution. We ascribe to each ray a characteristic wavelength, critical in computing ray refraction and grating diffraction. Each ray can also be assigned a screen color that governs how it will appear in Layout diagrams and spot diagrams. A ray is a mathematical abstraction of the real physical properties of light wave propagation and the

associated complexity of wave mechanics, boundary conditions, attenuation, absorption, aperture diffraction, polarization, and the like. Ray calculations are useful when the features of the optical system are very much larger than the wavelength of light. The job of a ray tracer is to construct a sequence of geometric ray segments through a succession of optical surfaces, with each change of direction obeying the appropriate law of reflection or refraction at each surface. The usefulness of a ray tracer depends entirely on the ray approximation being valid.

Ray tracing can supplant or replace geometric aberration theory. In aberration theory, deviations from idealized Gaussian image points or wavefronts are written as algebraic expansions into increasing powers of field angle and pupil location. Aberration estimates are most useful for optics that are slow and narrow in field so that the expansions converge rapidly and are dominated by the lowest power terms. If the system is axisymmetric and the pupil is filled, the aberrations divide neatly into those that affect image quality (chromatic, spherical, coma, astigmatism) and those that affect image location (defocus, field curvature, and distortion). Aberrations help guide the design process because they have differing sensitivities to stop location and lens bending. In contrast, computer ray tracing deals with a more general class of optics, including those with no particular symmetry or those with complicated pupil shapes.

Electromagnetic wave theory is far more powerful, but more complex, than geometrical ray methods. Electromagnetic solvers deal explicitly with boundary conditions, polarization, diffraction, interference, attenuation, and vector fields throughout illuminated volumes. They implicitly include multipath phenomena inherent in optical resonators, laser cavities, waveguides, and fiber optics. Efficiencies of reflection, refraction, and transmission can be modelled in an exact way.

Ray tracing is not capable of determining the detailed electromagnetic fields within an optic, but it can give a relative estimate of the intensity distribution of light based on the density of ray hits in a spot diagram. In ray tracing the relevant approximation is to describe each portion of a wavefront as being a localized plane wave. Converging rays become increasingly concentrated as they proceed towards a focus, and increasingly diluted as they proceed away from a focus. A measure of the intensity of the beam is this degree of concentration: the number of rays per unit cross sectional area. The best way to convey this information visually is with a spot diagram (see Chapter 14, "Plot 2D" and Chapter 15 “MultiPlot”).

Fortunately most optical design tasks can be adequately treated using a ray tracer because most optical systems are very much larger than the wavelengths λ of light they convey. Indeed, even for a fully diffraction limited optic, ray tracing is commonly used to evaluate its geometrical aberrations, independent of diffraction. If the RMS geometrical wavefront error is less than 7% of a wavelength, then the optic is said to be highly corrected and its performance is close to the best that may be achieved for that aperture. This edition of BEAM FOUR has features that compute and helps you minimize geometrical wavefront errors.

Ray tracing is scale independent. Any convenient (but consistent!) unit of length can be used to measure the positions of surfaces and rays. Meters, millimeters, and inches are popular choices. The unit of surface curvature is always the reciprocal of the unit of length, so that curvature is always reciprocal to radius of curvature.

FURTHER READING

For the user wishing to become familiar with optics, optical devices, and the many ways to discuss and evaluate image formation, the introductory text by E. Hecht "Optics" (4th Edition, Addison-Wesley 2001) is highly recommended.

A more advanced text for the technical engineer is "Modern Optical Engineering" by W.J.Smith (4nd Edition, Mc-Graw-Hill 2008). He presents the broad subject of optical design and engineering, with the references, formulas, and worked examples showing how image quality can be evaluated and quantified in theory and in the laboratory.

The mathematical foundation of many ray tracing programs, including BEAM FOUR, is the one developed by G.H.Spencer and M.V.R.K.Murty: "General ray-tracing procedure," J.Opt.Soc.Amer. 52#6 (672-678) 1962. In just 7 pages they lay out the whole method.

A practical engineering guide to optical system specification, analysis, design, and testing is "Optical System Design" by R.E.Fischer and B.Tadic-Galeb (McGraw-Hill, 2000).

The mathematical foundations of optics are thoroughly explored by M. Born and E. Wolf in "Principles of Optics" (7th Edition, 1999). A particular strength of this reference work is the careful derivation of aberrations from electromagnetic wave theory.

A compendium of detailed lens designs, with discussions, advantages, disadvantages, and design strategies is presented by W.J.Smith in his "Modern Lens Design" (2nd Edition McGraw-Hill 2005). Here, Smith gives a comprehensive inventory of classical and modern lenses for cameras, eyepieces, telescopes, magnifiers, relays, etc.

A useful guide to aberrations is the book “Aberration Theory Made Simple” by V.N. Mahajan (SPIE, 1991) who extends Seidel aberrations to include centrally obscured annular pupils and Gaussian apodized pupils.

Finally, the publications and conferences of the Optical Society of America (OSA) and the Society of Photo-Optical Instrumentation Engineers (SPIE) are valuable resources. Visit them at www.osa.org and http://spie.org.

Chapter 6: COORDINATE SYSTEMS

Optics coordinates: The simplest optical systems are coaxial. All their optical elements are lined up along a common axis. Following Born & Wolf, we refer to this common axis as the Z axis. The X and Y axes are perpendicular to Z and to each other. The origin of the coordinate system is the point X=0, Y=0, Z=0. We shall use these upper case letters to describe coordinates measured in this overall reference frame or "lab frame." For the simplest coaxial system, each optical surface has a location along the Z axis where its surface intersects the axis, but each surface's X and Y coordinate locations are both zero.

In a more complicated optical system, the surfaces need not be lined up in any particular way, nor need they be oriented the same way. The point at which each surface cuts its own axis of symmetry is called the *vertex* of that surface. We describe the location in space of each surface by the coordinates (X,Y,Z) of its vertex, measured with respect to the lab system origin.

Optical surfaces can be arbitrarily oriented in space. Starting with a surface aligned with the lab coordinate frame, three successive rotations describe its orientation:

- **Tilt** pivots a surface around its +x axis (same direction as lab +X axis);
- **Pitch** then rotates it around its own (now possibly tilted) +y axis;
- **Roll** rotates the surface about its own tilted and pitched +z axis.

Tilt, Pitch, and Roll are specified in degrees. For axisymmetric surfaces (paraboloids, for example) Roll has no effect. Roll is a significant parameter only for non-axisymmetric surfaces such as cylinders, torics, and diffraction gratings, and surfaces whose edges are defined as square, rectangular, or elliptical rather than circular.

The figure below is an oblique view of the consecutive rotation angles Tilt, Pitch, and Roll being successively applied to a simple flat square surface. The square frame at the lower left lies in the (X,Y) plane and is not tilted. Going towards the upper right, the next frame is tilted by +30 degrees. The next frame is tilted and also pitched by +30 degrees. The final frame is, in addition, rolled by +30 degrees. (This figure is a layout of the four surface optics file ANGLES.OPT that ships with this product.)

As with X,Y,and Z, we use upper case initial letters T, P, and R to emphasize that these rotations are successively applied starting from the lab frame.

Alternatives to this (Tilt, Pitch, Roll) system are described in Appendix 2.

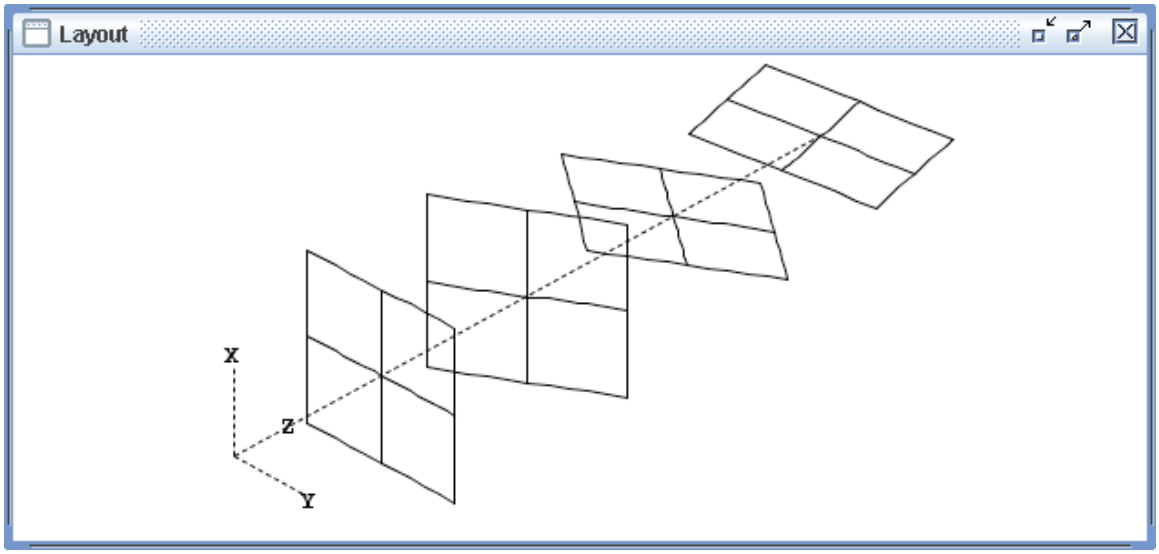


Fig 6-1 Oblique view of a square surface that is initially not rotated (lower left). Tilt of +30 degrees turns it about its x axis which is the same as the lab X axis, here vertical. Pitch then turns it about its own y axis. Finally Roll rotates it about its own z axis, its own axis of symmetry as shown in the upper right.

Ray vertex locations: A ray propagates through an optical system in a sequence of straight line segments. Each sequence begins at a point in space specified in Cartesian coordinates X0, Y0, Z0 that you input to the ray tracer via your .RAY table. A ray changes direction only at points where it has intercepted an optical surface. Such an intercept is called a *ray vertex*, in analogy with the vertex of a polygon or polyline. A complete output trace of each ray start is the sequence of its successive ray vertices. In the starting (lab) frame, we denote these ray coordinates with upper case letters:

X1, Y1, Z1 at the first surface;
X2, Y2, Z2 at the second surface; etc.

It is also sometimes useful to describe a ray vertex location in the frame of that optical surface, rather than in the lab frame. For this purpose we use the lower case symbols x,y,z,u,v,w so that for example y2 describes the y-coordinate value for a ray vertex at surface 2 measured in the local (x,y,z) frame defined by surface 2. The abbreviation "f" or "final" can be used to indicate the final surface number of a trace: for example Xfinal means X8 if there are just eight surfaces being traced, or X9 if there are nine.

Throughout BEAM FOUR, these coordinate descriptors X5, y2, etc are available to use in your .RAY tables, graphical display axis definitions, histograms, and the like. You may use these descriptors as column heads in a ray table where you'd like to see the computed numerical values. You may also use them in the dialogs that set up the plots and graphs. For example you may set up a plot of all each ray's X5 value plotted as a function of its y2 value, or see a histogram of all the ray table's Xfinal values.

Ray segment directions: The direction of each ray segment is described by three numbers U, V, and W. These are the X, Y, and Z components of the unit-length vector that points along the ray. They are commonly called ***direction cosines*** due to the fact that each number is the cosine of the angle between the ray segment and each coordinate axis. The numerical values of U, V, and W always lie between -1 and +1. A value of -1 indicates that the ray is headed opposite to the axis, while +1 means the ray is heading the same way. Zero of course indicates the ray is headed at right angles to the axis. Because a ray segment has to be pointed somewhere, and has unit length, $U^2 + V^2 + W^2 = 1$. This condition is established using the ray start information that you supply in your RAY table (see below) and continues to be true throughout a ray trace.

- Example: a ray headed towards +Z has (U, V, W) = (0, 0, +1).
- Example: a ray headed towards -Y has (U, V, W) = (0, -1, 0).
- Example: a ray headed 37 degrees away from the +Z direction towards the +X direction has (U, V, W) = (0.6, 0.0, 0.8).

The ray directions change, of course, as a ray propagates through an optical system. The numbering of the direction variables is the same as the numbering of the surfaces that launched each segment. In particular, the zero suffix identifies a starting direction for a ray, and therefore indicates ***input data*** that you provide to the ray trace process. So, W0 is the Z-component of a ray's direction as it starts its journey towards the first optical surface. Surfaces 1, 2, 3... are the computed directions of rays ***departing*** from surfaces 1, 2, 3... and are therefore ***output data*** from the trace and will be filled in by BEAM FOUR whenever the InOut ray trace function is chosen. For example, U5 describes the lab frame X component of the ray direction as it leaves surface 5. Vfinal is V at the final surface (both arriving and departing: never different at the final surface!).

Frequently you will want to read out a ray direction in the coordinates of the surface that launched it. This coordinate frame will usually differ from the lab frame: it will be translated from the lab origin, and it will possibly be tilted, pitched, and rolled. In the surface frame we use the lower case letters (u, v, w) again, combined with a number describing which surface is the source of the ray segment of interest. So, for example, u5 is the component of a ray's direction projected onto the x-axis surface of surface #5, as the ray departs surface #5.

In addition to these ray quantities, an ***optical path*** output variable is available to the .RAY table and graphical output functions. Optical path is a cumulative measure of the distance travelled on a ray's journey through your optical system. Each leg's linear length is multiplied by the refractive index of that leg. Optical path is proportional to the time delay experienced by a photon propagating through your optic. The usefulness of this statistic is not in its average value over a group of rays, but rather lies in its variation among rays that link an incoming wavefront to an outgoing wavefront. A well corrected optic will have a variation in optical path that is only a fraction of a wavelength, when considering a group of rays sharing common object and image points. The optical path is set to zero where each ray starts. It accumulates for each ray as the ray propagates from one surface to the next. To use this diagnostic, access the computed values of P1,

P2...Pfinal at the various surfaces graphically by using P1 or P2 or Pfinal as a variable name to be plotted, or by putting field headers "P1" etc into your .RAY table. See Chapter 10 for an example.

Closely related to optical path is ***wavefront error***, abbreviated WFE. This quantity is computed from each ray trace exactly like optical path, but instead of representing the delay between ray start and ray finish surfaces, it includes corrections for ray group inclination and curvature both for incoming and outgoing rays. Furthermore it subtracts away the average optical path for every group of rays, so only the ray-to-ray differences survive. In this way WFE represents the deviation of each ray from an idealized spherical or planar incoming wavefront and an equally idealized spherical or planar outgoing wavefront. This quantity then represents the variation in delay or path between the ingoing and outgoing wavefronts.

Total path delay has no effect on optical performance, but variations in this delay over the pupil have important consequences. Specifically, these variations directly show the degree to which the optic can be regarded as diffraction limited. It is commonly regarded that an optic whose root-mean-square WFE is less than 0.07 waves is diffraction limited and is therefore capable of essentially ideal performance.

WFE is a computed diagnostic of your optical system that BEAM FOUR provides in two ways: as an output datum on a ray-by-ray basis in your ray table, and in diagnostic plots such as Plot2D, Plot3D, and histograms.

Moreover, WFE minimization is by default built into the AutoAdjust feature. To design an optical system whose WFE is minimized, there is no need to create a goals column for WFE --- those goals are always zero. Instead, install a WFE field and remove any other goal fields or goal field headers that may be present in your .RAY table. AutoAdjust will remain active provided you have adjustable optics parameters (one or more has a "?" tag). The AutoAdjust action is then available to you to help you tune an optic for least WFE.

Because WFE is a measure of the variation between rays within a group having a common object point and image, it is important to fill your pupil with a representative set of rays if the computed WFE is to be a useful diagnostic. The default assumption is that all the rays in your .RAY table contribute to a common wavefront. If this assumption is correct, and if your pupil is properly filled, then the WFE diagnostic will be accurate and useful without further elaboration.

On the other hand if there are several field points within your .RAY table, then the default assumption is incorrect and a bit of intervention is needed. In the WFE field of your .RAY table, provide tag letters "a" ... "Z" to each ray's WFE field to indicate the object group to which each ray belongs so that all rays tagged "a" belong to one wavefront, etc. The total root-mean-square WFE is calculated from the deviations in WFE within each group while ignoring differences between the groups.

Chapter 7: The File Menu

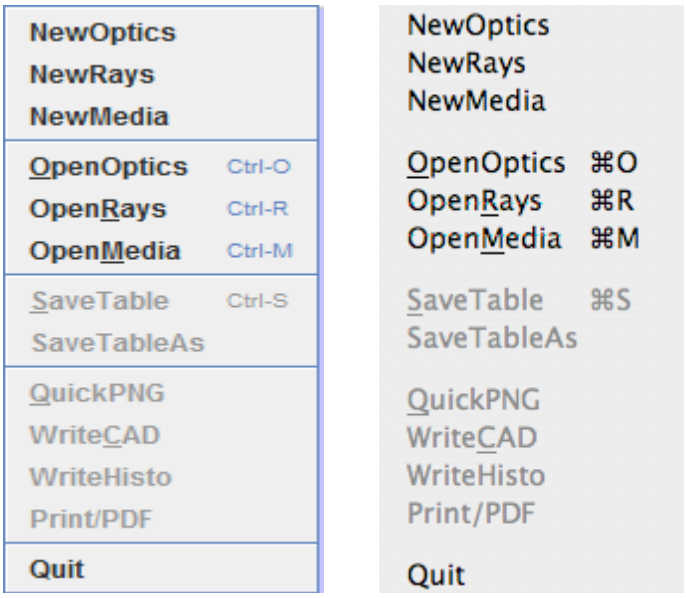


Fig.7-1: File menu on Windows (left) and Mac (right).

First stop on the BEAM FOUR menu bar is the File menu which centralizes the file reading and writing activities. Items that are grayed out are temporarily not available, either because some key information is missing, or because the input file has already been loaded. When you start the program, all the file input menu items ("NewOptics" etc) are available but none of the file writers have any output to deliver and are therefore grayed out. You may open an existing optics file by selecting **File::OpenOptics** or create a new file by selecting **File::NewOptics** at the BEAM FOUR menubar. Drag-and-drop is also supported, so that to load an optics file you may drag it from its on-screen directory and drop it into the BEAM FOUR workspace. The other data file types (**Rays** and **Media**) work the same way. After you have loaded an optics table, for example, NewOptics and OpenOptics become grayed out because only one optics table can be loaded at any given time. To conduct a ray trace you will need an optics table to describe your optical system and a ray table to specify how you want it illuminated. If you just want to look at a layout of an optic, and don't care about rays, you can load the .OPT alone and Run::Layout provided that each optical surface has a declared Diameter specifying the size of each drawn surface. (Diameters aren't necessary for Layout if you load your .RAY table, because the rays themselves can guide Layout's diameters.)

The default keyboard shortcuts use the control key in Windows environments and the command key on the Mac OS, as shown in the menu.

The tables are designed to handle numerical data. Three numerical formats are supported: the “decimal point period” floating format common in the U.S. and U.K, the “decimal point comma” international decimal numerical format used elsewhere, and the “e” or “E” exponential notation. BEAM FOUR automatically recognizes all these formats on input. To specify one or another of these in output fields, put a period, or comma, or “e” or “E” into the ruler line governing the field. Examples are shown in the following chapters.

Data tables are saved using the **File::SaveTable** and **File::SaveTableAs** menu items. The first of these saves the file with the same filename as it had when loaded or when most recently saved. SaveTableAs gives you a dialog where you may specify a new destination directory and a new file name. Save has the keyboard shortcut Control-S. Save and SaveAs are grayed out when there is no data table currently selected.

The File menu offers a window-capture quick-save feature that lets you document your work as a sequence of window-specific screen shots in PNG format. **File::QuickPNG** is enabled when any window, text or graphics, is currently selected (i.e. highlighted). Choosing QuickPNG triggers a bitmap capture of the selected window, and saves it to disk with a filename that you specify in a popup dialog. The .PNG extension marks it as a Portable Network Graphics compressed bitmap format compatible with a wide variety of graphics software and browsers. To review any of your captures, double-click it and your browser or default viewer application will display it.

For some purposes you will want to save your graphical output as a file, either for high-resolution reproduction or for use within CAD programs. BEAM FOUR supports three of the most popular graphics formats: Postscript, Plotter, and DXF (both 2-D and 3-D). To use this feature, first go to **Options::CAD** and choose your desired output format and orientation (Landscape or Portrait) and click OK. Then, later, with your specific graphic window selected (clicked, so that its frame becomes highlighted), click **File::WriteCAD**. This will give you a **File:Save** dialog to choose your destination directory, file name, and extension. You can set up your preferred **Options::CAD** at any time, but the menu item **File::WriteCAD** is grayed out until you have a graphic that is on-screen and selected.

The **File::WriteHisto** menu item lets you write an ASCII text file listing the numerical contents of a 1-D histogram, or 2-D histogram, or MTF plot, whichever type of plot is currently selected in the main window. This feature provides the histograms developed within BEAM FOUR for your use elsewhere. The feature's menu item is grayed out if there is no histogram currently selected.

The **File::Print/PDF** menu item lets you print any selected text or graphic window. A printer dialog will pop up. It lets you specify a local or a network printer and set up its printing format, paper supply, etc, as permitted by your printer and operating system. It is grayed out if no window is currently selected. To create a .PDF file, click the PDF button built into the Mac OS X print dialog, or in Windows scroll your printer selector to your installed PDF file creator software.

Finally, **File::Quit** exits the program. It is never grayed out but if you have unsaved work in any of your optics, rays, or media data tables, a reminder will pop up allowing you to save your work.

Chapter 8: The Edit Menu and the Editor

Because the main avenue of input to BEAM FOUR is through its data tables, it is obviously important to have an easy-to-use editor built into the application. The editor that is part of BEAM FOUR has many features that you will find familiar from having used simple text editors for email and other tasks. Here in BEAM FOUR you normally type in overwrite "Table" mode, mark text with the mouse, cut, paste, delete, and copy to/from the clipboard as you would with any text editor. However there are some added features designed specifically for working with tables, where it is important to keep things organized into columns. For this reason, in Table mode, characters don't insert, they always overwrite, leaving rightward fields unaffected. Also, deleting a character doesn't pull the entire line leftward, but instead acts more like backspace: it overwrites the character left of the caret. Mouse mark-and-drag marks entire records, not pieces of records, so that the integrity of the table columns will be maintained under cut, paste, copy, and delete. The tab key is similarly field-aware: tab moves your caret rightward to the beginning of the next field and shift-tab moves leftward to the beginning of the previous field, without disturbing the table.

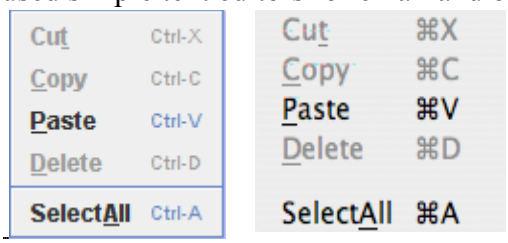


Fig 8-1: Edit menu: Windows (left) and Mac (shown on right).

Finally there are special combination keystrokes that are specialized for table editing. These keystrokes act on the data field at the caret position and help you lay out your data tables. They serve to narrow a field giving it fewer characters in the table, or widen a field to allow for more digits, or split a field into two, or copy the contents of a field downward into the record below. (Mac users: Mac OS reserves the function keys F1...F12 for other functions, and the Alt key is the same as the Option key.)

- F7 or Ctrl/Cmd-LeftArrow: narrow the field
- F8 or Ctrl/Cmd-RightArrow: widen the field
- F9 or Alt-RightArrow: split the field into two
- F10 or Alt-DownArrow: copy the field into the cell below
- Ctrl/Cmd Z: undo & redo the most recent change.

If you have a group of neighboring fields you want to copy down the page, temporarily delete the ruler line colons that separate the fields.. (This tricks the editor into thinking it is just a single wide field.) Then copy the whole group using Alt-DownArrow. Then replace the colons in the ruler line to reestablish the separate fields.

On-screen reminders of these special table editor functions and keys pop up when you select **Options::Editors** or when you choose **Help::Special Keys**.

Options::Editors lets you set up how the editor works. First and foremost, the editor has two distinct modes. The Table mode is best for keeping all the data aligned in the tables; you can recognize it by its blinking block caret. Alternatively the Text mode works more like a text editor: it inserts characters at its vertical-bar caret, and pulls characters left when deleting characters. You might want to use this mode to clean up tables that have been mangled by text processing elsewhere.

In Options::Editors, the first few checkboxes allow you to show or hide the directory path of a file that each editor is editing. You can choose to show or hide the row and column location of the editing caret. The appearance of text is controlled by choosing the editor font point size and boldness, and the screen pixel smoothing can be turned on or off, for whichever looks better on your display.

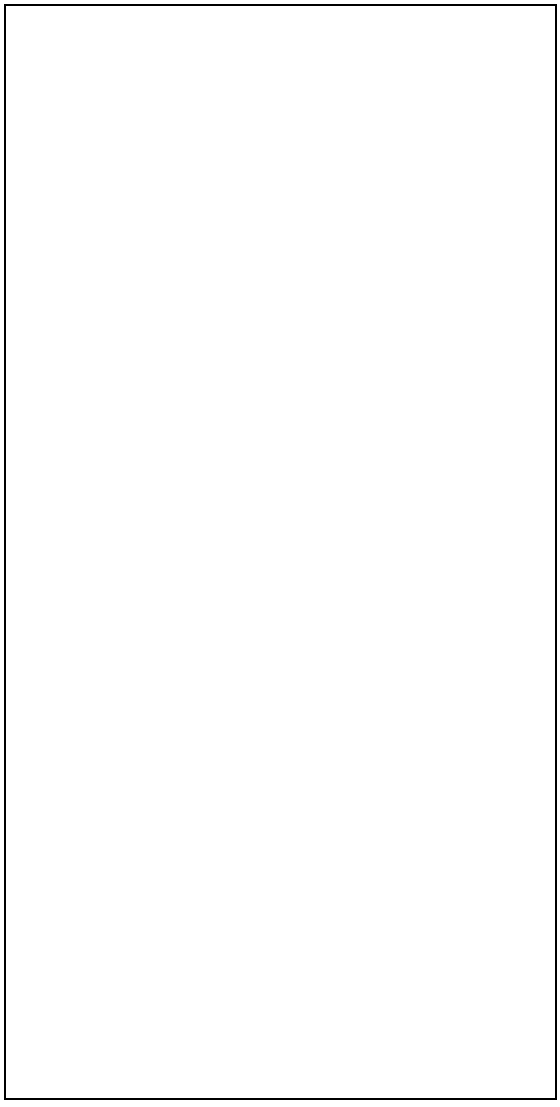


Figure 8-2: The **Options::Editors** dialog box that lets you customize the editors. At the top are reminders about the four special keystrokes used for managing your table layouts and a reminder about inserting or duplicating lines of text. The usual editor mode is Table mode, but there is an alternate Text mode that allows insert and delete. The checkbox "Show file paths?" lets you turn on or off the long filename format, while "Show row & col" controls the display of the editor caret location. (Editor rows for title, headers, and ruler are abbreviated T, H, and R.) The function "Expand commas?" lets you read in a comma-delimited text file and have it parsed into the successive fields defined by your headers and ruler. The "Default fieldwidth, chars" box lets you specify the initial field widths when a new editor is invoked with the File::New function. When editor contents are copied to your clipboard, you can specify keeping the delimiter colons if your copy destination is a text document, or tabs which are appropriate for copying to a spreadsheet. Finally you may specify your preferred font size and appearance with the bottom three options.

Options::Editors offers you a choice of how its text will be transferred onto the clipboard when it is marked and copied out to other software. In the native table format, colons separate the data fields to help you distinguish where they start and stop. So if you are copying text onto the clipboard to insert it into a document for ***visual*** use, choose "Colons, for text" since this choice will preserve the visual context. However, the spreadsheet world is different. There, data fields are automatically distinguished by nonprinting (hence invisible) tab characters. So if your clipboard transfer is destined for inserting data into a ***spreadsheet***, choose the option "Tabs, spreadsheets" so that the fields will be automatically recognized and properly parsed into data columns in the destination spreadsheet computation.

Before ending each editing session, remember to update the control number in the upper left hand corner of your table, which is the very beginning of your title line. This control number should be between 1 and 999. It specifies how much of the table BEAM FOUR should process: how many optical surfaces, rays, or refractive media you want evaluated or processed.

Obsolescence and your intellectual property: We've all experienced that sinking feeling upon discovering that engineering work from years ago has become inaccessible owing to data format changes by CAD, spreadsheet, and word processor standards. With BEAM FOUR this kind of issue is not so serious a problem: all your files are native plain ASCII characters. ASCII files are the closest thing the digital world has to a universally legible format. The data files you create in BEAM FOUR will likely outlive any software you currently own. Nonetheless we urge you to keep hard copy printouts of important files, because digital storage media and hardware that reads them are not particularly long lived. Who today can read a seven-track computer tape or an eight-inch floppy disk, even if its files are simply strings of ASCII characters?

Chapter 9: Optics and Optics Tables

To describe an optical system, we use a list of surfaces, written in the same sequence that light will arrive at each one. These lists are called optics tables. They are simply ASCII text files with extension .OPT. You may open an existing file by selecting **File::OpenOptics** or create a new file by selecting **File::NewOptics** at the BEAM FOUR menubar. Drag-and-drop is also supported, so that to load an optics file you may drag it from its on-screen directory and drop it into the BEAM FOUR workspace. In either case an editor window will open on screen. You may edit optics files using the keyboard, the mouse, and the Edit menu commands that send individual file records to and from your computer system's clipboard. Since they are ordinary text files you may also edit them with any other text editor you might have on hand.

Optics tables are organized on a line-by-line record basis. You may cut and paste individual records, or groups of records, using your computer's clipboard and the usual Edit commands: **Cut, Copy, Paste, Delete**, and **SelectAll**. Optics files are saved to disk using the **File::Save** or **File::SaveAs** commands. Files with the needed information can be imported from other software. Because they contain only standard ASCII characters, they are platform-independent. And of course via the clipboard you may freely copy and paste them into other documents such as email, reports, and spreadsheets.

Because the optics table is the main avenue of input to BEAM FOUR, it's important to understand how the table is organized. Dozens of examples are furnished as part of this distribution as files having extension .OPT.

An optics table has a three-line preamble followed by a list of the optical surfaces in the same order that light will encounter them. (If a surface is encountered twice by each ray, it must be listed twice.) Almost all the necessary optical surface parameters have likely default values, so even though there are upwards of 100 possible parameters per surface, you will usually need to specify only a few.

Example 9.1 To get started, look at Fig 9-1 below, which shows a very simple biconvex lens (filename: LENS.OPT) being illuminated with a diverging fan of rays (filename: LENS.RAY). A quick look at the table shows that it defines three optical surfaces: a lens surface, another lens surface, and a final surface whose type is ignored: it has no action since the trace stops there. The refractive indices approaching each surface are listed in the leftmost column: 1.00 (or equivalently, blank) represents air; 1.662 is presumably some refractive glass medium of which the lens is made, and the final index is again air. The surfaces are located along the Z axis at 2.8, 3.2, and 6.0 units. (Units don't matter in geometric ray tracing; they could be inches, millimeters, whatever, as long as they are consistent.) The curvatures of the front and back lens surfaces are +0.5 and -0.5 reciprocal units, zero being flat. The surfaces are spherical because no conic, toric, or polynomial entries are present.

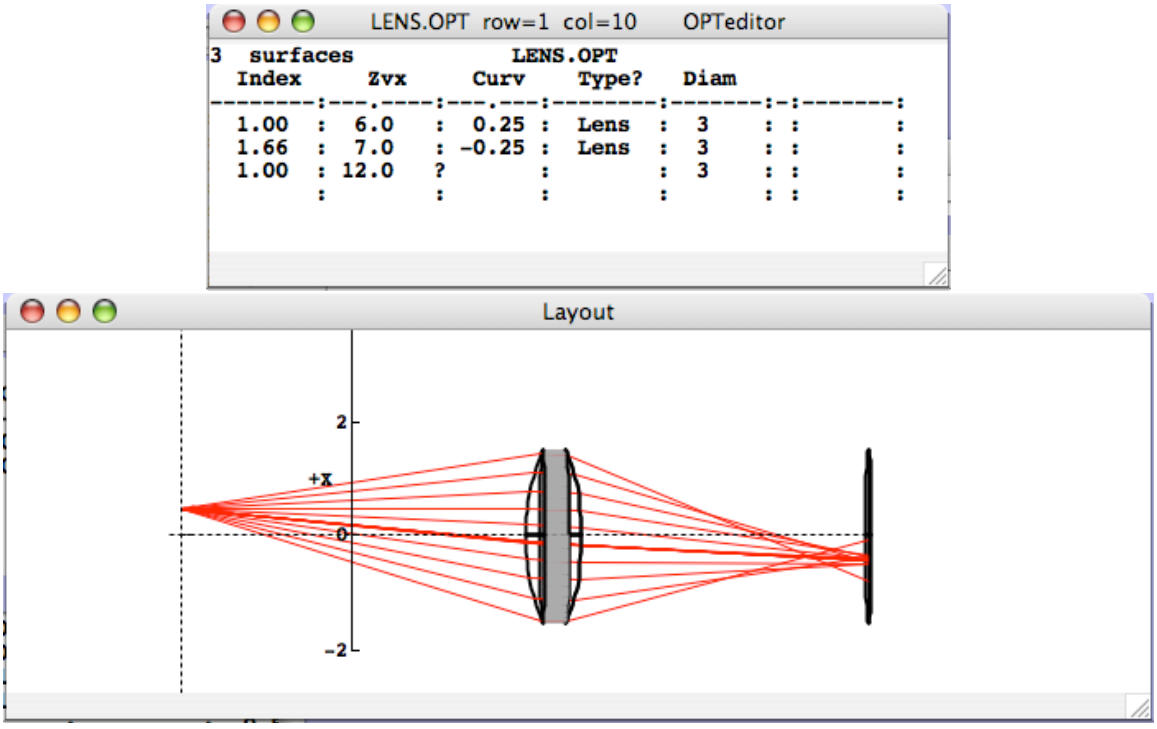


Fig 9-1 Top: Example of a simple optics table. The preamble is the top three lines: a title line, a row of field headers that define the table contents, and a ruler line whose colons delineate the data fields. The colons below the ruler line colons are there to guide your eye (and your typing!). There are three optical surfaces here (see upper left corner) --- by type they are a lens surface, another lens surface, and a final nameless surface. The final surface need not have any declared type since there is no ray action to perform when the ray stops. Below: a layout diagram of this LENS optic, illuminated by a .RAY file (see Chapter 10).

We now turn to a more detailed description of each part of the optics table, and follow up with a few further examples that illustrate some of the features of BEAM FOUR.

Title line (Line #1): This top line of a table has two purposes. Its leftmost item entry must show the number of successive optical surfaces that are to be traced. This can range from 1 to the maximum allowed, 99 for optics, 999 for other tables. The remainder of the top line can carry any sort of information. In our examples, we use it to prominently display the filename of the table. Other information may be kept here too: author's name, project, revision number, revision date, and so forth. The example shown here specifies that three optical surfaces are to be read by BEAM FOUR in computing its ray trace: two successive lens surfaces plus a final surface. If there were additional lines of data in the table, beyond the number of records specified in the title line, they would be ignored by BEAM FOUR. Having this control number available makes it easy to stop the trace partway through a complicated optic: just set the control number at the upper left hand corner to equal to the surface number where the trace is to stop. Leading spaces are OK.

This arrangement also makes it easy to use space below the data records for additional text information about the project. The editor can handle more than 1000 lines of text, so there is a bit of room for commentary and design notes below the active surface listing.

This way, brief text notes can be kept with the technical optics definition as a project develops. However, changes to the field widths applies to the entire table, so before using this notation feature be sure you are happy with your field width assignments.

Header line (Line #2): Optics tables are fully programmable. The individual data columns can be set up in any way that makes sense to you. BEAM FOUR of course needs to know what specific information the columns contain. The column headers in Line #2 convey this information, by showing a column header word or abbreviation. Leading and trailing blanks in these field abbreviations are ignored. Usually only the first one or two characters serve as a sufficient indicator to BEAM FOUR, but for easy human recognition you may prefer to enter more nearly complete header words. For example, to abbreviate "index of refraction" approaching a surface you may simply type "I" or "i" or "Index." A list of abbreviations and meanings is given below.

In the LENS.OPT table shown previously, the leftmost data field has been reserved for refractive index information about the medium approaching each surface. An empty data entry in this field defaults to 1.00, the refractive index of air.

Of the many data field titles understood by BEAM FOUR, you usually need to use only a few. Every data field has a reasonable default value that is applied when its field title is missing, or when its field title is present but the datum in the table is blank. For example an all-reflective optical system is one that has no need for refractive index information because there is no refraction. This can be specified by putting the value 1.00 into the "Index" fields, or by leaving them all blank, or omitting the "Index" field entirely.

In the same spirit, the default location of every optical element is the origin, {XYZ} = {0,0,0}. So if your optical train is coaxial, so that for example all the element vertices lie along the Z axis, then you need not specify any X or Y vertex locations and you will not need the X or Y field titles. A plano optical surface has no curvature, and zero is the default value for curvature, so again you need specify only your curved surfaces. Similarly if you are not using holographic optics, or polynomial aspheres, or other exotica, you need not specify (or even know about) those parameters and your optics table need not have any field headers describing them.

Ruler line (Line #3): The third line of an optics table is a ruler that shows how your data columns are separated. The hyphens in the row make the table look tidy, but actually BEAM FOUR ignores the hyphens. It is the *colons* in this row that indicate to BEAM FOUR how to parse the fields of the table.

Two other symbols can be placed into any ruler field to format the output from the program when an item in a field is being AutoAdjusted. A period or comma lets you show where you want BEAM FOUR to locate its decimal point when it is delivering its numerical output data to the table. Alternatively, the letter "e" or "E" anywhere in a ruler line field lets you stipulate than when BEAM FOUR displays a computed value, it should use exponential "e-notation" rather than regular decimal notation. There is no need to use

periods or E-notation indicators for your data input: BEAM FOUR handles all reasonable data reasonably.

In the file LENS.OPT shown earlier, the first data column has the header "Index" which reserves this column for the specification of refractive index in the medium approaching each surface. An "Index" datum should be left blank or set to 1.00 to indicate that rays approaching that surface travel through air (or vacuum, if you are working in the ultraviolet where refraction is measured that way). A numerical entry under this header (for example "1.662") is taken to be a refractive index value that applies to all rays independent of any wavelength information supplied with each ray. All nonzero numerical values (including negative numbers! BEAM FOUR handles negative refraction as well as ordinary positive refraction) are valid and will be properly refracted. A zero or blank entry is interpreted as an index = 1.000.

Before exploring further examples of .OPT files we'll cover the three general kinds of information that these tables convey: **action headers** that control with the ray tracer action will be at each surface; **geometry headers** that describe the location, orientation, and shape of each surface; and **periphery headers** that govern the physical boundaries of each surface within its vertex frame of reference. See Table 9-1 below.

Any data put into a field with a blank header -- like fields 6 and 7 of LENS.OPT above – will be ignored by BEAM FOUR. This feature lets you temporarily turn off an entire column of input data, so that it reverts to the default.

Action headers let you control what action the ray tracer will take when encountering each optical surface. The governing header is "Type" or simply "T" beneath which you specify each surface as a lens, a mirror, an iris, or a retroreflector. Alternative headers "M" and "L" are synonyms of "Type". The default surface action is refraction, so if you have an all-refractive optic, you won't need this header or field. But if you want to include nonrefractive irises or other elements, you will need it. Major types available to each surface in your .OPT table are "L" or "l" for “lens” (including transmission gratings), "M" or "m" for “mirror” (including reflection gratings), "I" or "i" for iris (no change in ray direction but defined maximum and/or minimum radii for ray passage). A fourth type of surface is "R" or "r" for an ideal retroreflector that simply reverses the direction of each ray (unphysical, but handy for locating virtual foci). A phantom surface is one that does not affect rays but merely provides a reporting point for ray coordinates; if you need one, use a lens surface with no change of refraction, or an iris with no declared diameters. A coordinate break surface pair CBin & CBout interrupts and resumes ray propagation at a different location or orientation. See example 9.19 below.

The remaining action headers let you supply additional information about the chosen action. For a lens surface, the refractive index of the medium approaching the surface is specified in an "Index" field (a field headed with the word "Index" or "I" or "i"). For monochromatic traces this can be a numerical value like 1.665, or for polychromatic traces it can be a glass name chosen from the list of media in your .MED table.

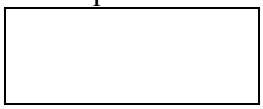
Table 9-1: Optics Headers and Input Field Abbreviations	
Surface Action headers …. and the data in their fields	
Type, M, L	lens, iris, mirror, retro, lensarray, scatter, CBin, CBout; default= lens.
I, i	index of refraction in the medium approaching the surface; default=1.0
Gx, Gy	diffraction grating groove density in x and/or y direction
O, o, order...	order of diffraction for each diffractor, all rays; default = 0
VLS1 ... VLS4	varied line space grating polynomial coefficients; default = 0
HOEx1... HOElam	holographic optical element constants (see Appendix 5)
Scat, scatter…	RMS Gaussian forward scatter angle in degrees; default=0
Nx, Ny	Number of array elements along x and y axes
Surface Geometry headers …. and the data in their fields	
X, Y, Z	lab coordinates for vertex of each surface
T, P, R	lab coordinates for tilt, pitch, roll angles in degrees
C, curv, ...	curvature of a surface; value = 1/radius of curvature * positive value: curved towards local +z direction * negative value: curved towards local -z direction * zero or blank: a flat (plano) surface, default.
Asph, Ax, Ay...	asphericity: departure from spherical profile = Shape - 1 * default value = 0, a spherical surface * Ax and Ay specify the two biconic asphericities
S, s, shape, ...	shape: departure from parabolic profile = Asphericity + 1 * default value = +1, a spherical surface Use A, or S, or neither (default = sphere), but not both.
Cx, cx	Forces toric (without Ax) or biconic (with Ay and Cy) Curvature "Cx" is in xz plane with cylinder axis in y direction.
Cy, cy	With Cx, forces biconic curvature; use with Ay
A1, A2, A3,...A14 polynomial coefs	* surfaces of revolution: coefficients of $r, r^2, r^3, ...r^{14}$ * toric surfaces: coefficients of $\|y\|, y^2, \|y\|^3, ... y^{14}$
Zern0, …Zern35	Zernike polynomial coefficients, see Appendix 3. These require a specified Diameter for the Zernike surface.
Surface Periphery headers …. and the data in their fields	
D, Dia, Diam...	Diameter of the outer periphery of the surface
Dx, Dy	Diameters for x and y axes separately
d, dia, diam...	inner diameter for blind center of an optical surface or iris
dx, dy	inner diameters for x and y axes separately
OffOx, OffOy	offset in x or y for outer periphery from vertex location
OffIx, OffIy	offset in x or y for inner perphery from vertex location
F, f, form, figure	field specifying square/rectangular periphery; * "S" indicates square or rectangular outer periphery, * "s" indicates square or rectangular inner periphery, * "Ss" or "sS" indicates both peripheries square or rectangular. * default = round (circular or elliptical).
Ns, Nspi, ...	number of spider legs in an iris (default = 0)
Ws, Wspi, ...	width of spider legs in an iris (default = 0)

The action fields "Gx" or "Gy" let you convert an existing lens or mirror surface into a parallel groove diffraction grating in transmission if it's a lens, or reflection if it's a mirror. Data in these fields will be the numerical groove density in the local-frame x or y directions, expressed in units consistent with your ray wavelengths (grooves/micron if your wavelengths are in microns). The field "Order" or simply "O" describes the order of diffraction to be evaluated, usually a small positive or negative integer or zero if you'd like it turned off. The related action fields "VLS1" through "VLS4" let you convert a uniformly ruled grating into a varied line space grating, with polynomial spatial variation coefficients of x^1, x^2, x^3, x^4 for Gx, or similarly the first four powers of y for a Gy grating. Additional details for diffraction gratings are presented in Appendix 4.

The action fields "HOExxx" let you set up a holographic optical element. These fields are explained and illustrated in Appendix 5 "Holographic Optical Elements."

Geometry headers let you set the position, orientation, curvature, and other properties of each surface. The headers X, Y, and Z let you locate the vertex of each surface in space, using its lab frame coordinates. T, P, and R specify its tilt, pitch, and roll angles in degrees, as explained in Chapter 6 "Coordinate Systems". The default values are all zero, so if you have a simple coaxial optic (nothing decentered or tilted) you need not use X, Y, T, P or R; simple string your surfaces out along the Z axis.

The profile header "C" or "Curvature" lets you specify the curvature of each surface in your optic. The numerical value of curvature is 1/(radius of curvature) and its units are the reciprocal of the length units in which your X, Y, and Z coordinates are expressed. For a flat (plano) surface, the curvature is zero. This is the default value. For a surface that curves towards the local +z direction, the curvature sign is positive, and is negative for a surface curving towards the -z direction. Spherically curved surfaces are the most common surface profiles in optics, and the spherical profile is the default. The equation for the spherical surface departure from a plane is



where $r^2=x^2+y^2$ and C is the curvature of the surface.

Aspheric profiles of several kinds are supported by BEAM FOUR. The simplest of these are the conic sections of revolution --- a group that includes the ellipsoids and the hyperboloids as well as the special cases of the paraboloid and the sphere. To gain access to these surfaces, introduce a header "Asph" for asphericity into your optics table. Asphericity measures departure from a sphere: Asph<-1 are the hyperboloids, Asph=-1 is the paraboloid, -1< Asph < 0 are the prolate ellipsoids, 0 is the sphere, and Asph>0 are the oblate ellipsoids (see Appendix 1). The equation for these surfaces is

where again $r^2=x^2+y^2$, C is the curvature of the surface, and Asph is the asphericity.

Extreme shapes can be generated with this function for special purposes. One example is the **cone**, a surface used in axicons and energy concentrators. A cone is a hyperboloid whose vertex curvature C is infinite (an infinitely sharp tip). Although infinity cannot be represented in BEAM FOUR, very large numbers are easily written in scientific notation. Use any huge curvature, say 9E9, for a cone opening toward +z, or -9E9 for a cone opening toward the opposite direction -z. Then evaluate your asphericity from the formula $-1-\tan^2(\alpha)$ where α is the semiapex angle of the cone (the angle between its axis and its side). For example, a 45° cone (90° full apex angle) has an asphericity equal to -2.

Some workers prefer to use a different measure of conic constant, namely "Shape" equal to Asph+1. Shape measures departure from a paraboloid. BEAM FOUR recognizes "Shape" and or "S" as a profile header and you may use that if you prefer. Don't mix them in the same optics table however. The default value for Shape is +1.0, a sphere.

Some workers use the symbol k or κ to indicate a general conic constant. There is unfortunately some ambiguity in the literature as to whether this is to be interpreted as asphericity or shape or eccentricity, all of which are conic constants. For clarity we do not support k or κ but only the uniquely defined asphericity and shape quantities. Appendix 1 details the connections among these parameters.

The conic sections of revolution can be extended in complexity by adding polynomial terms. The equation for the conic surface including polynomial terms is

Polynomial terms are useful on their own, without curvature or asphericity, in polynomial optics such as Schmidt correctors. More often they are combined with curvature and asphericity to provide small high-order corrections to a surface. The default values for A1...A14 are zero and you will usually need to use only the first few even coefficients A2, A4, and A6 since most optical surfaces will be very nearly approximated by the conic aspheric profile.

Another family of surfaces is the **toric family**. A toric surface is the surface swept out when a function z(y) is rotated about an axis that is parallel to the local y axis. This surface function can be zero, circular, conic, or include polynomial terms:

When this z(y) curve is revolved about an axis parallel to the local y axis, the resulting toric surface is exactly circular in its xz plane, and is exactly the specified function in the

yz plane. BEAM FOUR recognizes a toric surface by the presence of a Cx header with a nonblank entry. Cx is the reciprocal of the radius of the circle in the xz plane, to which the appropriate sign is attached: positive bending towards +z, or negative for –z.

