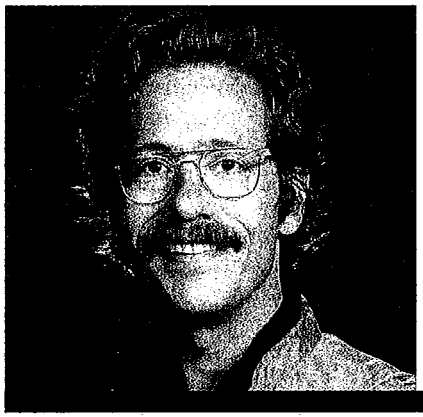


# ESSAY 5
# Ray Tracing Using a Computer

MICHAEL LAMPTON

In the preceding chapter you learned how an optical device produces an image: rays proceed from the object, encounter one or more optical surfaces that reflect or refract them, and finally arrive at a surface where they form an image or become visible. A professional optical engineer designing a camera lens or a microscope may have to examine the detailed paths of thousands of individual rays going through each trial system to evaluate its image quality. To handle this workload, rays are traced using a computer.

A ray tracing program has to deal with two entities: the optical system and the rays passing through it.

For a system producing a real image (e.g., a movie projector or a camera lens) the screen or film is included as the final optical surface. Information about image quality is obtained from the positions of the final rays, ignoring their arrival directions. However, for systems that produce a virtual image, such as a telescope or microscope, the final element is an "eye plane" where the viewer's eye will be located. Here, information about the image quality is obtained from the final ray directions rather than the positions of the final rays.

A ray tracing program needs the starting point and direction for each ray. The position and direction are customarily specified in Cartesian vector form: $X$, $Y$, and $Z$ locate the point in space from which the ray is launched towards the optical system, and $U$, $V$, and $W$ are the components of the unit vector along the ray. A ray aimed along the $+Z$-axis has a direction vector $U = 0$, $V = 0$, $W = 1$. The length of the ray direction vector is always unity.

The ray trace program called "TRACE" is listed in ■ Figure E5.1. The code is written in BASIC. Separate stages of the computation are marked by groups of line numbers: the setup of the optical train is done in the 2000's, the ray is set up in the 3000's, and the loop that manages the actual trace is in the 4000's. The remaining groups of statements are subroutines that do the numerical work. Like professional ray tracers, this program has messages that catch common error situations. If a surface intersection occurs behind the start of a ray segment, the message "backward" is displayed. If a ray is refracted into a lower-index medium at too large an angle, it may not be transmittable due to total internal reflection; a statement in the refraction subroutine handles this situation.

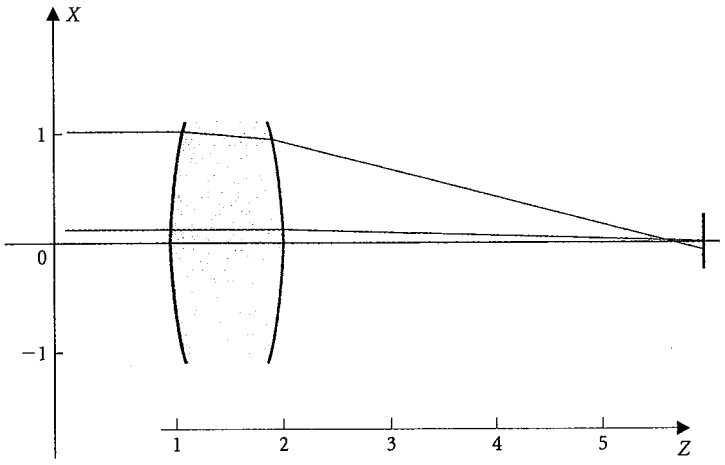
This program deals with optical systems that are coaxial: the surfaces are all lined up along the $Z$