We emphasize that the local variables x, y, and z are all measured with respect to the vertex of the surface and are oriented so that z lies normal to the surface at its vertex. These local coordinates differ from the lab coordinates by translation and orientation. In the case of the toric surfaces, Roll may be used to rotate the toric about its z axis and thereby place its principal x and y axes at any angle with respect to the lab frame.

Here are a few specific examples of toric surfaces.

- A large class of cylinder surfaces is available by explicitly setting Cx=0. In this way the surface z(x,y) will depend only on y via the formula above and will be a constant with respect to x. Its axis will be parallel to the x axis. It will be a circular cylinder if Asph=0 and if the polynomial coefficients are all zero. By choosing a nonzero value for Asph, the group of conic cylinders (elliptical, parabolic, hyperbolic) is produced. By selecting nonzero values for one or more polynomial coefficients A1...A14, polynomial cylinders can be created. Setting Cx=0 and C=∞ (any huge value such as 9E9) along with an Asph < -1 yields a **dihedral** -- the figure formed by joining two planes along a common edge. Choose Asph = $-1-\tan^2(\alpha)$ where α is the dihedral half-angle, as with cones.

- A simple circular cylinder surface can be produced by forcing C=0 and assigning no polynomial coefficients, and then using Cx to control its curvature. This way the cylinder axis lies parallel to the y axis and its (x,z) intercept is a circular arc whose curvature is Cx (i.e. its radius is 1/|Cx|).

- If both C and Cx are nonzero, but the polynomial coefficients are all zero and the asphericity is zero, a conventional circular torus results.

- All these capabilities can be combined to define a general conic or polynomial toric surface. Here, we use C, asphericity, and polynomial coefficients to form the z(y) curve and then use Cx to impose the figure's curvature in the xz plane.

- Because the C and Cx descriptors apply only in the local frame of an individual surface, the entire toric figure can be rolled to any angle you might need by introducing a Roll datum to that surface. For example, if Roll=90deg, the simple circular cylinder controlled by Cx will be curved in the lab (Y,Z) plane and will have its axis in the lab X direction.

Example 9.2 We illustrate three of these features in Figure 9-2 below. At the lower left is a hyperbolic cylinder, using Cx=0 but Curv=+10.0 and Aspher=-2.0. The cylinder's axis is parallel to the x axis, vertical in this presentation. This cylinder is general, allowing variants from circular through conic to polynomial generators. In the middle is

a circular cylinder using Cx=+1.0 but Curv=0. At the upper right, a circular torus is shown, with Cx=+1.0 and Curv=+5.0.

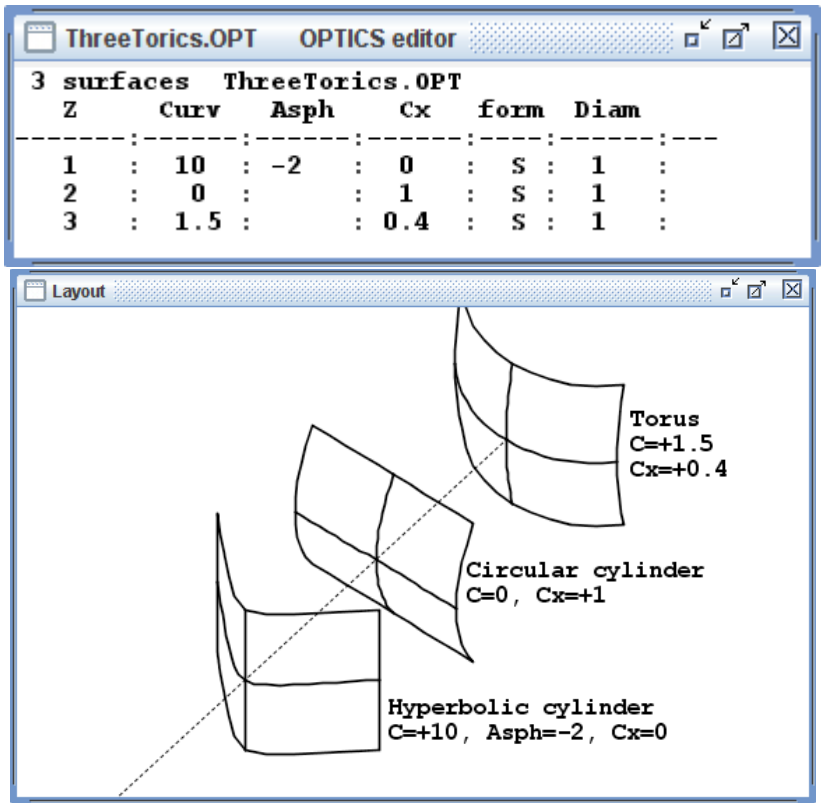


Figure 9-2 An oblique view of three toric surfaces: a general cylinder (here, hyperbolic) with its axis in the x direction (here, upward); a circular cylinder with its axis in the y direction, and a circular toric section at the upper right.

Beyond torics, there are a multitude of other surface descriptors in use within the optical community. Biconic surfaces (see BICONIC.OPT) are built into Beam Four: these use all four specifiers Ax, Cx, Ay, and Cy. Zernike polynomials (Appendix 3) are built into Beam Four: they are most commonly used to describe the wavefront variation over a pupil, and can be added to any surface to serve as a wavefront corrector. Alternative mathematical formulations of surface profiles (not part of Beam Four) are the Cartesian Oval (R. Descartes, 1637) and its generalizations, the cone concentrator (e.g. Winston, Appl. Opt. 17#5 1978; Eichhorn, Appl. Opt. 21#21 1982), the aplanatic aspheres (Mertz, Appl. Opt. 18#24, 1979), the B-spline (e.g. Chase, Proc. SPIE 4832, 2002), the super-conic and sub-conic (Greynolds, Proc SPIE 4832, 2002), the "exact" aspheres of Lynden-Bell and Willstrop (MNRAS 351, 2004), and the polysag family of surfaces (Terebizh, astro-ph/0707.1731 2007).

Periphery headers are optional. They let you define the physical boundaries of each surface, describing it as circular, elliptical, square, or rectangular. In the absence of a periphery specification, rays will use your optical surfaces wherever they fall, subject only to restrictions imposed by geometry (e.g. if a ray misses a sphere, it cannot intercept it, and is lost). By imposing one or more specifications you gain control over which rays

will proceed through the ray trace at each defined surface, according to where they intercept that surface. In this way you can explore issues of vignetting and field coverage using realistic optical element dimensions and shapes. Peripheries apply to any type of optical surface: lens, mirror, iris, array, or retroreflector.

The most basic periphery datum is the Diameter field. Its header can be abbreviated D or Diam. Upper case “D” indicates outer diameter. It creates a circular zone with the specified diameter beyond which rays fail to continue their progress in the ray trace. A blank datum under this header turns the feature off so that no rays are stopped. Its shape is by default circular, and it is centered on the vertex of its host optical surface.

A blind center zone can be introduced into any surface by putting a "diameter" field into your table and an appropriate nonzero value into that surface's field. The lower case "d" indicates that this is an inner periphery. It can be abbreviated "diam" or simply "d". Rays that intercept the surface within a centered circle of this diameter will be stopped.

Elliptical peripheries can also be produced. Instead of using a D column, use two columns: a Dx column and a Dy column. Distinct numerical entries in these columns will yield an elliptical aperture with distinct diameters in the x and y direction. If either of these is left blank, the default is the Diam (if specified) or the other D value, making the aperture circular. Similarly, lower case dx and dy specify create data columns where you can specify distinct inner diameters for an elliptical central blind zone.

These same rules apply to apertures and peripheries that are square or rectangular. To create a rectangular or square element, insert a data column whose header is "f", "F", or "figure" or "Form" or some such, and in this field give those optical elements that are to be square or rectangular the designation "S" for the outer periphery, or "s" for the inner periphery, or "Ss" for both outer and inner peripheries. A blank datum leaves a surface with its default circular or elliptical form. Then, assign the diameters "D" or "Dx" and "Dy" as you would for a circular or elliptical element. Again, "d" and "dx" etc can be used with the square or rectangular figures to specify a blind center that is rectangular.

If you furnish the Diameters for all your optical elements, there will be enough information for BEAM FOUR to show a layout diagram of your optical system even with no rays being present and with no .RAY table having been loaded. To create this layout without rays, click Run::Layout. Of course, with no rays, there will be no ray traces computed. But the artwork for Layout will be generated and displayed. For most purposes though you will use a .RAY table to to set up the rays that will illuminate your optic. This way, even without specified element diameters, the rays themselves furnish the needed Diameter information and allow a layout diagram to be constructed. The only surface types that cannot be drawn in this way (with automatic ray Diameters) are the iris and the spider.

Offset or off-axis optics can also be modelled. The fundamental coordinate system for optical elements is based on the vertex positions of the elements --- the vertex being the point on a mathematical surface where the axis of symmetry lies. In centered optics, the

periphery of the optic is equidistant from that vertex. But in off-axis optics, the working surface can be displaced so that its center lies at a local x, or y, or both, that is nonzero. To specify a optic whose periphery is offset from its vertex, create a field header "OffOx" and/or "OffOy" (not case sensitive) and for each item in the table that is offset, fill in the offset distances in x or y for that optic, measured in the vertex frame from the optic's vertex. Similarly, if there is a central hole or other inner diameter feature to a surface, the additional fields "Offix" and "Offiy" (not case sensitive) are available to offset the inner boundary from the vertex. These offsets work for boundaries that are circular, elliptical, square, or rectangular. The next three examples illustrate these ideas.

Example 9.3 The first of three off-axis paraboloid (OAP) examples is the simplest. We begin with a simple parabolic reflector located on our coordinate axis, illuminated by a ray group that is offset to one side so that only an off-axis portion of the paraboloid is actually in use. Figure 9.3 shows the optics table, the ray table, and a layout of this elementary system. Note that just as with a fully illuminated paraboloid, these rays arrive at a perfect focus provided of course that they are incident parallel to the paraboloid axis.

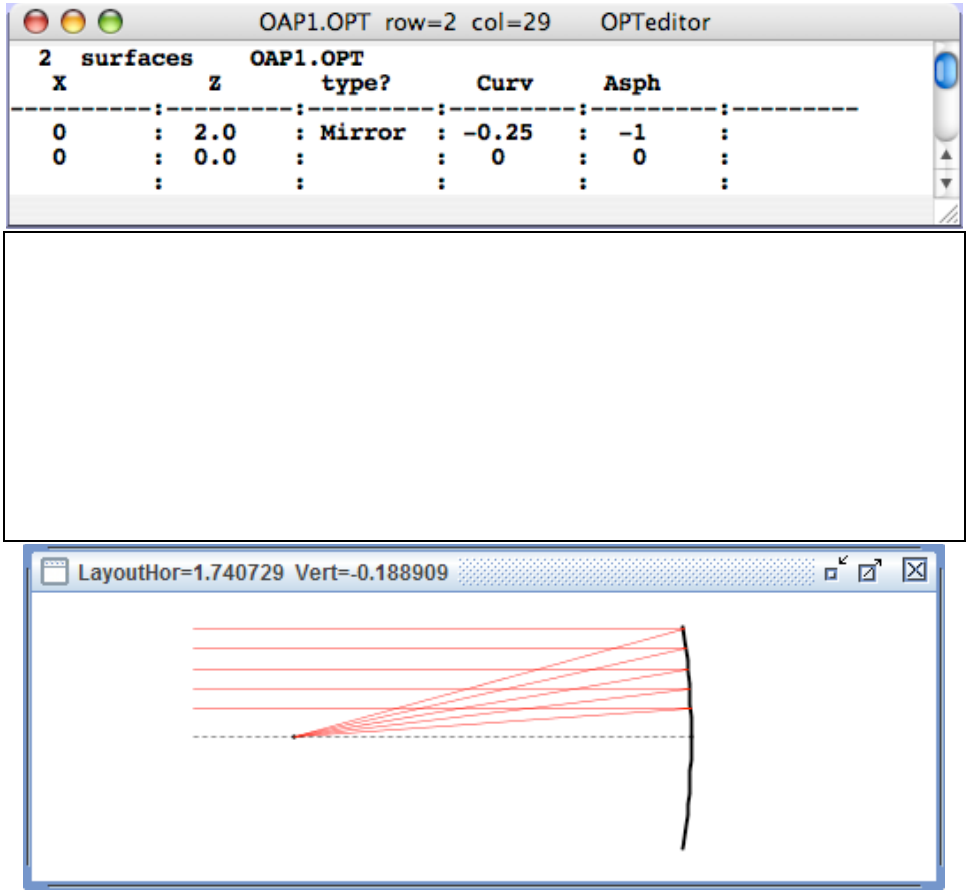


Fig. 9-3: A parabolic mirror is illuminated by a ray group that is offset from its axis. The horizontal dashed line is the lab z axis, which like the other drawing features can be switched on or off at Options::Layout. The default behavior of Layout is to show the entire axisymmetric surface even if only a portion is illuminated.

Example 9.4 This second OAP example shows how to set up the periphery of the mirror. Here we define a mirror Diameter of 0.5 and a periphery offset of 0.35 so the entire mirror lies off its optical axis. The trace results for this ray group are unchanged, but if the beam were enlarged, intercepts beyond the mirror edge would fail.

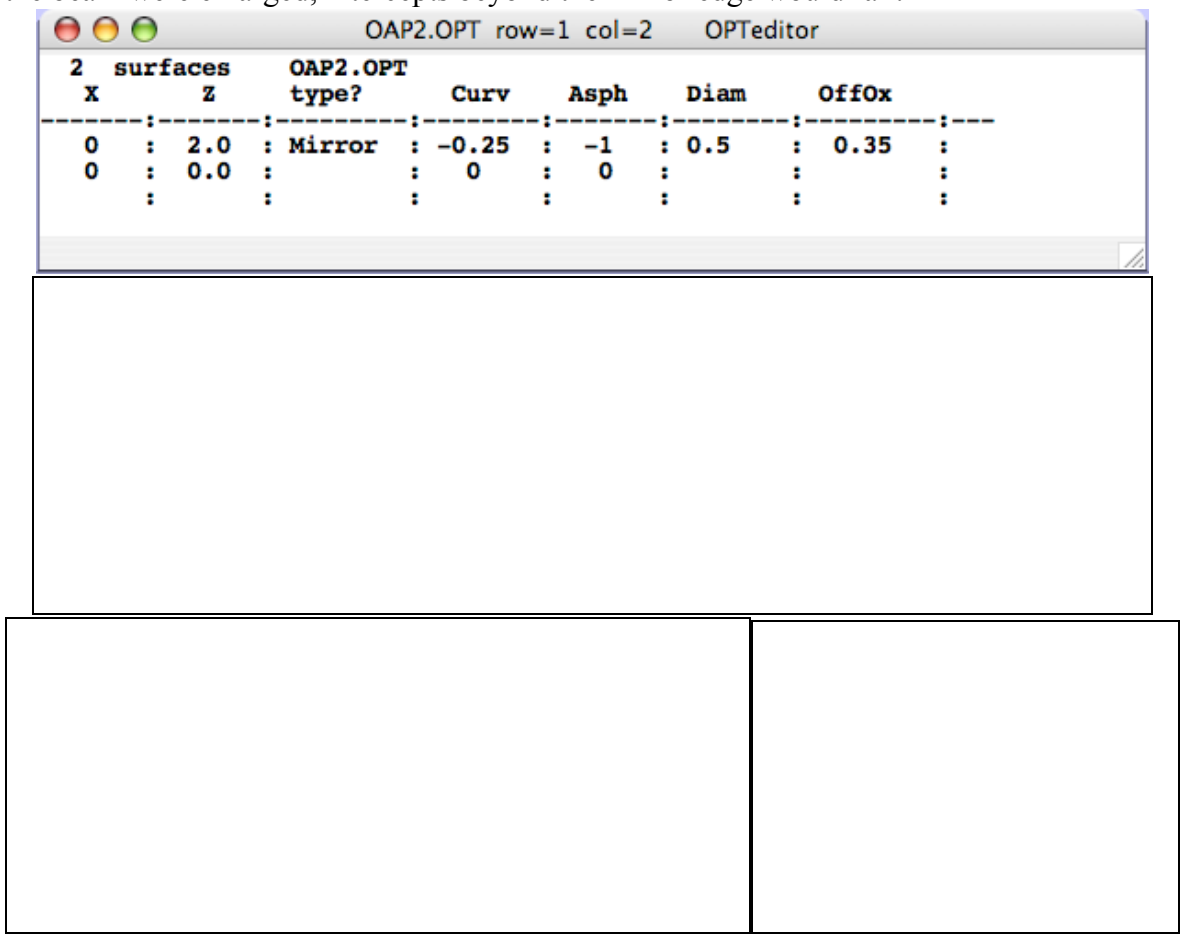


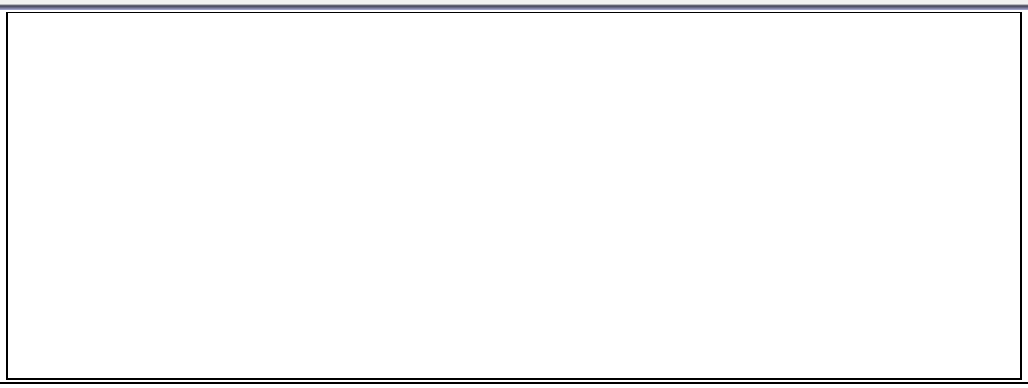
Fig. 9-4: The parabolic mirror is again illuminated by a group of offset rays, but now the periphery of the mirror is defined, using **Diam** and **OffOx** parameters. With the periphery defined, only the portion of the optical surface within its periphery is active, and only that portion is shown in Layout. The mathematical vertex of this paraboloid still lies on the lab Z axis. An oblique Layout view is shown in the above right panel.

Example 9.5 This third OAP example is like the second but now we have moved the rays and the mirror to a new lab coordinate system centered on the rays rather than on the mirror vertex. The ray trace is unaffected except of course the lab frame final intercepts are listed in this new coordinate system.

OAP3.OPT row=1 col=3 OPTeditor

2 surfaces OAP3.OPT

X	Z	type?	Curv	Asph	Diam	OffOx
-0.35	2.0	Mirror	-0.25	-1	0.5	0.35
-0.35	0.0		0	0		



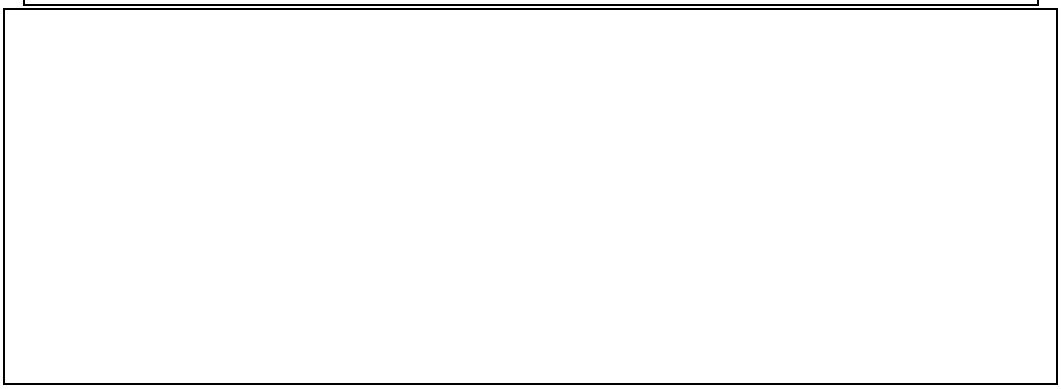


Fig. 9-5: An off-axis parabolic mirror is traced again, this time redefining our coordinate system so that it is centered on the incoming ray group, putting the rays on axis. The mirror vertex and the focus are moved off the lab Z axis to a value X=-0.35.

Example 9.6 Optical surfaces of type **Iris** have these properties of outer and inner dimensions but unlike lens or mirror surfaces do not modify the directions of rays – they serve only to enable or disable a ray encountering the iris. In addition to its outside Diameter and inner diameter properties, an iris can have a set of equally spaced radially-oriented spider legs that block light. The **Nspi** and **Wspi** field headers (see Table 9-1 and Fig 9-6) allow you to enter the number of legs and the leg width into your optics table.

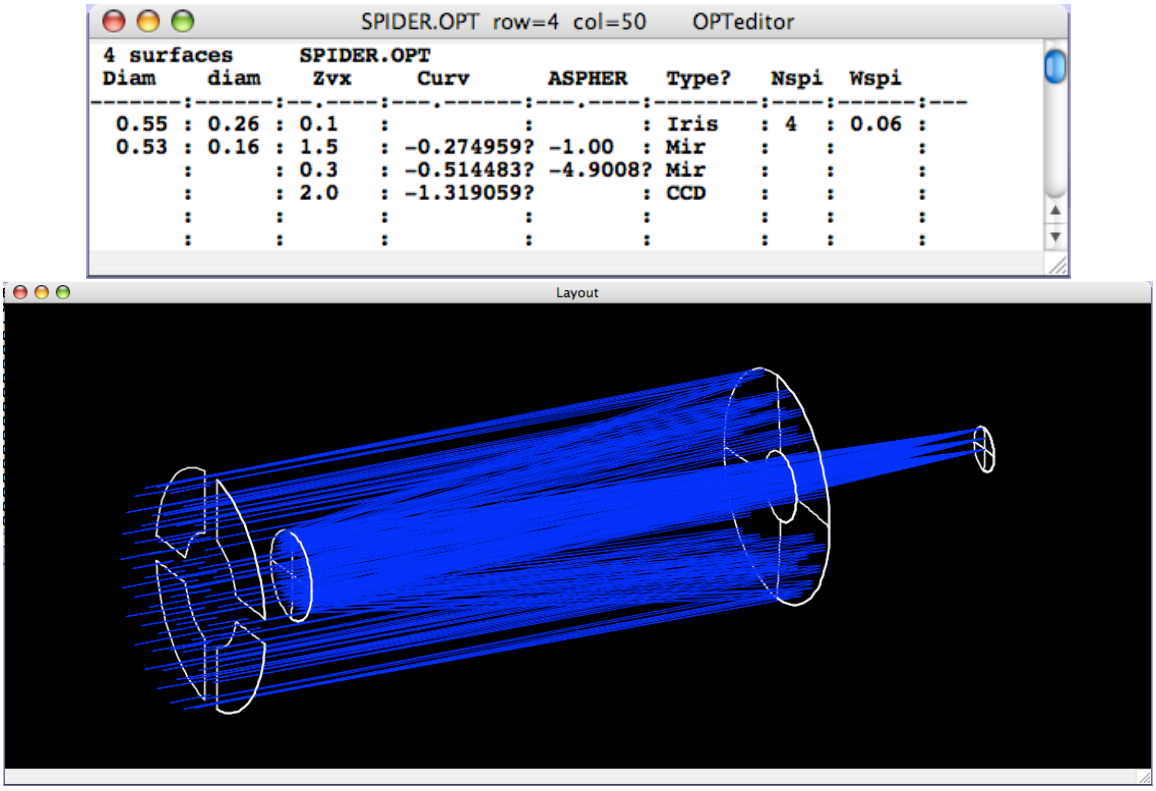


Fig 9-6 A cassegrain imager with a four-legged spider as the initial iris surface. The number and width of the spider legs are specified in Nspi and Wspi respectively. Note that in this optics table the final surface is given a type "CCD" – this is meaningless for the ray trace, since the final surface has no ray bending action. You may assign your final surface however you like, as a memory aid.

Iris surfaces (including spider legs, if any) need not be planar. They enjoy the same geometric freedom that other optical surface enjoy, and therefore can be conic, toric, polynomial, etc. See example 9.10 for a conical spider.

More complicated peripheries can be built up by stacking irises atop any target optic. For example, an octagonal mirror edge can be modelled as a square mirror to which a square iris of the same size, rolled 45°, has been added. Similarly one or more irises with defined inner diameters (but no outer Diameters) can be applied to an optic to create blockages wherever needed. If the surfaces to be stacked are flat, they need not be separated by any distance whatsoever: they may have identical locations and orientations.

Beyond these individual lens, mirror, and iris elements, BEAM FOUR can generate and use arrays of lenses, mirrors, and irises. An array of elements is a flat two-dimensional rectangular grid of identical elements. To set up an array, you will need to declare a Diameter (or separate Dx and Dy fields) to describe the boundary of the rectangular grid, and the numbers Nx and Ny of array elements along the x and y axes. The array types are LensArray, MirrArray, and IrisArray. The lenslets and mirrors are quadratic figures of revolution or cylinders and have parameters Curv, Cx, and Asphericity. The individual irises in an iris array can be circular or elliptical and are described by diam or dx and dy. To make the array elements square or rectangular, put a lower case "s" into their Form field.

Now we turn to some examples with additional details that illustrate these features.

Example 9.7: The files PRISM.OPT and PRISM.RAY show the basics of doing polychromatic refraction using GLASS.MED. The first field of the optics table "Index" calls out the glass name to be looked up as a row in the media table, and the ray table calls out each ray wavelength to be looked up among the fields of that media table.

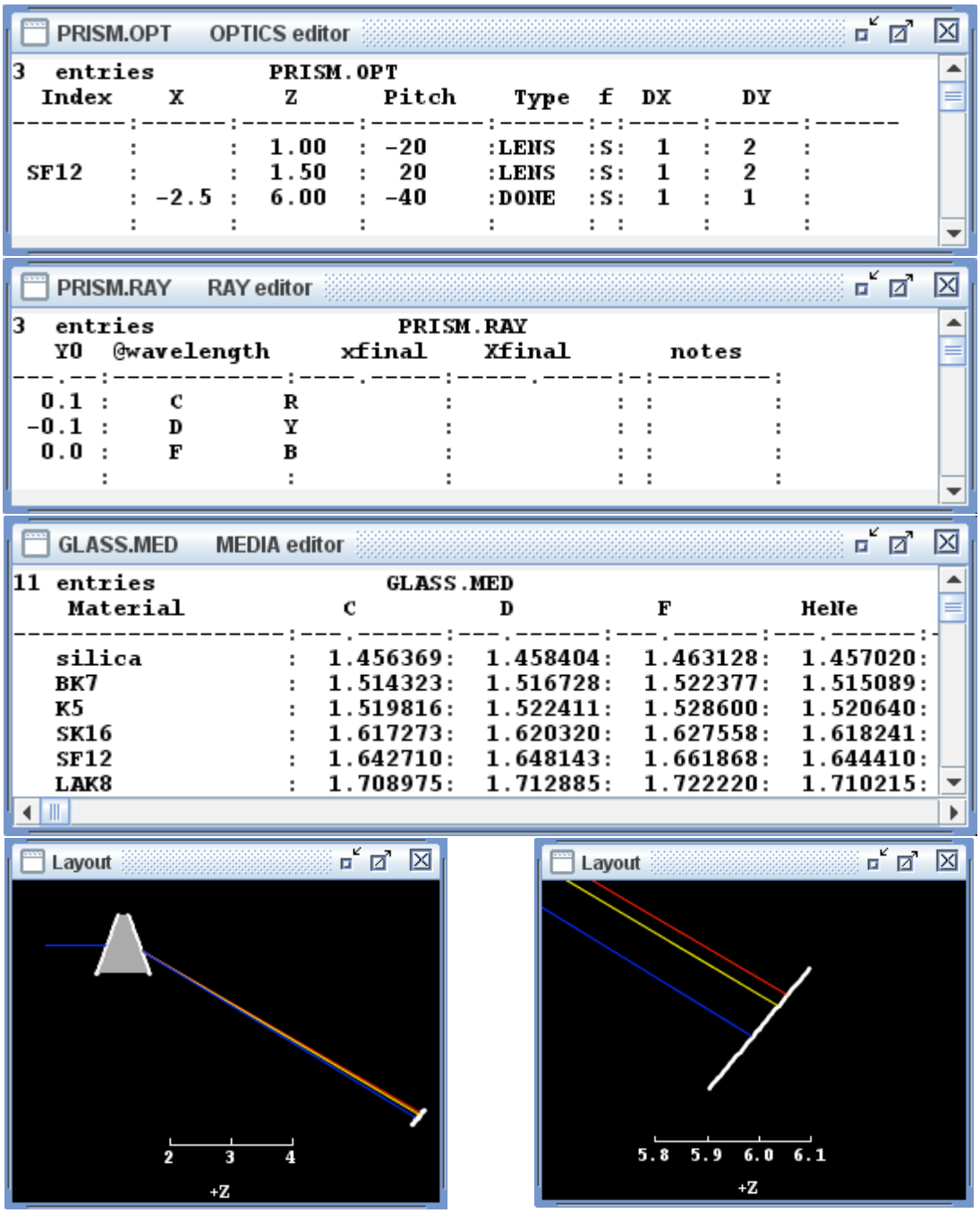


Fig 9-7: A simple prism with rays at three wavelengths disperses its light. In the .RAY table, the Fraunhofer "C" line lies at the red end of the spectrum and is color coded "R" for display purposes; similarly the sodium "D" lines are yellow, tagged "Y", and the "F" line lies at the blue end of the visible spectrum, tagged "B" for display.

Example 9.8: Below, in ACHRO.OPT, the refractive index field is supplying glass names rather than numerical refraction index values, just as in PRISM.OPT above; again the ACHRO files require GLASS.MED to provide this lookup table. The second field "Z" provides Z axis coordinates of the successive optical surfaces. The X and Y data are not specified in this table, and therefore will default to zero so that there are no decenters. Moreover there are no tilt or pitch columns, so the tilt and pitch angles default to zero. This optic is therefore fully coaxial.

The third column in ACHRO.OPT is headed "Curv" indicating the table location for the curvature of each surface. A zero or blank data entry under this header indicates a flat (plano) surface. A positive entry would indicate the surface to be curved towards the local +z direction, and a negative entry indicates curvature towards -z. Because no columns are provided to describe asphericity or shape, and because there are no polynomial or toric descriptors anywhere in the table, the surfaces will all be taken to have the default spherical profile, as with the simple LENS.OPT example shown earlier.

The fourth field in ACHRO.OPT is headed "Mir/Lens" which identifies a column to declare what the surface action is to be. An equivalent header is "Type." Since BEAM FOUR uses only a many characters as necessary to parse your data, any word that starts with "M" or "L" or "T" will serve. Under this header, an entry that begins with "m" or "M" indicates that the surface is a mirror and the trace is to follow a reflection at this surface. "L" or 'l' indicates a refractive lens surface. "I" or "i" indicates the surface should be traced as an iris -- stopping or passing a ray but not changing its direction. Each iris let you specify outer and inner diameters, a format "S" or "s" or "Ss" meaning square or rectangular (default is round), and also a number of equal spider legs and their widths.

"Retro" in your "Type" column specifies that a surface is a retro reflector -- it sends each ray exactly back on itself. This is mathematically impossible (it violates the principle of stationary phase for real rays) but occasionally useful to convert a virtual image into a real image, for analysis. Do not use "retro" in calculating optical path or wavefront error. However a retro surface is very useful if you want to originate rays at an internal pupil: start them there, and send them backwards through your optic to an external Retro surface out front. They will return to those pupil locations exactly and will then trace through your optic. If you are using this feature you may want to make your Retro surface invisible in Layout::Options so the rays appear to be simply incoming.

Your final surface is defined by the guide number at the upper left corner of your optics table. At the final surface, the entry under your "Type" header doesn't matter since the ray won't be redirected further. In our examples we use "film" or "CCD" or "Done" but it could be blank, or anything, since no ray bending happens here.

The fifth field in ACHRO.OPT is "Diameter" which reserves space in the table to show what the design diameter of each optical element is. Diameters that are left blank won't limit the trace in any way. Those that are specified will allow only rays to proceed that

would be transmitted by an actual optic having the specified working diameter. The letter "D" is how BEAM FOUR recognizes this specification; the letters that follow it are simply reminders as to what it stands for.

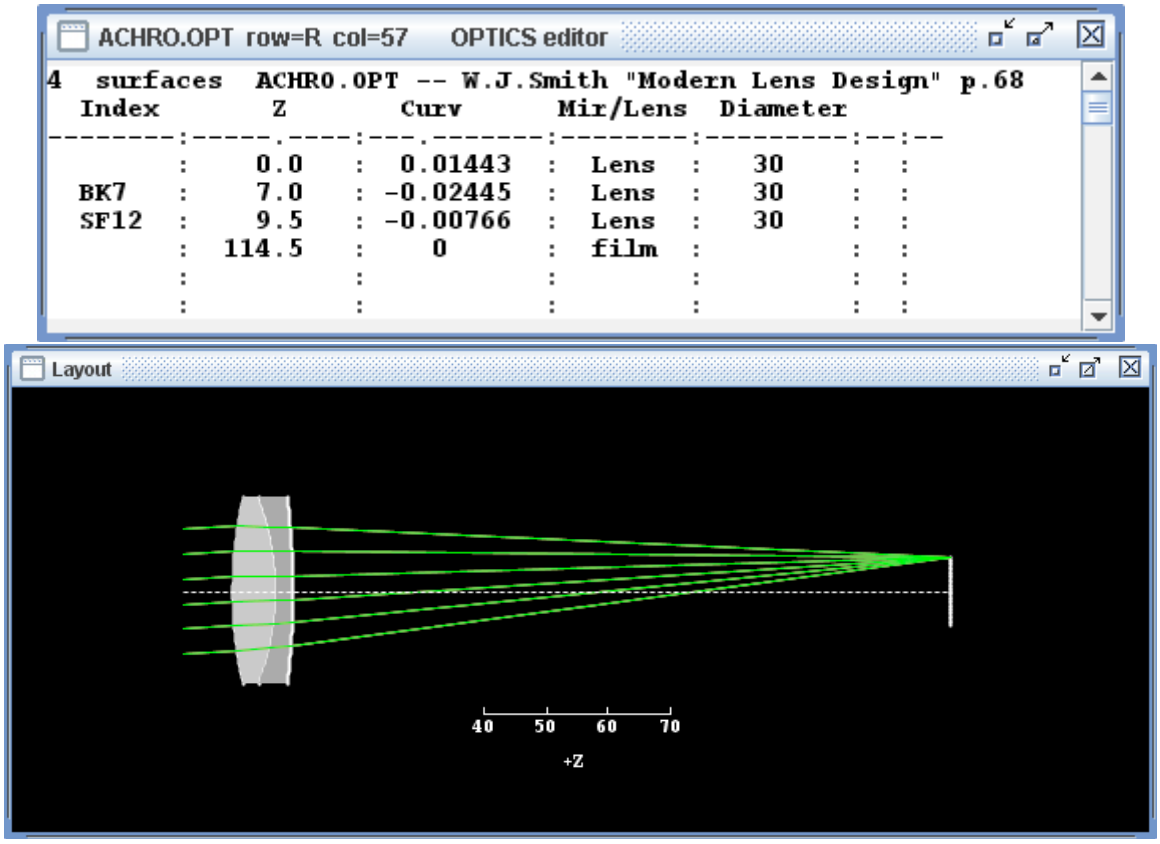


Fig 9-8 **Upper**: Unlike LENS.OPT shown earlier, in ACHRO.OPT the glass names are listed in the "Index" field, so that refractive index information can be looked up in a .MED table when the ray traces are run. Where the glass name is left blank, air is assumed. To accomplish the lookup, each ray in the ray table will need an assigned wavelength. See the discussion of wavelengths in Ray and Media chapters for the details. **Lower:** A layout diagram with ACHRO.OPT and a set of incoming rays color coded red, green, and blue. The rays are drawn in the same order that they appear in the ray table, and these three ray groups have traces that are so nearly identical that the blue set overwrites the others at this moderate layout zoom setting. When you zoom in on the rays you can see that their individual traces are not, in fact, identical.

Example 9.9 A simple Cassegrain telescope with an eyepiece is modelled in the file CASS.OPT shown below. The eyepiece is built from a pair of achromatic lenses.

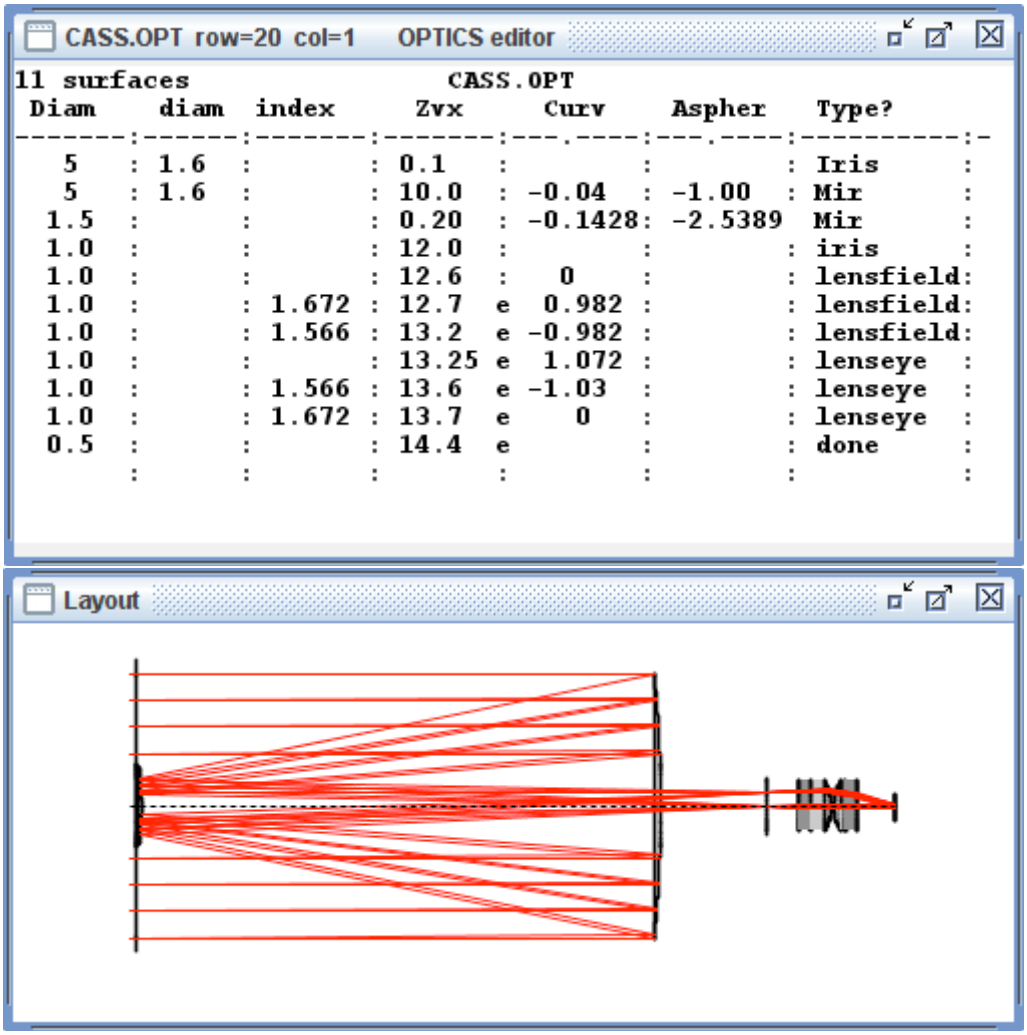


Fig. 9-9 The file CASS.OPT shows an iris with a central blockage and a primary mirror with a central hole. These have a specified outer Diameter, and also a specified inner "diam" (note the lower case "d") within which it is blind. To do a proper study of the vignetting (off axis transmission) you may need a series of iris surfaces that constrain working rays to lie within assigned boundaries.

One feature illustrated in CASS.OPT is the specification of a central blind zone in an optic. Whereas "Diameter" or "Diam" or "D" (upper case D) describe the outer diameter of a circular optic, the lower case "diameter" or "d" specify the diameter of its blind center. This is the second field of the table in CASS.OPT shown above. For elliptical apertures you may specify DX and DY separately, and dx and dy separately also. Rectangular apertures are specified by adding a “Form” column containing “S” and/or “s” for square.

Example 9.10 A more complex telescope of the "three-mirror anastigmat" type is illustrated in Figure 9-10 below. Here, the incident light passes through a strongly-curved conical 3-legged spider iris representing the structure that supports the secondary mirror. Then, rays pass to the primary, secondary and tertiary mirrors to arrive at a focal plane. This final surface is assigned a Type "CCD" but again the final surface type is ignored by the ray tracer: Type has no effect on the ray arrivals at a surface.

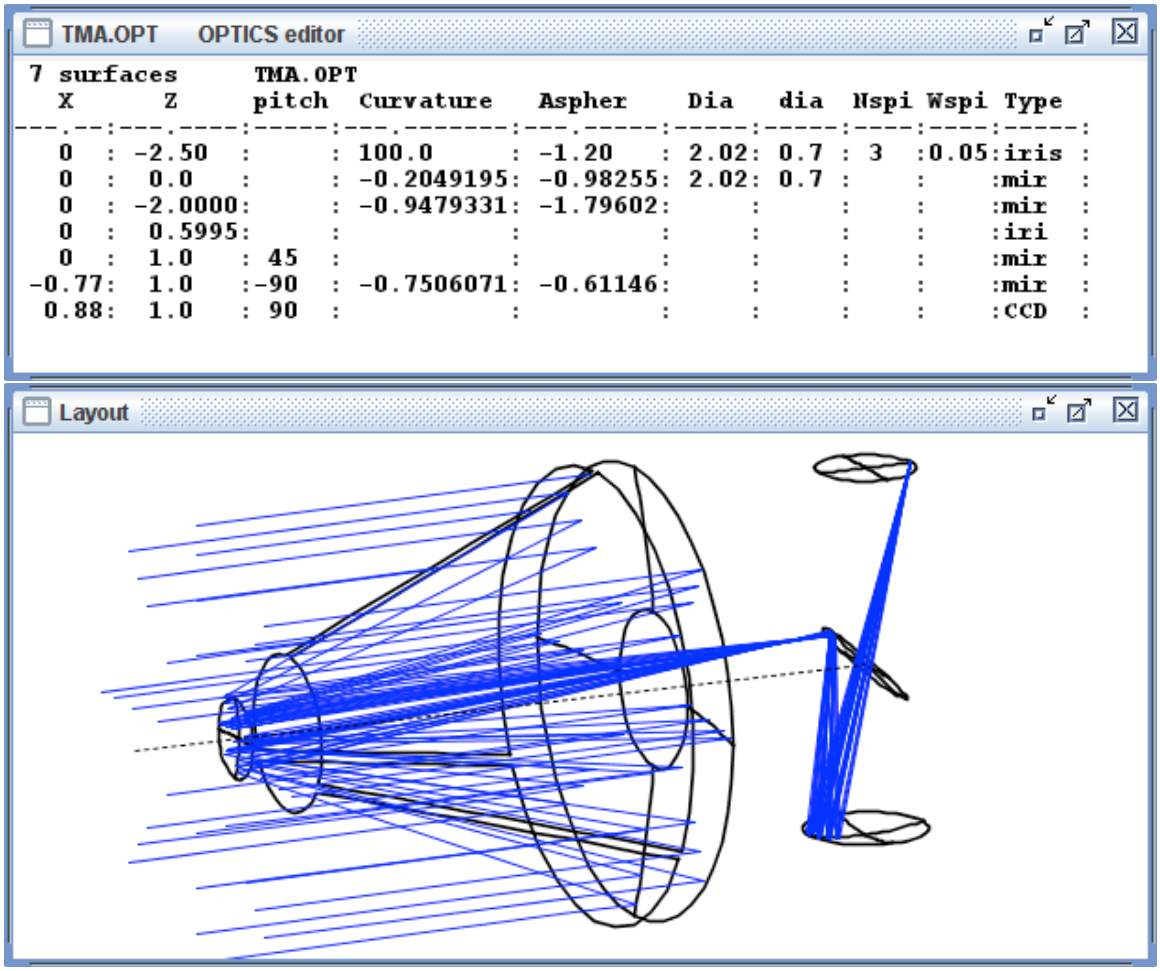


Fig. 9-10: A three-mirror anastigmat telescope with a three-legged spider that models the obstruction caused by a three-legged secondary mirror support.

In this example, the spider is far from planar: it is assigned a huge positive curvature and an asphericity making it essentially a cone. Irises and spiders respond to all the surface descriptors available to active optical surfaces: cylinders, torics, polynomials etc.

Example 9.11 A relay is an important component of a complex system. It moves a real or virtual image from one location to another. A fully reflective Offner relay uses three reflections from mirrors that are spherical and concentric. An example of an Offner relay is shown in Figure 9-11.

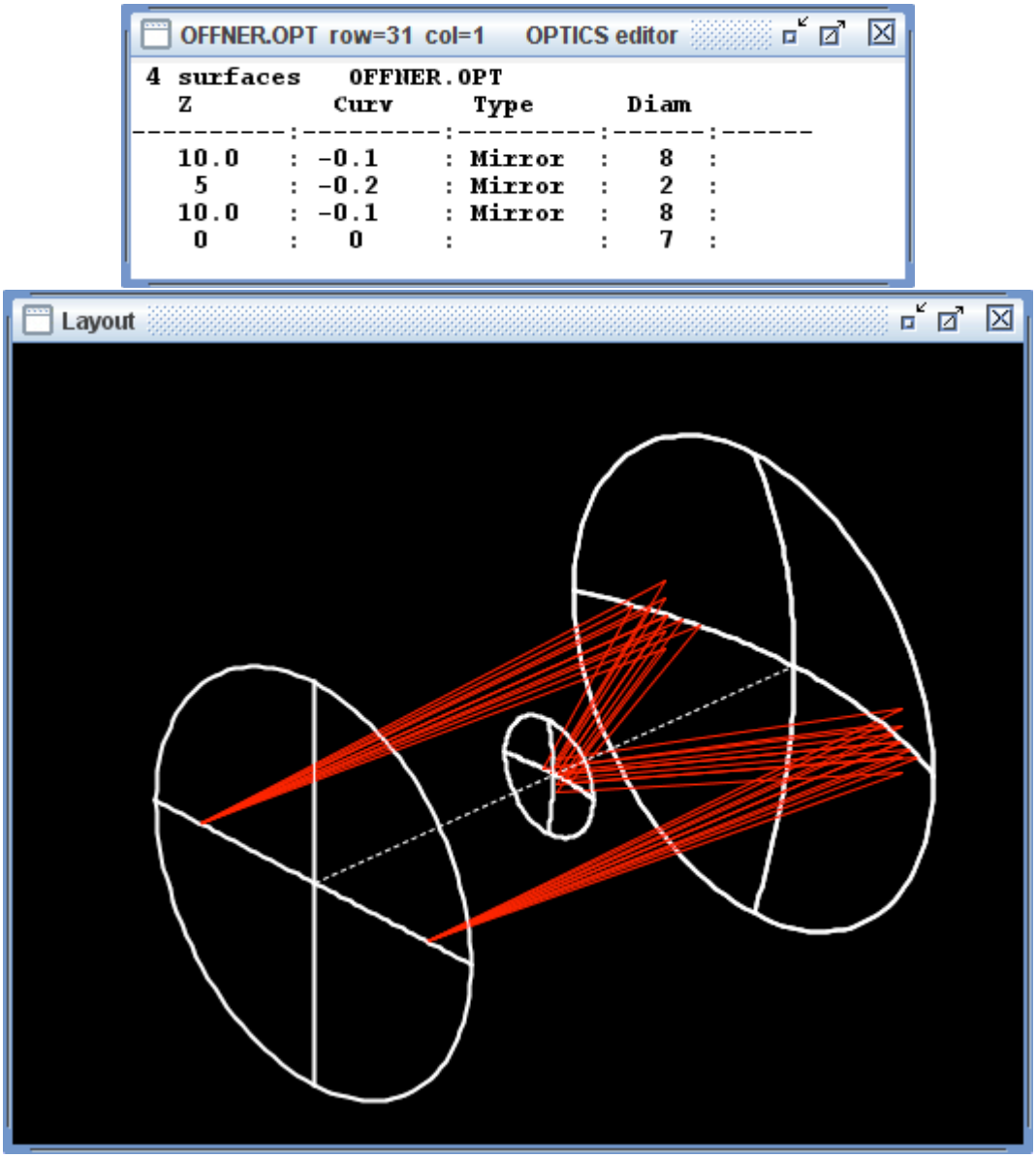


Fig. 9-11 An Offner relay has three reflections from spherical surfaces. The first and third reflections are from a single concave mirror, and the second takes place at a smaller concentric convex mirror having twice the curvature.

Example 9.12 The Offner relay is a convenient demonstrator of the offset dimensioning described earlier in this chapter. Only two zones of the concave mirror are in use in an Offner relay and these can nicely be divided up by identifying two offset portions of the parent sphere. In Fig 9-12 we illustrate the use of the offset OffOy. Notice that the first and third reflective surface have the same Z, Curvature, and Diam --- they differ only in their y axis offsets OffOy.

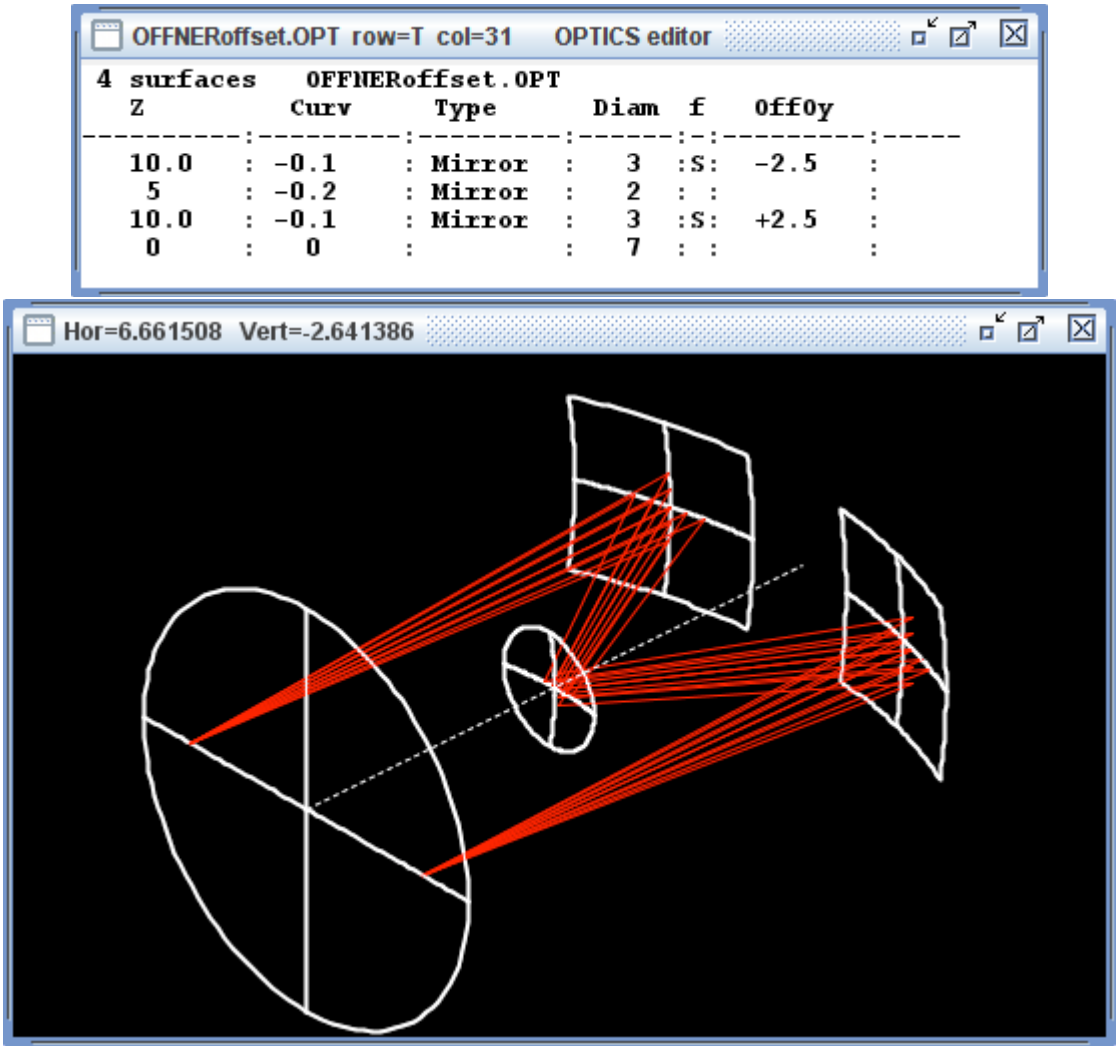


Fig. 9-12: Similar to 9-11 but the active portions of the concave mirror have been isolated by means of defining OffOy, by setting Form="S", and choosing smaller Diameters.

Example 9.13 The Hartmann test for concave mirrors reveals slope errors over the mirror surface. The setup places a point source of illumination at or near the mirror's center of curvature, and an opaque screen with an array of holes located near the mirror.

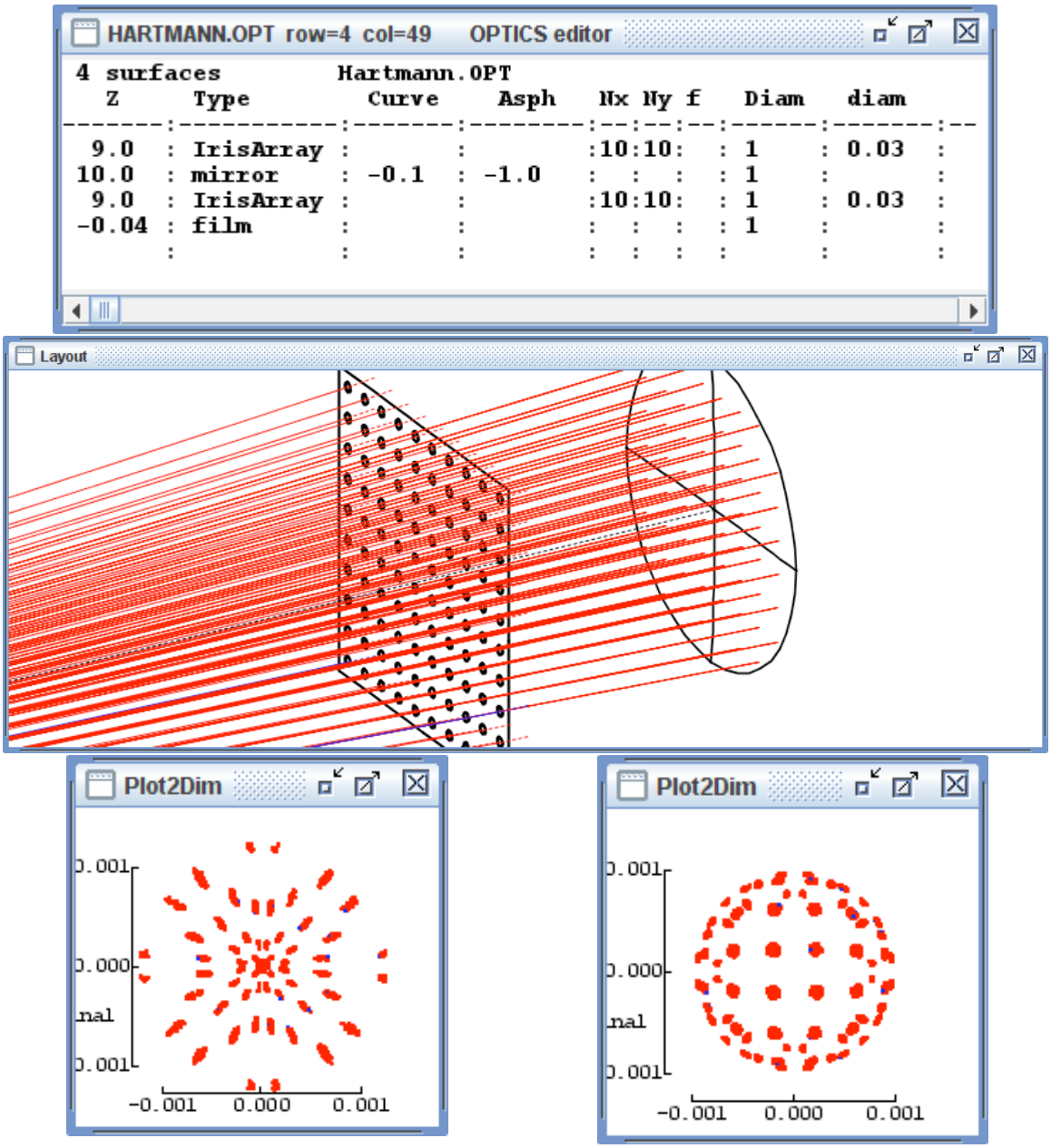


Fig. 9-13: A Hartmann test for slope errors in a concave mirror involves a perforated screen near the mirror, and a light source and imager near its center of curvature. Top: HARTMANN.OPT file showing a parabolic mirror fed by an IrisArray with 10x10 holes in the diverging and converging light. Center: portion of a layout showing the screen and mirror. Bottom: the spot diagram at paraxial focus, and the spot diagram with the image plane set to Z=-0.04 units.

An imager near the center of curvature records the pattern of beamlets in the converging light returning from the mirror. For an ideal spherical mirror, the pattern is regular and symmetrical both inside and outside focus. For an ellipsoidal, parabolic, or hyperbolic mirror the patterns inside and outside focus differ in a quantitative manner. At paraxial focus, the spots are highly concentrated at their center; at edge focus the paraxial rays cluster at the edge of the diagram.

The Hartmann test is useful because it is very sensitive to small slope errors in the mirror being tested. To demonstrate this, add a few parts per million of a Zernike coefficient (Appendix 3) to the mirror and see the remarkable changes to the spot pattern.

Example 9.14 A Shack-Hartmann wavefront test uses an array of lenses to image a grid of wavefront sections onto an image sensor. If the wavefront is flat, the rays will be parallel and the image spots will form a regular grid defined by the lenslet grid. If the wavefront is irregular, the spot pattern will be irregular. Figure 9-14 shows a 5x5 lens array set up for this task.

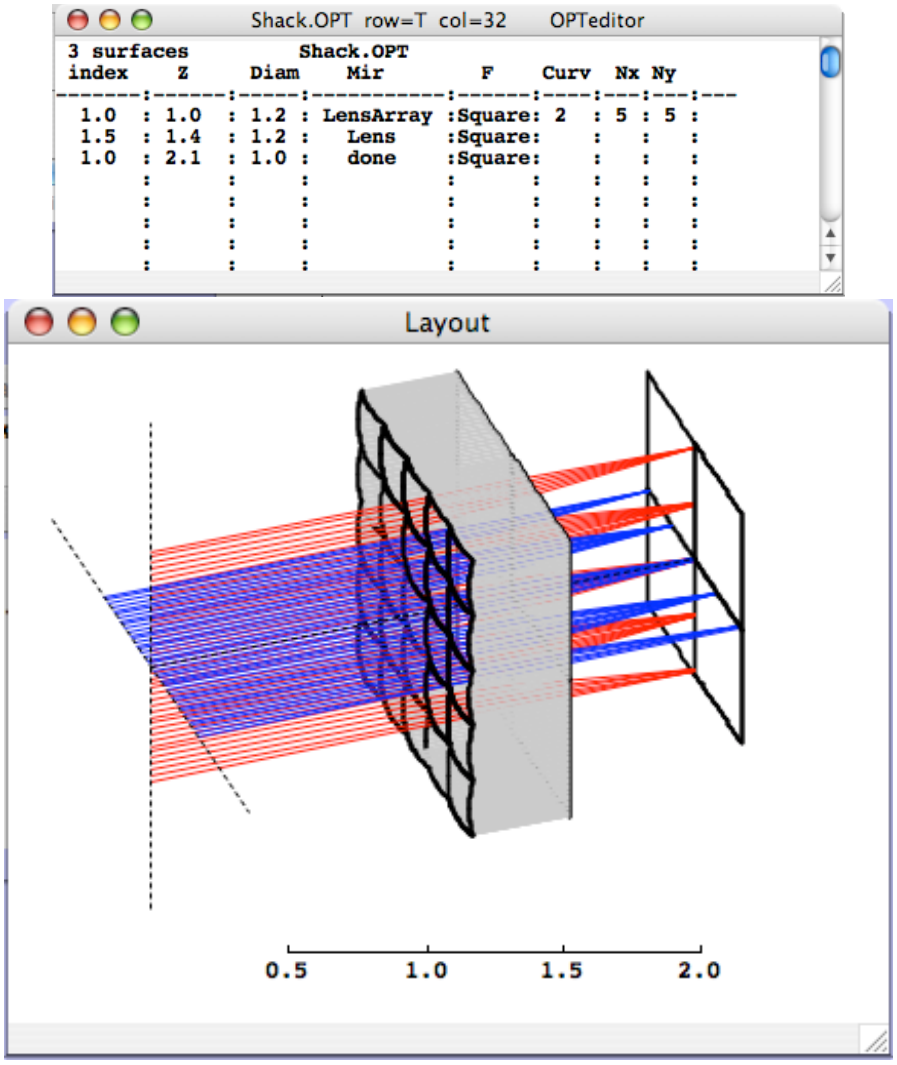


Fig. 9-14: The Shack-Hartmann sensor uses a precise grid of lenslets that bring parallel rays to form a grid of image surfaces, as shown.

The Shack-Hartmann test is far more efficient in its use of light than the Hartmann test because nearly all the incident light is sent on to the image. This fact allows its application in real-time wavefront sensing where speed is essential and light levels are often limited.

Example 9.15 A compound mirror may be described as a ***group*** of surfaces. Each group acts as a single ray intercept opportunity in the sequential flow of the trace. The specific surface intercept within a group is chosen by each ray according to "or" logic: its intercept must fall within the element's Diameter and have the shortest ray travel distance, i.e. the intercept chosen is the nearest legal surface intercept. Use a Group column to assign a common name to elements in each group. Grouped mirrors must appear successively in the .OPT table, and each member must have a specified Diameter to establish its legal intercept zone. Fig 9.15 below shows an incoming beam divided into three portions by three mirrors, each with its own orientation and other properties. The elements of a group can be separately tuned using AutoAdjust (Chapter 20).

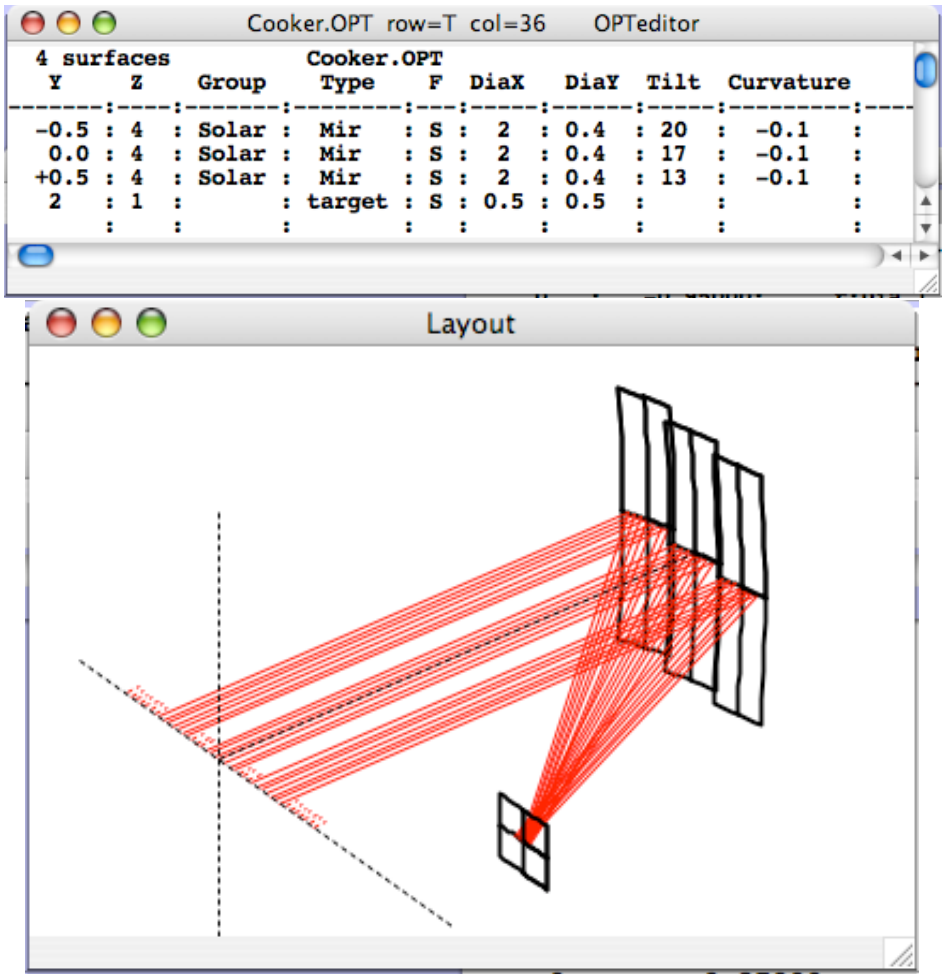
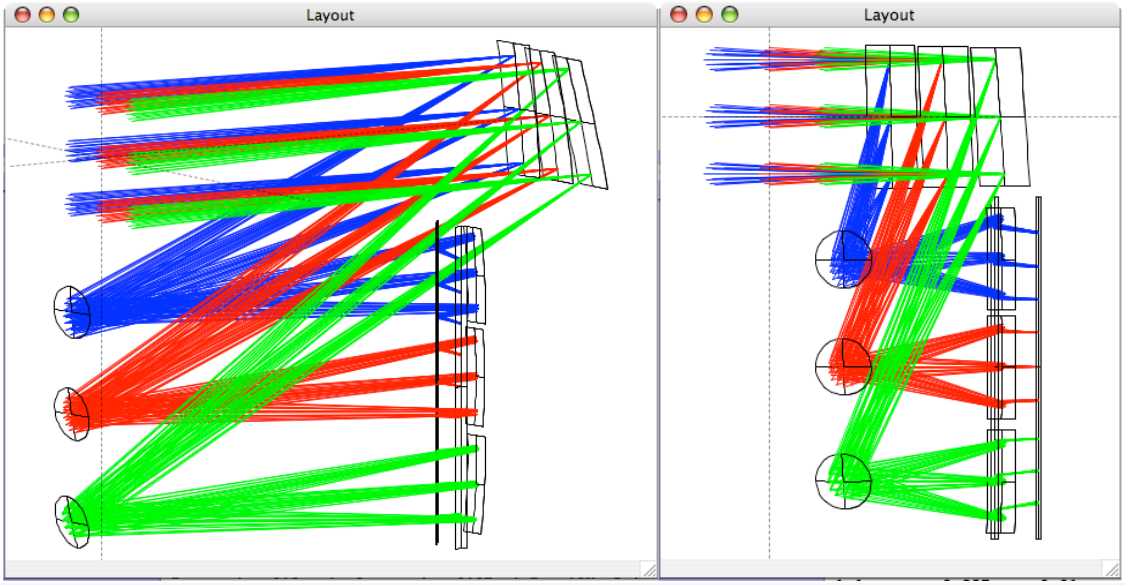


Fig 9-15: Mirror Groups. Here, three mirrors form a group that gathers light from an incoming beam. Unlike an array, each mirror surface is separately specified.

When using groups, the .RAY table output fields and the graphic diagnostics report ray results based on the ***group number***, not on surface number. In the example above, there are just two groups of surfaces: the first group "Solar" comprising three mirrors, and the second single surface (a group of its own). Rays that arrive at the final surface will report "OK 2" since they successfully landed on the final group, #2.

Example 9.16 An image slicer is an optical tool that divides a two-dimensional object field into slices and rearranges them end-to-end. It's used in astronomy to format an extended field --- a galaxy for example -- for spectroscopy. We use 3 groups of mirrors.



Slicer.OPT row=6 col=62 OPTeditor

11 surfaces Slicer.OPT

X	Y	Z	DiaX	DiaY	Pitch	Tilt	Type?	Group?	F	Curve
	-1	10	2.5	0.9	6	7.0	Mir	Slicer	S	-0.035
	0	10	2.5	0.9	10	4.6	Mir	Slicer	S	-0.035
	+1	10	2.5	0.9	14	2.2	Mir	Slicer	S	-0.035
-2.5	2.0	-2	1	1	6	7.0	Mir	Pupil		0.080
-4.4	2.0	-2	1	1	10	4.6	Mir	Pupil		0.080
-6.4	2.0	-2	1	1	14	2.2	Mir	Pupil		0.080
-2.5	2.0	7	1.8	0.5	3	6	Mir	field	S	-0.100
-4.4	2.0	7	1.8	0.5	0	6	Mir	field	S	-0.100
-6.4	2.0	7	1.8	0.5	-3	6	Mir	field	S	-0.100
-4.4	2.2	6	6	0.25		-39	Mir		S	
-4.4	3.0	6	6	0.25		-90	slit		S	

Fig. 9-16 shows a schematic three-path image slicer. The input field has a 3x3 grid of spots focused onto the three slicer mirrors. Those illuminate three round pupil mirrors (left side of upper left oblique view) which in turn relay the spots onto three field mirrors. The nine output beams are seen in the upper right end view.

In the highly simplified example Slicer.OPT shown here, there are just three slices, defined by three mirrors located at the focal plane of a telescope. We color code the rays blue, red, and green to help visualize the slices. Each slicer mirror is separately pitched, tilted, and curved to illuminate its own dedicated pupil mirror. These pupil mirrors do all the work: they relay the incoming spots into the output zone, lined up in a vertical stack of images. Each pupil mirror has a dedicated field mirror to make the outgoing light converge into the pupil of a downstream spectrograph. A final narrow flat mirror redirects the light from all groups out of the side of the apparatus.

Example 9.17 Another use of groups is to allow a pair of hemiellipsoids to be joined, making a complete ellipsoidal reflector. In BEAM FOUR, surface profiles are defined as single-valued functions z(x,y). This definition safely avoids false ray intercepts with phantom hyperboloid sheets or farside ellipsoid vertex hits. Sometimes however you may want to use that farside hemiellipsoid, as in this example (Fig 9-17). Here, the first reflector surface listed is a hemiellipsoid whose vertex is at Z=0. Its mate is a second mirror at Z=8 with the same shape but with opposite curvature. The focus locations are given by the conic equation formulas (see Appendix 1) based on curvature and shape.

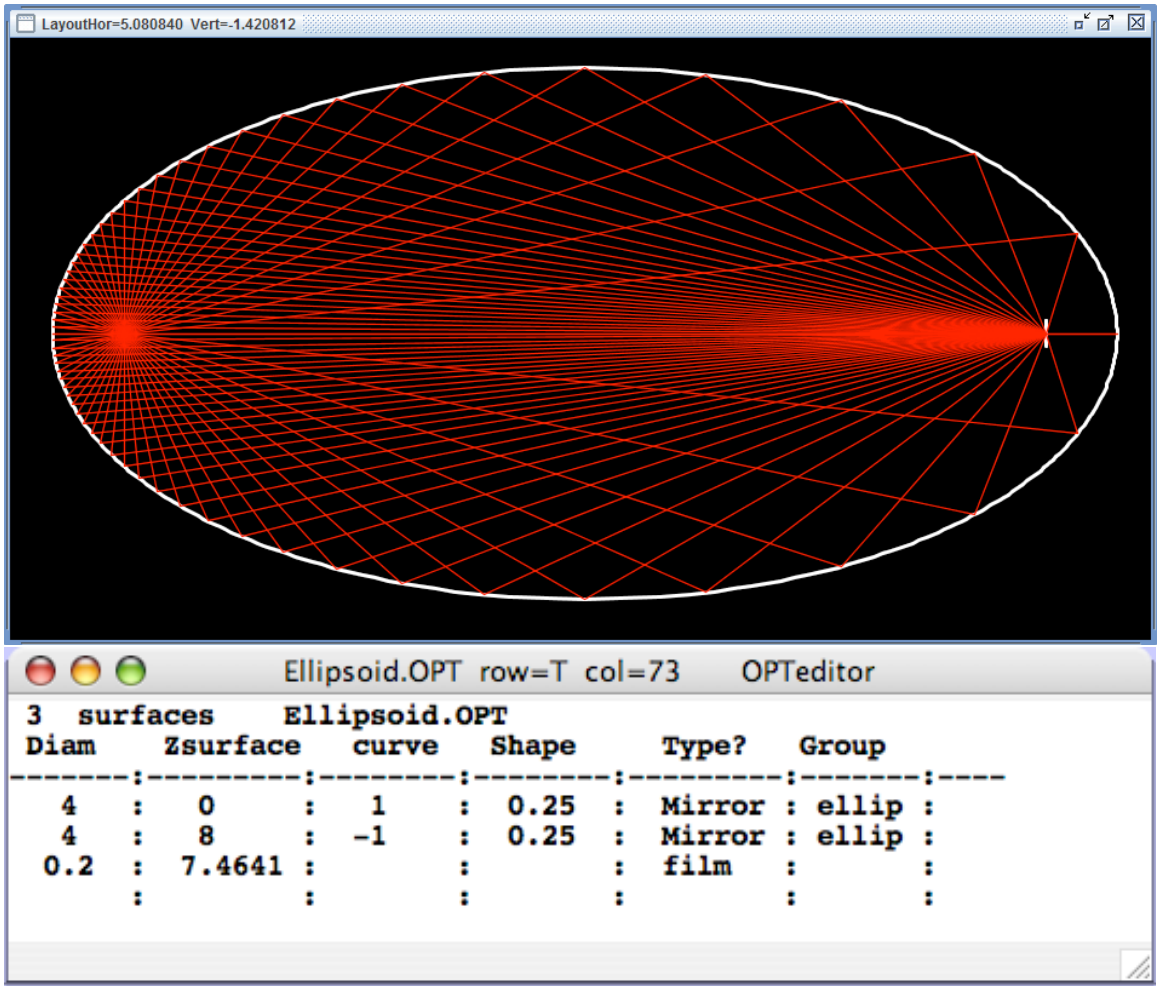


Fig 9-17 Top panel: layout view of the pair of joined hemiellipsoids, with rays emitted from one focus into all directions. They arrive at the opposite focus. Middle panel: the .OPT file showing the Group column that joins these two surfaces into a common "or" logic relationship so that an intercept with either one is a valid reflection. The group name is unimportant but members of a group must have the same name and must appear in successive rows in the .OPT table.

Example 9.18 Forward scattering surfaces are built into BEAM FOUR. As an example we show an ideal hyperbolic lens collimator, a forward scatter surface that deviates rays in direction by an amount of 1.0 degrees RMS in each of the U and V directions, and a hyperbolic lens camera that records the deviated ray positions. If you want a reflective scatterer, just combine this forward scatter function with a mirror or retroreflector. Wavefront error determinations are not compatible with nondifferentiable optical path disruptors like this one.

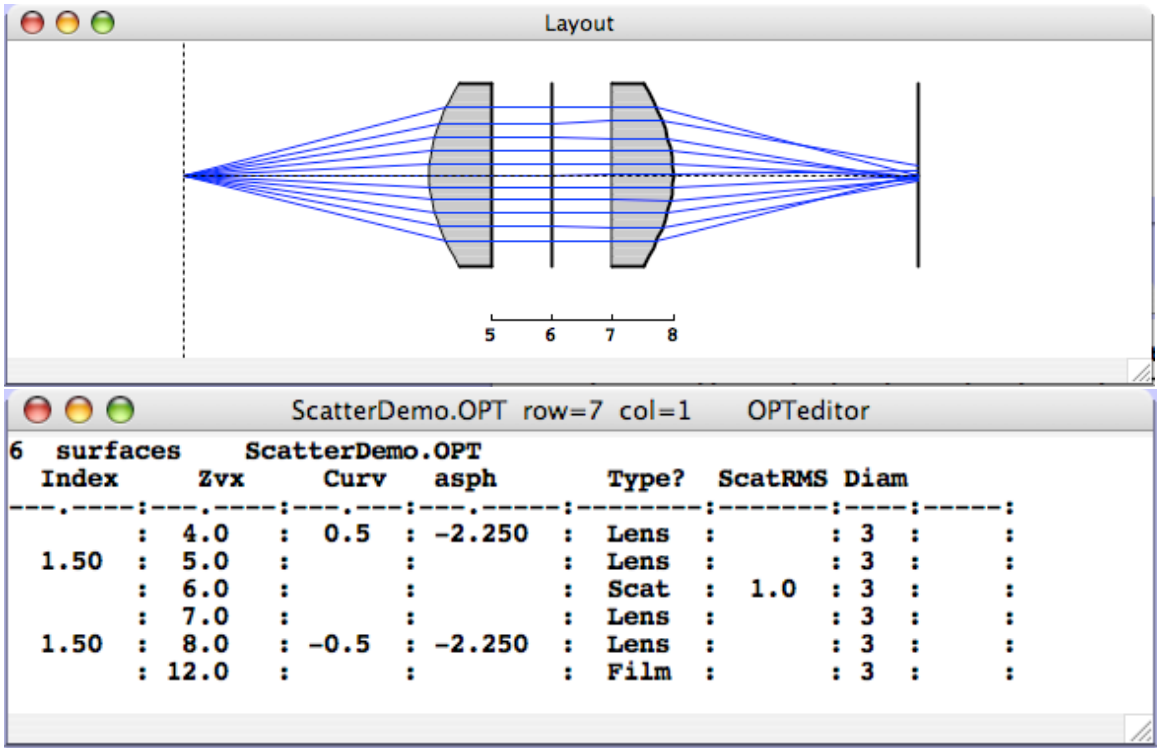
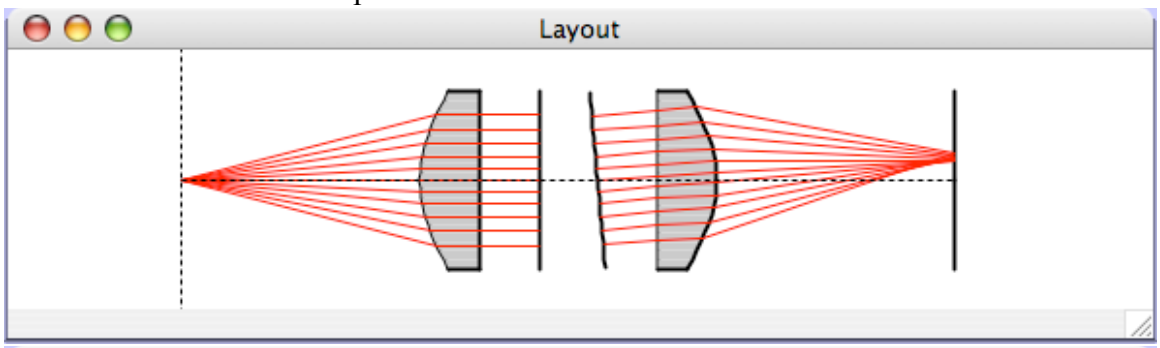


Fig. 9-18 shows a layout of an idealized collimator lens, a scattering surface (surface #3) and an idealized camera lens. The Type designator is "Scat" identifying surface 3 as a forward scatter surface, and its parameter "ScatRMS" is where you specify the RMS 1-D angle in degrees. Here it is set to 1.0 degrees.

Example 9.19 A coordinate break is sometimes needed as a programmable interface between two optical subsystems, especially if each is complicated and it is easiest to locate them on a common axis even though there may have to be some adjustments made in their relative displacement or orientation. As an example we show below an ideal hyperbolic lens collimator on the left, a coordinate break with a z-axis gap and an angle, and a hyperbolic lens camera. The collimator and camera are coaxial in the table, to simplify the specification, but the rays are re-aimed by the coordinate break, introducing an effective bend into the optic.



CoordBreak.OPT row=12 col=43 OPTeditor

7 surfaces CoordBreak.OPT

Index	X	Z	Curv	Asph	Type?	Pitch	Diam
		4.0	0.5	-2.25	Lens		3
1.5		5.0			Lens		3
		6.0			CBin		3
		7.0			CBout	5	3
		8.0			Lens		3
1.5		9.0	-0.5	-2.25	Lens		3
		13.0			Film		3

Fig. 9-19 shows a layout of an idealized collimator lens, a coordinate break and an idealized camera lens. The coordinate break can introduce displacements X,Y, and Z as well as reorientations of the ray group Tilt, Pitch, and Roll. Rays arriving at **CBin** are emitted from **CBout** with the same local coordinates.

The mechanism of the coordinate break is mathematically simple. At the first surface **CBin,** the local ray coordinates {x,y,z,u,v,w} are captured, and the ray propagation is interrupted. Then, at the immediately following surface **CBout,** the local ray coordinates are taken from **CBin** and emitted from the **CBout** frame as if they were local to **CBout**. This is a kind of a transporter jump: without having to propagate, the rays find themselves in a new coordinate environment, ready to go to work.

If **CBin** and **CBout** were at the same location and orientation, they would do nothing. But if they are located differently, or oriented differently, or both, then rays would vanish from their **CBin** arrival locations and find themselves re-emitted from **CBout** in a new state.

There are many uses for coordinate breaks. For example when tolerancing an optic, it is often of interest to see what the effect is by introducing an angle, or lateral displacement, or axial displacement, into the optical train. Of course this may always be accomplished by modifying the lab coordinates of the individual optical surfaces, but if they are grouped together it is far simpler to move the light rather than to move the surfaces. A spacing change can be introduced by using a **CBin/CBout** pair with differing Z locations, while keeping the nominal prescription unchanged. In this way the tolerance work is isolated from the nominal design parameters.

Another example is to provide a rotation about the optical axis for a group of successive system elements. Many optical elements are axisymmetric: rotation will not affect them if they are aligned coaxially. But prisms (such as Risley prisms) deviate light, and their rotation in surface groups is a key performance requirement. It is usually easier to rotate the light, using coordinate break Roll angles, than to reconfigure the tilt and pitch of the prism surfaces (their individual surface rolls do nothing if they are *locally* axisymmetric).

Coordinate breaks are usually planar, meaning the **CBin** and **CBout** surfaces should have zeros for all their curvature parameters. For most purposes your CBs should be planar. However BEAM FOUR does not forbid CBs being curved, and with care you may use curved breaks to fit between closely spaced curved surfaces. Be sure that the curvatures and x,y,z offsets are identical, to eliminate the possibility of introducing false power into the system. Pure roll redefinitions are safe to use in this curved coordinate break environment.

Other Examples are presented in the collection of demo files that are supplied with each BEAM FOUR distribution. Chapter 28 lists many of these files with short descriptions. They illustrate holographic optical elements, Zernike polynomials, and a variety of other useful optical items. Those files include a group of null test files whose purpose is to evaluate the accuracy of ray traces: if the optic being traced has an exact analytic solution, a null test file can be created that demonstrates how closely this solution is achieved. Use the null tests to evaluate BEAM FOUR and other ray tracers you have.

Chapter 10: Rays and Ray Tables

A ray table tells how your optical system is to be illuminated, that is what the ray coordinates are that you want launched into your optic. These are ***input data.*** It is also usually set up to show selected ***output data:*** information about ray positions and directions at specified internal surfaces within your optical system, and/or at your final surface where your trace ends. Because it combines input and output ray data, the ray table is the most important way to summarize the results of a trace. An example of a ray table is LENS.RAY, shown below. It illuminates a simple lens with a fan beam of rays originating at X0=+0.5. Many other examples are provided with BEAM FOUR.

Ray tables are text files. They may be copied to the clipboard from which they may be combined with other files, inserted into e-mails, word processed, and generally treated as you would treat other text. Tables generated by other programs can be read into BEAM FOUR using its **File:OpenRays** function. There is a particularly convenient way to perform i/o to and from spreadsheets via the clipboard as described in Chapter 26.

To load an existing .RAY table, choose **File:OpenRays** or simply drag it from its directory onto the BEAM FOUR workspace. Or, to load an empty ray table, choose **File:NewRays**. In either case an editor window will open on screen, with the table editor functions outlined in Chapter 8.

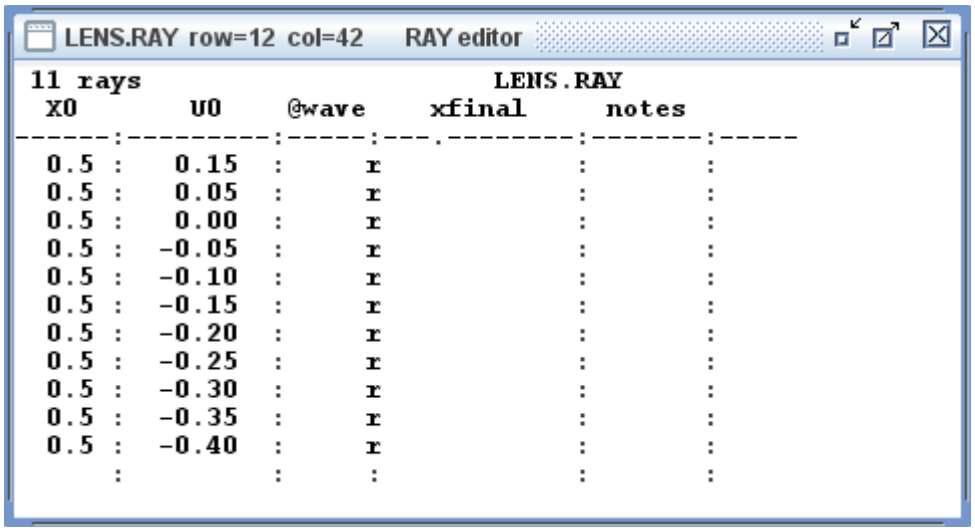


Fig. 10-1 The file LENS.RAY demonstrates a simple fan beam. All rays here start at at height X0 = 0.5 units above the axis of symmetry (the Z axis). They are aimed into a range of directions given by the list of U0 values (see text). Individual ray wavelengths are not specified, but the tag character "r" on each ray wavelength has BEAM FOUR colors each ray red when drawn in a spot diagram or layout diagram. The xfinal field represents the final ray height. Its presence here will capture the xfinal data when a ray trace is run using Run::InOut. The "notes" field gives an abbreviated explanation of the fate of each ray -- see Table 10-1.

Like optics and media tables, a ray table has a three line preamble followed by data records. Each line in the table represents one ray in your illuminating beam. Table entries below your declared number of rays are ignored by BEAM FOUR: if your table has 800 rays in it, but the control number in the upper left corner says "11," then only the first 11 rays will be traced. You can start with a tiny ray group and add complexity later.

Title Line (line #1): The top line in the preamble is the title line: its initial characters should specify a number between 1 and 999, the number of rays to trace, and the remainder of the title line can be used for any purpose. In our examples we put the file names into the title, but you may use this title area for any purpose that you find useful.

Ray Field Headers (Line #2): The second line of every ray table carries the ray field headers. BEAM FOUR uses this information to figure out what data you are specifying and what output data you want to have computed and saved in the table. The first two or three characters of each field header word are significant to BEAM FOUR, although in our examples we usually add enough extra characters to make these abbreviations more easily recognizable to people. Leading and trailing blanks in the headers are ignored. There is no requirement to place your fields in any particular sequence across the width of the table, but in our examples we usually keep the input data towards the left and the output data towards the right. Fields whose headers are blank can be used as separators or for your own notes or comments.

Some of these fields (in this example X0, U0, @wave) are ***input*** data to BEAM FOUR. Ray starts that are not specified in the table (for example: Z0 -- what is its value?) are assigned reasonable default values, typically zero, that are listed below. The idea is to have you provide only the non-default information in the ray table.

The other fields (in this example: xfinal, notes) reserve space for ***output*** data. You use these by typing in the field header names that you'd like to see be presented when your ray trace runs. Leave the data fields blank since they will be filled in by BEAM FOUR when you request a trace using the Run:InOut menu item.

In the example LENS.RAY shown above, the first field header is X0 which means that data in this field will be understood as X coordinate ray heights at surface zero -- the starting surface. The optical surfaces are numbered 1, 2, ... up to the final surface, with zero referring not to an optical surface but rather the ray start position from which the rays are launched before any surface intercepts have occurred. So, X0 will carry input data that you specify, telling BEAM FOUR each ray's X starting coordinate. In this example all the ray starts have the same value, 0.5 units, so all rays start at the same height off the Z axis.

In this example there is no Y0 or Z0 field. So, these default to zero for all rays.

The header for the second field in this example is U0. This is a list of the ray starting directions, and the data furnished gives a fan of ray directions. Positive values of U0 are ascending rays and negative U0 values are descending rays. U0 is a direction cosine, the

component of the unit ray direction along the X axis. V0 and W0 are along the Y and Z axes.

The ray direction V0, by its absence, defaults to zero. W0 too is absent; it is usually allowed to default to the value +sqrt(1 - U_0^2 - V_0^2) that completes the unit direction vector. Its default sign is positive, i.e. rays are initially launched into the +Z direction rather than the -Z direction. (These various defaults can be modified using Options::RayDirections as described in Chapter 24 "Options.")

In this example the third field "@wavel" is a place to specify the wavelengths of the separate rays. Here these wavelengths are blank, which is perfectly OK for a monochromatic ray trace where the glass lens refractive index is specified numerically. However there is a tag field in use: the tag "r" marks each ray's color as red when displayed in layouts and spot diagrams. For other color options, see "tag fields" below.

In this example the fourth field, headed "xfinal," is a request to see the individual ray x values at the final surface, whatever its number might be. The final surface number is the guide number appearing in the upper left hand corner of your optics table. This means that even though your optic might be complicated and have many successive surfaces, you can trace it partway by specifying whatever final surface you want there.

In this example the fifth field, "notes", is an output field. It provides a place for BEAM FOUR to furnish an abbreviated explanation of why any ray failed to reach its final surface when Run::InOut is run. It is very useful in debugging a ray trace. For example if you get a blizzard of "Dia 02" notes, you will want to enlarge the specified Diameter for surface number two, or re-aim your rays to allow them to trace through that surface. See Table 10-3 below for further details.

Generally, if the first character of a ray table header is X, Y, Z, U, V, A, W, or P, or their lower case equivalent, that determines the kind of ray coordinate for that field (see Chapter 6, "Coordinate Systems"). Upper case is lab frame. Lower case is local surface frame, displaying coordinates of each ray with respect to the vertex position and orientation of that surface.

Table 10-1: Ray Field Headers for Input Fields	
X0, Xinitial, Xi, ...	Lab frame X coordinate for ray start
Y0, Yinitial, Yi, ...	Lab frame Y coordinate for ray start
Z0, Zinitial, Zi, ...	Lab frame Z coordinate for ray start
U0, Uinitial, Ui, ...	Lab frame U coordinate for ray start
V0, Vinitial, Vi, ...	Lab frame V coordinate for ray start
W0, Winitial, Wi, ...	Lab frame W coordinate for ray start
Xg, Yg, Zg, Ug, Vg, Wg	Lab frame goals for RMS calc or autoadjustment
xg, yg, zg, ug, vg, wg	Final-surface frame goals for RMS or adjustment
@, @wave, wave, wavelength	wavelength for each ray
O, o, order, ...	Order of diffraction for each ray

Ray Wavelength Input: Ray wavelengths are required inputs if your refractive index values are to be looked up in a .MED glass table, or if you will be tracing diffractive optics. A ray wavelength field is a field whose header word starts with the "@" character or with the "wa" characters. In our examples we use "@wave" or "@wavel" to make it less mysterious.

For looking up refractive index data in a .MED table, the wavelength identifiers can be letters, words, or numbers. The lookup process is a simple search for an exact character match between the wavelength name in your ray table and the wavelength name in your .MED table. Any words are OK provided that the spelling and case agree exactly. Leading and trailing blanks are ignored.

To ray trace a diffractive optic, each ray will have to specify its *numerical* wavelength in the @ field. Moreover, the units of wavelength must be consistent with the specified grating groove density. For example, if the groove density is specified at 1.2 grooves per micron, then the wavelengths must be expressed in microns, not nanometers or microinches or Angstroms.

In the example LENS.RAY there is no need for a ray wavelength field since the refractive index in LENS.OPT is specified numerically. No lookups are needed to get the refractive index for each ray. The wavelength field is there just to allow the rays to be assigned color in the graphics output diagrams, where they look nicer in red (tag letter "r") than they would in plain black and white.

In the example ACHRO.RAY we have assigned ray wavelengths to be letter codes, called Fraunhofer designations that indicate particular spectroscopic wavelengths. This choice of descriptors is handy, and consistent with the glass table GLASS.MED that is furnished with each BEAM FOUR product. Further information on ray wavelength designators is provided in Chapter 11 "The Media Table" and Table 11-1.

The ray wavelength field ***tag character*** (the final character in the field, replacing the colon) specifies ray colors in the screen graphics for Layout and Plot. Color tags are "r" = red, "g" = green, "b" = blue, "y" = yellow, "m" = magenta, "c" = cyan. Upper case is equivalent to lower case. Without a specified tag letter, the default ray graphic color is the opposite of the black or white background that you have chosen for your graphic. In many of our examples we link the ray color tag to the actual ray wavelength, so that the rays of red light appear red on screen in layouts and in plots. However there is no particular requirement that the plotted ray color tags be associated with ray wavelengths. Each ray can have its own color tag, independent of wavelength. In this way you can mark specific rays to identify them more easily. For example you may wish to have your chief ray or particular marginal rays specially marked.

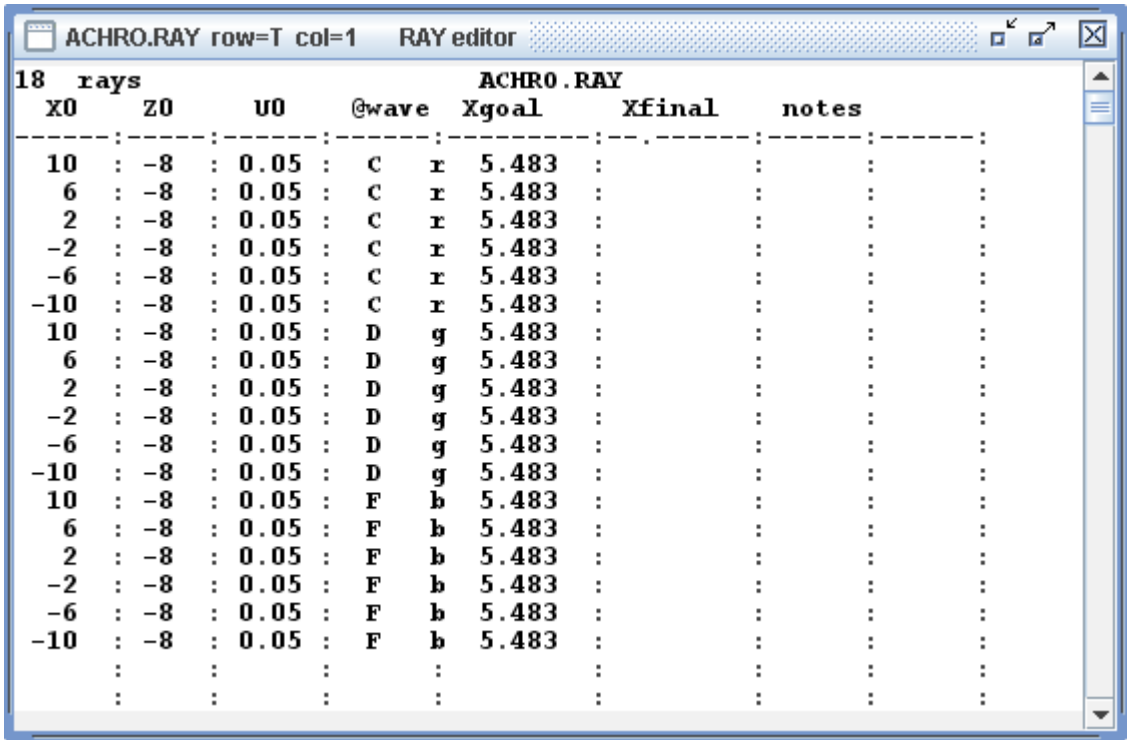


Fig. 10-2 The file ACHRO.RAY sets up a group of rays all entering the lens at the same off axis angle (here, U0=+0.05). The first six have wavelength C and are color coded red ("r") in the plots and diagrams. The second six have wavelength D and are color coded green "g". The third set have wavelength F and are color coded blue "b".

Setting up Initial Ray Directions (U0, V0, W0): Table entries of this type are the means to initially direct your rays toward your optic. Ray segment direction vectors {U, V, W} are always internally normalized to have unit length, and BEAM FOUR takes care of this normalization automatically and invisibly using the ray start information that you provide. For the most part, you can simply specify one or more of these parameters in your ray table, and let BEAM FOUR handle the details from there by means of its default choices. For completeness we list eight situations with increasingly detailed ray direction specifications, and the resulting direction vectors, in Table 10-2.

Table 10-2: User Specified Ray Directions

User specifies....	Makeup Normalization?	Actions taken
none	none	{U0,V0,W0} = {0, 0, ±1} ±sign per Options::DefaultRays
U0 only	W0	U0 gets limited to ±1; V0 = 0; W0 = ±√(1-U0²) ±sign per Options::DefaultRays
V0 only	W0	V0 gets limited to ±1; U0 = 0; W0 = ±√(1-V0²) ±sign per Options::DefaultRays
W0 only	V0	W0 gets limited to ±1; U0 = 0; V0 = ±√(1-W0²) ±sign per Options::DefaultRays
U0 and V0	W0	U0²+V0² gets limited to 1; W0 = ±√(1-U0²-V0²) ±sign per Options::DefaultRays
U0 and W0	V0	U0²+W0² gets limited to 1; V0 = ±√(1-U0²-W0²) ±sign per Options::DefaultRays
V0 and W0	U0	V0²+W0² gets limited to 1; U0 = ±√(1-V0²-W0²) ±sign per Options::DefaultRays
U0, V0 and W0	all	{U0,V0,W0} gets normalized; if this fails, it is replaced by {0, 0, ±1} ±sign per Options::DefaultRays

Ray Generators: A basic tenet of geometrical ray tracing is that a distribution of rays should sample your pupil and report fairly on your optical system aberrations. Several kinds of initial ray groups are needed frequently, and BEAM FOUR has four automatic ray generators to help you populate your .RAY table for ray starts or goals. The types are (1) an equally-spaced one-dimensional fan beam or ribbon beam, (2) a two dimensional uniform rectangular ray grid, (3) a two dimensional uniform circular ray pattern, and (4) a two dimensional circular Gaussian ray pattern. These generators can be accessed at the **Options::Ray Generators** menu item. To use one of them, follow these steps:

1. Create or open a ray table in the usual way. Set it up with fields for the ray start information that is appropriate to your optical task. For example you might want to illuminate a circular entrance pupil with a dense grid of rays starting in your XY plane, so be sure your ray table has field headers for X0 and Y0, and that your field widths and decimal point locations are appropriately set to receive your new data. High accuracy is available if you allow enough digits for the results.

2. Choose Options:Ray Generators and select your generator type. A dialog will pop up showing the options available for each generator. Each ray generator will extend your table as far down the page as necessary to accommodate its rays.

1D Ray Pattern: The 1D Ray Pattern Generator (Fig 10-3) offers you the choice of spanning the X, Y, Z, U, V, or W ray start or goal, using the radio buttons at the top of its dialog. For ray starts, these are lab frame. For goals whey are whichever frame you specified in your ray table (Xg=lab, xg=local). The coordinate center and span can be separately specified, as can number of rays generated. To fit this pattern into a pupil of height H with equal ray weights and areas, use span = H·(N-1)/N.

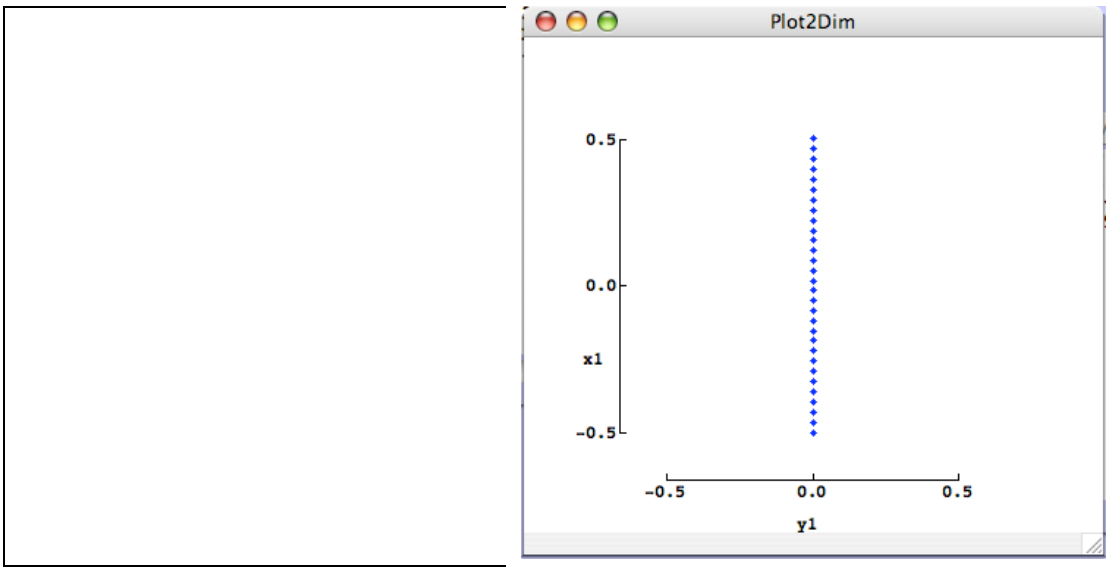


Fig 10-3: Options::RayGenerator dialog for the 1D Ray Pattern. Output can go to the ray start or the ray goal coordinate chosen. The radio button group selects which ray coordinate is to be populated. Its center value, span, and ray count govern the result. Your ray table will receive the rays starting at the row requested.

2D Rectangular Pattern: The 2D Rectangular Pattern Generator is similar to the 1D generator but produces a grid of ray starts spanning two dimensions: X,Y; X,Z; Y,Z; U,V; U,W; or V,W. Again, the top radio button group sets your selection. The centers, spans, and number count of uniformly spaced values per axis are specified in the six data entry boxes of the dialog. To fit this pattern into a rectangular pupil HxW with equal ray weights and areas, choose your spans equal to $H\cdot(N_h-1)/N_h$ and $W\cdot(N_w-1)/N_w$.

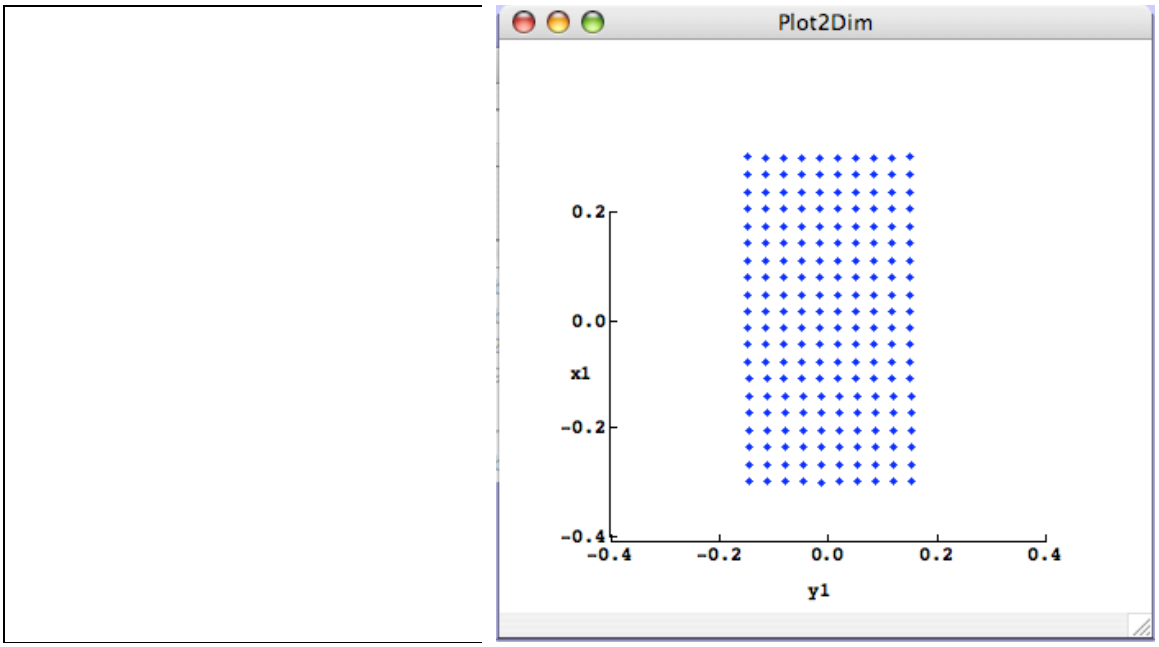


Figure 10-4: **Options::RayGenerator** for the 2D Rectangular Pattern generator. The radio button row selects the pair of ray coordinates that you want the generator to populate. The first three data entry boxes set up the center value, span, and count for the first coordinate. The second three data entry boxes handle your second ray coordinate. This example produces a 20x10 grid of ray starts in (X,Y). Be sure your .RAY table has columns for the two coordinates you are generating (for example, X0 and Y0 fields or Xg and Yg fields). Click "OK" to run the generator.

2D Circular Uniform: The 2D Circular Uniform generator (Fig 10-5 below) is a bit more complicated. It is based on the hexagonal number sequence (1, 7, 19, ...) that fills a circular pupil with a nearly uniform density of points arranged in circles of increasing size. To use this generator, first decide which coordinate pair should be populated, and decide what the position offsets and radius in the lab frame should be. Then choose how many circles and therefore rays you would like to put into your pupil. Assuming that your .RAY table is set up to receive your chosen coordinates -- for example X0 and Y0 -- - when you click OK these ray starts will be computed and entered into your .RAY table beginning with the row you have specified in the dialog box (default is row 1). To fill a circular pupil of radius P with equal ray weights, choose $R_{outer} = P \cdot N_{rings}/(N_{rings}+0.5)$.

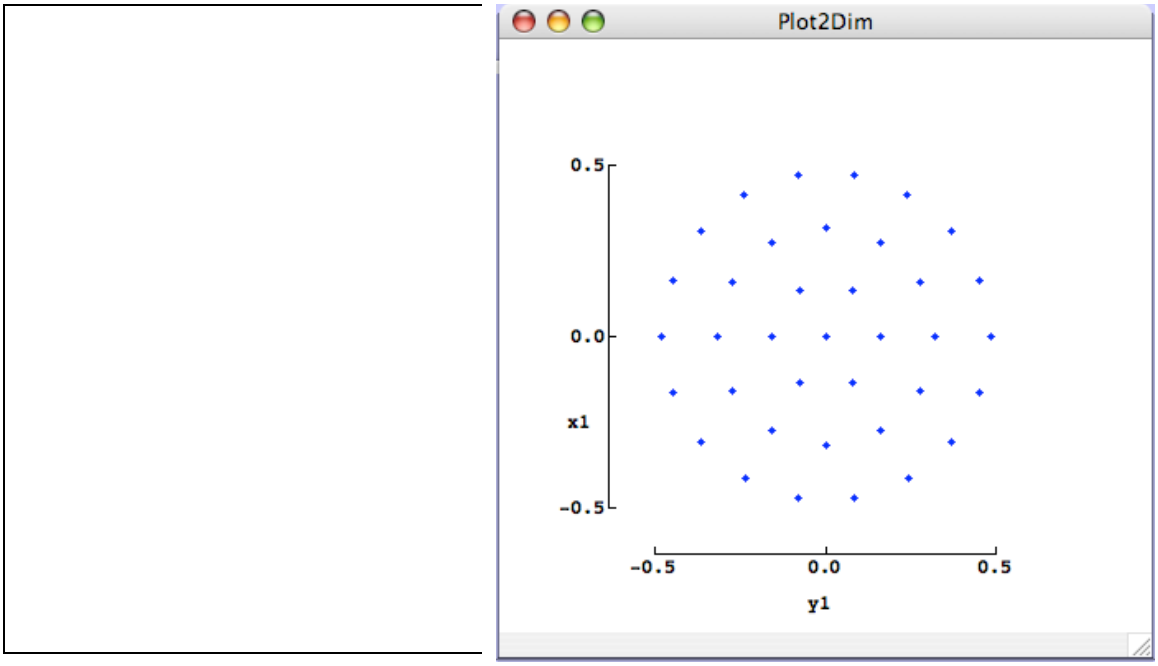
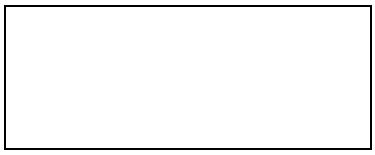


Figure 10-5: **Options::RayGenerator** for the 2D Circular Uniform generator. First be sure your .RAY table has columns for the two coordinates you are generating (for example, X0 and Y0 fields). As with the other ray start generators, the radio button row selects the pair of ray start coordinates to be populated. Two data entry boxes set up the center value offsets for the first and second ray coordinates. The third data entry box sets up the radius of the outermost circle of the pattern. The spinner control lets you specify the number (1...17) of concentric ray start circles that you want generated. Click "OK" to run the generator. The right hand plot shows three circles totalling 37 rays.

2D Circular Gaussian: This ray generator is also used to fill a circular pupil, but with a ray population that is concentrated towards the center. There are N concentric rings containing 6, 12, 18, …rays for a total of $3N+3N^2$ rays. We assign each ring group a fractional probability proportional to how many rays it has. The radius enclosing a Gaussian probability P is $s\cdot\sqrt{(2*\ln(1/(1-P)))}$ where s is the standard deviation. It follows that the ring radii are linked together nonlinearly, with



This is the formula used to generate the successive ring sizes. Options and results are shown in Fig 10-6. To fit this pattern into a circular pupil of radius P containing 99% of the light with equal ray weights, choose Router = $P\cdot\sqrt{(\ln(1+N)/\ln(100))}$.

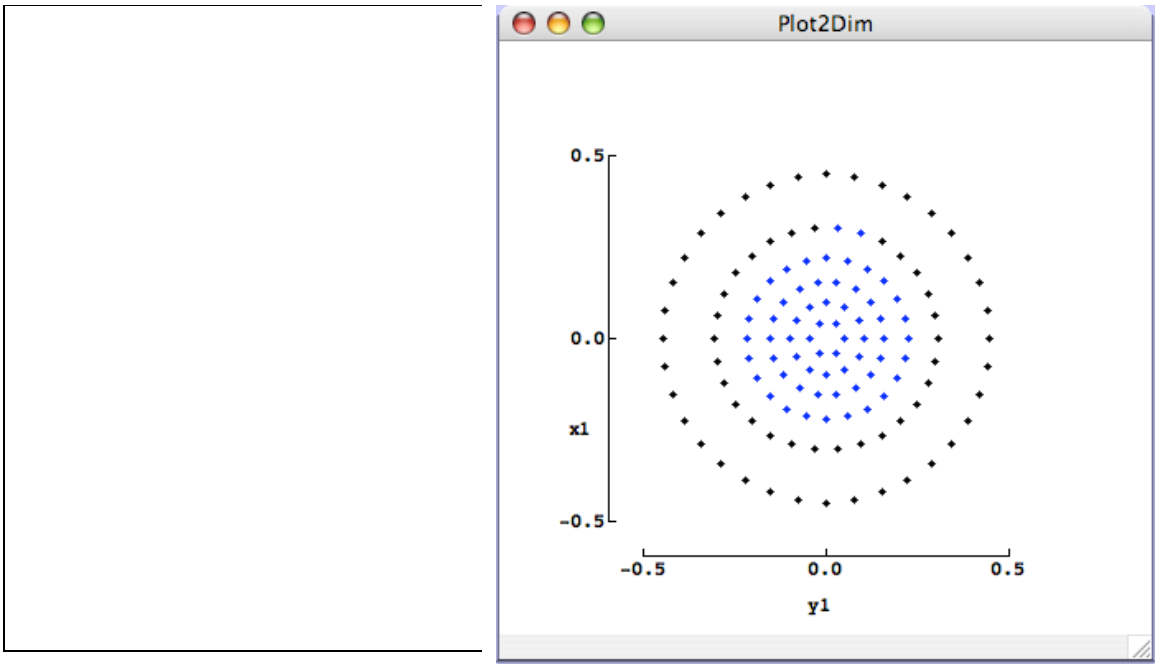


Figure 10-6: **Options::RayGenerator** for the 2D Circular Gaussian generator. This works like the Circular Uniform generator, except that its ray density is based on a two dimensional Gaussian probability density rather than being uniform.

Fixed Goals: You can specify one or more goals for each ray. These input data are final ray coordinates (most commonly xgoal and ygoal) specifying where you would like the rays to go. They serve as target values when Run::InOut has its RMS calculator turned on, or when you run the AutoAdjust feature which adjusts optics or ray parameters to minimize the RMS deviation between the rays and their goals. Table 10-1 shows the nomenclature. For examples see Chapter 12 "Run::InOut" and Chapter 20 "Run::AutoAdjust." Rays can be individually aimed at goals to fill an internal pupil using AutoRay, Chapter 21.

Floating Goals: Goals for groups of rays can also float, i.e. adjust themselves to a common value within each group. This value is the average of all the group ray coordinate values. This feature is used when you want to know or minimize the scatter about the mean group position but don't care where that position lies. Goals are grouped by tag letter. All rays in the first group might be tagged "a" and the next group "b" and so forth; actually any letters will do. There is no need to supply numerical values in the tagged fields since these will be computed and displayed as part of the averaging process for each group.

Ray Table Outputs: As mentioned at the beginning of this chapter, the ray table can be set up to receive selected output data from any ray trace. To do this, put appropriate output field labels into your table header. These abbreviated labels follow the same name conventions as the input field labels, except of course that output data are not at surface 0 but rather are at surface 1, 2, ...final. So X1, X2, ... through Xfinal are the lab frame X coordinate values of ray intercepts at surfaces 1, 2, ... final. Headers A1, A2 etc request angles between each incoming ray and its surface normal.

Table 10-2: Ray Field Headers for Output Fields	
X1, X2, ... Xf	Lab frame X intercept value at surface 1, 2, ...final
Y1, Y2, ... Yf	Lab frame Y intercept value at surface 1, 2, ...final
Z1, Z2, ... Zf	Lab frame Z intercept value at surface 1, 2, ...final
U1, U2, ... Uf	Lab frame U direction after surface 1, 2, ...final
V1, V2, ... Vf	Lab frame V direction after surface 1, 2, ...final
W1, W2, ... Wf	Lab frame W direction after surface 1, 2, ...final
x1, x2, ... xf	Local frame x value at surface 1, 2, ...final
y1, y2, ... yf	Local frame y value at surface 1, 2, ...final
z1, z2, ... zf	Local frame z value at surface 1, 2, ...final
u1, u2, ... uf	Local frame u value at surface 1, 2, ...final
v1, v2, ... vf	Local frame v value at surfaee 1, 2, ...final
w1, w2, ... wf	Local frame w value at surface 1, 2, ...final
A1, A2, … Af	Angle (deg) between incoming ray and surface normal
P1, P2...Pf, p1, p2, ...pf	Optical path to surface 1, 2, ... final surface
WFE	Wavefront error
N, n, note	Note describing fate of each ray

Ray Notes Output: One very useful output data field is the "notes" field. Notes tell you what happened to each ray during its trace. A ray that traces all the way to your final surface will be noted as "ok 18" or whatever the number of your final surface is. Otherwise, its note will be an abbreviated indication of what went wrong. The table below lists these messages.

Table 10-3: Ray Table Notes Abbreviations and Meanings	
OK NN	Ray ran OK to final surface whose number is NN
mis NN	Ray missed surface NN: failed to intercept it
bak NN	Ray intercept lies behind the ray start, so no legal forward intercept
Dia NN	Ray intercepted surface NN beyond its allowed outer Diameter
dia NN	Ray intercepted surface NN within its forbidden central diameter
Ord NN	Diffraction order specified is not allowed at this angle
TIR NN	Total internal reflection, no refracted solution at this angle

By inspecting a list of notes results you can usually tell what is wrong (or right!) with your optical setup. If your notes show a lot of similar complaints, that usually tells you just what needs fixing. For example if you get a list of "Dia 6" entries, you need to open up the diameter of surface number 6, or re-aim the rays so that they can get through it. A series of "mis 4" entries suggests that your surface 4 is located somewhere to the side of rays leaving surface 3. The “bak 7” result is telling you that your ray would have to go backwards to reach surface 7. To remedy those situations you will want to reorganize the surfaces or re-aim the rays to allow the trace to proceed.

Optical Path Output: This quantity is a diagnostic of the total delay a wavefront experiences passing through an optical system. The path starts at zero where the ray originates, and accumulates by an amount equal to the linear path length of each segment multiplied by the refractive index of the medium in which that segment lies. For each ray traced, the optical path to surface number 1, 2, etc., is computed and can be displayed in the .RAY table using the appropriate field header P1, P2, ... up to Pfinal.

Wavefront Error Output: One way to quantify the overall performance of an imaging optic is by means of its wavefront error. This is the relative deviation in optical path that each ray experiences, compared to the average of other rays from the same object point, corrected for inclinations and curvatures of the incoming and outgoing wavefronts. An optical system is said to be well corrected if its worst wavefront errors are smaller than about a quarter wavelength of its light, or its RMS WFE is less than 7% of the wavelength. By default, there are no tag letters for your WFE field, and all rays in your .RAY table constitute a single group for evaluating the average optical path. For this default to be useful, the rays should all originate at a common object point at a finite distance or at infinity. If several object points are represented within your .RAY table, you should break the WFE calculation into groups by tagging each ray's WFE field with a group letter, "a" ...."Z". To compute the RMS WFE, remove any goals fields and run InOut with Options::InOut having RMS output display enabled.

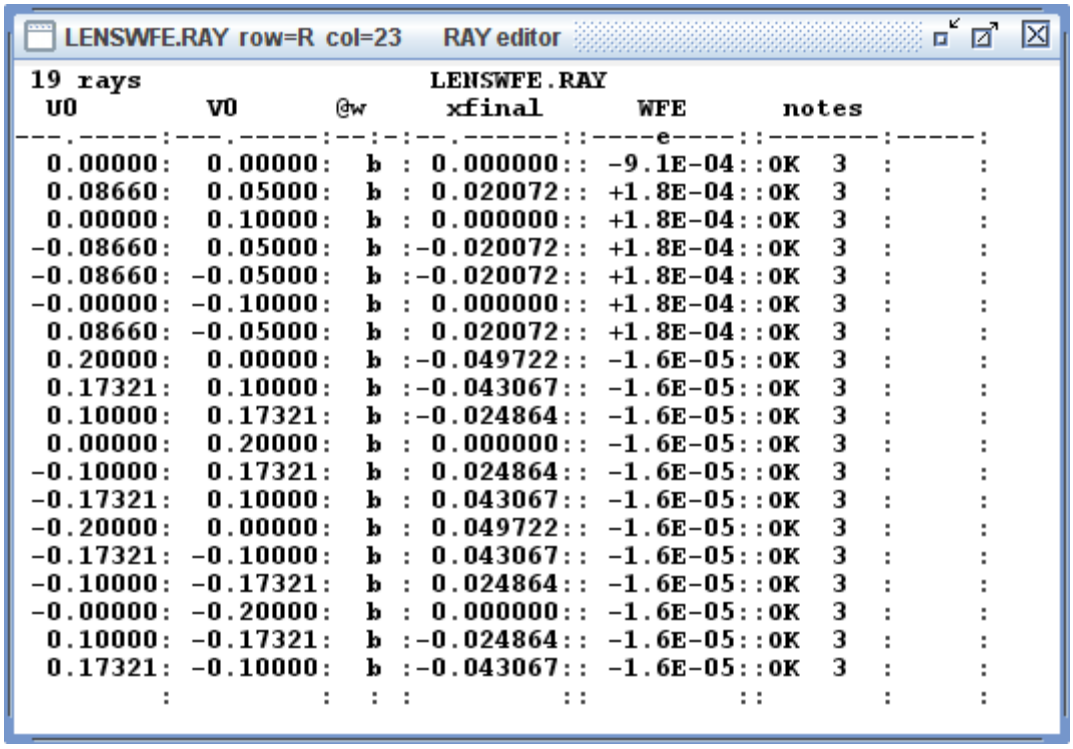


Fig. 10-7. An example of a .RAY table that has a WFE field to display the wavefront error contributed by a simple lens with a single on-axis object point. The initial ray directions U0 and V0 were installed using the 2D Circular Ray Pattern generator, here with only 19 rays in only three zones --- but to be useful for WFE estimation, the pupil should be more fully illuminated using dozens of rays in five or more zones.

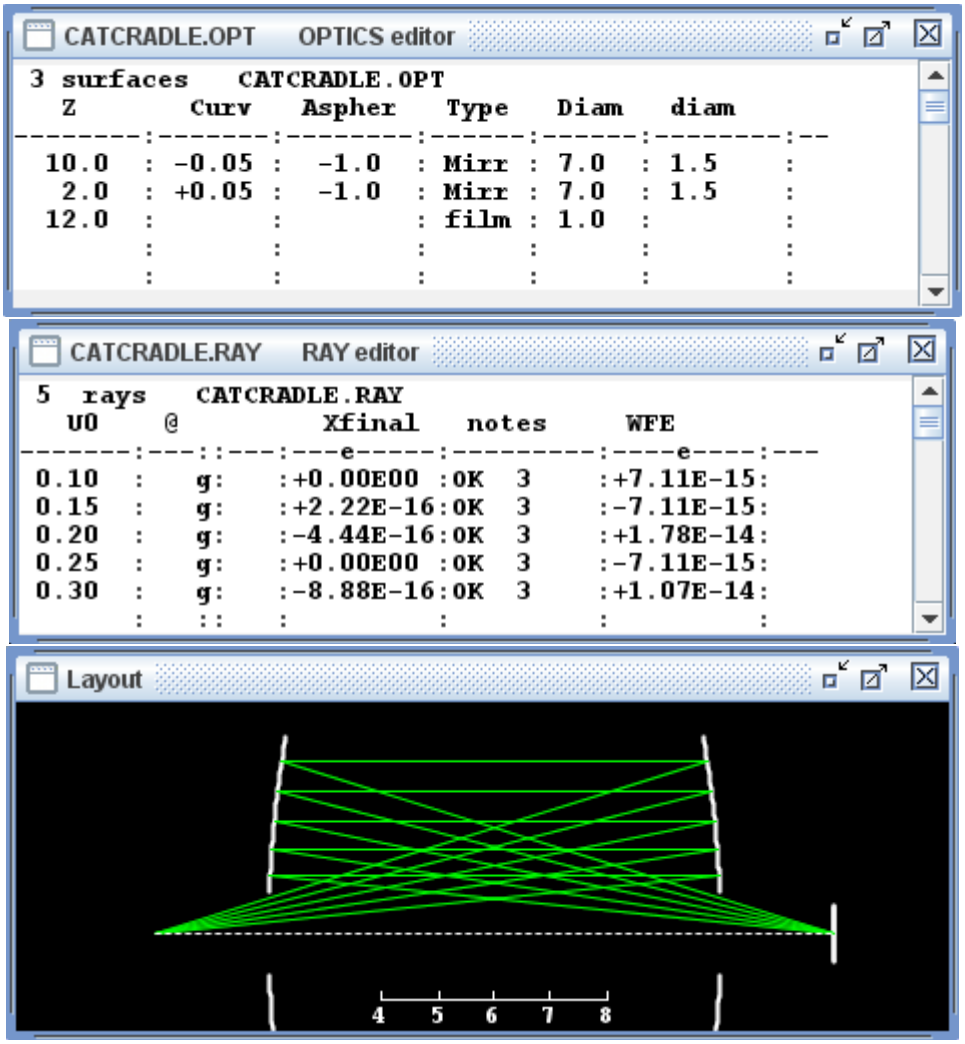


Fig 10-8 The CATCRADLE relay optic transfers a point object to a point image without spherical aberration. Here we trace it with a few rays to show its lack of WFE. Both the Xfinal field and the WFE field in the ray table have been set up with an "e" in their rulers to obtain exponential notation output --- useful when dealing with very small numbers, which here arise entirely from numerical errors in the floating point computation.

Wavefront Error Minimization: AutoAdjustment can be invoked to minimize WFE. To optimize an optic for least WFE, the implicit WFE goals are all zero, and no explicit goal field is recognized or needed. Delete or behead any explicit goal fields in your ray table. **Run::AutoAdjust** (see Chapter 20) will minimize the total squared WFE. As with any optimization, it is important to fully illuminate your optic with a representative set of ray start locations and directions. In this way your pupil will be properly filled and your working field will be occupied by test image points that span your image space.

Chapter 11: Media and Media Tables

Media tables resemble the other text tables used with BEAM FOUR. There is a three line preamble (title, headers, and ruler) followed by a collection of up to 999 glass records that hold the refraction data for the glasses. Media tables are used for looking up the refractive index for a given ray (whose wavelength is specified in your ray table) for a given glass type (whose name is specified in the leftmost field of your optics table).

Media tables make polychromatic ray traces possible. To carry out a monochromatic ray trace, you could just load the numerical value of the refractive index in the medium approaching each surface into the optics table under the field header "index." That way every ray will undergo the same refraction. In a polychromatic ray trace, however, each individual ray needs to look up the refractive index appropriate to its own wavelength. The media table is this lookup table.

Four media files are supplied with BEAM FOUR. One of them, GLASS.MED, has eight glasses and three plastic optical materials at four wavelengths in the visible range. The others list glass manufacturer's refraction data. You may create as many media files as you need, using refraction data from whatever sources you trust. Also, if you need to interpolate refraction data, you can construct interpolated refraction values and build your own media tables. Two examples are given in Chapter 26 using a spreadsheet to perform parabolic interpolation and Sellmeier interpolation.

GLASS.MED MEDIA editor

```
11 entries                    GLASS.MED
    Material                      C          D          F          HeNe
---------------------------:---.------:---.------:---.------:---.------:------:
   silica                  :  1.456369:  1.458404:  1.463128:  1.457020: 67.8 :
   BK7                     :  1.514323:  1.516728:  1.522377:  1.515089: 64.2 :
   K5                      :  1.519816:  1.522411:  1.528600:  1.520640: 59.5 :
   SK16                    :  1.617273:  1.620320:  1.627558:  1.618241: 60.3 :
   SF12                    :  1.642710:  1.648143:  1.661868:  1.644410: 33.8 :
   LAK8                    :  1.708975:  1.712885:  1.722220:  1.710215: 53.8 :
   sapphire                :  1.764943:  1.768138:  1.775580:  1.765963: 72.2 :
   SF11                    :  1.775986:  1.784458:  1.806455:  1.778621: 25.6 :
   polymethylmethacrylate  :  1.488   :  1.491   :  1.496   :           : 61.  :
   polystyrene             :  1.585   :  1.590   :  1.604   :           : 31.  :
   polycarbonate           :  1.581   :  1.586   :  1.598   :           : 34.  :
                           :          :          :          :           :      :
                           :          :          :          :           :      :
                           :          :          :          :           :      :
```

Fig. 11-1 The file GLASS.MED has refraction data for eight glasses and three plastics at four wavelengths. The glass names are listed down the first field, and index of refraction is listed for wavelengths C, D, F, and HeNe. Untitled fields such as the rightmost field are ignored by BEAM FOUR. In this example, that rightmost field lists the Abbe v index, which is a measure of freedom from chromatic dispersion, given by $(n_D-1)/(n_F - n_C)$.

The media tables give you freedom in choosing the appropriate way to describe wavelengths and glass names. The names of your glasses are listed down the leftmost field of your glass table. These names are completely arbitrary -- none are built into BEAM FOUR except the blank field whose refractive index is always 1.000. Of course, your glass names cannot be purely numeric, since a numeric value in your "index" field will be interpreted as an actual refractive index, not a glass name. Leading and trailing blanks are ignored. However capitalization and punctuation are significant, so that Bk-7 and BK-7 and BK7 are all different glasses. At trace time, it is this list of glass names that will be searched when a non-numeric refractive index is encountered in the index field of your optics table. Your glasses may be listed in any order. You may list as many glasses as you like (up to 200), but make sure that every glass that appears in your optics table appears somewhere among your listed media.

There are two ways to specify wavelengths. For visible wavelengths and common optical glasses, the usual way is to specify the Fraunhofer designator, which is a letter identifying an emission line lamp wavelength. The other way is to write out the wavelength numerically, for example in microns. This numerical description is mandatory in optics that use diffraction and which therefore have to evaluate the wavelengths numerically. Whichever method you use, be consistent between your ray table and your media table where the wavelengths will be looked up. Because this is a simple string lookup, be consistent in your spelling: 0.633 and .633 are different lookup wavelengths. You may list as many wavelengths as you like , but make sure that every wavelength appearing in your ray table in its "wave" field is listed somewhere in your media table.

Table 11-1: Fraunhofer Lines

Designator	Wavelength, microns	Element
r	0.7065	He
C	0.6563	H
C'	0.6438	Na
HeNe	0.6328	laser
D1	0.5896	Na
D	0.5893	Na
D2	0.5890	Na
d or D3	0.5876	He
e	0.5461	Fe
E	0.5270	Fe
F	0.4861	H
F'	0.4800	Cd
g	0.4358	Hg
G	0.4308	Fe
h	0.4047	Hg
H	0.3968	Ca+
K	0.3934	Ca+

Unlisted wavelengths: Often you will need to trace an optic at wavelengths not listed by the glassmaker. See Chapter 26 “Spreadsheets” for some interpolation formulas.

Chapter 12: The RUN Menu: InOut

Once an optics table and a ray table have been defined, the main menu RUN command pops up a list of ray tracing tasks. The first of these is **InOut**, so named because its job is to fill in all the data fields that you have specified in the ray table. Recall that each ray table contains your ray-start input data, and can also have places for ray coordinate output data to be computed during an InOut ray trace. The InOut function does two things:

- A ray trace is computed;
- The data requests in your .RAY output column headers get filled in.

Once the data are computed and displayed, you can save the ray table to have a record of the result of the ray trace.

Recall that one available type of .RAY output data field is "Notes." Here, each ray that runs through the entire optic without error will get the note "OK" but each ray that fails to trace through your optic to your final surface will deliver an abbreviated message to this field of your ray table, to help diagnose what its problem might be. For example if an iris or other optical element is undersized, you can expect a flurry of "Dia NN" messages showing the number NN of the offending surface. Table 10-3 in Chapter 10 lists the ray table notes and their meanings.

If one or more goals have been specified in the ray table, then InOut can also calculate the root mean square deviation between the final ray coordinates and these goals. This result is shown in a pop-up box "RMS from goals" that appears when goals are present:

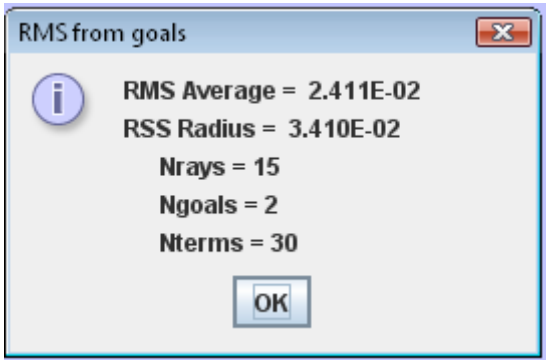


Figure 12-1: This popup dialog displays the computed root mean square deviation between the ray goals and their final ray coordinates. When Ngoals=2, RSS **radius** is also shown; it is $\sqrt{2}\times$RMS average. Nterms is the number of terms used in the calculation = number of good rays × number of goals per ray. A caution statement will appear if WFEs are mixed with other goals (usually unwanted).

There are can be two RMS values posted in this dialog. “RMS Average” is the root mean square deviations of all the ray coordinates from their respective means, and is available for any number of goals. “RSS Radius” is posted when there are just two goals (usually xgoal and ygoal, or ugoal and vgoal), and is the Pythagorean sum (RSS) of the pair of deviations. It is larger than the RMS average of the two deviations by a factor of $\sqrt{2}$.

This posted RMS value can be used to compare alternative adjustments of an optical system. RMS reporting can be switched on or off using the Options::InOut dialog.

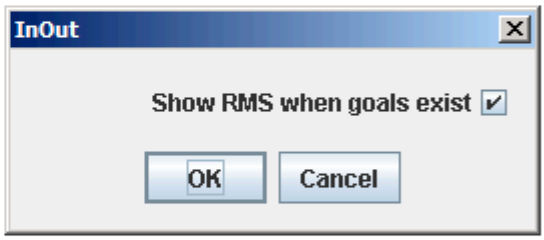


Figure 12-2: The **Options::InOut** dialog that you use to enable or disable the display of RMS ray departure from the user specified ray goals.

Setting Up Goals: The RMS calculation evaluates the square root of the average of the squared deviations between the ray goals and the ray coordinates at the final surface. To set up this calculation, furnish your ray table with ray goals: fixed, floating, or both. Figure 12-3 shows an example of a Cassegrain imager. Its first six rays have a common fixed goal of 0.05 units and the next six rays use a tag "z" to set a common floating goal which evaluates as shown. Run::InOut shows the RMS is 1.6×10^{-6} units.

The reason you might want both fixed and adjustable goals is this. Having some group of rays specify a fixed ray goal value is the way to specify an overall system focal length. Other rays (representing other field positions) might want variable (but common) field positions will allow them to have distortion yet have imperfect distortion. Those floating goals can enforce a nice tight bomb pattern but be tolerant of the actual absolute goal location. Often, one field location will have a fixed goal, and all other field positions will have floating goals.

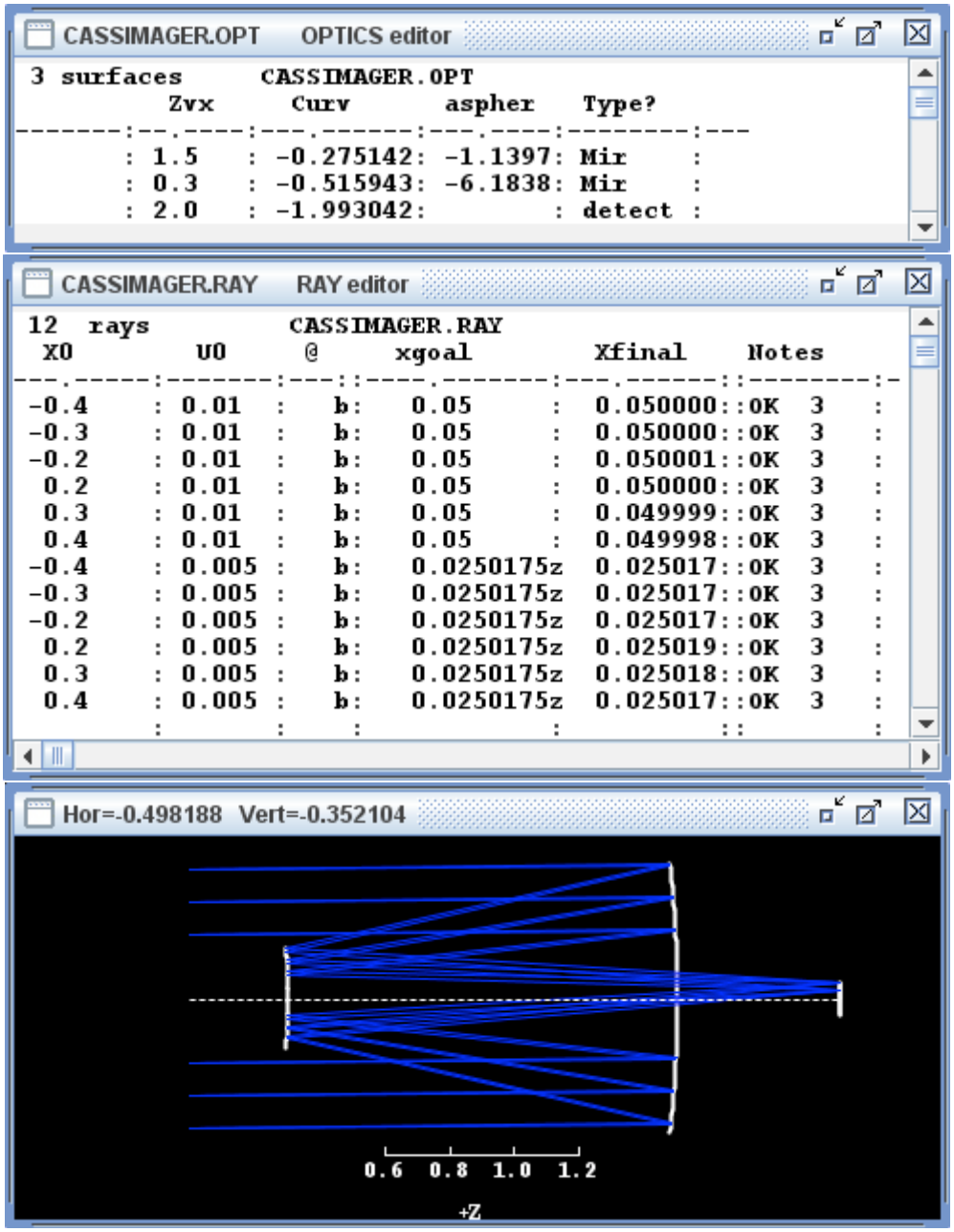


Fig 12-3 An imager is optimized in linear blur with one ray group goal set to a fixed 0.05 units while the other ray group tagged with "z" has a common unspecified floating goal.

Chapter 13: The RUN Menu: Layout

An extremely convenient and rapid means of veritying that a trace has done its intended job is to examine the results pictorially. Selecting **Run::Layout** creates an on-screen diagram of your optical system and its rays. Layout does two things:

- A ray trace is run;
- A graphics window is opened and an optical layout is drawn.

Layout lets you manipulate your view of the optical system using your mouse. To zoom in or out, first click the mouse to set your zoom center, then use the mouse wheel or use the function keys F7=ZoomIn, F8=ZoomOut. Vertical zoom is F5 and F6 or Shift + mouse wheel. To view different parts of the optic, drag the graphic around its window by holding down the left mouse button. To twirl the optic and view it from various elevation and azimuth angles, drag it with the right mouse button. These manipulations are available to all the BEAM FOUR 3-D graphics, including the little cube demo **Run::Demo**.

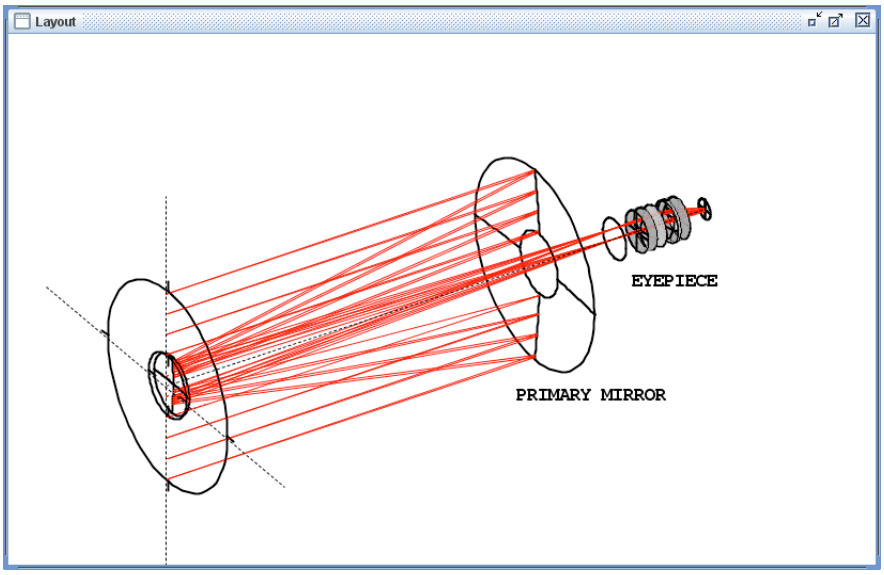


Figure 13-1: Layout of a Cassegrain telescope. Features like the axes and ruler are optional elements of layouts. Annotation such as "EYEPIECE" can be typed directly onto the graphic once you have set your zoom setting and orientation. The annotation font size is set in the Options:Graphics dialog for all graphics. The displayed color of each ray is set by the ray's **tag letter** "r", "g", "b" in the ray table's "wave" field.

Once you have your layout properly sized and oriented, you can annotate your diagram with text. Click your mouse to position the blinking caret appropriately, and start typing.

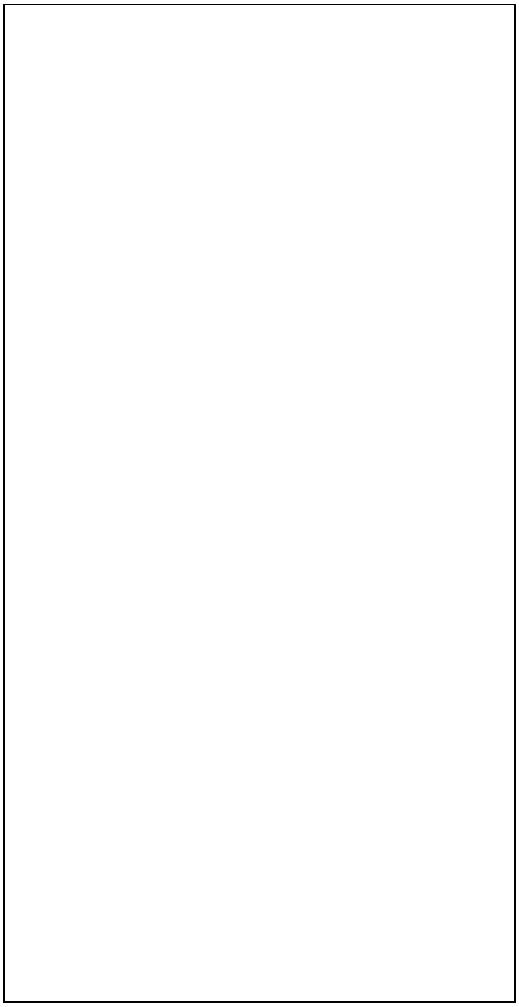


Figure 13-2: The **Options:Layout** dialog lets you set up your preferences for the features of the Layout function. The Layout elevation and azimuth angles are the angles in degrees from which you initially view the optical system. (Using your right mouse button you can of course modify your viewing angle dynamically.) The ArcSegments lets you specify how many line segments are used to represent curved arcs in your layout. Four checkboxes let you retain your current pan & zoom settings, show refractive parts with gray shading and/or connector lines, and control the visibility of retro surfaces. The six "axis" buttons let you choose the orientation of your optic. The onscreen rulers and coordinate axes can be turned off, or on, with the five "Axes" check boxes. Then, each optical surface is drawn as eight arcs identified by eight points of a compass; they can be selectively turned off or on with the bottom row of checkboxes. The default is to have them all turned on. The artwork line weights in pixels are specified in the bottom three boxes for rays, surfaces, and coordinate axes. Finally, the display can be set to white, black, or black w/ stereoscopic artwork using the “Format” buttons.

The annotation font size and weight are chosen at **Options::Graphics**, (see below) and apply to all the graphics screens, not just the Layout screens.

Diagrams produced by Layout may be captured and saved in a variety of formats. For most purposes the **File::QuickPNG** function is the easiest. QuickPNG is available for every BEAM FOUR window, text or graphic. It saves an image of the window in .PNG format with a filename you specify using the usual pop-up file save dialog. There are several other graphic output formats built into BEAM FOUR for working with computer-aided design and drawing projects. See Chapter 23 for more details of the CAD graphics output capabilities.

Layout has a host of customization features available at its **Options::Layout** dialog. There you can set up a preferred initial viewing angle, and choose how the optical system will be oriented with respect to the viewing window. "Layout Elevation" and "Layout Azimuth" are the initial view angles, in degrees. There is a radio-button selector for

which optical system axis is to point upward in your layout diagram. **Options::Layout** offers checkboxes to control the appearance of your layouts, including features such as on-screen horizontal and vertical rulers and your optical system's geometric axes. Three display formats are available; see below.

In Layout, optical surfaces are drawn as a set of four radial arcs and four circumferential arcs. Normally these layout arcs are all turned on, but you may modify these settings by deselecting the appropriate checkboxes in the **Options::Layout** dialog to show simple side views, cross section views, and cutaways. These are illustrated in Figs 13-3 to 13-6.

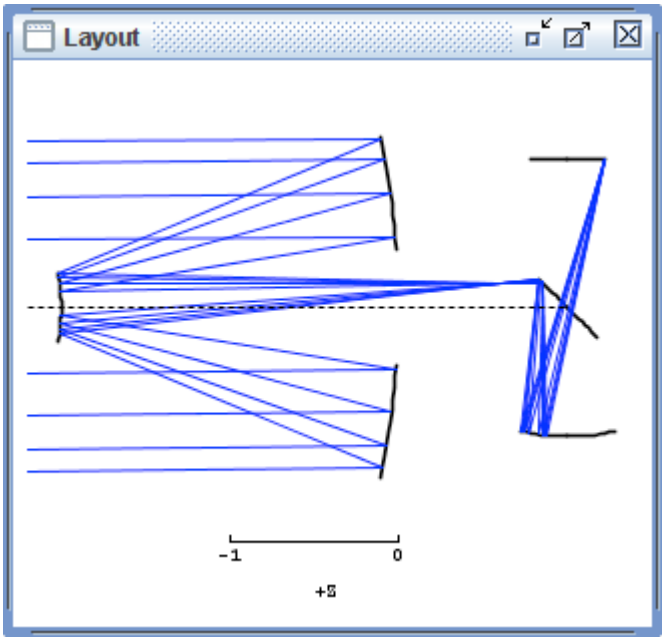


Fig. 13-3 To obtain a cross section view in Layout, such as the one shown here, set the Elev and Azim options to zero, and turn off all the "Arcs to be drawn" checkboxes except the N and S arcs as shown in the Options dialog in Fig 13-2.

Options::Layout has a Format selection window to let you customize the display format. Default is a white background format, where the optical elements are drawn in black and the rays are drawn with their specified colors or black, if unspecified. A black format is similar but the background is black, the optical elements are drawn in white, and the rays again are drawn with their specified colors or white if unspecified. A stereo format is also available, in which the background is black, and the optics and rays are drawn twice: once in blue, and again in red but from a slightly different vantage point. The difference in view directions is controlled by the stereo parallax parameter specifying the angular separation of the red (left eye) and blue (right eye) images, in milliradians. For most viewers a good choice of parallax is one that makes the red and blue images separate by a very slight amount, just a few screen pixels. The stereoscopic illusion is most effective with a full screen in a darkened work environment. Red/blue eyeglasses are offered by many web game retailers and video game outlets. A negative parallax value accommodates reversed glasses. See Chapter 29 "Viewing Stereoscopic Displays."

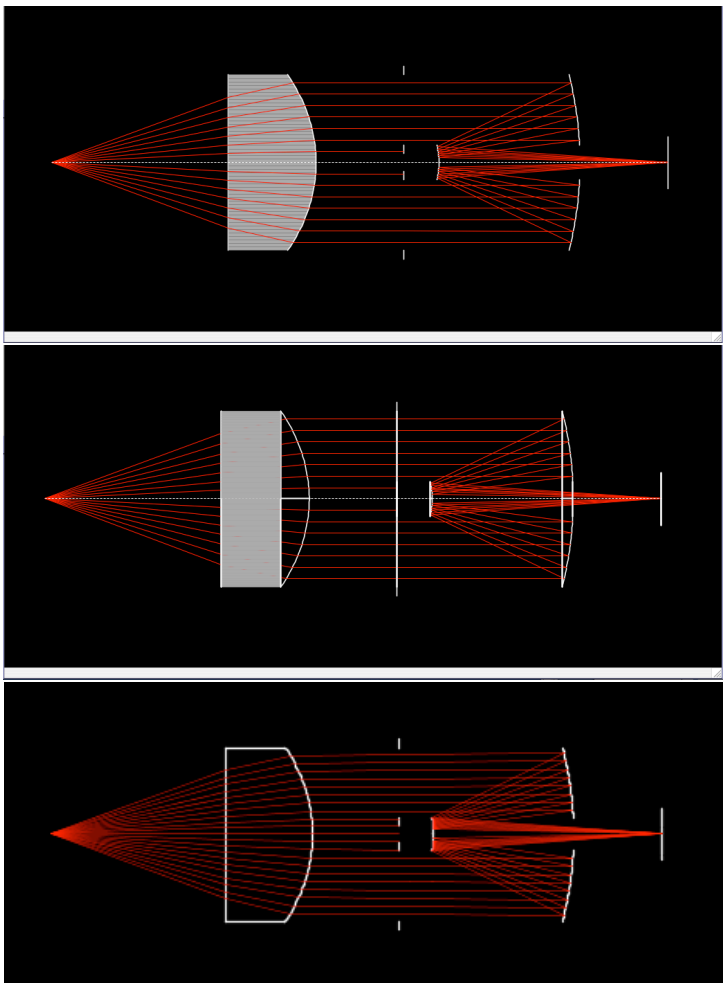


Fig 13-4 Layout options with a lens, an iris, and a pair of mirrors. **Top**: a cross section view shows refractors with solid gray and shows openings in irises and central mirror holes. Enable the "Refractor shading" option and disable all arcs except the North and South arcs. **Middle**: an external side view is obtained by keeping the "Refractor shading" option so you can't see the rays through the side of the lens, and enabling all the Layout arcs. The refractive bulge on the lens surface is transparent, so you can see rays within the lens before they emerge. As with a true normal side view you can't see iris or mirror holes (you can of course see everything by rotating your view point). **Bottom**: If you prefer to turn off refractor shading, you can turn on refractive connectors to outline each lens.

Fig13.4.OPT row=T col=1 OPTICS editor

6 surfaces Fig13.4.OPT

index	Z	Type	Diam	diam	Curve	Asph	F
	1.0	lens	1.0		0		
1.62	1.5	lens	1.0		-1.2343	-0.5919	
	2.0	iris	1.0	0.2			
	3.0	mirr	1.0	0.2	-0.500	-1.0671?	
	2.2	mirr	0.2		-2.120	-2.5531?	
	3.5	FP	0.3				S

Fig 13-5 This is the optics file that illustrates many of the Layout options.

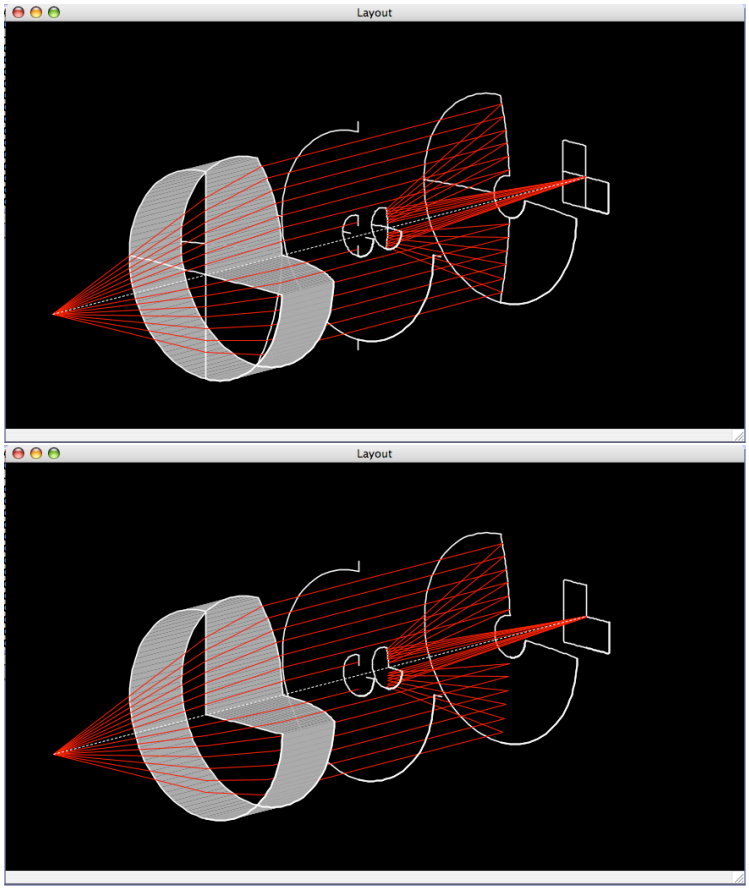


Fig 13-6 By disabling just one of the arc groups in Options::Layout, you can prepare a corner cutaway view of an optic. **Top:** we show the effect of disabling the "NE" (northeast) arc group: all radial arcs N, E, S, W are enabled, and all the peripheral arcs except NE are enabled as well. **Bottom:** in this layout the radial arcs S and W have also been disabled, so that the enabled arcs are just N, E, NS, SW, NW. This way, the periphery is emphasized.

We caution that the Layout function is not suited for precision measurement of surface locations or ray intercept coordinates. Rather, it is a quick-and-dirty cartoon generator that is based on the connect-the-dots principle. All the arcs plotted are really sequences of connected straight line segments, and the circular-looking curves are really polygons. These are quick to plot and they will reveal optical issues immediately. But mathematically they do not stand up to close scrutiny. If you want to investigate the Layout graphics closely, request plenty of arc segments in Options::Layout. Better is to evaluate the ray intercept (X,Y,Z) values using the full double precision accuracy of the InOut tables. Use the "bed of nails" concept: have a grid of parallel rays intercept the target surface and read out the ray intercept coordinates in the .RAY table using the **Run::InOut** function.

Additional customization features that apply to all BEAM FOUR graphic screens are available at the **Options::Graphics** dialog. (We describe these features here since this is the first chapter dedicated to a graphics output function.) With **Options::Graphics** you can choose your preferred mouse wheel directions for ZoomIn/ZoomOut, and set up the font size and style of the text that is automatically generated for each graphic. You may also specify the font size and style for the annotations that you add by typing onto any graphic, and the startup size (in pixels) for each graphic window.

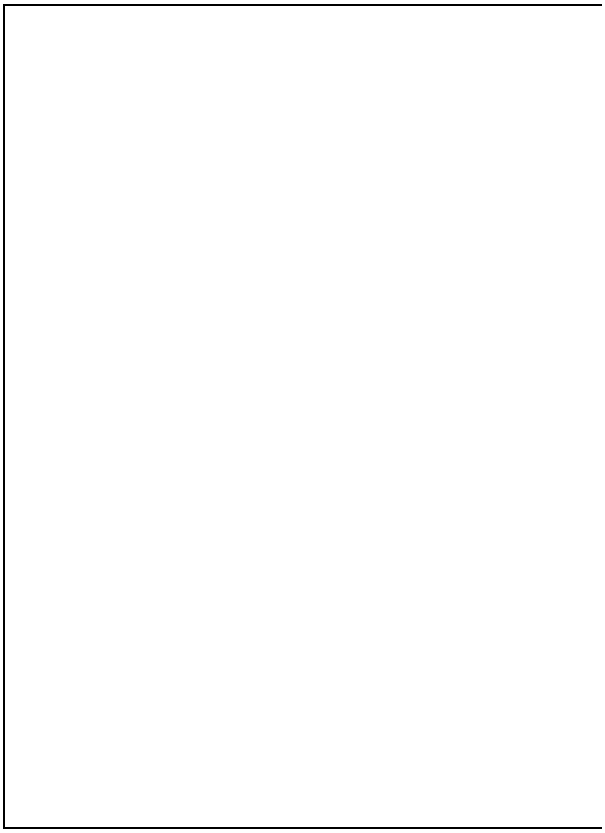


Fig 13-6 The **Options::Graphics** dialog box offers control over all the output features that are in common among the graphics display functions. Wheel direction buttons let you choose pull or push to zoom in on a graphic. The graphic font size and boldness controls the text that the graphic itself generates; the annotation font size and boldness govern the text that you type onto a graphic to annotate it. Initial window size sets the size of a window when it is first created, but of course you can minimize, maximize, or stretch it however you like using the mouse. Pixel smoothing enables a software tool that reduces jaggies on most computer systems.

Chapter 14: The RUN Menu: Plot 2D

The Plot2D function provides a way to view relationships between any two ray trace variables. Its most common use is to produce a **spot diagram** which shows the positions of the arrivals of rays at the final surface of a ray trace. A spot diagram gives an immediate qualitative impression of the size and shape of the point spread function of the optic under idealized conditions (no diffraction, no seeing, no motional blurring).

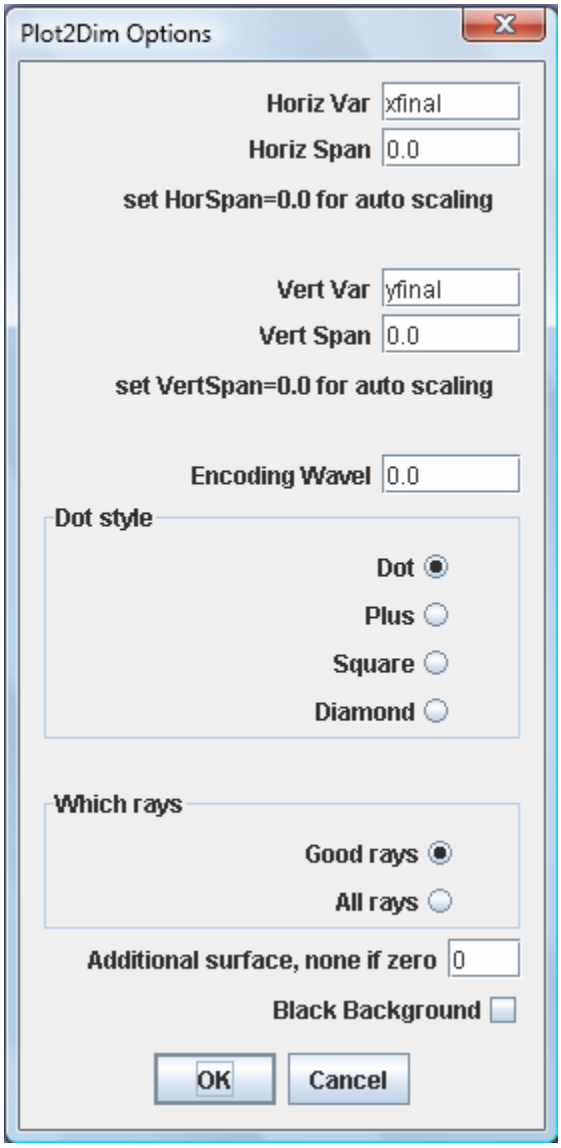


Figure 14-1. The **Options::Plot2D** is the control panel for setting up your plot variables and ranges. The horizontal and vertical plot variables can be any of your ray coordinates (x0, v3, yfinal....). Each variable allows you to specify a range, or span, of values to be plotted. If you set span=0.0 (the default) you enable automatic scaling that shows all the available data. The plot symbol can be selected from the four radio buttons marked "dot style". The plot symbol color follows the ray color assigned by tag letter in the .RAY "wave" column, making it easy to identify groups of rays that are functionally related. "Which rays" lets you choose to plot only those rays that proceed all the way to your final surface, or all rays. The "Additional surface" option lets you superpose ray spots from another optical surface onto the current defined surface spot diagram. Finally, you may set your preference for white (default) or black background for your plots with the checkbox near the bottom.

To set up the Plot2D function, start at the Options::Plot2D menu option and use the dialog (shown above) to choose your Plot2D settings. The (xfinal, yfinal) pair of coordinates to plot is a common choice since these are the local frame values of ray intercepts at your final optical surface. Bear in mind though that all the internal ray

variables are available for plotting on either axis, so you can make a wide variety of diagnostic plots, not just spot diagram.

To get a useful spot diagram it is important to flood your entrance pupil with a broad enough span of rays in all relevant dimensions. For a telescope focussed on infinity, you'll want to set up a range of X0 and Y0 ray start coordinates that overfill the pupil. For a microscope or copy lens you'll need a suitably large span of U0 and V0. Take care to define your optical diameters to properly limit your pupils. A small number of preset rays in your .RAY table is likely to be insufficient to densely probe your optic. Once you have a few rays tracing properly, use the **Run::Random** feature (available to all graphics screen displays) to populate your ray space with randomly generated ray starts. In this way your Plot2D will build up in density and reveal fainter outlying regions of your spot diagram.

Once you have set your **Options::Plot2D**, produce your Plot2D with the .OPT and .RAY tables that you have loaded. By default, every good ray is shown as a dot or other selected symbol on the diagram. The color of the symbol is the color assigned to its ray. Good rays are those that navigate your entire optic successfully. Each InOut ray trace note that is not "OK" will show the surface where it failed. Use these ray notes, and the Layout function, to diagnose where and why they are failing.

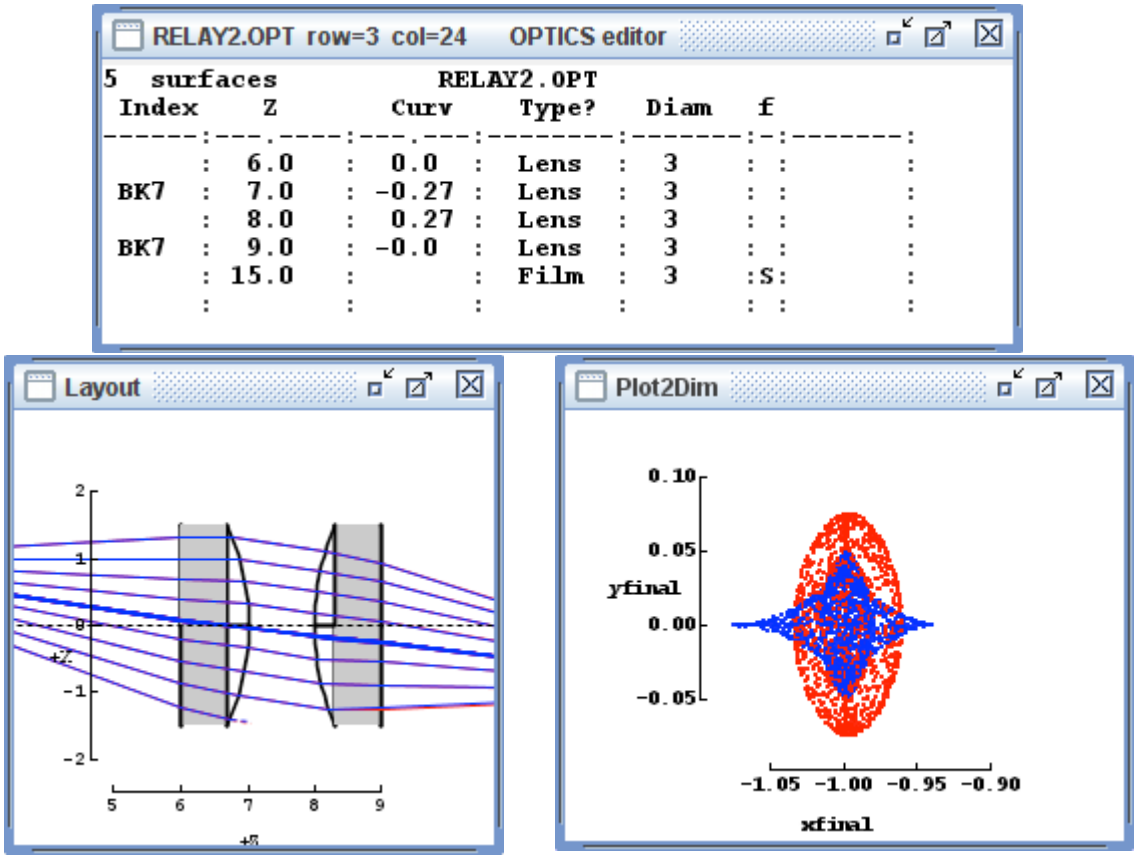


Fig 14-2 This two-lens relay has significant chromatism. The spot diagram (lower right) with a few thousand random rays contrasts the very different images in red and blue light.

Other kinds of diagnostics are easily generated using Plot2D. For example a simple relay lens would ideally have its xfinal remaining essentially constant while the incident ray angles vary. A plot of xfinal versus U0 would then ideally be a flat horizontal line. A linear trend in this pattern is an indication of defocus. A curvature in this pattern shows the presence of aberrations.

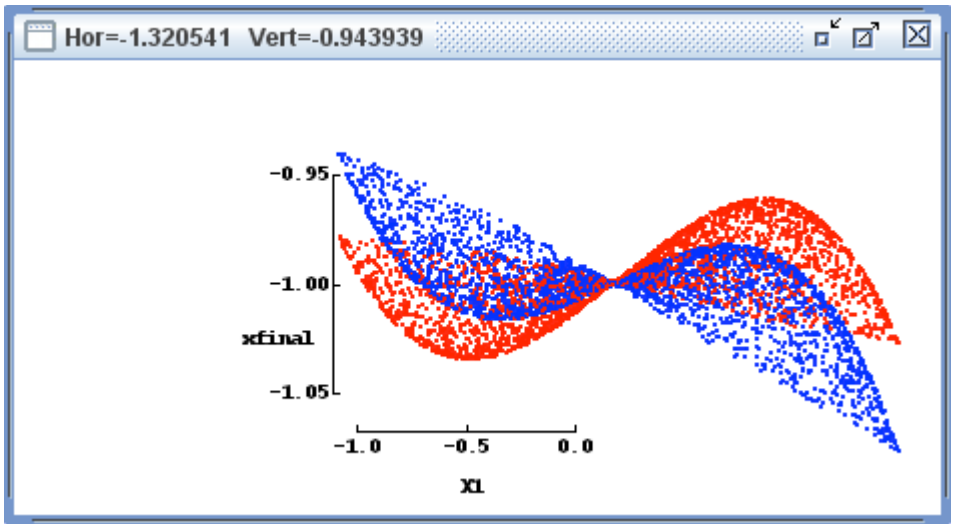


Fig 14-3 Plot of final ray intercept height xfinal as a function of initial ray intercept height X1. Red and blue spots are traces of rays having wavelengths C and F, each filling the pupil of the lens pair.

Chapter 15: The RUN Menu: MultiPlot

MultiPlot does just what you'd think: it steps through a series of parameter values and builds a Plot2D for each. These plots can be arranged side by side or top to bottom on the screen, making a stepped sequence of spot diagrams. It also can step two parameters and make a 2-D grid of spot diagrams. The center of each box is the centroid of its rays.

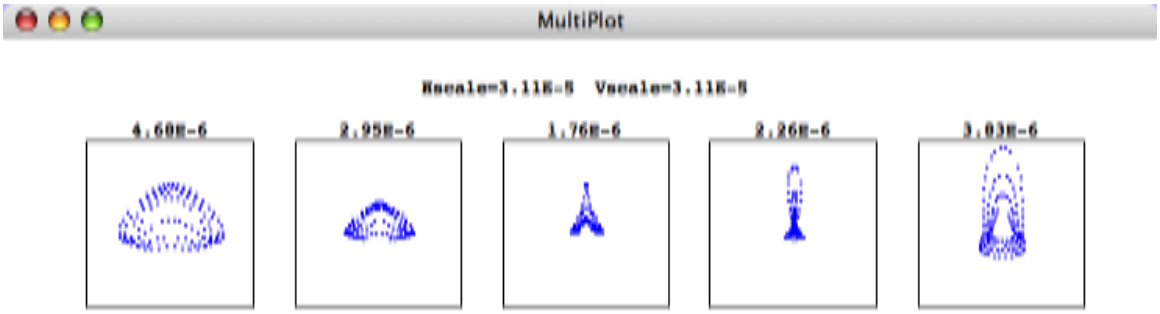


Fig 15-1 MultiPlot (above) shows a sequence of spot diagrams from our TMA telescope example. This diagram is termed a "through-focus" chart because the successive spot diagrams are constructed by shifting the optical system focus through a sequence of settings. Setup for MultiPlot uses the popup dialog **Options::MultiPlot** seen at left. In this example there is just one horizontal step variable: Z2 (the secondary mirror position) and it is assigned the replacement values shown. Here we have no vertical stepped parameter. Diagram scaling is controlled by the hor span and vert span values: zero or blank gives automatic scaling, or you can enter a specific value (e.g. your pixel size) to show scaling relevant to your system. Ten different numerical diagnostic data are available to title each plot box. Here, "m" (rms) is checked which produces the rms values shown in the top figure.

To create a **MultiPlot**, start with an ordinary spot diagram using Plot2D. Set up your ray table to illuminate your pupil from a single object point, and set your Plot2D ray variables to display a nicely filled aperture. Then decide what ray start parameters and/or optics parameters are to be stepped through ranges to explore your optical system.

The **step parameter** can be a ray starting value (X0, Y0, Z0, U0, V0, or W0) or optics table parameter (X1, X2, X3, etc; Y1, Y2, etc; Z1, Z2, etc; pitch tilt or roll (P1, P2, ..., T1, T2, etc), curvature (C1, C2...), or asphericity (A1, ...). Up to nine successive plots can be displayed along either or both axes. Your step pattern modifies your system (just temporarily!) from its nominal design, a convenient way to explore "what-ifs."

In MultiPlot, the displayed spot diagram **ray variables** within each box can be any of those accessible to Plot2D, namely ray coordinates in global (upper case) coordinates or local (lower case) coordinates, or optical path. See Chapter 14 "Plot2D" for full details. Spot diagrams most commonly are built to illustrate the final-surface ray coordinates xfinal and yfinal if there is a real finite image, or ufinal and vfinal if the image is at infinity. As with any spot diagram, it's essential that your illuminating beam fill your optic's entrance pupil so that the pupil and its aberrations are well sampled. Use plenty of rays. The built-in ray generator can be used to set up a pupil containing dozens or hundreds of ray starts, sufficient to detail your point spread function or wavefront error.

The simplest MultiPlot is a **through-focus diagram**. An example is shown in Fig 15-1. The idea is to illustrate how the point spread function varies as the optical system focus is changed. So, to create one of these, first identify some optical parameter that controls your system's focus. This is often the Z axis location of the final focal surface, or it can be some internal element surface curvature or surface location that controls focus, such as in the TMA example above where the secondary mirror's axial location is represented by Z2. Then fill in the rest of **Options::MultiPlot** for your computation.

How much image space should the individual plot boxes span? To start you'll want to leave the box spans empty, or 0.0, which gives automatic sizing, fitting your largest spot diagram into the boxes. This will give you an overview of the diagram shapes. The box size is displayed at the top of the finished chart. Later, you may wish to set specific nonzero values for the box spans to show a nominal round-number size or a pixel size for comparison.

A compound MultiPlot is produced by choosing two step parameters. An example is shown in Fig 15-2, which is a sequence of through-focus plots for each of three field positions. Again, **Options::MultiPlot** sets up this additional parameter, as you can see in the left hand portion of the figure. The three rows of spot diagrams show that this TMA telescope is well corrected at off axis angles of U0=0.005 (bottom row) and U0=0.01 (in the middle row) but not nearly so well corrected further off axis at U0=0.015 (top row).

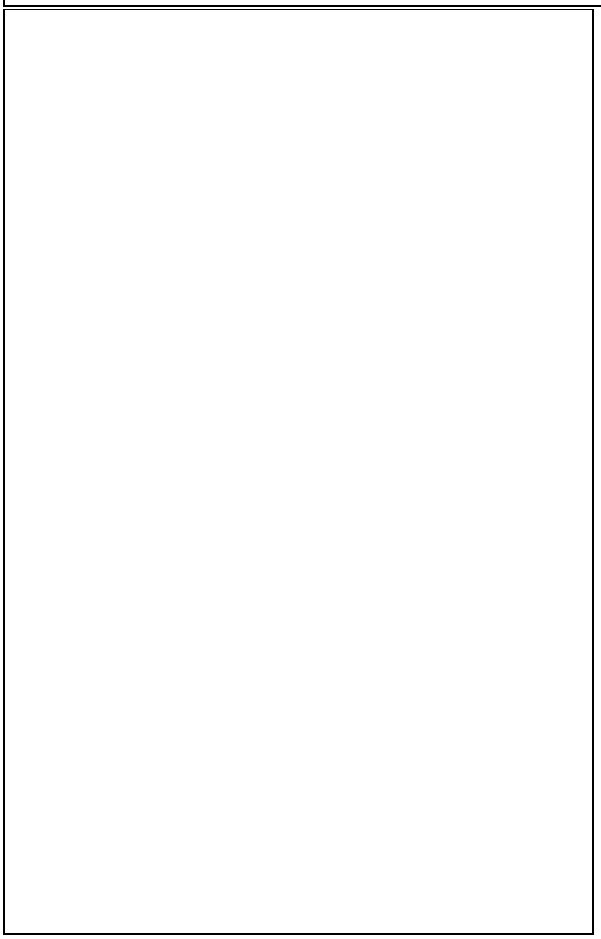


Fig 15-2 MultiPlot (top) with a 5x3 grid of spot diagrams computed from the TMA telescope example. The horizontally stepped parameter is again the secondary mirror location Z2 which controls telescope focus. We have added a vertically stepped parameter: the ray group off-axis angle U0 which explores the vertical field of view. In the **Options::MultiPlot** dialog at left you see how these are introduced: the three U0 values diagrammed are 0.015, 0.010, and 0.005. As many as three optics or ray-start parameters can be stepped at once.

Statistics can be calculated and displayed for each box. These can be selected using the checkboxes along the bottom row of the **Options::MultiPlot** dialog:

H: the value of the topmost horizontally stepped variable for each box;
V: the value of the topmost vertically stepped variable for each box;
n: the number of ray hits shown in each box;
h: the mean of the horizontally plotted ray variable = horizontal center value;
v: the mean of the vertically plotted ray variable = vertical center value;
hh: the root mean square of the plotted h deviations from their mean;
vv: the root mean square of the plotted v deviations from their mean;
hv: the root mean of the (h.v) deviations, negative if <h.v> is negative;
s: the root of $hh^2 + vv^2$ which is the average radius in two dimensions;
m: the root of $0.5*(hh^2 + vv^2)$ the average radius projected onto one dimension.

When you are displaying more than a few plot boxes, they are small and there isn't room for more than one or two of these results to be shown. In a pinch you can reduce the display font size with **Options::Graphics**.

You can of course display just one plot box. With the step variables left blank, a 1x1 MultiPlot acts very much like Plot2D but with spot diagram statistics available. It also offers a way to temporarily commandeer one or two optics parameters or ray start coordinates: just fill in one or two step variables and the desired center values.

Chapter 16: The RUN Menu: Map

The preceding chapter helps you explore the field of an optical system or some range of system parameters: it shows how the spot diagram of a single field point varies. The Map function is similar, but instead of showing the spot diagram details, it shows a color coded chart that indicates how the wavefront error or point-spread function varies with ray or optics parameters. Before using Map, set up your optics and ray information for a single field point, and verify that your ray trace proceeds successfully, i.e. most of your rays reach your final surface. Then, Map facility can show you the effects of varying one or two parameters over specified ranges. This setup is done in the Options::Map dialog, sbown below in Fig 16-1.

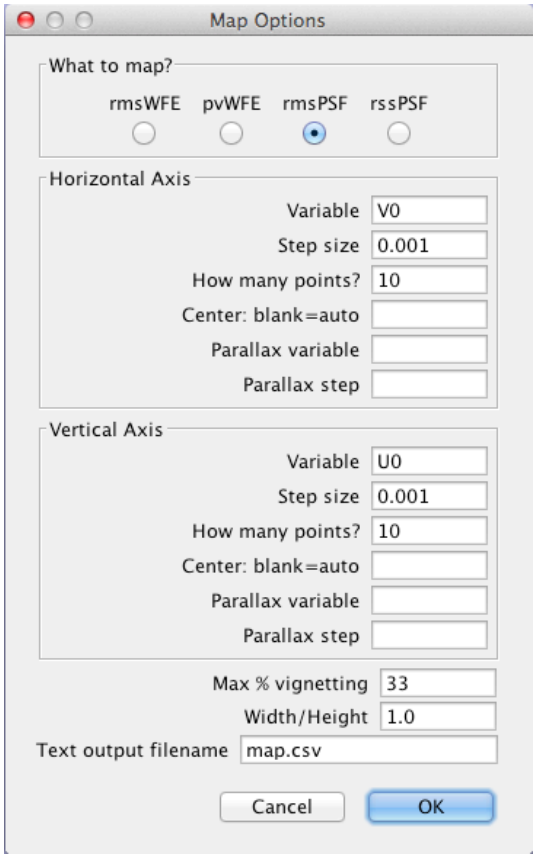


Fig 16-1: The Options::Map dialog lets you set up the Map parameters. The four radio buttons gives you the choice of mapping the rms WFE, the peak-valley WFE, the rms 1D PSF or the rss 2D). Each axis has its own plot variable that can be chosen from among the six ray start variables (X0, U0, etc) or from the positions, orientations, curvatures, or asphericities of any of the optical surfaces (Y2, C4, etc). If you are exploring ray starts and are not beginning your trace at the entrance pupil, you may want to a parallax variable to keep your rays centered on the pupil while being launched from a range of positions or angles. Up to 100 by 100 map points may be specified. You may specify a maximum percent vignetting, beyond which the map cells will turn gray as a warning of light loss. You may specify the map's aspect ratio, i.e. ratio of width to height, to help illustrate the relative range of field spans for the two axes. Finally, if an output filename is specified, the results will be saved as text.

Parameters to be mapped can be any combination of surface parameters (locations, tilts, curvatures, aspherics, refractive indices provided they are not looked up from a media table, etc) and ray start coordinates (X0, U0, etc). The option "Max % vignetting" allows you to tolerate a fraction of rays getting lost at each parameter step. The default value, zero, causes any lost ray to turn its map cell gray, meaning no valid WFE or PSF for that setting. If your application tolerates a few percent lost rays you can change this option to suit your needs. We caution that the lost rays are probably the most discrepant rays with

regard to WFE and PSF, and their loss will surely bias your map towards smaller values of WFE or PSF.

The text output feature allows you to save the map results in numerical form for further work. Each plot point produces one text record containing seven values: the values of the stepped variables on the horizontal and vertical axes, the ray count for each map point, the centroid {X,Y,Z} for those rays, and the value being mapped. If you specify a filename ending in ".csv" then the output fields will be comma separated values, for easiest entry into spreadsheets. Otherwise the fields are spaced to fixed width.

```
V0,   U0,  Ngood,   Xf,   Yf,   Zf,  rmsPSF
  -0.004500000     -0.004500000       364       62.807327       62.805067    -11125.102301       0.009876693
  -0.003500000     -0.004500000       365       62.780266       48.827327    -11124.939401       0.009564741
  -0.002500000     -0.004500000       365       62.759947       34.865382    -11124.818163       0.009273229
  -0.001500000     -0.004500000       366       62.746426       20.914747    -11124.737799       0.009056140
```

Fig 16-2 Illustrating the first few lines of a text output listing.

Once Map is set up, the Run::Map menu item starts the mapping process. In a map with 41x41 field points, like the present example, the computation may be time consuming, and BEAM FOUR shows its the map progress as successive field points are added and the thermometer is updated. The color scheme assigns blue to the least-blur map locations and red to the worst-blur locations. The units are the same as used in your geometrical definition of your optic (meters, mm, etc).

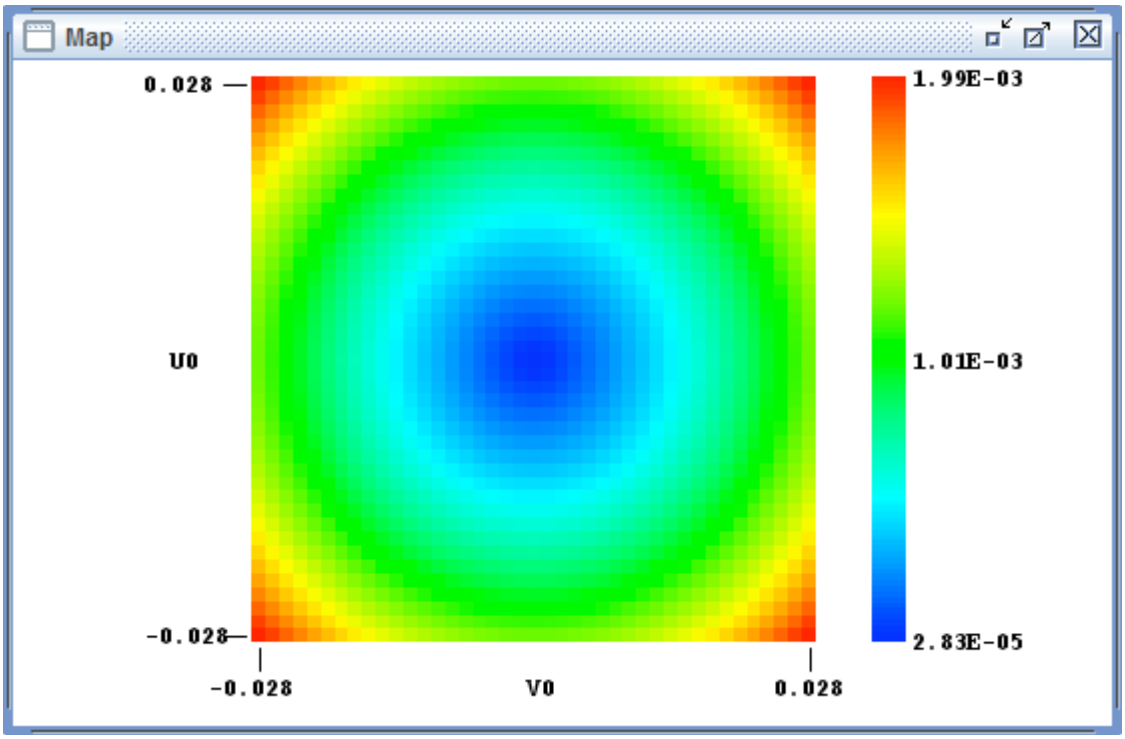


Fig 16-3 Example of a map, showing the RMS WFE for an aspheric singlet lens as a function of the off axis illumination angles V0 and U0. Optics with greater complexity have correspondingly more complicated maps.

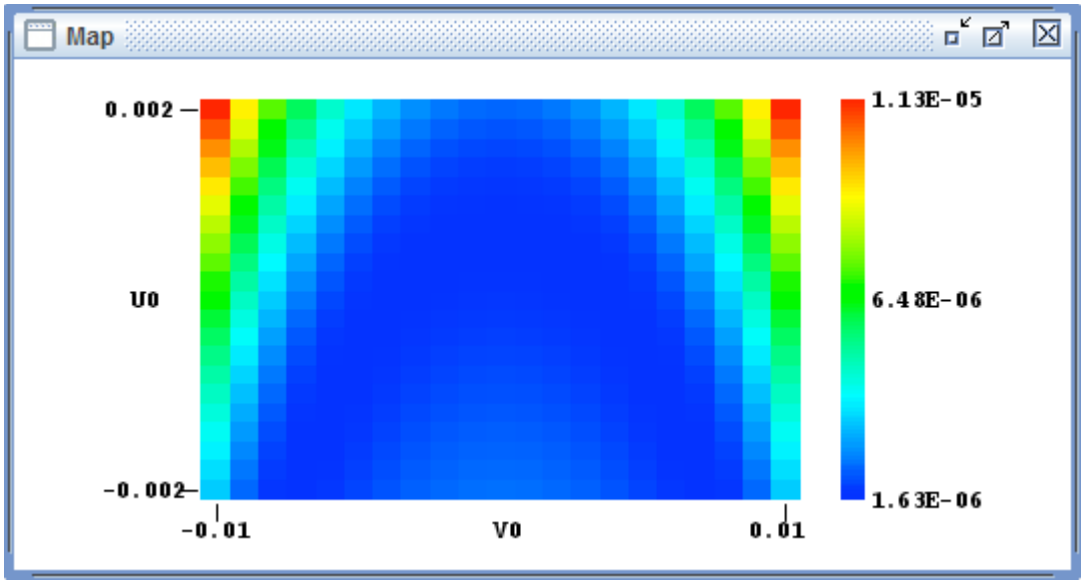


Fig 16-4 Another example, showing the RMS WFE for a three-mirror anastigmat telescope as a function of the off axis illumination angles V0 and U0.

Choosing your parallax values: parallax is needed when your ray starts do not lie in the same plane as your entrance pupil. The idea is of course to displace the ray starts for each off-axis illumination angle so that the pupil remains properly illuminated. At small angles, the parallax needed per step is –Ustep*L where Ustep is the step size in U0, and L is the separation between the ray start plane and the pupil plane. However if there are intervening optics, or if high accuracy is needed, it is important to check the pupil stabilization. To do this, in your .OPT file temporarily set Nsurfs to the pupil surface, and run Map, saving its output file as (say) Test.csv. Then examine Test.csv: if the pupil has been stabilized, the centroids Xf and Yf will be independent of U0 and V0.

Chapter 17: The RUN Menu: Plot 3D

Plot3D serves the same purpose as Plot2D but is most useful when there are two independent variables that jointly control some feature of your trace. As an example we may ask, how does optical path vary over the pupil of the relay optic shown in the previous chapter? Figure 17-1 below shows Pfinal as a function of the X1 and Y1 coordinates of each ray's trace. The red and blue ray groups are separated owing to the greater greater optical thickness (higher refractive index) of the lenses in the blue. The variation in path over the pupil is evident.

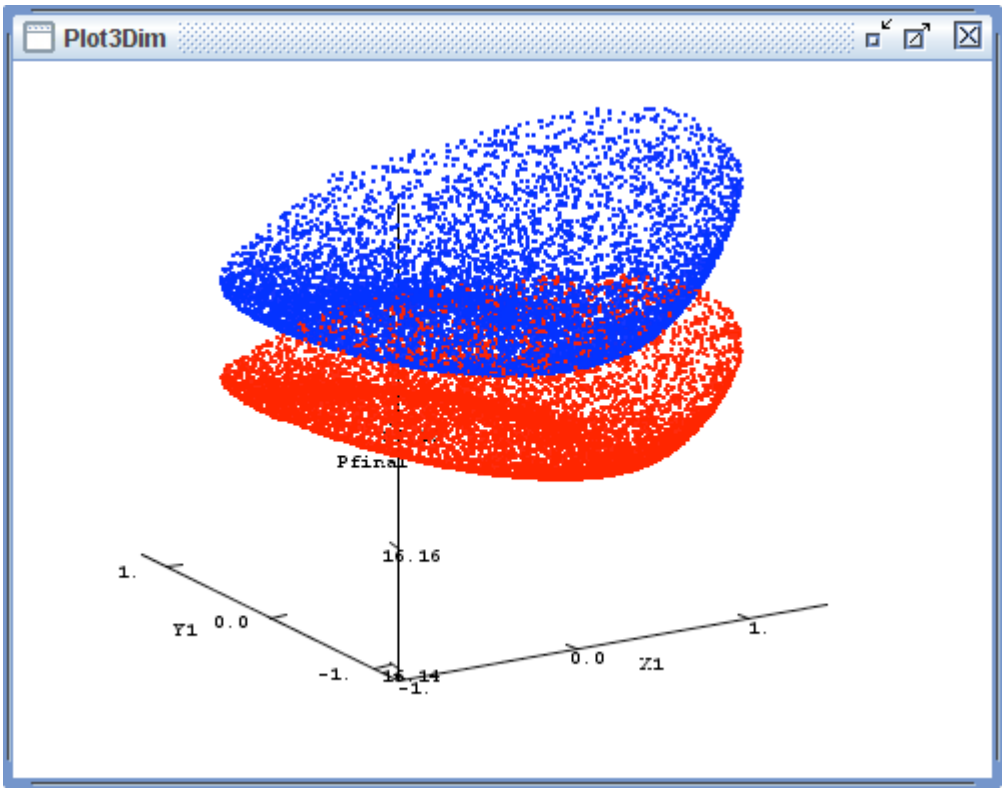


Fig 17-1: Example of the Plot3D function. Each plotted point has three display coordinates selected from each ray's trace, using **Options::Plot3D**. Each coordinate can be chosen from more than a dozen ray variables at each optical surface, so that highly customized diagnostic plots can be generated. Here, the **Run::Random** feature has been run to augment a small number of user-specified rays, to fill in the optical relationships among the chosen ray variables.

In an optic that is to be nearly diffraction limited it is important that the wavefront error remain nearly constant over the pupil, for all wavelengths and field positions of interest. The Plot3D function is well suited for constructing such displays. Figure 17-2 below shows a peak-to-valley wfe that is about 150 nm, nicely suited for work at visible and near infrared wavelengths. To make a plot like this, use the Options::Plot3D dialog to set the three plot variables: A=X1, B=Y1, and C=WFE. You can use the default spans to get

a reasonable image, or tailor the individual spans manually for an improved presentation. The Run::Random feature is used to populate the diagram with many thousands of rays.

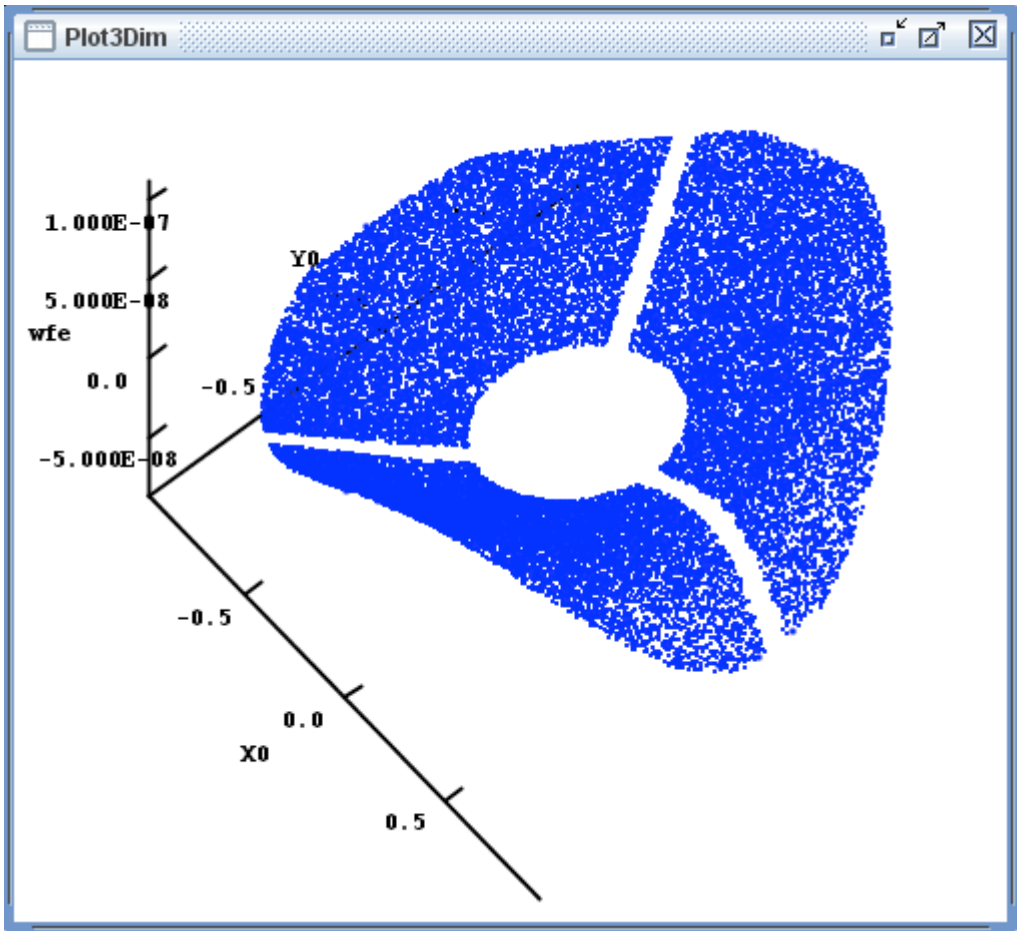


Fig 17-2: Wavefront error over the pupil of a three-mirror anastigmat telescope that has a central obstruction and a three-legged spider, for a single off axis field point.

The Plot3D output function is set up at the Options::Plot3D dialog (see Fig 17-3 below). There, the three plot variables are specified, along with their plotted spans. Setting a span to zero (the default) lets BEAM FOUR select an automatic span value that shows the range of the ray variables presently represented within your current ray trace. The View Elevation and View Azimuth let you choose your starting viewpoint angles in degrees, but of course you can modify this view by right-dragging the plot with your mouse. Four dot styles are available for plotting ray triplet coordinates, chosen with the radio buttons. The "Which Rays" dialog lets you elect to plot all rays that have proceeded far enough to define the three specified coordinates, or restrict the plot to good rays only, i.e. those that propagate all the way to the final defined surface.

Like Layout, three drawing formats are available: white background, black background, and a stereoscopic display. See Layout (Chapter 13) and Stereo (Chapter 29) for details of these features.

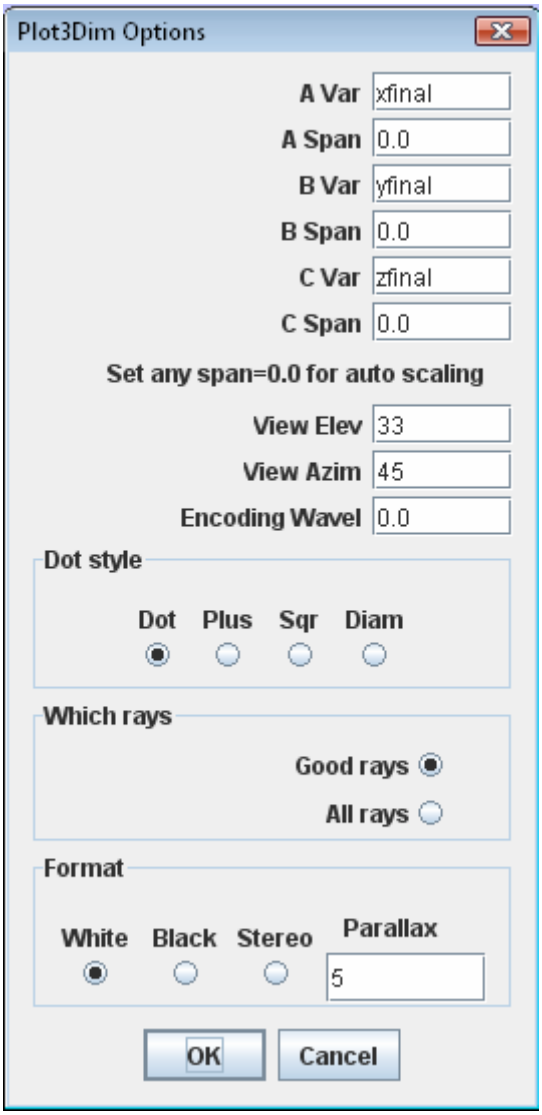
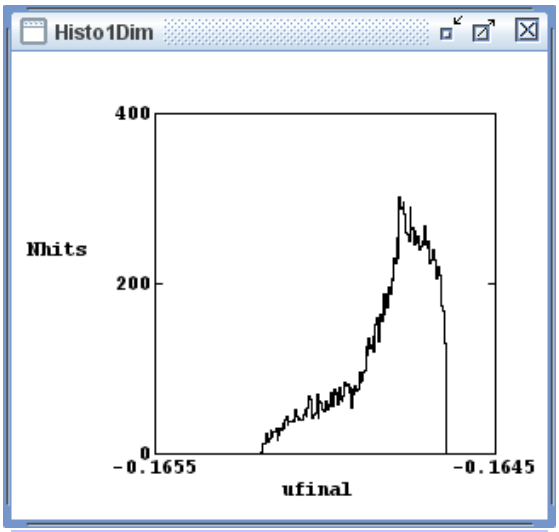
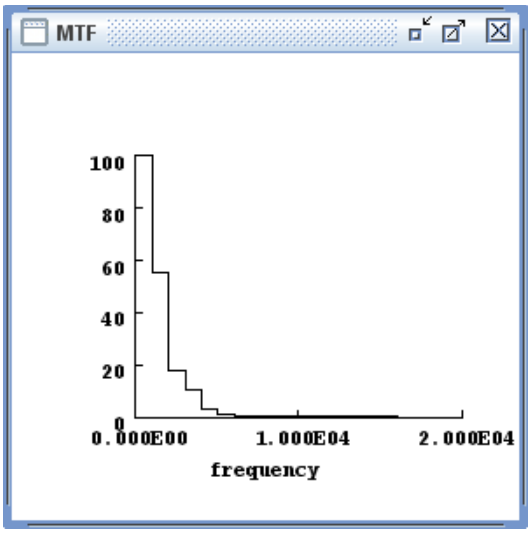


Fig 17-3: The **Options::Plot3D** setup dialog. The three display axes are identified here as A, B, and C. Specify which ray variable is to be mapped onto each axis. You can also set its display span. The default span is zero, which gives automatic scaling for that axis, based on the range of ray coordinate values found in your current optical trace. The initial view direction is specified by the elevation and azimuth angles, in degrees. Once displayed, the view direction is easily changed with the mouse right-button twirl feature, letting you get the best view of your data. The "Dot style" box lets you choose between four plot symbols: a dot, a plus sign, a small square, and a diamond. In the "Which rays" box you can elect to plot all rays that reach the three surfaces specified in A, B, and C, or plot only the good rays (those that arrive at your final surface). Finally there is a display format box, letting you specify a white background, a black background, or a stereoscopic display with red and blue plots superposed onto a black background with a selectable parallax. (You'll need red/blue spectacles to see this.) For best results, choose a parallax that gives a slight divergence between the red and blue plots.

Chapter 18: The RUN Menu: Histo 1D and MTF

A histogram is a way to visualize the probability distribution of a single variable. In optics, when a pupil is illuminated in a realistic way, a histogram of any one of the resulting ray variables yields a view of how the intensity of light is distributed with respect to that variable. BEAM FOUR has a built-in one dimensional histogram function that lets you gather a distribution of any computed ray variable and display it. This can then be used qualitatively, illustrating the breadth of an image feature, and quantitatively to evaluate the moments of the distribution or --- with the included MTF (modulation transfer function) feature --- the Fourier spectrum of the distribution showing the relative amplitudes of the spatial frequencies in the distribution.





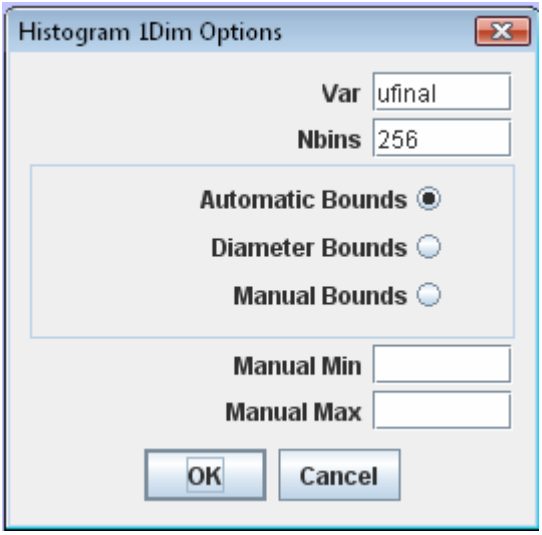
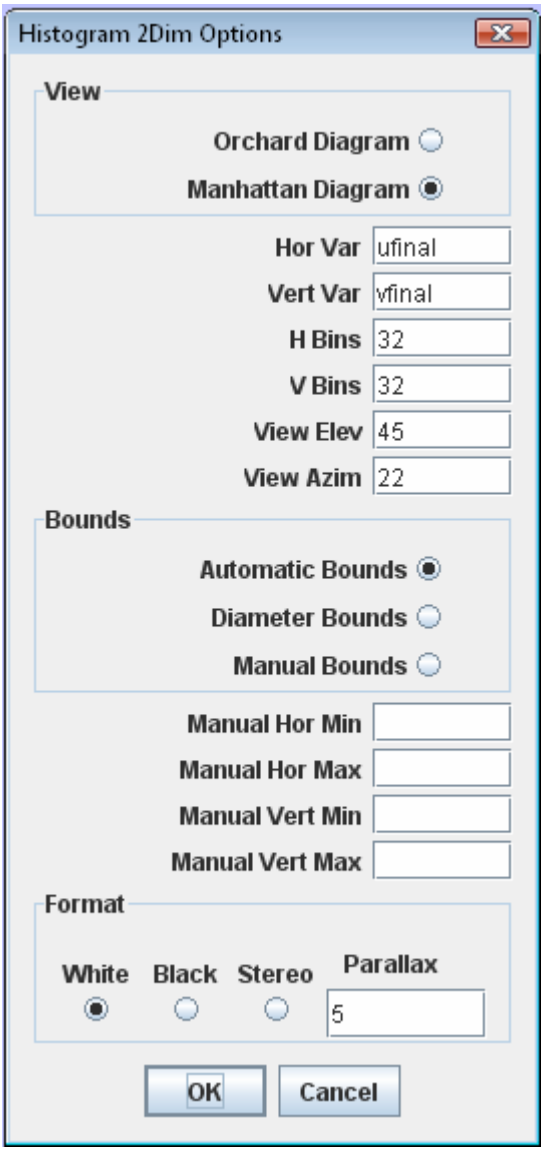


Fig 18-1 The basic one-dimensional histogram is set up using the Options::Histo1D dialog shown at left. The resulting 1D histogram, built from 40000 random rays, is shown at the upper left. Its chief use is quantifying the distribution of energy along any one axis within an image. Once it is computed, its MTF can be calculated with Run::MTF and the results are shown at the upper right. The histogram counts can be saved as a numerical text file using File::WriteHisto.

The automatic span feature of Histo1D yields reasonable resolution on the histogram but not particularly good resolution in MTF. For finer MTF plots choose a wider histogram span i.e. set manual bounds further below and above the image centroid position.

Chapter 19: The RUN Menu: Histo 2D

A two-dimensional histogram is a view of the relative frequency of any pair of ray variables. BEAM FOUR generates these. The example here shows the distribution of (ufinal, vfinal) values produced by the CASS.OPT telescope illuminated by the ray file CASS.RAY at 3 milliradians off axis. Compare with the 1D histogram results in the preceding chapter.



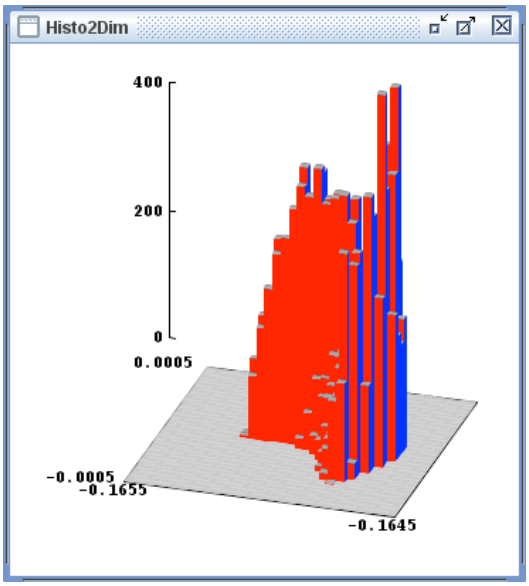


Fig. 19-1 Example of the Histo 2-D function. Two ray variables are chosen at the dialog Options::Histo2D. Then, Run::Histo2D traces the table rays and produces the histogram. Additional rays can be generated using Run::Random to populate the histogram more fully. Graphic display features Pan, Zoom, and Rotate use the mouse buttons and wheel, as with Layouts. Each graphic can be written to disk in a variety of CAD formats described in Chapter 23 "CAD Functions" or numerically using File:WriteHisto.

Chapter 20: The RUN Menu: AutoAdjust

Often an optical designer will have a conceptual optical train but will need computational help in setting some of its parameters. Element locations, surface curvatures, and aspheric coefficients are likely examples. BEAM FOUR has a built-in AutoAdjust capability to fine tune an optical system to reduce the sum-of-squares of an optical demerit function defined on the final ray coordinates, with the adjustable parameters being tagged in the .OPT table. This least squares optimizer works for parameters that are continuously differentiable (curvatures, locations, and most of the surface descriptors in the .OPT table) but not for discontinuous choices such as specific glass name or various numbers of optically active surfaces. So the autoadjust process is useful when you have a feasible optic --- one that at least moves its rays from their starting points to the final surface --- but not when you lack a starting point design.

AutoAdjust can adjust ray starts as well as optics parameters. By marking some ray starts as adjustable, you can discover what aiming conditions rays must satisfy to reach their intended targets. This can be particularly useful if a number of ray starts are ganged together (see below) so that an entire beam can be targeted in an optimum way.

Designating adjustable parameters is accomplished the same way for optics tables and ray tables. In either case, the tag letter or symbol is the key. The tag is the character that follows a parameter value and shows the end of the data field. The default character is a colon, indicating that the parameter is not adjustable. Editing the table to replace one or more data tags with question marks or tag letters makes them autoadjustable.

Independent variables are tagged with question marks. Each "?" variable is adjusted independently, without a direct connection to any of the other adjustables.

Ganged variables are tagged with letters. Every variable within an optics or ray table field that is tagged with an "a" for example will receive the same change in value as every other "a" variable in that field. Any letter will do. If your co-named variables share the same starting value, they will remain equal throughout the autoadjustment process. If they are set up with some initial difference, they will retain that difference. Up to 26 ganged adjustables can be simultaneously present within any one field --- but in almost all optical design situations just a few will do the job.

Anti-ganged variables can also be set up. If a variable is tagged "R" for example, then other parameters in the same optics or ray field tagged with "r" will be linked to the "R" adjustable but with the opposite sign. Changes to one will occur to the other but with the opposite direction. This way you can set up symmetrical optics for autoadjustment and have them remain symmetrical throughout the adjustment process.

Focussing an on-axis point object takes just two steps. We assume here that your optic is coaxial and that your image centroid will therefore lie at (xfinal, yfinal) = (0, 0). We also assume that you have identified a few optical parameters that you want BEAM FOUR to adjust, and that you have tagged each adjustable parameter with a question mark tag. Set up two goal columns, xgoal and ygoal. Fill in zero values for both goals for each ray: xgoal and ygoal for a real image, or ugoal and vgoal for an image at infinity. (If you have a virtual image, interpose a retroreflector to make it real.). Then click Run::Autoadjust. BEAM FOUR will compute a sequence of improvements -- adjustments of the parameters that reduce the root-mean-square difference between the final ray coordinates and the goals specified.

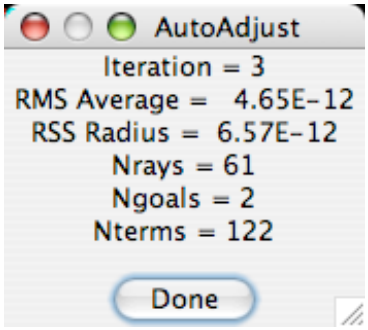


Fig 20-1 AutoAdjust popup dialog. It counts off the iterations run and displays the number of comparison points (good rays × goals per ray). The successive values of the RMS average deviation between rays and goals are shown. RSS radius (root sum square of x and y blurs) is shown only when Ngoals=2. A caution appears only when WFEs are mixed with other goal types (usually unwanted). While running, the button function is [STOP]. When complete, the button changes to [DONE].

Focussing while holding magnification constant is only slightly more complicated. You will need to set up some ray starts representing two object locations, for example one location on axis and another location off axis. Then for the first set of rays, show goals that are zero, and for the second set of rays show goals that are nonzero.

Focussing an off-axis object takes just three steps. We assume here you don't know or care where the optimum location (x,y) where the focus will fall, but you simply want it to be as tiny a spot as possible. We assume you have a few unknown optical system parameters that you'd like BEAM FOUR to adjust for you. Here is what to do.

1. Set as goals xgoal and ygoal, and fill in goal values of 0.0 for these columns. Note the use of lower case here: ray coordinates measured in the frame of the final surface.

2. Set as adjustables the X and Y values for the final (focal) surface, in addition to any other adjustables you have chosen to be optimized. This means put columns in place in your .OPT table headed by X and Y if you do not already have them, and fill in starting values for X and Y for your final surface if they are not already specified. Put question marks into the tag columns for Xfinal and Yfinal.

3. Run AutoAdjust. During this run, you will see the location of your final focal surface X and Y be adjusted, along with your other optical parameters. The final values of X and Y will be the locations where the centroid of the spot pattern lies since in the frame of the focal surface the average xfinal = yfinal = 0.0.

Examples: We now turn to a few examples showing how the tables are to be set up for an adjustment run. Bear in mind that these toy examples are highly simplified, to show the syntax of the table entries compactly. Realistic optimization will usually require a large number of rays that your pupil and span a representative range of field positions.

Example 20.1: To begin, Fig 20-2 shows a positive power lens making an image of a point source at infinity: the incoming rays are parallel but are located at various heights. Notice that the final (third) surface has a Z axis location that is initially set to 4.00, but it has a question mark tag, indicating that it is to be adjusted so that the best focus position can be found. The bottom figure is a layout showing this starting situation. The rays are good, i.e. they reach the final surface.

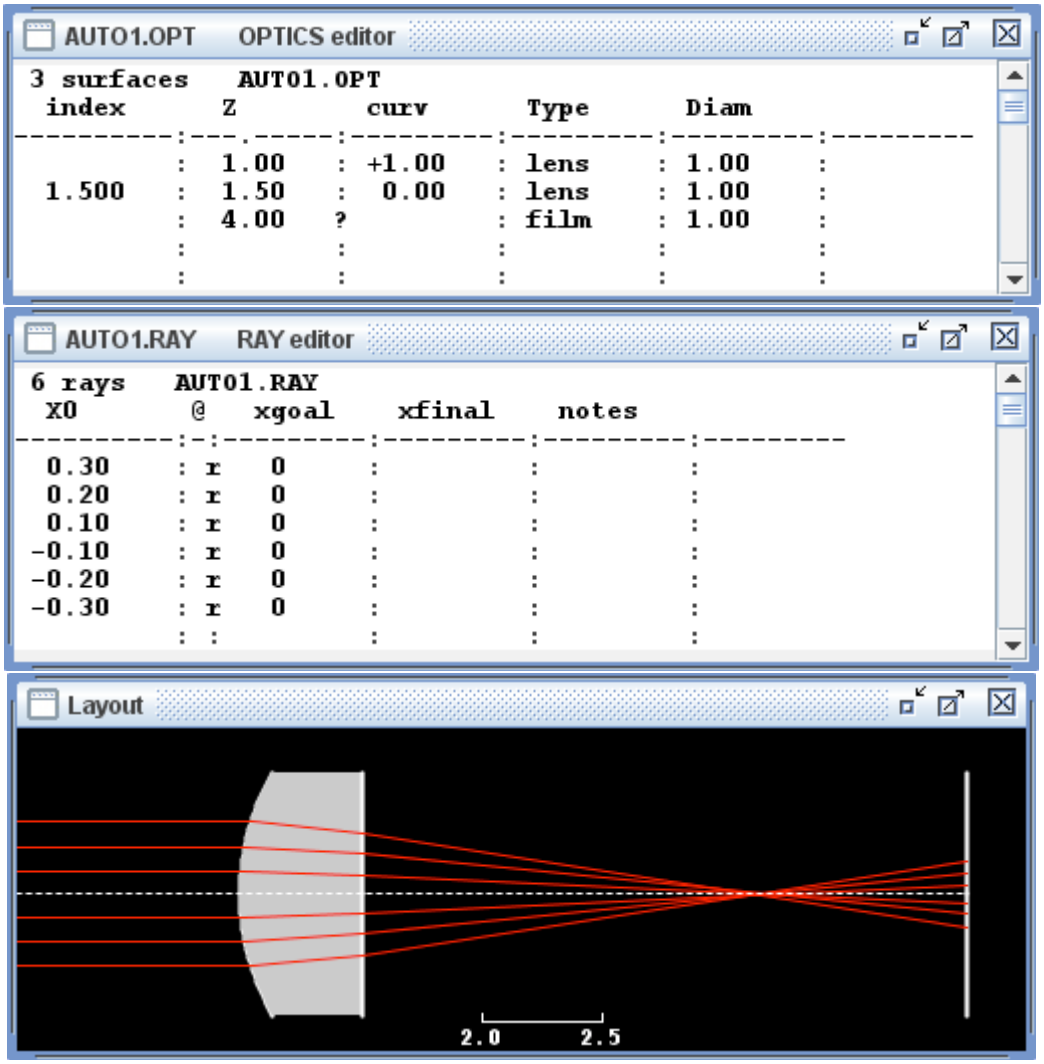


Fig 20-2 A positive lens (convex & plano) is illuminated with parallel rays. Initially it is seriously out of focus. Top: initial optics table. Middle: ray table. Bottom: layout.

After clicking Run::InOut, the Z value of the final surface becomes 3.12613 (Fig 20-3).

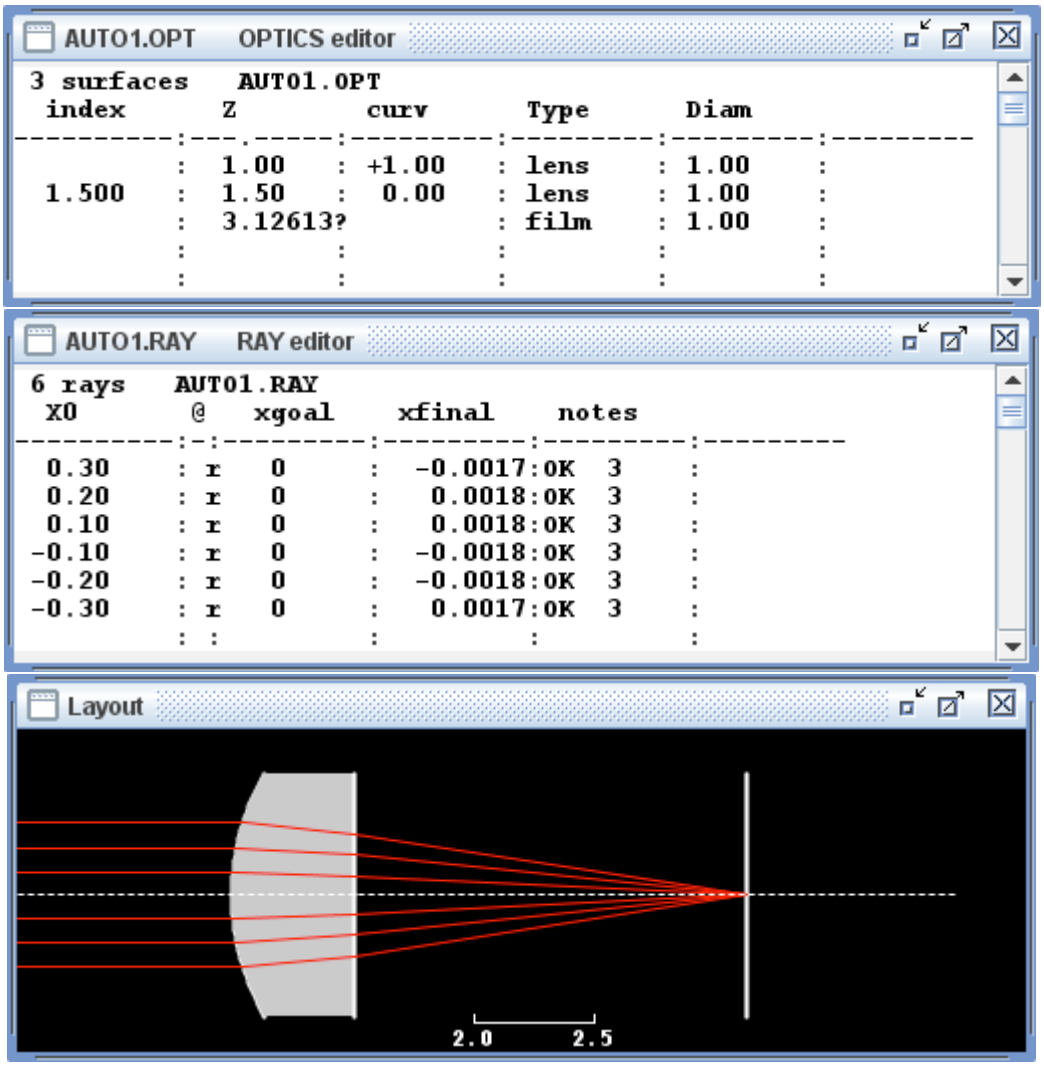


Fig 20-3 After autoadjustment, the optics table (top), ray table (middle), and layout are as shown here. The optimization has minimized the least squares deviation of the ray final values from their common declared goals, all equal to 0.0.

Example 20.2: Next let's look at another situation, namely moving a lens. In Fig 20-4 we see an .OPT table set up so the front and rear surface locations of the lens are adjusted together. The .RAY table is the same as Fig 20-3. After adjustment the lens has moved to a new Z location as shown in the middle and bottom parts of the figure.

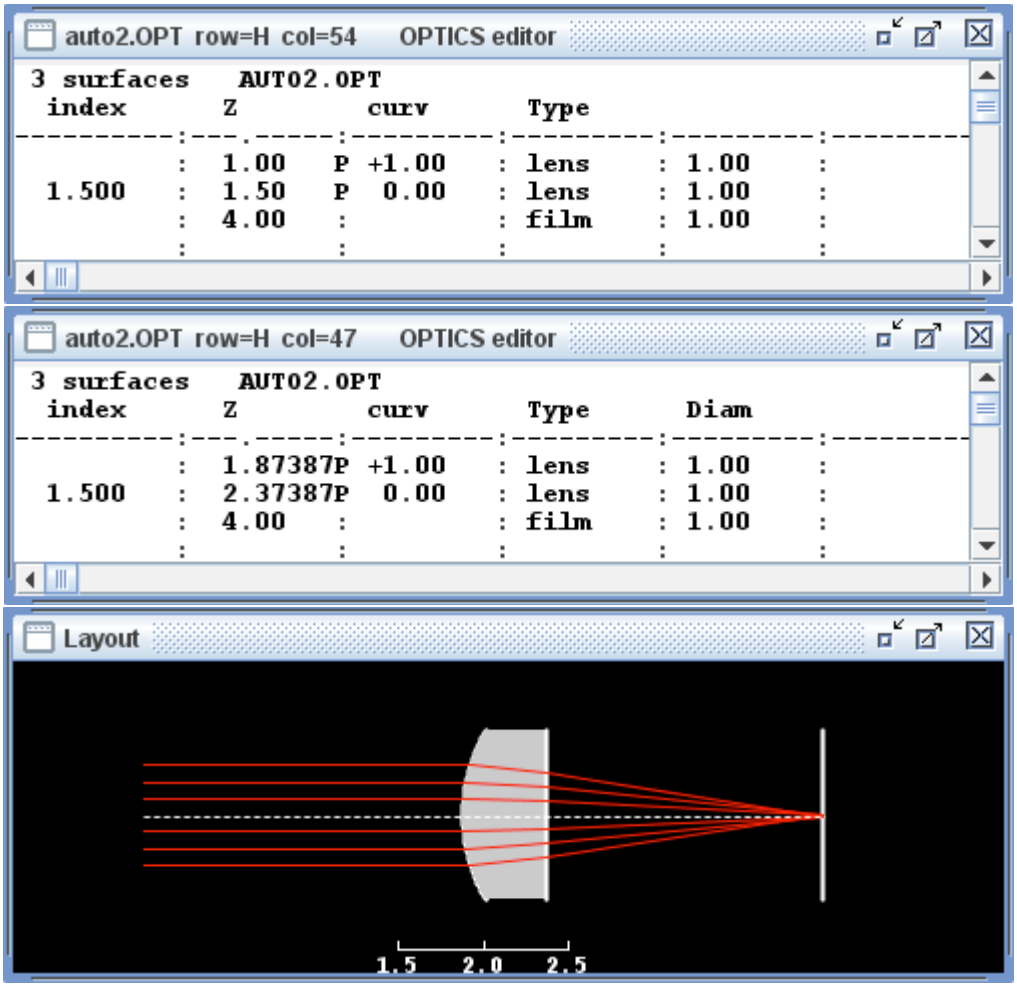


Fig 20-4 The front and rear surfaces of the lens have Z positions that are tagged with the same letter (here, "P") which gangs them together. Middle: after autoadjustment, they have migrated to the positions shown. Bottom: Layout after adjustment.

Example 20.3: Next let's alter a lens curvature, changing its power so as to place its focus at a specified location. In Fig 20-5 we show the .OPT table and its layout (rays are as before); here the front surface curvature is tagged with a question mark making it adjustable. After adjustment, the .OPT table and its layout are altered as shown.

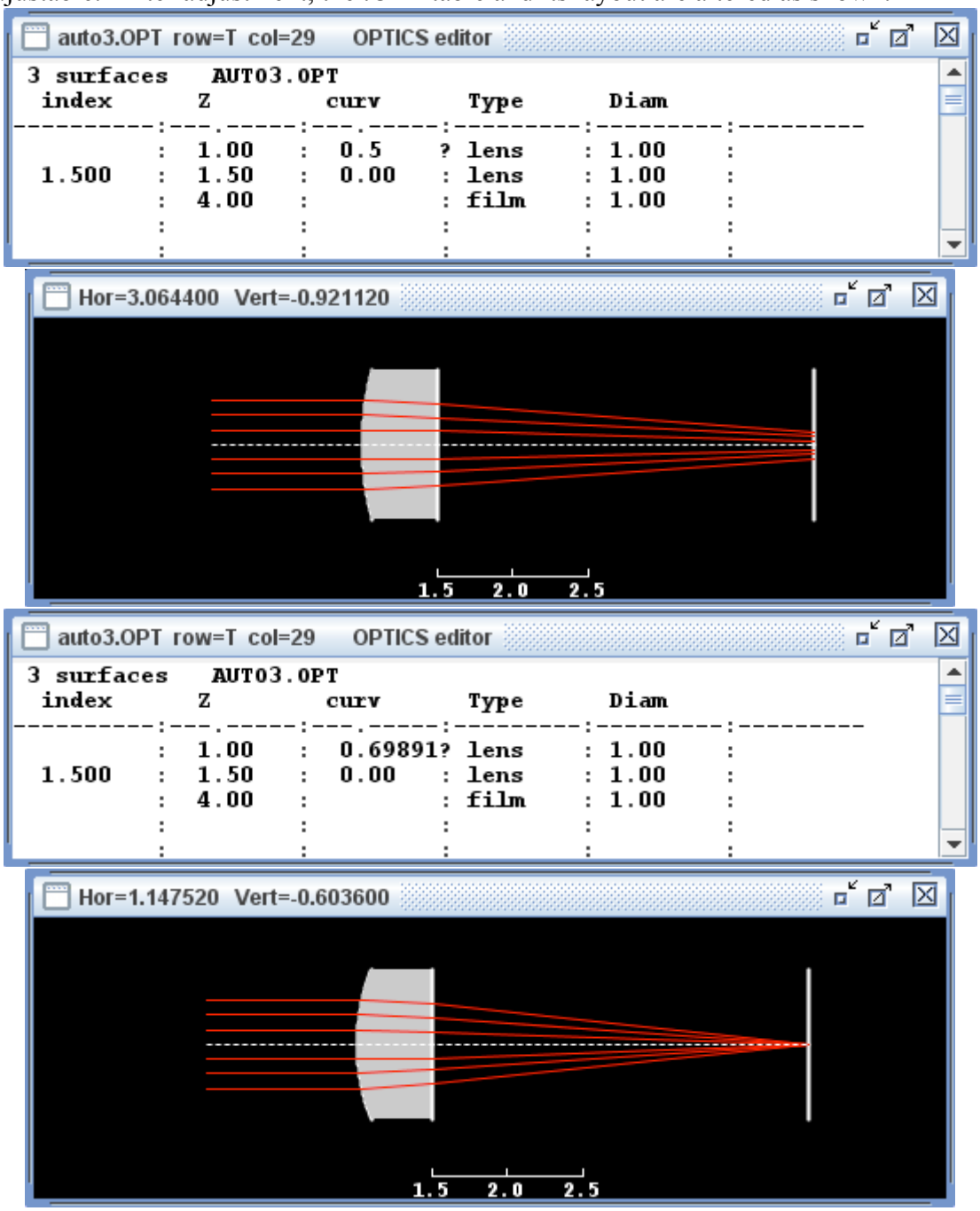


Fig 20-5 The curvature of the first lens surface is adjusted here, to place the lens focus at a given location (here, Z=4). Top pair: before adjustment; bottom pair, after adjustment.

Example 20.4: Next let's simultaneously adjust the front and rear curvatures of our positive lens to make a doubly convex lens that gives the smallest spot height for our given ray bundle. In Fig 20-6 we show the initial .OPT setup: both front and rear curvatures are tagged with question marks. The .RAY table is as before.

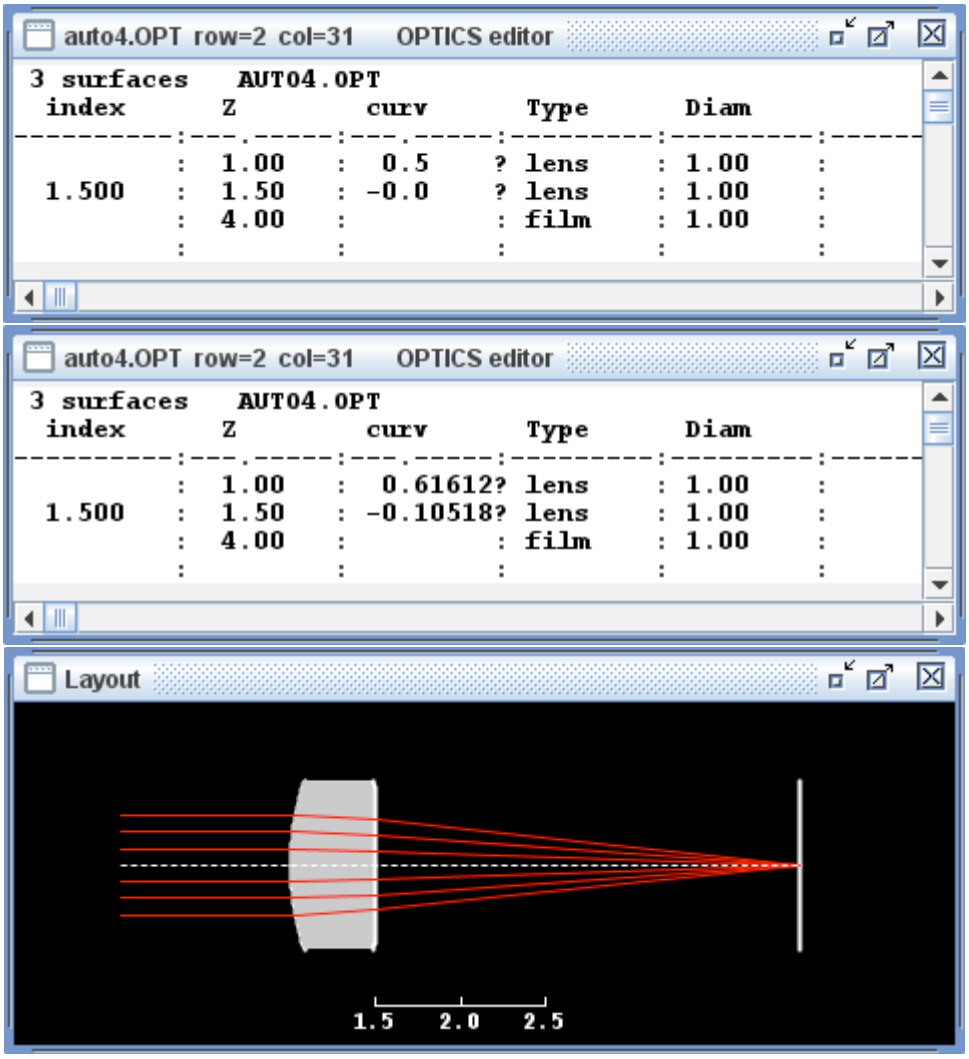


Fig 20-6 Simultaneous adjustment of front and rear curvatures of a lens. In the setup (top) the .OPT table tags both curvatures with question marks. The initial curvatures need be only good enough to allow the rays to make through the system. After adjustment (middle, bottom) the curvatures are optimized for this ray group.

Example 20.5: Now let's look at a classic discovery (R. Descartes, 1637) that spherical aberration can be entirely eliminated from a lens by making it an appropriate conic section of revolution. Here, we illuminate a plano-convex lens with parallel rays and mark its rear surface curvature and asphericity as adjustable, and then optimize. In Fig 20-7 we show the result.

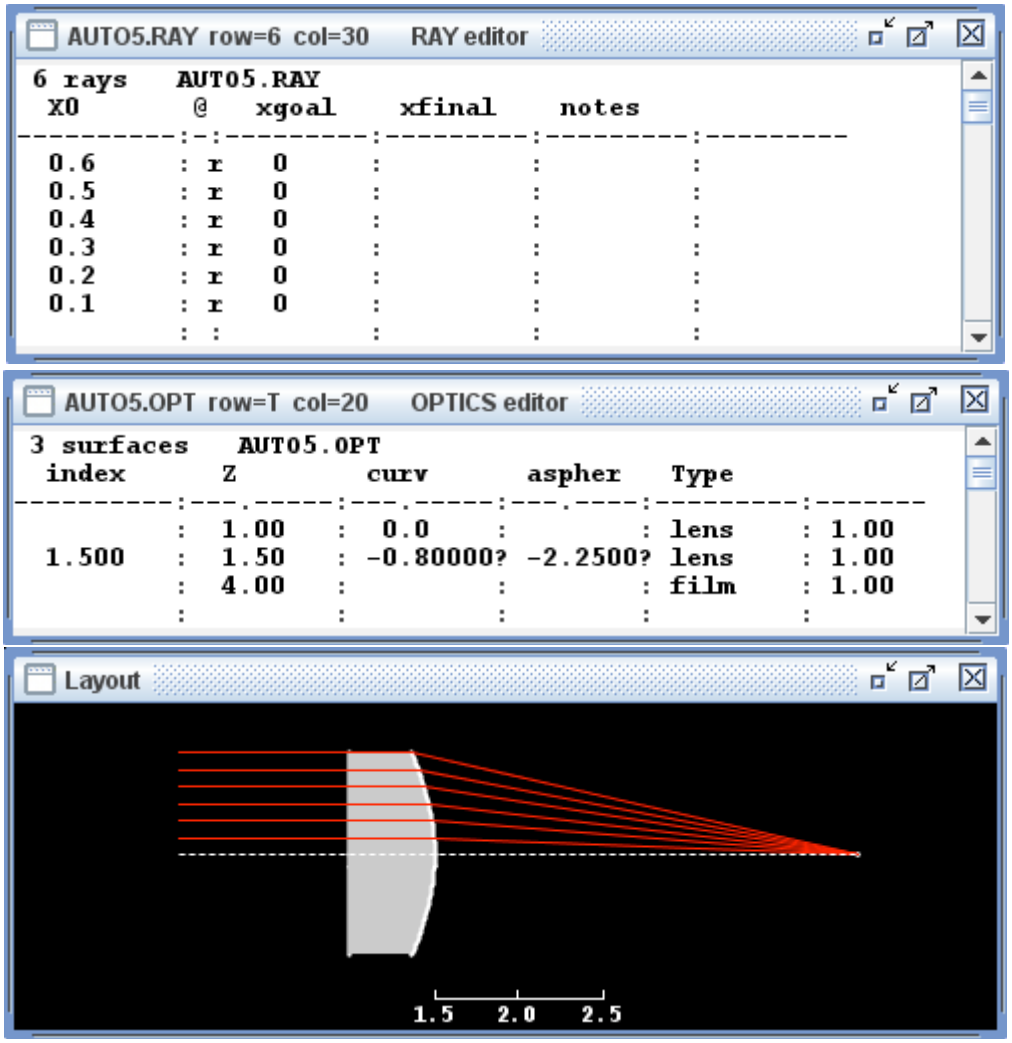


Fig 20-7 Adjusting a plano-convex lens to eliminate its spherical aberration by means of its conic constant. Top: ray setup; bottom pair, after adjustment.

The analytic solution for this problem is Curvature = -Power/(N-1), which is true for any lens whose rear surface alone is curved. The Descartes solution for zero spherical aberration yields Asphericity = $-N^2$. You can verify that these conditions are met in this example.

Example 20.6: Next let's look at a refractive relay lens with equal conjugates: the object and image distances are the same and the magnification = -1. In Fig 20-8 we mark the front and rear lens curvatures as adjustable using tags "q" and "Q" (any letter will do). These curvatures will remain opposite if they are started opposite in value. Because up to 26 letters are available, up to 26 variables in a field could be tagged, although usually only a few are needed. At the same time the asphericity is adjusted for both surfaces, tagged "h" and "h" so they remain equal if started equal. The result is a back-to-back Descartes relay free of spherical aberration.

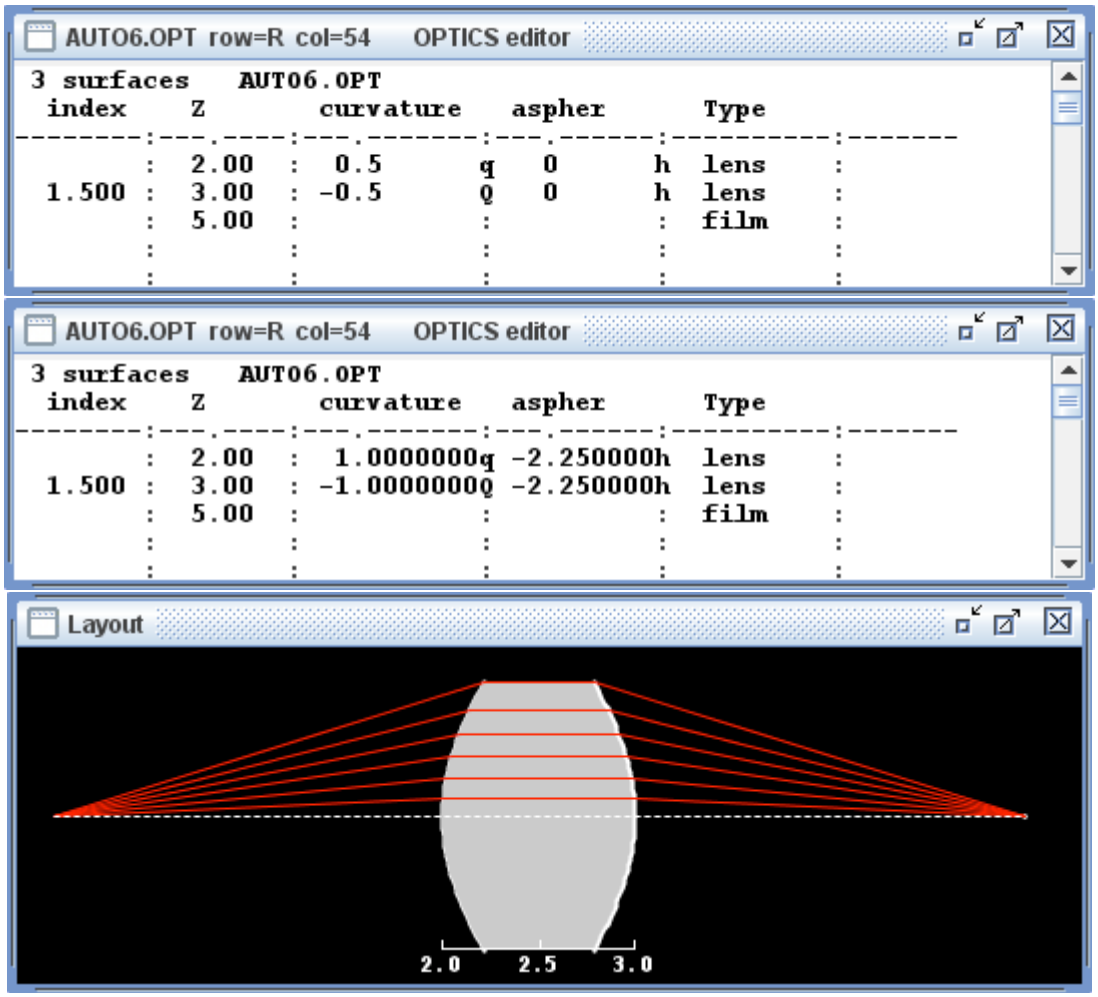


Fig 20-8 Adjusting two curvatures to be linked together but with opposite changes is done using tag letters of opposite case. Top: .OPT setup, note the "q" and "Q" tags. At the same time the asphericities are tagged the same to make them change together. Middle: optimized optic. Bottom: layout of the optimized optic.

Example 20.7: Next let's look at an autoadjustment set up using floating goals. Our example will be a Cassegrain imager working over a wide enough field that its field curvature and distortion are both significant. We fix focal length but ignore distortion. The fixed goal for the first eight rays fixes the target focal length at 0.05/0.01 = 5 units, while the floating goal (rays 9-16) accommodates the distortion at mid field height.

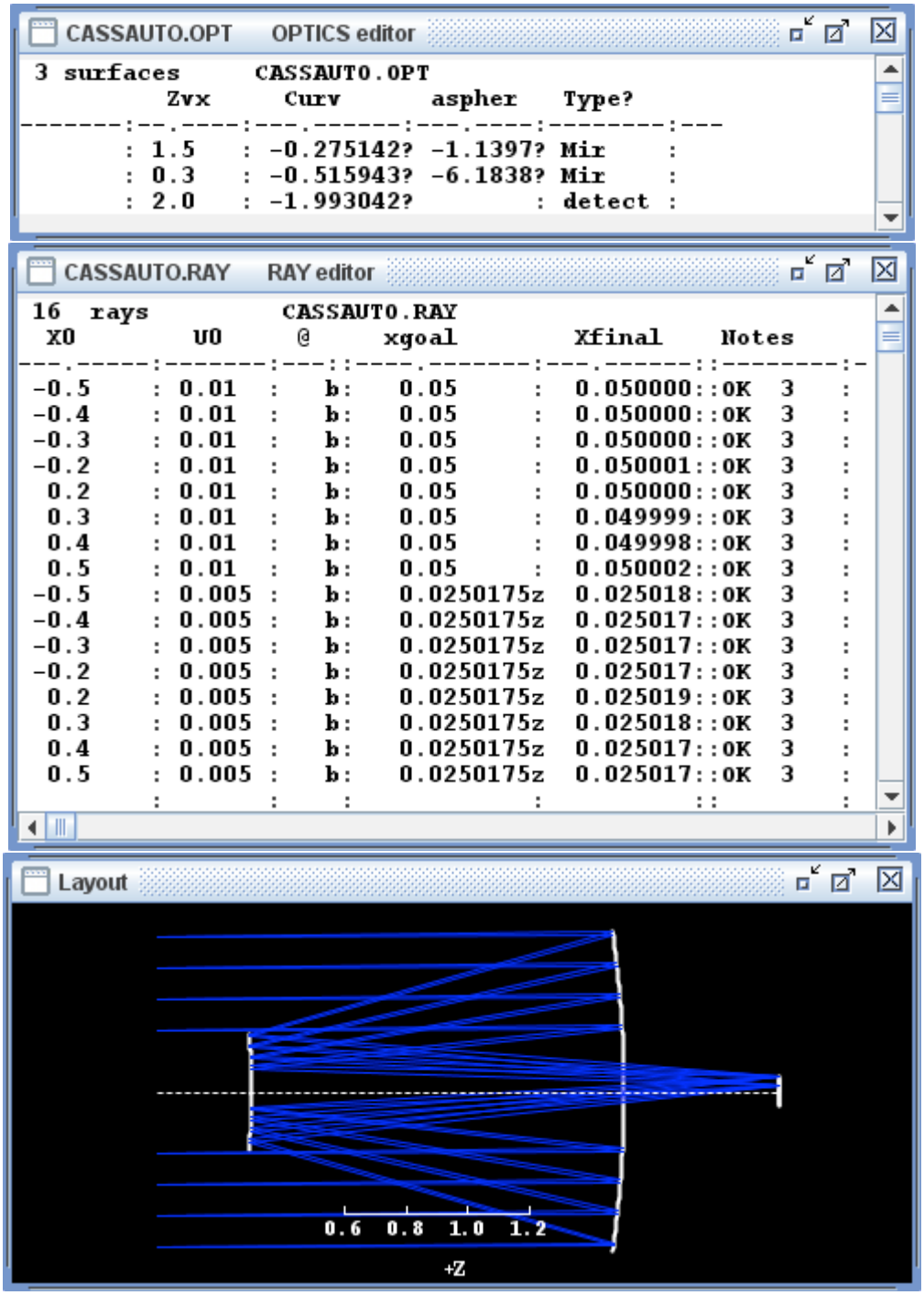


Fig 20-9 An imager is optimized in linear blur with one ray group goal set to a fixed 0.05 units while the other ray group tagged with "z" has a common unspecified floating goal.

Example 20.8: Finally let's work an example in which the ray starts are adjusted to deliver their final image spot to a given goal position on the focal plane. This is the reverse of the previous examples – the optic is fixed but the ray starts are the unknowns being solved for. We'll use the Cassegrain imager from the previous example but it will have fixed parameters (no question mark tags). The ray table is started with viable rays sharing the same U0 value in each group, but tagged so they vary together during adjustment. AutoAdjust fixes up the ganged ray starts to minimize the RMS discrepancy between the xfinals and the xgoals.

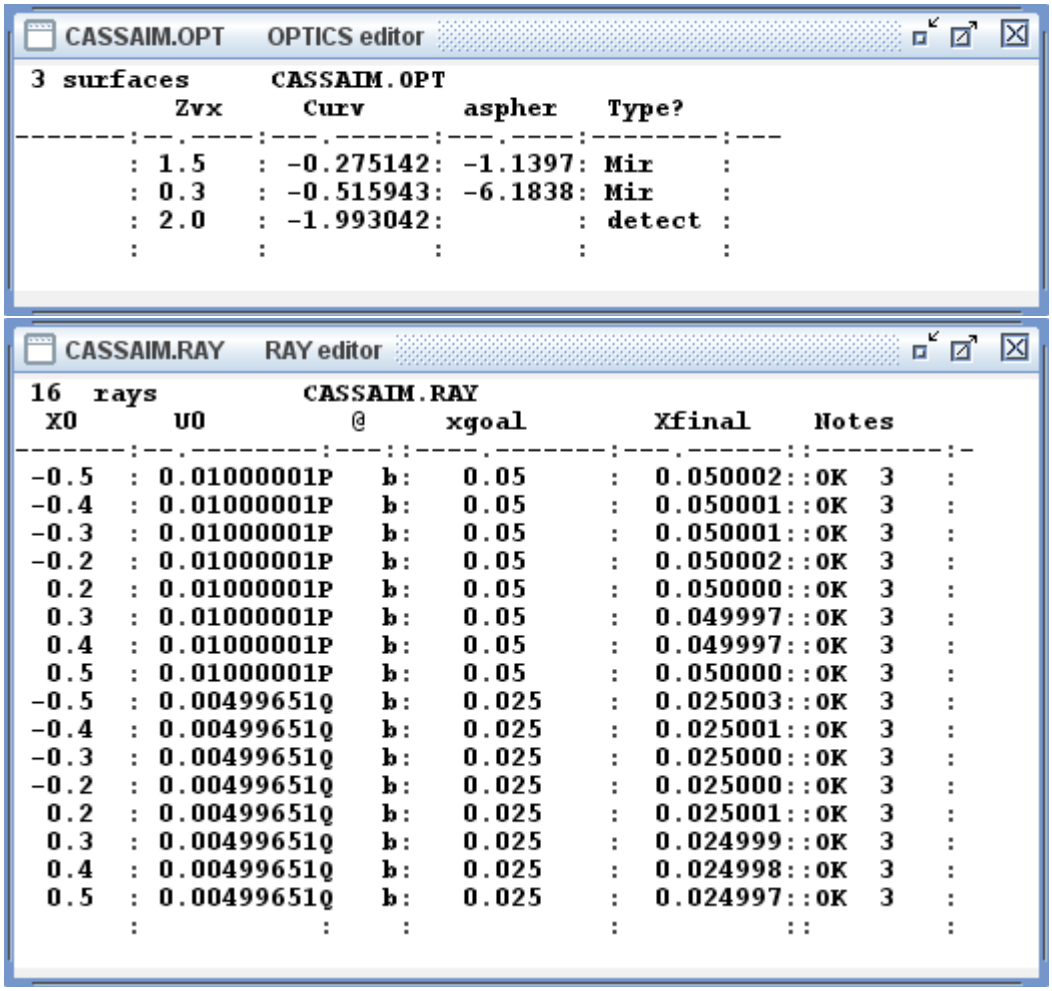


Fig 20-10 A Cassegrain imager is held fixed while its ray start directions are adjusted to best meet the given goal positions. The top eight rays form one group whose xgoal is 0.05 units, and the second eight form a second group whose goal is 0.025 units. After running AutoAdjust, the best fit values of the ray start U0 values appear in the ray table, as shown.

To be more realistic it is important to fill the pupil of the optic with a representative group of rays, not just the eight rays spanning a range of X0 shown here. Properly filling the pupil will help give a fair estimate of the input-output relationship for your optic, including the various aberrations that may be present.

Options: For most purposes, AutoAdjust can be run at any time you have set up your optics table or ray table appropriately. There are three optional parameters, however, that might need some fine tuning for special purposes. These are shown in Fig 20-11 below, with the help of the **Options::AutoAdjust** dialog. The first of these parameters is "Step Size" with a default value of 1E-6. It is used in taking finite-difference derivatives of your optical demerit function. This value is appropriate when the numerical values for the things being adjusted are of the order of 1: spacings, curvatures, and angles. If you are planning to autoadjust quantities of much smaller or larger magnitude you may wish to change the step size so it is of the order of 1E-6 of the things being adjusted. The second parameter is the maximum number of iterations that AutoAdjust will perform before it exits. Usually only a few iterations are needed. On large multivariate problems you might need to raise this ceiling. Of course you can abandon a run at any time to make changes in the optic, the rays, or the adjustment parameters as needed.

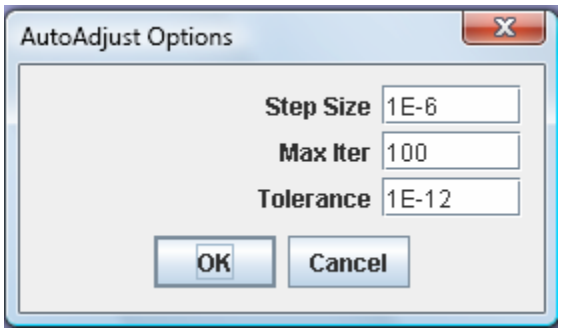


Figure 20-11: **Options:AutoAdjust** is a dialog with just three user settable parameters. Step Size fixes the numerical increments used in determining ray derivatives. MaxIter sets a ceiling on how many iterations are run automatically. Tolerance sets the smallest reduction in the computed RMS that will continue the optimization iterations.

Minimizing wavefront error (WFE) is accomplished without setting any explicit goals, since the goal is always to have zero wavefront error. If your ray table has any goal columns, delete them, or at least behead them (blank their field headers) for the WFE work. Because WFE is determined from the differences in optical path among rays that contribute to a given field point, it is important to declare how the WFE entries in your .RAY table are grouped: one group per field point. Use common tag letters at the ends of your WFE fields to show this grouping. Then, in your .OPT table, identify any adjustable parameters in the usual way: question mark tags for isolated adjustables, and tag letters for parameters that will move together as a gang. Then click **Run::AutoAdjust** to start the optimization process.

Troubleshooting AutoAdjust

- Turn off all Diameters since optimization will stop rather than lose any rays;
- Use plenty of good rays to make the problem well-defined;
- Start with just one or two of the most critical parameters adjustable;
- Do your fine tuning after a close approximate adjustment is achieved;
- Use Layout or Ray "notes" to see where a trouble spot might lie;
- Check robustness by repeating the run from several starting points.
- If **Run::AutoAdjust** is gray: goals present? adjustables present? Rays OK?

Chapter 21: The RUN Menu: AutoRay

AutoRay is a feature that sets up individual ray starts that deliver rays to specified internal goal positions. Unlike AutoAdjust, AutoRay solves for one ray at a time, holding the optical system fixed. Its usual use is to populate an internal pupil with a homogeneous pattern of rays. Because the pupil is internal, the ray start coordinates (positions X0,Y0 or directions U0,V0) have to be solved for each ray.

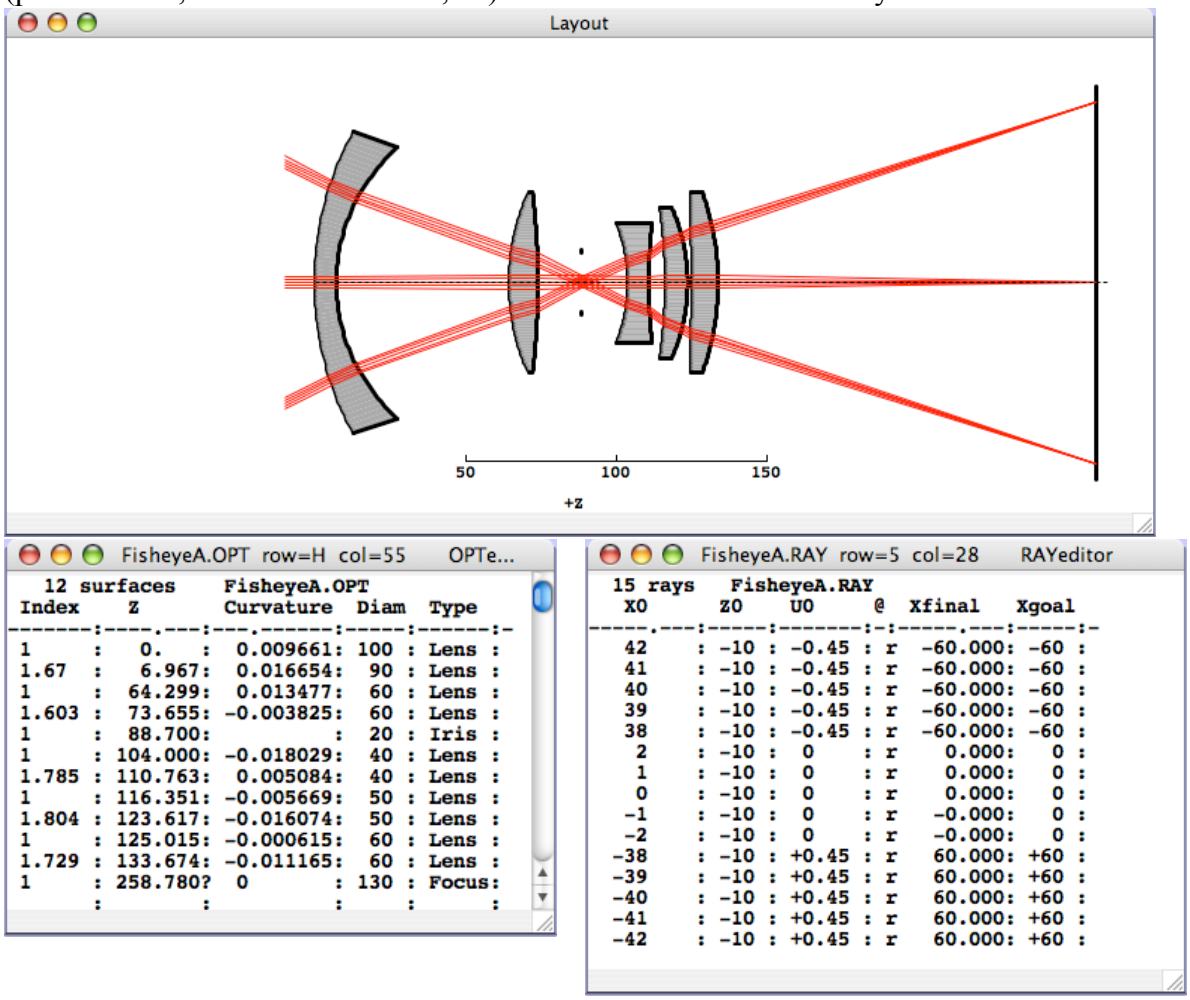


Fig. 21-1: A wide field lens with an internal pupil (the 20mm diameter iris that follows the second lens element). The initial rays fail to fill that pupil. AutoRay adjusts your ray starts to deliver the pupil coordinates that you specify, 1D or 2D.

Consider the five-element wide field lens in Fig. 21-1. Rays start outside the lens in three parallel groups. But these initial rays do not fill the internal pupil (surface 5). To assess the lens performance, we need to populate that internal pupil with a uniform distribution of rays, given by a ray generator as ray goals. How to arrange each ray start to (a) deliver its ray group direction (easy!set U0, V0) and but also (b) land at the right place on the internal pupil? This is determined by X0 or {X0,Y0} through a rather nonlinear optical calculation for each ray. That calculation is what AutoRay performs.

Setup for AutoRay is shown in Fig 21-2 below. Here, three key temporary changes have been introduced to the tables:

- In the optics table, the number of surfaces has been reduced to equal the internal pupil surface, so that ray goals will apply here, not at the eventual focal plane. You may also need to remove, or behead, Diam and diam fields to allow adjustment to proceed;
- In the ray table, the first ray in the column(s) to be adjusted is given a question mark tag, marking it (and those below it) as AutoRay adjustment targets;
- In the ray table, goal coordinates are entered, representing positions wanted for each ray when it arrives at the pupil.

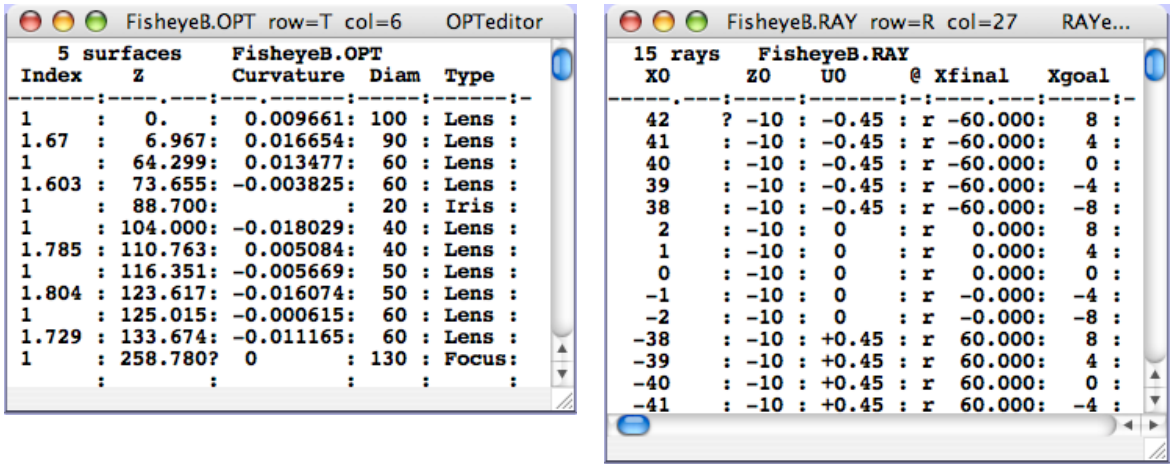


Fig. 21-2: Illustrating the changes to the .OPT and .RAY tables, set up to run AutoRay.

With this setup made, click Run::AutoRay. The ray table's X0 values will recalculate and display themselves, and the Xfinal values will correspond to the Xgoal values. Ditto for Y0, Yfinal, and Ygoal if you have done a 2D setup.

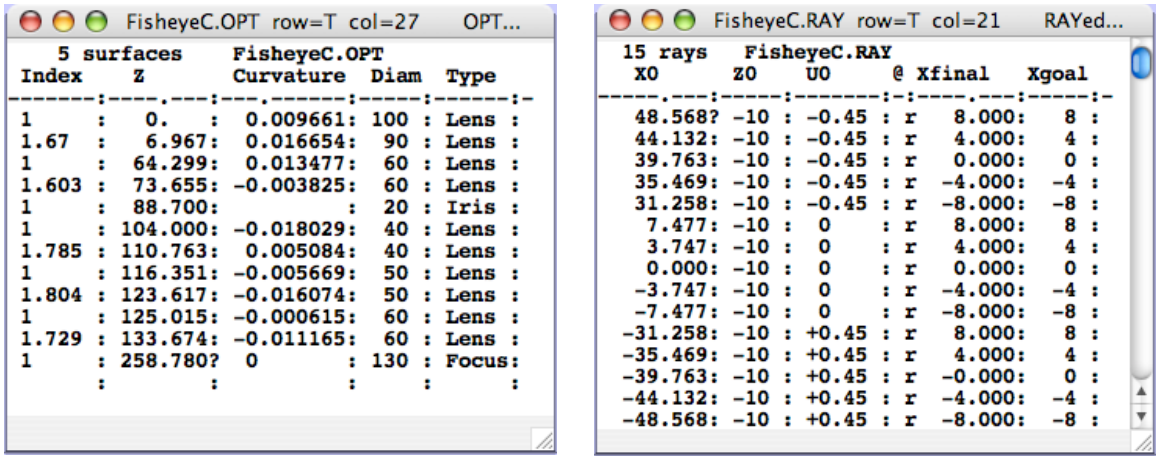


Fig. 21-3: Showing immediate results of having just run AutoRay. The new X0's bring the the surface 5 (pupil) Xfinals into agreement with the specified Xgoals.

The final step is to undo the three temporary alterations introduced for the AutoRay run.

- In the optics table, restore the surface count and Diams;
- In the ray table, replace your question mark tag(s) with colons;
- In the ray table, replace your pupil goals with your focal plane goals.

These changes are illustrated in Fig 21-4 below.

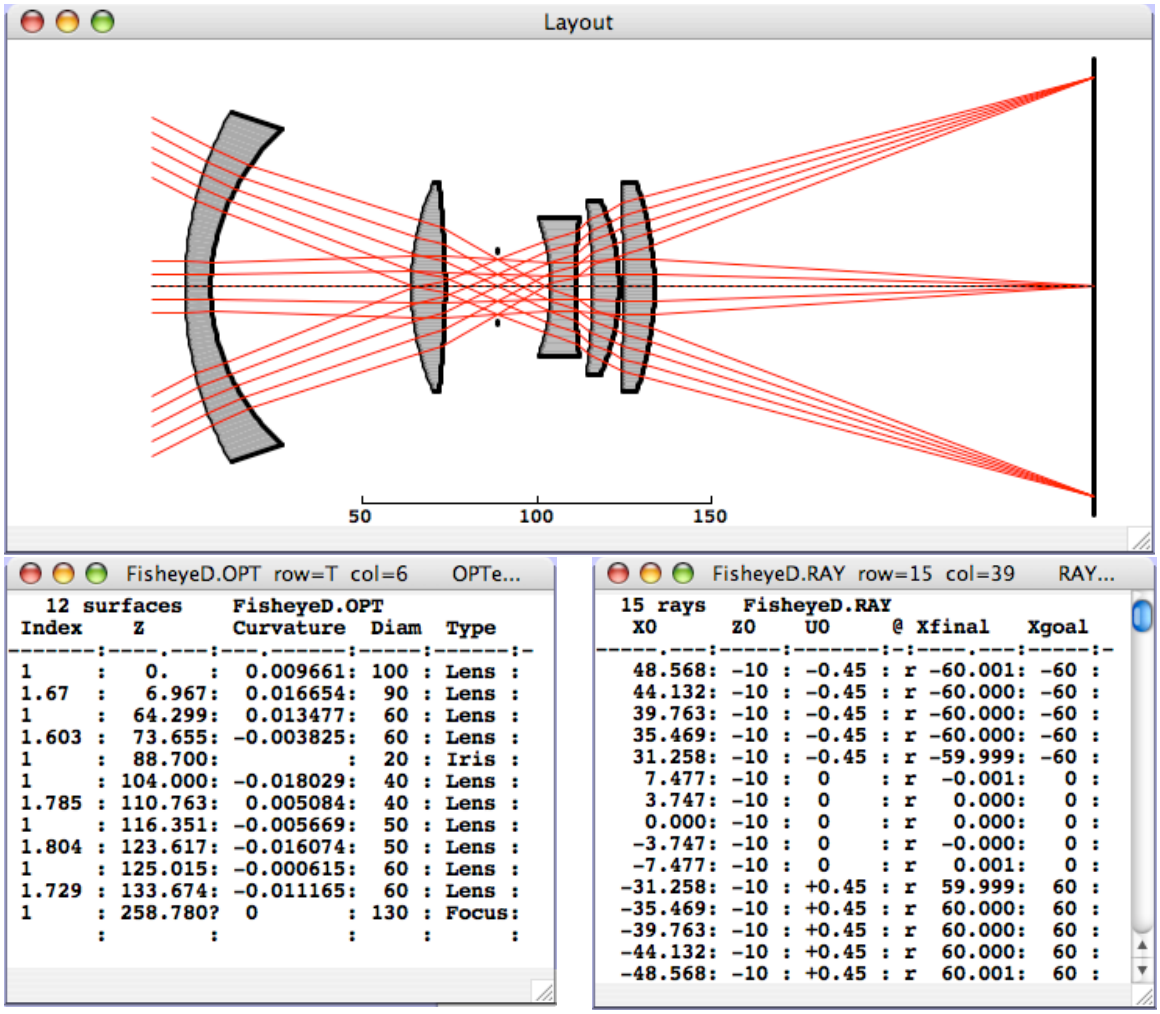


Fig 21-4 Illustrating the system performance with adjusted ray starts. Notice that the internal pupil is now uniformly filled.

The example here shows how to use AutoRay to fix up ray start positions to fill a one dimensional internal pupil. This task is appropriate for an optic focused at infinity. For an optic with a finite object distance, like a microscope, the ray start positions correspond to field locations and the pupil must be filled by setting ray start directions, not positions. AutoRay solves this problem as well: just substitute your {U0,V0} pairs as the adjustment targets, not your {X0,Y0} pairs.

Chapter 22: The RUN Menu: Random

Once you have set up an optic and an illuminating beam, you will possibly want to fill your pupil with a huge number of random rays to obtain estimates of any downstream ray variables or combinations of ray variables. The **Run::Random** menu function does exactly this function for layouts, plots, and histograms. For most on-screen graphics, the Run::Random menu item becomes ungrayed, allowing it to be clicked and begin delivering randomly generated rays into your optic.

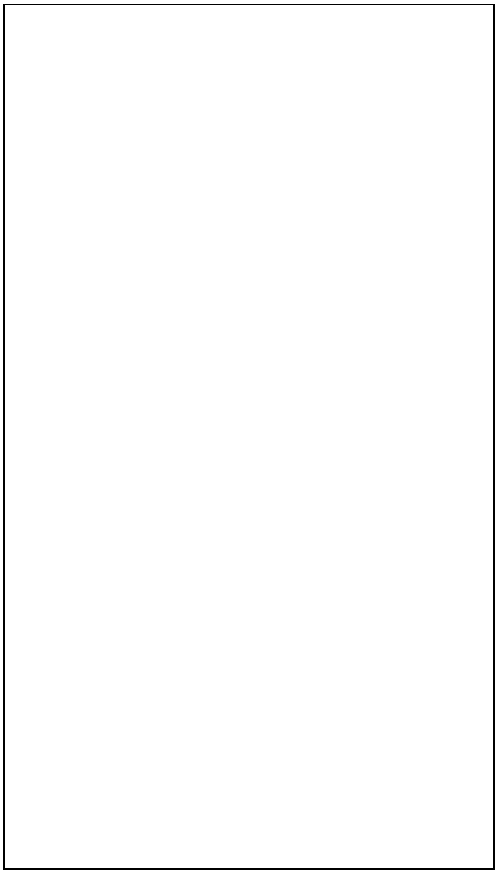


Figure 22-1: The **Options:Random** dialog box lets you set up how frequently your screen graphic will be refreshed with a stream of random rays. You may also specify when the ray generator should stop: at a given maximum number of ray starts, or a number of successful ray finishes. The random ray start locations, given by (X0,Y0,Z0) can be generated as continuous distributions from within your .RAY table's extreme values, or as a discrete triplet taken verbatim from a random choice of one of your table rays. Similarly the ray start directions given by (U0,V0,W0) can be chosen to be continuous or discrete. Five probability distributions are available: most common is the uniform distribution which gives equal probability for all ray start values within your start range. Cosine is centrally peaked, and Quartic Bell is bell-shaped and more concentrated. Gaussians and Lorentzians offer specified concentration = half width of your span divided by the 50% probability width of the distribution.

The range of the continuous random ray coordinates is controlled entirely by the range of ray starts found in your ray table. That is, the X0 value of each random ray is a uniformly distributed random number lying between the extreme values of X0 found among your current rays. Similarly for Y0, Z0, etc. If you have several wavelengths appearing in your ray table, then each random ray will be assigned a random choice from among those discrete values. If your ray table is set up to span your optic's entrance pupil, then the resulting random rays will also span this pupil. Be careful to fully define your pupils with circular or elliptical apertures if they are not rectangular, since these random ray starts will fully span a rectangular space.

The computational speed for random ray generation depends on the complexity of the optical system being traced and on the complexity of the graphic being updated as well as the processor performing the calculations. Choosing a large number of random rays per screen refresh (for example 10000 rays) makes for the quickest calculation, while a small number (for example 100 rays) leaves more opportunities for other tasks to use your computer's resources. For a simple optic such as the one or two-lens relay, and a simple display such as Plot2D, you can expect to see the order of 10,000 rays per second being traced and plotted on a typical personal computer.

Chapter 23: CAD Graphics Output

All the BEAM FOUR screen graphics functions have the ability to create precision CAD graphics files that can be exported into a variety of drawing, drafting, and illustration software. Unlike the WYSIWYG window copiers File::QuickPNG and File::Print/PDF, the CAD output files are complete images, not cropped by the current view. Using CAD, WYSIWYG is furnished by your CAD software. The file output types supported are…

- Encapsulated PostScript
- Plotter (sizes A, B, C, D, E)
- DXF for two dimensions
- DXF for three dimensions

The **Options::CAD** dialog lets you set up your CAD selection.

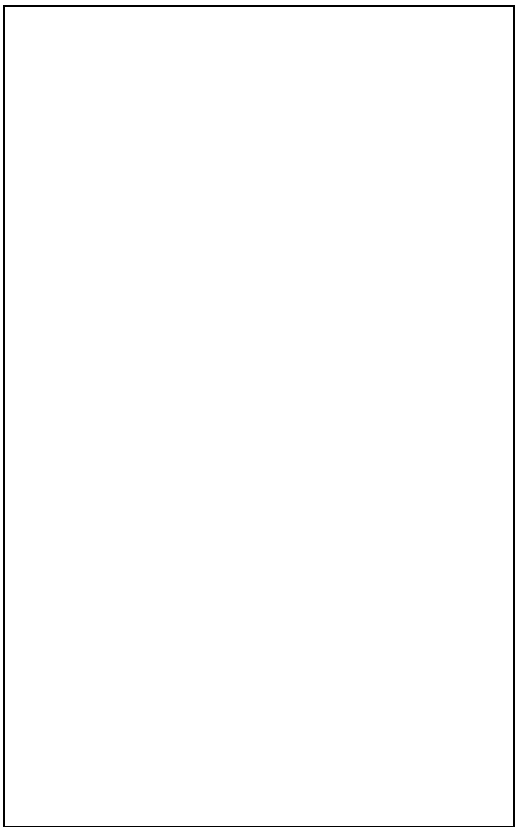


Figure 23-1: The **Options::CAD** dialog lets you specify your preferred graphics output format for use in other applications. Encapsulated Postscript (.EPS) files are best used with document preparation software which can read them directly. Plotter files (.PLT) are of course compatible with mechanical plotters but are also read by many CAD software packages, as are the .DXF file formats. Plotter files are digitized into plotter units, while DXF files carry the user coordinates and dimensions, in your choice of 2D or 3D format. DXFs are commonly used when drawing layers need to be assembled with a common dimensional plan.

The PostScript file format was developed by Adobe Systems Incorporated as a document description language that can be used by printers and by document creation software. It has become widely used because it is independent of platform and has remained highly stable over many years of use. BEAM FOUR produces encapsulated PostScript files.

They are usually given the filename extension .EPS and are easily imported into almost any kind of page composition software. When you are prompted for a file name, include the suffix .EPS to assure portability. PostScript files are readily converted to widely usable .PDF page description format files using Adobe Illustrator or Distiller. If you are using Mac OS-X you have a .PDF file writer built into your print utility and can use that feature to create .PDF files of any of your work.

Plotter files conform to the Hewlett-Packard Graphics Language (HPGL) and are most commonly used in a technical CAD environment. These files contain sequences of instructions that move one or more plotting pens. They can be plotted directly by most kinds of plotters, and are readable by most varieties of CAD software.

DXF files were developed by Autodesk Incorporated for its AutoCAD series of products and have become highly standardized throughout the drafting industry. Technically, a DXF file can contain a diverse collection of database information such as fonts, shadings, line styles, etc. What BEAM FOUR supplies is the "Entities" portion of a DXF file, which is listing of the vectors to be drawn and the 2-D or 3-D coordinates of each endpoint of each segment. This format is best used in a setting where the optical surfaces and rays are to be imported into a more complex drawing showing all aspects of an assembly (mechanical, electrical, ... as well as optical). Unlike the other graphics formats, the coordinate numbers in a DXF file are expressed in absolute user units, not page coordinates or plotter pen steps. This fact allows drawings from several sources to be precisely combined (provided of course that the coordinate systems are consistent!).

DXF viewers are commonly available from several CAD suppliers. For the PC platform, Autodesk offers the free “DWG True View” product. For Mac users there is the free viewer from eDrawings “Viewer for Mac OS X” versions 10.4 and up. Each offers ways to explore and measure 2D and 3D DXF files.

Some editions of BEAM FOUR support a "Quad list" graphics output format which is specific to this application (not an industry standard). A quad list is a text file that lists the sequence of vectors generated within BEAM FOUR in creating its 3-D vector artwork. Each vector is a "quad" containing four pieces of information: the X, Y, and Z values of a point in 3D user space, and an action descriptor for that point. Actions are starts or ends of line segments, vertices of polylines, displayed characters, color or shading information, or comments.

Finally, the two radio buttons at the bottom of the Options::CAD dialog allow you to specify your choice of orientation on the page, portrait or landscape.

Chapter 24: The Options Menu

The Options menu lets you set your preferences for the many functions that BEAM FOUR provides. Each menu item opens a dialog with controls for setting the appropriate detailed preferences that apply to each ray tracing function. Your preferences are stored every time you make a change, and your most recent set of user options will be reloaded each time you start the program. Changes in your Editors preferences are sent immediately to all editors currently on-screen and will apply to future editors as well. Changes in any specific graphics display apply to future displays and are also sent immediately to the display of that type if one is currently selected (in front of the other windows), but not to others in the background. This feature allows you to have several output display formats on screen at the same time.

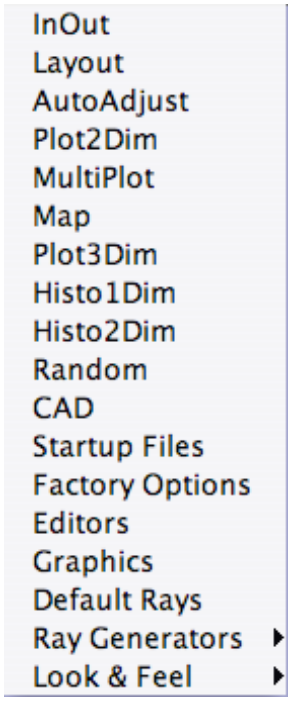


Fig. 24-1: The Options menu.

Options::InOut has just one checkbox, enabling the display of the root mean square (RMS) deviation of the ray final coordinates from their user specified goals. See Chapter 12, The Run Menu: InOut.

Options::Layout offers a very complete way to specialize the Layout artwork to best meet your needs. See Chapter 13, The Run Menu: Layout.

Options::AutoAdjust lets you set the parameter step size, maximum number of iterations, and tolerance for the auto adjustment process. Full details are shown in Chapter 20, The Run Menu: AutoAdjust.

Options:Plot 2D is the setup panel for the Plot 2D display, described in Chapter 14, The Run Menu: Plot2D.

Options:MultiPlot gives you access to the multiple plot features of BEAM FOUR. See Chapter 15, The Run Menu: MultiPlot.

Options:Map presents you with the setup panel for the Map display, described in Chapter 16.

Options::Plot 3D is the setup panel for the Plot 3D display, described in Chapter 17, The Run Menu: Plot3D.

Options::Histo 1D is the setup panel for the Histo1D display described in Chapter 18, Run Menu: Histo1D and MTF. Histo1D is also the gateway to the MTF function. If there is a Histo1D window currently on-screen and selected, **File::WriteHisto** writes its histogram as a text file of ASCII numbers in your current directory.

Options::Histo 2D is the setup panel for the Histo2D display described in Chapter 19, Run Menu: Histo2D. Like Histo1D, if a histogram is currently available, you can save its numerical contents as a text file using **File::WriteHisto**.

Options::Random lets you set the ray refresh rate and the random ray stopping point, either in terms of the number of ray starts or the number of ray finishes. The capabilities are described in Chapter 22, Run Menu: Random Rays.

Options::CAD is the setup panel for choosing the CAD output artwork format. The CAD and graphics output formats supported are briefly described in Chapter 23.

Options::Startup Files lets you specify those files that you may want to load automatically at startup: .OPT, .RAY, and/or .MED.

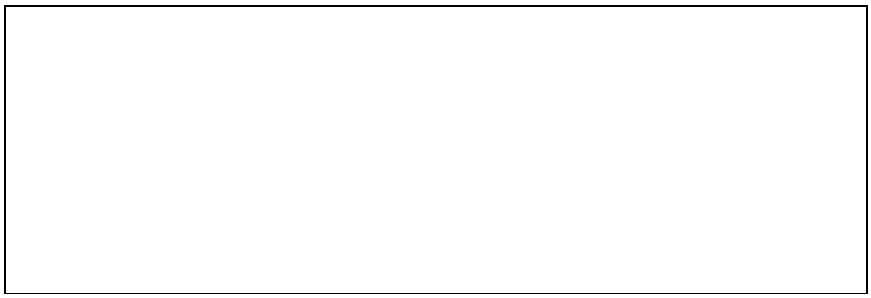


Fig 24-2: The Startup Files dialog. The “Current” button captures the names of files that are currently loaded into the editors, for quick startup next time. The center button lets you name your preferred startup files. Cancel closes the dialog without making changes.

Options::Factory lets you overwrite your accumulated Options with the initial factory settings. You might want to do this if you want to return your BEAM FOUR to its configuration as shipped, for example to diagnose some unexpected behavior.

Options::Editors allows some customization of the editors that are built into BEAM FOUR. These include choosing the information to show in the title bar of each editor, choosing font size, and display pixel smoothing. See Chapter 8 "Editors."

Options::Graphics provides a way to customize your graphics output, including font size, initial window size, and pixel smoothing. Two fonts are used: a basic Graphic font that is employed by each graphic screen in labelling axes and dimensions, and a separate font that you use when annotating a graphic after having set its pan and zoom. Indeed you can have several annotation fonts successively by changing it partway through your annotation. **Options::Graphics** is introduced in the concluding section of Chapter 13 “The Run Menu: Layout.”

Options::Default Rays lets you set your preferred ray start default directions, including the random ray generators. See Chapter 10 "Rays and Ray Tables" for examples.

Options::Ray Generators gives you access to deterministic ray generators that can supply ray start coordinate groups to your .RAY tables. These are: 1D ribbon or fan beams, 2D rectangular ray groups, or 2D circular nested groups of rays, either uniform or Gaussian. See Chapter 10 "Rays and Ray Tables" for examples. These ray generators are accurate but you must allow enough digits in their output fields to achieve high accuracy.

Options::Look and Feel offers a collection of appearance options and behavior options appropriate to your operating system. The L&Fs "Metal" and "CDE/Motif" are built into Java. Others are gathered from your computer's operating system. On MS-Windows platforms you will likely see “Windows” and “Windows Classic” offered. Newer Macintosh computers have the Mac OS X look-and-feel offered.

Chapter 25: The Help Menu

The Help menu has three entries:

Show Table Error is enabled when a table is loaded that contains a data entry that fails to parse correctly as a valid input field. This is usually due to a typographical error in the table. Select this menu item if you receive an error message such as "optics syntax error" or the like. This menu item will pop the offending table to the front, and will place the blinking caret at the beginning of the offending data field. If there are no syntax errors in the tables you have loaded, this menu item is grayed out.

Special Keys brings up a reminder dialog about the special key assignments for specific editor functions that help to manage your BEAM FOUR data tables. These are:

- F7 or Ctrl/Cmd-Left: narrow the field
- F8 or Ctrl/Cmd-Right: widen the field
- F9 or Alt-Right: split the field at the caret
- F10 or Alt-Down: copy field contents down the page.
- Ctrl/Cmd Z: undo and redo the most recent table change.

This window also reminds you how to **insert lines** into your tables while preserving your layout of data fields: copy one or more existing lines to the clipboard, then paste them back into your table.

This same window shows reminders for graphics, using a two button one-wheel mouse:

- Left button Drag: retrace & pan
- Left button Click: set caret for annotation and zoom center
- Wheel or F7/F8: retrace and zoom in/out
- Shift + Wheel or F5/F6: zoom the vertical magnification only
- Right button Drag: twirl vertically and/or horizontally.

Mac users: the Mac OS does not allow applications to access the function keys F7, F8 etc so you will need to use the Cmd and Alt key alternatives shown here.

Zoom In vs Zoom Out: the mouse wheel zooms graphics in and out. You may choose to have the wheel pull action cause a zoom in, or have a wheel push action zoom in. Go to **Options::Graphics** and select the wheel action polarity you prefer.

Run::Demo is a simple graphics demo for the mouse controls. A colored cube is displayed within a graphics window and it can be panned, zoomed, and twirled.

About gives the release and copyright information for the product and your host Java.

Chapter 26: Spreadsheets

A spreadsheet like Microsoft Excel© is an easy way to generate and analyze numerical data. It can be used to generate optics tables, ray tables, and media tables that BEAM FOUR can read, and conversely they can be used to manipulate and process the numerical output that is produced by BEAM FOUR. In this chapter we describe how to communicate from and to Excel, and show a few useful spreadsheet tasks.

Communication: There are basically two routes: clipboard and file. Clipboard transfer is quicker; file transfer gives you a more permanent record of the data transferred.

Clipboard Transfer of Spreadsheet Data Records into BEAM FOUR: In your spreadsheet, set up columns of data that mimic the format that you want your BEAM FOUR table to have. You have the full power of your spreadsheet at your disposal to create this information. The block of cells to be transferred can lie anywhere in your worksheet, but make sure that the sequence of fields generated corresponds to the entire sequence of fields that you will want in your BEAM FOUR data table starting in its leftmost field. Mark this block with your mouse as a rectangular block of cells and copy it to the clipboard using Ctrl/Cmd-C or **Edit::Copy**. Then, in BEAM FOUR, construct or adopt a table of the appropriate gender (.OPT, .RAY, or .MED) with your field widths set sufficiently wide and your field headers sequenced appropriately. Click your mouse in the table to put the blinking caret at column 1 in the topmost data record that is to receive your transfer block. This may lie anywhere below your ruler line. Hit Ctrl/Cmd-V or **Edit::Paste** to transfer the information from your clipboard into BEAM FOUR.

Clipboard Transfer of Entire Table into BEAM FOUR: Set up a worksheet to mimic your entire table layout. In cell A1, put the number of data entries that your table will contain, and in B1 put any other title block information appropriate to your table. In cells A2, B2, C2... install your data field headers. Your ruler cells A3, B3, ... can be left empty but they look nicer if each has a ruler segment in it, which you can enter by typing an apostrophe and a few hyphens ('------) into each cell. Cells A4, B4, etc are where you place your computed numerical data. Save your completed worksheet in the usual way, and then mark the table zone with your mouse and copy it to the clipboard. Then, in your BEAM FOUR destination table, put your blinking caret into the left end of the top line (the title line) and paste the clipboard contents with Ctrl/Cmd-V or **Edit::Paste**. When you transfer an entire table like this, BEAM FOUR uses the generic editor field width (set in Options:Editors:Default Fieldwidth), typically 10 characters per field. If you intend to display fields with more digits than this, you will want to boost the receiving field width accordingly before doing the transfer.

File Transfer of Entire Table into BEAM FOUR: This process goes just like the clipboard transfer, except when you have generated and saved your worksheet, save it again as a tab-delimited .txt file or as a comma-delimited .csv file. (BEAM FOUR reads and properly interprets either type of file). You may rename the file with an extension

.OPT, .RAY, or .MED extension to help you remember what it represents, or you can leave it with its native extension and read it into BEAM using **File::Open::All_Files.**

Clipboard Data Transfer into Excel®: This process is eased by the fact that Excel can by default read and properly parse clipboard text that is tab-delimited. Set up BEAM FOUR to generate tab-delimited clipboard output by going to Options:Editors and choosing Clipboard Output = tabs. Once set that way, blocks of records you mark and copy to the clipboard will be sent there with their colons or other tag characters converted to tabs, so that successive fields will be automatically distributed into successive spreadsheet cells when pasted into place within Excel.

File Transfer into Excel®: If your BEAM FOUR file uses colons as field delimiters, without other tag characters, then Excel can be set to identify and use those colons to identify the successive cell contents. In Excel, choose File:Open and select Files of Type = All Files (*.*), then select your .OPT, .RAY, or .MED file. A "text import wizard" will guide you through three steps: at step 1 choose Original Data Type = Delimited; at step 2 check the "Other" box and enter Other [:] for colon recognition; and at step 3 choose Column Data Format = General. Click [OK] and the file will transfer.

Spreadsheet Tasks: A spreadsheet is a useful adjunct in working out optical system setups (for example, from a first order paraxial description), or for generating ray starts according to a computational scheme of your own, or for interpolating refractive index data to wavelengths you need. We give a few examples here.

Parabolic interpolation: One kind of spreadsheet calculation is to interpolate a function between given data points. Usually, refraction data are tabulated at specific wavelengths defined by emission line lamps, whereas the optical designer may want refraction indices at some other wavelengths. A parabolic (quadratic) curve can be useful for this interpolation provided that the data are smoothly varying, as they will be over a limited wavelength range. Three data points determine the quadratic function. Here we abbreviate the three wavelengths X1, X2, and X3, and denote the refractive index values there as N1, N2, and N3. The interpolation function giving N as a function of any value of X lying in the interval spanned by X1..X3 is

To set up a worksheet to deliver this interpolation in a format that resembles a .MED file, put the number of glasses into cell A1 and the filename into cell B1. On your second row, put the three known wavelengths into cells B2, C2, and D2. Then put all the other wavelengths where interpolations are needed into cells E2, F2, etc. In the third row, put all blanks --- the colon line delimiters will appear when the tab delimited file is loaded. In the fourth row, enter your glass name in cell A4, and enter the known refractive indices into cells B4, C4, and D4. Then, into cell E4, enter the interpolation formula.…

+$B4*(E$2-C2)*(E$2-$D$2)/(($B$2-$C$2)*($B$2-$D$2))
+$C4*(E$2-B2)*(E$2-$D$2)/(($C$2-$B$2)*($C$2-$D$2))
+$D4*(E$2-B2)*(E$2-$C$2)/(($D$2-$B$2)*($D$2-$C$2))

...and then copy this one-cell formula into all the other column and row cells where interpolation is wanted. The spreadsheet takes care of the cell renumbering during the copying operation. To check your typing, put some interpolation wavelengths into your spreadsheet whose value is the same as a given wavelengths, and verify that the answers agree with the given refraction values. An example is shown below.

6	media		CALC.MED			
	0.4861	0.5893	0.6563	0.5000	0.5500	0.6000
K7	1.5170	1.5111	1.5085	1.5161	1.5131	1.5106
BK7	1.5224	1.5167	1.5143	1.5215	1.5186	1.5162
SK2	1.6149	1.6073	1.6041	1.6137	1.6098	1.6067
LF7	1.5875	1.5749	1.5709	1.5853	1.5788	1.5740
F4	1.6285	1.6164	1.6116	1.6265	1.6203	1.6155
SF2	1.6612	1.6475	1.6421	1.6590	1.6519	1.6465

CALC.MED MEDIA editor

```
6          media                    CALC.MED
           0.4861     0.5893     0.6563     0.5000     0.5500     0.6000
          :          :          :          :          :          :          :
K7        :1.5170    :1.5111    :1.5085    :1.5161    :1.5131    :1.5106
BK7       :1.5224    :1.5167    :1.5143    :1.5215    :1.5186    :1.5162
SK2       :1.6149    :1.6073    :1.6041    :1.6137    :1.6098    :1.6067
LF7       :1.5875    :1.5749    :1.5709    :1.5853    :1.5788    :1.5740
F4        :1.6285    :1.6164    :1.6116    :1.6265    :1.6203    :1.6155
SF2       :1.6612    :1.6475    :1.6421    :1.6590    :1.6519    :1.6465
```

Fig 26-1 Top: cells A1 through G9 of an Excel® spreadsheet prepared for six glasses. Bottom: the same file saved as type .TXT and renamed .MED, then dropped into the BEAM FOUR workspace. The tab characters in the text file are interpreted as colons on the ruler line and in the refraction data below. The rightmost colon on the ruler line serves to terminate the final field and should be supplied manually if it gets truncated.

Sellmeier interpolation: For refraction interpolation, the Sellmeier formula is commonly used throughout the optical and near infrared wavelength range. This formula has six coefficients B1, B2, B3, C1, C2, and C3, with the wavelength λ in microns:

Sellmeier coefficient values are published by optical glass manufacturers for their most popular glass formulations. A spreadsheet with this formula can yield accurate refractive index values over a wide wavelength range.

Here's how to set up a spreadsheet using a glass manufacturer's B and C values to carry out Sellmeier interpolation. First, arrange the spreadsheet to show the glasses you are interested in, with columns for B1, B2, B3, C1, C2, and C3. Here are the first few Schott glasses with the Sellmeier coefficients highlighted in blue:

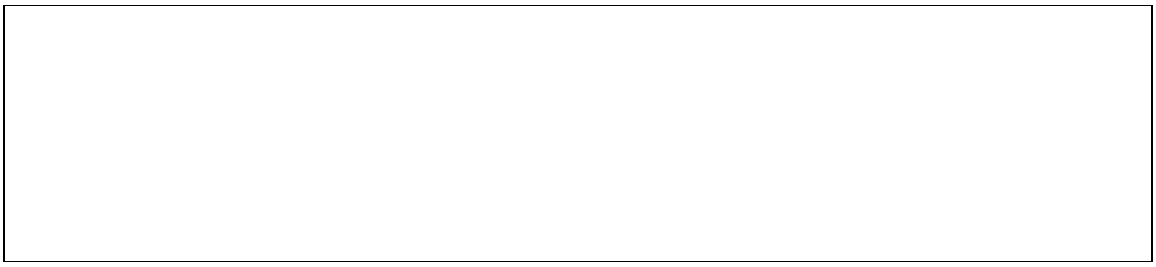


Fig 26-2: Screen snapshot of an Excel ® spreadsheet for doing Sellmeier interpolation.

We've used spreadsheet columns A and B to hold the Schott glass names and their **nd** values (their refractive indices at 0.5875618 microns). We arranged columns C through H hold the values of B1 through C3. In column I, we've started the interpolation work with a test **d** wavelength of 0.5875618 microns in cell I2, to allow us to compare with the given **nd** values in column B. Put the Sellmeier formula into cell I3…

```
+sqrt(1+I$2*I$2*$C3/(I$2*I$2-$F3)+I$2*I$2*$D3/(I$2*I$2-$G3)+I$2*I$2*$E3/(I$2*I$2-$H3))
```

This formula can be entered once, then copied down the page to interpolate all the glasses at this wavelength. Verify its agreement with column B. Then you can put a succession of other wavelengths into cells J2, K2, L2 etc, and copy the column I interpolation formulas over into J, K, L, etc. Hide columns C-H to make the results tidy. Finally, copy and paste the spreadsheet into a new .MED editor window and save it as a .MED file.

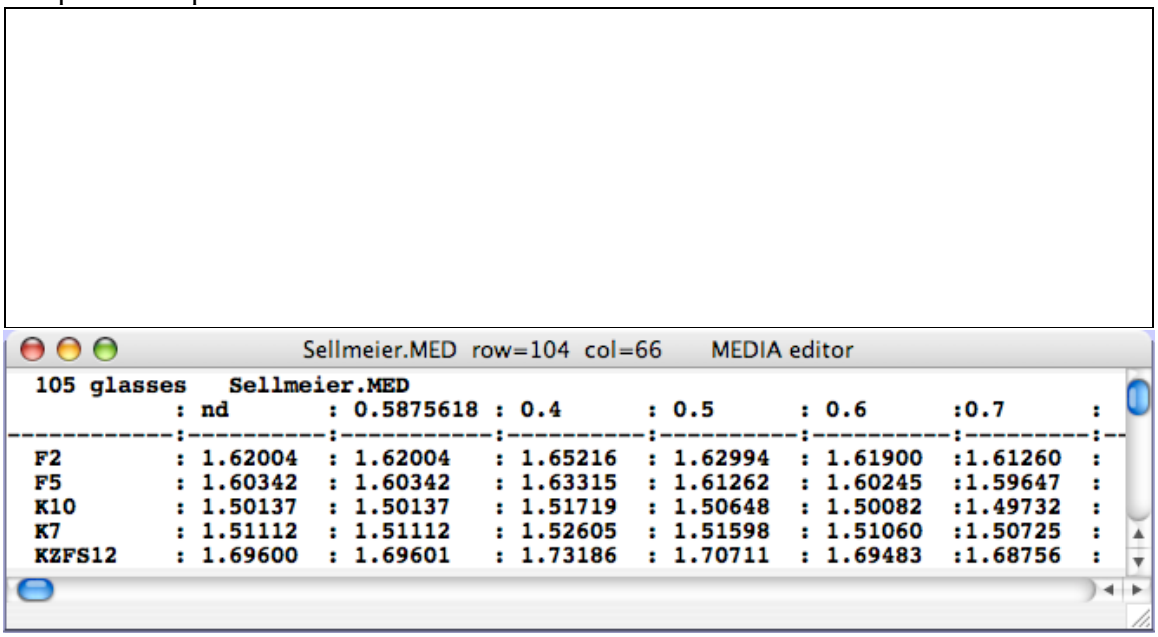


Fig 26-3 Top: spreadsheet interpolation results; bottom: MED file contents.

Format conversion: A spreadsheet is a convenient way to convert an optical system description from one tool's format to another. The changes that might be needed are unit conversions (mm, inches, meters), choice of showing refraction in the medium approaching vs. departing a surface, choice of using curvatures vs. radii, and choice of using absolute surface locations vs. spacing between surfaces. Here we show an example of a Ritchey-Chretien telescope with a two-lens corrector that demonstrates all four of these conversions.

FLATFIELD.xls

	A	B	C	D	E	F
1						
2	7 surfaces	FLATFIELD.OPT f/5 R-C=corr; 0,8,12,14mr; 22umRMS				
3	index	Z	Curve	Aspher	A4	type
4	------	---.----	---.-------	----.-------	----.------	---------
5	1	0	-0.046922	-1.1	0	mir PM
6	1	-7.46	-0.0778368	-11.6008	0	mir SM
7	1	-1.87	0.9906675	0	-0.144659	lens
8	FS	-1.85	1.5569145	0	0	lens
9	1	-1.74	0.2179004	0	0	lens
10	FS	-1.68	-0.511151	0	0	lens
11	1	-1.07	-0.0636044	0	0	sensors
12						
13						
14	FLATFIELD.ZMX					
15		Thickness				
16	Radius	preceding	Following	Conic	A4	
17	mm	mm	index	Constant		
18	-------	-------	-------	-------	-------	-------
19	-21311.96	0.00	1	-1.1	0	mir PM
20	-12847.39	-7460.00	1	-11.6008	0	mir SM
21	1009.42	5590.00	FS	0	-1.4466E-10	lens
22	642.30	20.00	1	0	0	lens
23	4589.25	110.00	FS	0	0	lens
24	-1956.37	60.00	1	0	0	lens
25	-15722.18	610.00	XX	0	0	sensors
26						
27						
28	7 surfaces		FLATFIELD.OPT			
29	approaching					
30	index	Z,m	Curve	Aspher	A4	type
31	--------	--------	-------	-------	-------	-------
32	1	0	-0.046922	-1.1	0	mir PM
33	1	-7.46000	-0.0778368	-11.6008	0	mir SM
34	1	-1.87000	0.9906675	0	-0.144659	lens
35	FS	-1.85000	1.5569145	0	0	lens
36	1	-1.74000	0.2179004	0	0	lens
37	FS	-1.68000	-0.511151	0	0	lens
38	1	-1.07000	-0.0636044	0	0	sensors

The top segment of the spreadsheet is written in B4 style: index is the refractive index in the medium approaching each surface, curvature has its usual meaning, and element locations are given in absolute meters. The middle segment converts this to an alternative format with units in mm, refraction departing each surface, and locations are determined from the thickness preceding each surface. The bottom segment converts back. Note that polynomial coefficients can have large conversion factors: going from meters to mm in A4 needs a factor of 1E-9.

Chapter 27: Sample Session

Here is a short session that illustrates a few of the features of BEAM FOUR. The files to be used will be the simple biconvex lens data stored in LENS.OPT (an optics file) and a ray file LENS.RAY. These files are supplied in the distribution package. We assume that you have installed the software as described in Chapter 4.

1. Run the BEAM FOUR program by double clicking on its icon.

2. At the top menu, select **File::OpenOptics** by clicking your mouse on the menu item "File" and pulling it down to "Open Optics." A dialog box opens showing the current set of .OPT optics files. Double-click the file LENS.OPT. It will open in an edit window. This file shows the list of optical surfaces in the sequence that light will visit them.

3. At the top menu, select **File::OpenRay**. Choose LENS.RAY. This file too will open, appearing in an edit window that shows the list of rays that are to be traced.

4. Now, most of the "Run" menu items become active, that is no longer grayed out. Select **Run::InOut** to run a ray trace and have the ray table's output fields fill in numerically with the results. If the results are useful, you can return to **File::SaveTable** to save the results of this calculation.

5. Select **Run::Layout**. A graphics window will open on the screen and a layout diagram of your lens and its rays will appear. You may adjust the zoom, pan, and orientation of this diagram using the mouse wheel, left mouse button drag, and right mouse button drag controls.

6. Turn on the random ray generator and watch your graphic fill in with random rays. Select **Run::Random**, which is available whenever a graphic window is topmost. Its job is to illuminate an optic with randomly generated rays that span the position and angle range of rays already defined within your ray table. An accompanying dialog shows the progress of the random ray production, listing the number of ray starts, the number of rays that completed their trace, and the ratio of successes to tries.

7. Now let's look at a spot diagram. First, construct a group of rays that fill the pupil of this lens: choose **Options::RayGenerators::2D Circular**. Select Coordinate Pair = (U, V). Select OuterCircleRadius = 0.1. Select 2 circles = 19 rays and click OK. This will write a group of 19 ray starts into your ray table, ready to run. Then ch**:Plot2D** and produce a spot diagram.

Chapter 28: Sample Files

Optics and Ray Files: A number of sample files are provided with each shipment of BEAM FOUR. Many of these illustrate one or more features of the software and are the examples shown in this Guide. Some of these could be useful starting points for your own work.

360.OPT and 360.RAY
ANGLES.OPT
ACHRO.OPT and ACHRO.RAY
BICONIC.OPT and BICONIC.RAY
CASS.OPT and CASS.RAY
CYLINDER.OPT and CYLINDER.RAY
CZERNY.OPT and CZERNY.RAY
DISK.OPT and DISK.RAY
DROPLET.OPT and DROPLET.RAY
ELLIPSE.OPT and ELLIPSE.RAY
ERFLE.OPT and ERFLE.RAY
FISHEYE.OPT and FISHEYE.RAY
FZP.OPT and FZP.RAY
GRAZE.OPT and GRAZE.RAY
HOE.OPT and HOE.RAY
HUBBLE.OPT and HUBBLE.RAY
HYPER.OPT and HYPER.RAY
LENS.OPT and LENS.RAY
NEWTON.OPT and NEWTON.RAY
PRISM.OPT and PRISM.RAY
ROWLAND.OPT and ROWLAND.RAY
SPIDER.OPT and SPIDER.RAY
TORIC.OPT and TORIC.RAY
XYZFIT.OPT and XYZFIT.RAY

Media: Four media tables are furnished for use at visible, NIR, and NUV wavelengths:

GLASS.MED with 8 glasses and 3 plastic materials;
SCHOTT.MED with 105 glasses (see www.us.schott.com);
OHARA.MED with 120 glasses (see www.oharacorp.com);
HOYA.MED with 94 glasses (see www.hoyaoptics.com).

You of course may augment these tables, or build your own, based on optical media that you use and emphasizing the wavelength regions of interest in your work.

Null Test Files: Testing the accuracy and robustness of your software is important, and it is valuable to run a series of test calculations whose answers are known in advance. Because accuracy can depend significantly on surface type and angle of incidence (near normal vs. grazing incidence), test cases should explore these variables. We supply a few test files called "null tests" for which each ray's final coordinate is zero if the calculation is performed properly. These files “xxxNULL” are described below and come with each distribution.

CASSNULL.OPT and CASSNULL.RAY
-- This is a classical Cassegrain telescope comprising a concave parabolic (conic) primary mirror and a convex hyperbolic secondary mirror, ideally aligned and focussed. In parallel incoming light a perfect focus should be formed.

GRAZE1NULL.OPT and GRAZE1NULL.RAY
-- A grazing incidence telescope with both conjugates at infinity is formed by two confocal parabolic reflectors at grazing incidence. The paraboloids are implemented using conic sections having asphericity = -1. For incoming rays parallel to the axis, outgoing rays should also be parallel to the axis.

GRAZE2NULL.OPT and GRAZE2NULL.RAY
-- Similar to GRAZE1NULL except that this test sets up a finite image conjugate which necessitates a hyperbolic secondary mirror implemented as a conic, and for variety we implement the primary mirror using a polynomial.

GREGNULL.OPT and GREGNULL.RAY
-- This is a classical Gregorian telescope comprising a concave parabolic (conic) primary mirror and a concave ellipsoidal secondary mirror, ideally aligned and focussed. In parallel incoming light a perfect focus should be formed.

HYPER1NULL.OPT and HYPER1.RAY
-- This is a plano-convex lens with a hyperbolic surface profile chosen with regard to the refractive index of the lens material to form an ideal point image for incoming rays that are parallel to the lens axis. To satisfy this condition, the hyperboloid must point toward the finite conjugate and the asphericity must equal the negative of the square of the refractive index.

HYPER2NULL.OPT and HYPER2.RAY
-- This file represents a biconvex lens with both surfaces having hyperbolic profiles chosen with regard to the refractive index of the lens material. An ideal focus should be formed, given an ideal point source of illumination.

PARABNULL.OPT and PARABNULL.RAY
-- Here a concave parabolic (conic) reflector provides a perfect image point focus for illuminating rays that are parallel to the paraboloid axis.

POLY1NULL.OPT and POLY1NULL.RAY
-- This file explores a simple parabolic mirror implemented -- not as the usual conic section -- but rather as a polynomial surface of type A2. Polynomials are more general than conic sections of revolution and employ different internal mathematical components, and therefore need their own tests. Parallel incoming rays should form an ideal focus.

POLY2NULL.OPT and POLY2NULL.RAY
-- A complicated way to make a concave spherical reflector is to construct a concave surface out of polynomial terms to 14th degree, as is done here. Because this is finite, the accuracy of the approximation is limited. For an extreme edge ray in an f/1 system, the expected transverse aberration should be a computable fraction of the focal length.

SPHERENULL.OPT and SPHERENULL.RAY
- An ideal concave sphere should return all origin rays to its origin. Here the sphere is implemented the easy way, as a conic section of revolution with asphericity = 0.

TORIC1NULL.OPT and TORIC1NULL.RAY
-- A concave spherical mirror is implemented using an equal radius toric surface. Light originates at the origin of coordinates and should ideally return there upon being reflected by this special toric, exactly as for a sphere. The difference here is the internal mathematical components (interceptor, solver, and gradient computation) have a higher level of generality than the more elementary conic section solvers, and are therefore tested separately. This file makes a sphere the hard way.

TORIC2NULL.OPT and TORIC2NULL.RAY
-- Here a concave parabolic (conic type) toroidal reflector is illuminated by "wheel spoke" rays that radiate in a plane offset from, but parallel to, the principal axis. The parabolic conic surface should return all rays to the origin.

TORIC3NULL.OPT and TORIC3NULL.RAY
-- This is a concave toroidal reflector similar to TORIC2NULL except that its parabolic section is a special polynomial case, A2, rather than a special conic case with asphericity = -1.

ZERNNULL.OPT and ZERNNULL.RAY
-- a Newtonian telescope implemented using a Zernike surface of type Zern3 as its parabolic primary mirror. Zern3 is actually the sum of a paraboloid and a negative piston (see Appendix 3). The Newtonian telescope's tilted flat mirror is implemented by a Zernike mirror surface of type Zern1, pure tilt. This test is unusual because Zernike polynomials are commonly employed only as small deviations from nominally ideal optical surfaces. An ideal focus should appear to one side of the principal telescope axis.

Chapter 29: Viewing Stereoscopic Displays

3-D stereo display option is available for Layout, Plot3D and Histo2D functions. The stereo effect is produced by generating two views of a 3D line drawing. One view uses red on a black background, and the other uses blue on a black background. The two views are slightly separated in left-right viewing angle. Viewed through red/blue eyeglasses, the two scenes appear to merge into one 3D scene.

Parallax The parameter that controls the apparent angular separation of the two images is parallax, and can be set numerically in the Options dialog for each graphic. Parallax values that work best are those that provide just a few pixels separation between the extremes of the red and blue screen images. Start with a value of 5, and adjust zoom & twirl to determine the best parallax setting for your graphic. Traditionally the red filter is for the left eye, blue on the right; if your viewer is reversed, use a negative sign for your preferred parallax.

Viewers For best effect it is important that the red viewing filter have very little transmission of blue light (this is nearly always true) and the blue filter have very little transmission of red light (this can be a problem). The common red/blue lens combination works well; red/cyan suffers from added red leakage through the cyan filter, and red/green is poorer still at rejecting red light through its green filter.

The best viewers we've found is the Berezin #543 clip-ons and the Berezin 542B plastic frame eyeglasses. Inexpensive cardboard frame glasses with colored cellophane filters can also work. Try a web search for "3D glasses" to see what is available. Some vendors offer one pair of cardboard/cellophane glasses for just the price of postage. Your local video game store probably sells or gives away viewers too. To check the compatibility of your display screen colors and your viewing glasses, set up a stereo B4JE display, then close one eye, then the other. Each view should be single, not double.

Darkness Another factor that improves the 3D effect is your local lighting condition. Dark ambient lighting and a huge display in full-screen mode both help the 3D illusion.

Density When many rays bunch together, the red and blue artwork elements merge into a featureless magenta patch from which no 3D illusion is possible. 3D viewing works best when the line density on the screen is low enough that individual features are distinct.

Speed Finally note that 3D images take more than twice the computer time to render, so the machine speed is a stronger consideration than with the usual 2D screen images.

Chapter 30: License and Warranty

SINGLE USER LICENSE AGREEMENT: Stellar Software and its suppliers grant to Customer a nonexclusive and nontransferable license to use the Software and its Documentation. Customer may make any number of archival copies of the Software and Documentation for his/her own use and for backup purposes. The program may be used on only one computer or terminal at a time. Corporate owners may assign the software to any one employee, during which assignment the software may not be used by any other employee under this single user license.

BEAM FOUR is a proprietary product of Stellar Software. It is protected by international copyright law. Except as expressly authorized above, the customer shall not: modify the software; reverse compile or reverse assemble any portion of the software; rent, lease, distribute, sell, or create derivative works of the software or documentation, except by prior written agreement with Stellar Software. Rights of sale and distribution are retained by Stellar Software.

STELLAR SOFTWARE offers this software and documentation on an "as is" basis. A defective disk will be replaced by Stellar Software within 90 days of purchase, when accompanied by a proof of purchase. No other warranty of any kind, expressed or implied, is offered. In particular, Stellar Software does not warrant that the information distributed in the software or documentation will necessarily be suited to the specific needs of the purchaser. Stellar Software does not assume liability for incidental or consequential damages arising from the use of the product, even if Stellar Software has been advised of the possibility of said damages.

Appendix 1: Conic Sections

Conic sections are mathematical curves (parabolas, hyperbolas, circles etc.) that satisfy quadratic algebraic expressions. Geometrically they are equivalent to the intersection of a cone with a plane, hence the name. When a conic section is rotated about an axis, it sweeps out a surface in three dimensions (paraboloid, hyperboloid, sphere or ellipsoid). Surfaces of this type are very useful in optics. The parameters that represent conic sections depend on the coordinate system chosen. The table below can be used to convert a conic section description in one coordinate system to another. BEAM FOUR, in common with the optics industry, uses the vertex-origin Cartesian system.

	Center-origin Cartesian Coords	**Vertex-origin Cartesian Coords**	**Focus-origin Polar Coords**
Equation:			
Semimajor z axis:			
Semiminor x axis:			
Asphericity Asph:	$(B/A) - 1$	$S - 1$	
Shape S:	B/A	$Asph+1$	
Eccentricity e:			e
Curvature C = 1/radius		C	
Locations of foci:			$r = 0, \quad r = 2ae$
Distance between foci:			$2ae$
Oblate ellipsoid:	$B>A>0$	$S>1$, Asph>0	fails
Sphere:	$B=A>0$	$S=1$, Asph=0	$e=0$
Prolate ellipsoid:	$A>B>0$	$0<S<1$, -1<Asph<0	$0<e<1$
Paraboloid:	fails	$S=0$, Asph= -1	$e=1$
Hyperboloid:	$A<0<B$	$S<0$, Asph< -1	$e>1$

Beware of the term "conic constant" and/or the symbol K, k or κ. These descriptors are not well standardized. Some authors (e.g. Fischer & Tadic-Galeb) use them to mean asphericity (i.e.departure from a sphere). Others (e.g. Spencer & Murty; Wetherell & Rimmer) use κ to describe departure from a paraboloid. W.J.Smith (p.445) uses “p” for departure from a paraboloid. In BEAM FOUR, we use the descriptor "Asphericity" (abbreviated "Asph") to indicate departure from a sphere. We use "Shape" (abbreviated "S") to indicate departure from a paraboloid, so that S = Asph+1. Use either Shape or Asphericity, but not both, in your optics tables.

Ray Intercepts with Conic Sections: Because the conic sections of revolution are based on quadratic functions, there can be zero, one, or two intercepts for each ray. These possibilities are processed as follows:

- No real roots: this ray segment is declared to be a "mis."
- 1 or 2 roots: tested for primary hemisurface, i.e. the hemisurface whose vertex lies at z=0. If no survivors, this ray segment is also declared to be a "mis."
- 1 or 2 survivors: tested for positive ray length: rays cannot propagate backwards. If there are no survivors at this step, the ray segment is declared to be a "bak."
- 1 or 2 survivors: tested for intercept inner diameter. If no survivors: "dia."
- 1 or 2 survivors: tested for intercept outer Diameter. If no survivors: "Dia."
- 1 survivor: it is declared to be "OK." The trace proceeds.
- 2 survivors: the one with the shorter ray path is chosen by default. A tag character on the "mirror" field modifies this default. Use ">" for the longer path; or use "<" to get the root that lies closer to the vertex.

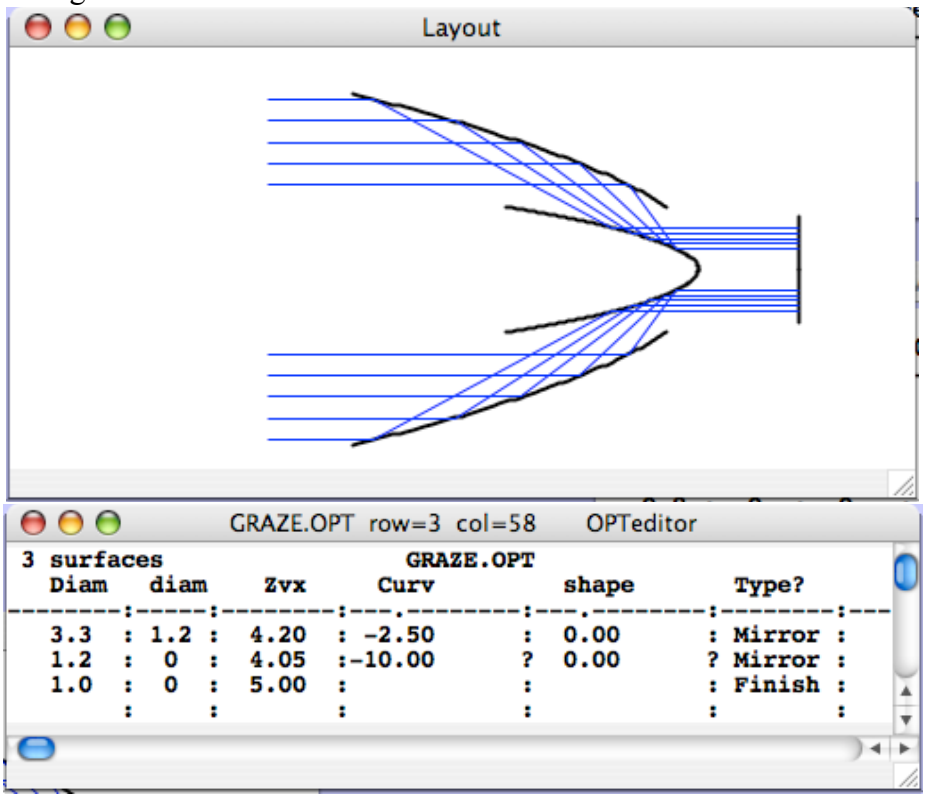


Fig A-1 Illustrating a pair of confocal paraboloids in GRAZE.OPT. The intercept desired at the secondary mirror is the one with the shorter ray length, and this default intercept is permitted by the secondary Diameter. Reduce that to 0.2 and you will see that only the longer ray path survives. Those rays can be enabled if you move the final Zvx to 0.0 or thereabouts to let these rays complete their journeys.

Appendix 2: Rotation Sequences

The orientation of an object in three dimensions can be described in a variety of ways. One such description as follows: take an arbitrary point P that is not the origin, and write its coordinates (x,y,z) in a frame of reference fixed in the object. Also, write its coordinates (X,Y,Z) in a lab frame of coordinates having the same center. The matrix M that converts (x,y,z) into (X,Y,Z) is uniquely defined by the orientation of the object's coordinate frame. Explicitly:

This description is a set of nine numbers, namely the matrix elements M_{ij}. Each matrix element has a numerical value between -1 and +1. Each represents the projection or dot product of a unit local coordinate vector (x, or y, or z) onto a unit lab coordinate vector. They are not independent: the sums of the squares along any column or row is exactly 1.0 because the length of the (x,y,z) vector and the (X,Y,Z) vector have to come out equal for any object vector. A matrix that satisfies these six constraints is said to be unitary. Nine numbers and six constraints mean that there are just three degrees of freedom for the three dimensional orientation.

Orientations can also be described using angles. For any rotation, there exists some direction in space (two parameters) and one angle around that axis that gives the rotation. In Cartesian coordinates it is usually more convenient to break this one eigenaxis rotation into three separate consecutive rotations about the original or partly rotated axes.

How many such descriptions are there? The first motion can be taken about the lab X, Y, or Z axes -- three possibilities. Whichever is chosen, the second rotation cannot be around that axis again, but now there are two new axes to choose from, plus two unused of the original three axes. If the second angle is one of the original three, it creates three new axes for a total of eight, seven of which are possible third axes, giving 42 possible descriptions. If the second angle is one of the two new axes produced by the first angle, it creates two new axes for a total of seven, six of which are possible third angle axes, giving 36 more descriptions. In all there are 78 descriptions.

THE (X, y', z") ROTATION DESCRIPTION

In BEAM products, and in much of the optics industry, we use the rotation-sequence described by successive rotations about the X, y', and z" axes. This is just one way of doing business, but it has the advantage that each coordinate appears once so that if you have a simple motion about just one of X, Y, or Z, there is a one-angle description of it,

and it is about the obvious axis. Another advantage of (X,y',z") over say (X,Y,Z) is that this is the way that actual mechanical goniometers work: each goniometer cradle carries the base of the next goniometer, so that the first motion is about a fixed lab coordinate direction -- the second motion is about an axis that depends on the first motion, and the third motion axis depends on both the preceding motions. In all, 18 of the rotation sequences are of the form (A, B', C") that can be created with actual goniometers. Of these, (X, y', z") is the easiest to remember.

Define these three angles in a positive right-handed sense:

Tilt	*t*	-- about lab +X axis
Pitch	*p*	-- about tilted +y axis
Roll	*r*	-- about tilted and pitched +z axis.

Abbreviate: $ct = cos(t)$
$st = sin(t)$
$cp = cos(p)$
$sp = sin(p)$
$cr = cos(r)$
$sr = sin(r)$

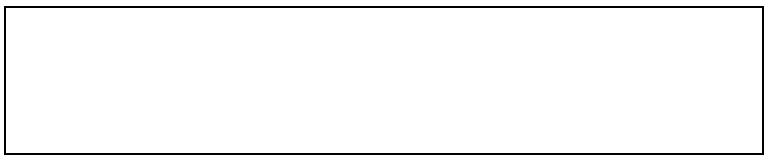
(Roll is needed for the general case but is not significant for axisymmetric surfaces.)

Then, any vector *(x, y, z)* seen in the frame of the rotated coordinates can be converted to its lab frame representation by the linear unitary matrix operator M:

For the case of a flat surface whose orientation is defined by the tilt-pitch-roll coordinates, its normal is the unit +z axis in its local (rotated) frame. In lab coordinates this +z axis, and hence the normal, is just the rightmost column of the matrix M:

$$X = sin(p)$$
$$Y = -cos(p) \cdot sin(t)$$
$$Z = cos(p) \cdot cos(t)$$

The inverse of any unitary matrix M is its *transpose*: its elements down the main diagonal (upper left to lower right) are the same, but the off diagonal elements get swapped across the main diagonal. The conversion of a lab frame vector into that vector expressed in the rotated coordinates is done with the transpose of M, usually abbreviated M^T:



THE (X,Y,Z) ROTATION DESCRIPTION

Another of the many three-angle descriptions is the (X,Y,Z) sequence: the first rotation is identical to tilt (right handed about the lab X axis), the second rotation is about the lab Y axis even though the optic is no longer aligned with the lab frame, and the third rotation is about the lab Z axis, ditto.

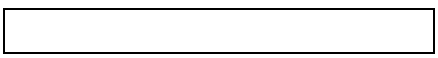
Call these angles *ax, ay,* and *az*, and abbreviate

cx = cos(ax)
sx = sin(ax)
cy = cos(ay)
sy = sin(ay)
cz = cos(az)
sz = sin(az)

Then, any point (*x, y, z*) in the frame of the rotated coordinates can be converted to its lab frame representation by the linear operator M(*Z*) * M(*Y*) * M(*X*) because successive motions in the lab frame multiply onto the left hand side (the lab side) of the building matrix. This product is:

There is a set of angles (*ax, ay, az*) that describes the same orientation as the set of angles (*tilt, pitch, roll*). Of course the angles are different, but the orientations are the same. This fact gives us a way to convert angles in one scheme to or from angles in another. We set up the rotation matrix for each scheme. All nine of the matrix elements are equal because the orientations are the same. So, solve either way for the set of angles. This plan converts any rotation description to any other.

Here's an example connecting (*ax, ay, az*) to (*tilt, pitch, roll*). M_{13} is sin(*pitch*) in the tilt-pitch-roll system. But M_{13} is also given by cos(*ax*) · sin(*ay*) · cos(*az*) + sin(*ax*) · sin(*az*). These must be equal. So, we have a formula for pitch, in terms of the three given angles ax, ay, az:



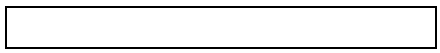
Similarly, note that the ratio of the sixth matrix element to the ninth matrix element is equal to -tan(tilt). So:

Similarly, the ratio of the second matrix element to the first matrix element is -tan(roll):

By shopping for the simplest expression of variables you need in one matrix, and using the equality of matrix elements, you can write down any needed conversion formula.

This works in both directions. For example let's find the angles (*ax, ay, az*) that correspond to a given trio of (*tilt, pitch, roll*). First, *ax* can be obtained by noting that tan(*ax*) is the ratio of the eighth to ninth matrix element:

The term sin(*ay*) appears by itself as the seventh matrix element, so



Finally, the tangent of *az* is the ratio of the fourth matrix element of M to its first element:

THE (Y,X,Z) ROTATION DESCRIPTION

Yet another of the specifications is the (Y,X,Z) system. The transformation to convert coordinates in this sequence is M = R(Z) * R(X) * R(Y) and it works out to:

Again, useful formulas going either way can be found by identifying corresponding parts of this matrix and the (*t,p,r)* matrix that performs the same transformation, from rotated to lab coordinates.

THE (Z, x', z") ROTATION DESCRIPTION

This is Euler's original set of angles and is widely used in dynamics although not much used in optics. Adopt the angle names *a* = roll about Z, *i* = pitch about x', and *p* = roll about z". Adopt the abbreviations

ca = cos(a)
sa = sin(a)
ci = cos(i)
si = sin(i)
cp = cos(p)
sp = sin(p)

Then the transformation M = R(Z) * R(x') * R(z") works out to:

Yet again, useful formulas going either way can be found by identifying corresponding parts of this matrix and the (*t,p,r)* matrix that performs the same transformation, from rotated to lab coordinates.

Appendix 3: Zernike Polynomials

BEAM FOUR supports the first 36 Zernike polynomials. Be careful with numbering: the "n" and "m" numbering system is from Born & Wolf while the index and descriptors are from J.C.Wyant's website, and from R.N.Wilson 2000, and from other sources, but is not well standardized. The appropriate BEAM FOUR field headers are Zern0 … Zern35. Always specify a Diameter when using Zernikes.

Zernike Polynomials

Born & Wolf numbering: n=radial degree; m=angular frequency

Index	n	m	Polynomial	Descriptor
0	0	0	1	Piston
1	1	1	$r\cos(\theta)$	x tilt, 0°
2	1	1	$r\sin(\theta)$	y tilt, 90°
3	2	0	$2r^2 - 1$	defocus
4	2	2	$r^2\cos(2\theta)$	astig 3rd order, 0°
5	2	2	$r^2\sin(2\theta)$	astig 3rd order, 45°
6	3	1	$(3r^3 - 2r)\cos(\theta)$	coma 3rd order, x, 0°
7	3	1	$(3r^3 - 2r)\sin(\theta)$	coma 3rd order, y, 90°
8	4	0	$6r^4 - 6r^2 + 1$	spherical 3rd order
9	3	3	$r^3\cos(3\theta)$	trefoil 5th order, 0°
10	3	3	$r^3\sin(3\theta)$	trefoil 5th order, 30°
11	4	2	$(4r^4 - 3r^2)\cos(2\theta)$	astig 5th order, 0°
12	4	2	$(4r^4 - 3r^2)\sin(2\theta)$	astig 5th order, 45°
13	5	1	$(10r^5 - 12r^3 + 3r)\cos(\theta)$	coma 5th order, x, 0°
14	5	1	$(10r^5 - 12r^3 + 3r)\sin(\theta)$	coma 5th order, y, 90°
15	6	0	$20r^6 - 30r^4 + 12r^2 - 1$	spherical 5th order
16	4	4	$r^4\cos(4\theta)$	tetrafoil 7th order, 0°
17	4	4	$r^4\sin(4\theta)$	tetrafoil 7th order, 22.5°
18	5	3	$(5r^5 - 4r^3)\cos(3\theta)$	trefoil 7th; 0°
19	5	3	$(5r^5 - 4r^3)\sin(3\theta)$	trefoil 7th; 30°
20	6	2	$(15r^6 - 20r^4 + 6r^2)\cos(2\theta)$	astig 7th order; 0°
21	6	2	$(15r^6 - 20r^4 + 6r^2)\sin(2\theta)$	astig 7th order; 45°
22	7	1	$(35r^7 - 60r^5 + 30r^3 - 4r)\cos(\theta)$	coma 7th order, x, 0°
23	7	1	$(35r^7 - 60r^5 + 30r^3 - 4r)\sin(\theta)$	coma 7th order, y, 90°
24	8	0	$70r^8 - 140r^6 + 90r^4 - 20r^2 + 1$	spherical 7th order
25	5	5	$r^5\cos(5\theta)$	pentafoil 9th order, 0°
26	5	5	$r^5\sin(5\theta)$	pentafoil 9th order, 18°
27	6	4	$(6r^6 - 5r^4)\cos(4\theta)$	tetrafoil 9th order, 0°
28	6	4	$(6r^6 - 5r^4)\sin(4\theta)$	tetrafoil 9th order, 22.5°
29	7	3	$(21r^7 - 30r^5 + 10r^3)\cos(3\theta)$	trefoil 9th order, 0°
30	7	3	$(21r^7 - 30r^5 + 10r^3)\sin(3\theta)$	trefoil 9th order, 30°
31	8	2	$(56r^8 - 105r^6 + 60r^4 - 10r^2)\cos(2\theta)$	astig 9th order, 0°
32	8	2	$(56r^8 - 105r^6 + 60\,r^4 - 10r^2)\sin(2\theta)$	astig 9th order, 45°
33	9	1	$(126r^9 - 280r^7 + 210r^5 - 60r^3 + 5r)\cos(\theta)$	coma 9th order, x, 0°
34	9	1	$(126r^9 - 280r^7 + 210r^5 - 60r^3 + 5r)\sin(\theta)$	coma 9th order, y, 90°
35	10	0	$252r^{10} - 630r^8 + 560r^6 - 210r^4 + 30r^2 - 1$	spherical 9th order

Appendix 4: Diffraction Gratings

The classical diffraction grating is a surface of arbitrary curvature carrying grooves defined by a stack of equally spaced parallel planes that intercept the surface. We use Gx and Gy to describe the number grooves per unit distance in x or y (local surface coordinate system), with the unit of length consistent with the unit of length in which the ray wavelength is measured. For example, if your wavelengths are expressed in microns, then express your groove densities in grooves per micron.

A diffraction calculation requires that the order of diffraction be specified. Order can be specified in either of two ways. In the **optics** table, a column headed "order" will determine the order of diffraction for that particular diffractor. If there is more than one diffractor in your optic, this is the way to define it. Or, in the **ray** table, a column headed "order" will specify the order of diffraction for each ray. This way, rays can be grouped by their orders of diffraction. The ray order method works only if there is just a single diffractor in the optics table.

If the grooves are associated with a **mirror** surface (any curvature, shape, etc) the resulting grating will be a reflection grating. Its type will be "mirror" and it will be distinguished from an ordinary grooveless mirror by having a nonzero Gx or Gy or both. Alternatively, if the grooves are on a **lens** surface (any curvature, shape, etc) the resulting grating will be a transmission grating. Its type will be "lens" and again it will be distinguished from an ordinary lens by having a nonzero Gx or Gy or both, and a nonzero Order. Of course any defined change in refractive index across the surface will also modify the ray direction, as it would on a diffractive lens surface.

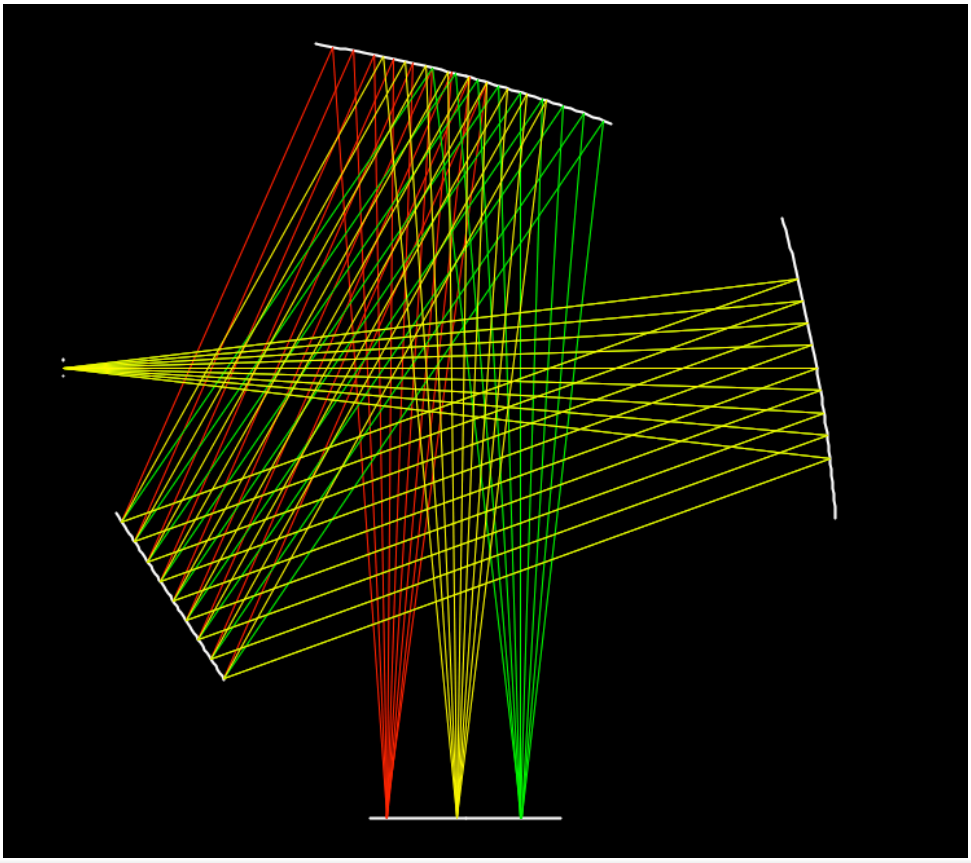
If the grating has grooves that are all parallel but the groove density varies smoothly across the face of the grating, it is termed a *varied line space* grating. Such gratings are commonly used to correct aberrations in spectrometers. The variation of groove density is described by a polynomial in x if the groove density Gx is nonzero and Gy is zero:

$$\text{density} = Gx + x \cdot VLS1 + x^2 \cdot VLS2 + x^3 \cdot VLS3 + x^4 \cdot VLS4$$

A similar expression applies if the groove density Gy is nonzero and Gx is zero. The polynomial coefficients VLS1 ... VLS4 are introduced into any optics table using column headers with these names “VLS1”, etc.

The intensities of the various diffracted orders are not computed by BEAM FOUR. For grooved surface relief gratings, the groove profile controls these intensities. For volume phase holographic gratings, the intensities are governed by the depth of the volume (normally a few microns), by the amplitude of the recorded refraction wave, and by the Bragg angle. Rigorous coupled-wave analysis can be applied to simultaneously solve for the amplitudes, phases, and polarizations of the diffracted beams.

Example: A crossed Czerny-Turner spectrometer uses a collimator mirror to illuminate a plane reflective diffraction grating, and uses a separate camera mirror to focus the diffracted spectrum. Figure A4-1 below shows a layout of this configuration. The files CZERNY.OPT and CZERNY.RAY are provided with this distribution.



CZERNY.OPT row=4 col=42 OPTICS editor

```
 5 surfaces    CZERNY.OPT       Crossed CzernyTurner
   X         Z        Curv     pitch     Type?     Diam    Gx    Order
----.----:---.---:--------:------:---------:------:-----:-----:
   0      : 0.1    :         :       : iris     :  2   :     :     :
   0      : 98.33  : -0.005  :  10   : mirror   : 40   :     :     :
 -30      : 14     :         :  33   : Grating  :      : 0.4 :  1  :
  38      : 52.5   : -0.005  :  75   : mirror   : 40   :     :     :
 -58.915 ? 52.5   :         :  90   : CCD      : 25   :     :     :
         :        :         :       :          :      :     :     :
```

Figure A4-1 illustrating a crossed Czerny-Turner spectrometer. Light enters from a slit or fiber at the left, and is collimated before falling onto the plane diffraction grating. A camera mirror at top focuses the spectrum onto a planar sensor at bottom.

Appendix 5: Holographic Optical Elements

BEAM FOUR can trace light rays diffracted from a holographic optical element (HOE). An HOE is a transmitting or reflecting optical surface that has been exposed to a pair of mutually coherent light sources and developed to photochemically fix their interference pattern. An HOE differs from a classical parallel-groove diffraction grating in that the HOE grooves can form a wide variety of curved patterns. In use, the pattern serves as a diffracting optic that when properly designed can offer desirable focal properties or dispersion characteristics.

The groove pattern is completely specified by specifying where in space each of its two wavefronts originated, and what the recording wavelength was that produced the interference pattern. Therefore BEAM FOUR has seven parameters that serve to completely describe the HOE. These optics-table parameters are:

HOELambda: the recording wavelength. This should be expressed in the same units that your ray table uses to describe the wavelengths of your rays in your "wave" column. Microns is the most common choice. Since HOEs are commonly recorded with a high power blue laser, HOELambda is usually in the range 0.2 to 0.5 microns. In your optics table column header, this variable can be abbreviated HOEL, but spelling it out may keep you from forgetting what it means.

HOEx1, HOEy1, HOEz1: The location in space of your first point source of coherent recording light. These coordinates are specified in the same units that your optics table uses, e.g. meters or mm. The coordinate frame is the local frame: 0,0,0 is at the center of the HOE face, and the z axis is the HOE center normal. Defaults are zero for unused coordinates. For a parallel recording beam you will specify a source point at infinity; so use any astronomically large number. For example a beam at 45 degrees in the (x,z) plane would have the source coordinates HOEx1=1e99, HOEz1=1e99.

HOEx2, HOEy2, HOEz2: The location in space of your second point source of recording light, as above.

The above seven parameters and the surface geometry serve to define the HOE pattern. To trace your rays, BEAM FOUR needs also to have you specify the order of diffraction that you want followed, and the numerical wavelengths of your individual rays. So you will need an ORDER column in your optics or ray table, and an @wave column in your ray table with the numerical wavelengths of your rays.

The HOE parameters can be autoadjusted to tune up an optical design. Once you have a feasible trace (that is the rays reach their destination surface), put a question mark into the tagfield of each variable to be adjusted, and choose the autoadjust function from the RUN submenu.

The HOE need not be recorded on a flat surface. The other surface descriptors continue to apply, so that you can for example employ an HOE pattern on a hyperboloid or torus.

Here is one example. A circularly symmetric ring-pattern HOE is produced by interference between two coherent on-axis point emitters whose recording wavelength is 0.4 microns. This HOE is to act as a positive (converging) lens for light at 0.55 microns. In the demo HOE.OPT the first surface is the HOE and the second surface is the exit side of its refractive substrate. Incoming parallel rays are brought to a focus on the third surface. The recording geometry is specified by the data HOEz1 and HOEz2, which give the effective locations on the HOE's z axis at which the point sources were placed. Since the point sources lie on the HOE z axis, the parameters HOEx1, y1, x2, and y2 are allowed to default to zero. (In a realistic recording geometry these would be two images of a single point source, seen through a beamsplitter). The table HOE.RAY illuminates the optic with five parallel incoming rays at a wavelength of 0.55 microns. The HOE design can be tuned up by autoadjusting the HOE parameters. Here, HOEz1 and HOEz2 are marked for autoadjustment to bring the 0.55 micron rays to a focus.

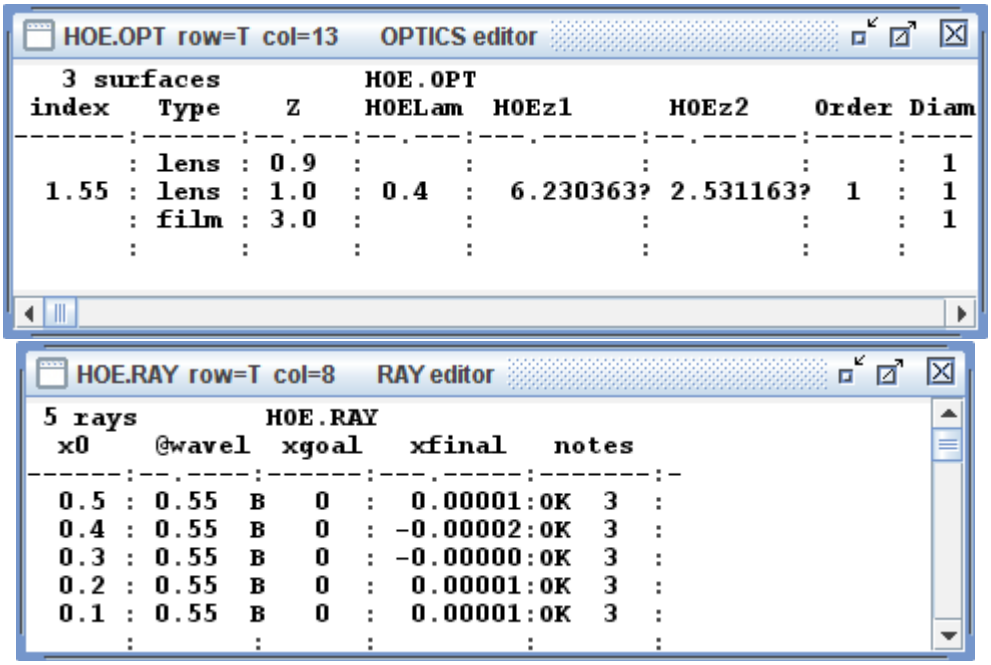


Figure A5-1 A simple HOE pattern recorded onto the front surface of a transparent plano window. The two recording point sources are both on-axis to create a circularly symmetric diffractor. The HOE acts as a highly chromatic lens, focussed here with collimated incident light at 0.55 microns.

An HOE that is closely related to this example is the Fresnel zone plate, which is obtained by setting HOEz1 or HOEz2, but not both, to infinity (say, 9e99).

Appendix 6: Reverse Ray Tracing

Although BEAM FOUR has no built-in tool for reverse ray tracing, it is a straightforward task to convert an optical prescription to its reverse. This is simply because BEAM FOUR is strictly sequential, and the reverse prescription is the forward prescription with the row sequence reversed. To produce a reversed .OPT file do these steps:

1. Rename your file XXX-rev.OPT
2. Using the table editor, copy the first surface entry to the bottom of the table.
3. Copy the next surface entry to just above the bottom of the table.
4. Repeat until each row has been copied just once.
5. Check the new sequence.
6. Delete the top group of surfaces including the original final surface.
7. Append a new final surface to the bottom of the list, located where the rays started.
8. For each pair of lens surfaces, move the refraction spec to the second surface.
9. If there are any CBin/CBout pairs, swap their names, but not their data.

You can use this .OPT file for any purpose. One purpose could be to check its validity using the optical reciprocity principle, that each ray introduced into the system's output with reversed direction should propagate to the original input location. Do this:

In the original (forward) trace, provide output fields for Xfinal, Yfinal, Zfinal, Ufinal, Vfinal, and Wfinal with plenty of digits accuracy, and run that trace so that each ray delivers its complete final state information.

Then with the original .RAY file:

1. Rename the file xxx-rev.RAY;
2. Delete the old columns X0 Y0 etc. and rename the old output column heads as inputs:
    Xfinal -> X0
    Yfinal -> Y0
    Zfinal -> Z0
    Ufinal -> U0
    Vfinal -> V0
    Wfinal -> W0
3. Change the signs of U0, V0, and W0 but not the new X0, Y0, Z0.
4. Add new fields for Xfinal and Yfinal.
5. Run a trace of your reversed optic with your reversed rays.

According to the optical reciprocity theorem, the reverse ray trace should place its final rays on the origins of the forward rays, but with reversed directions. This provides a check that everything has been properly reversed.

Index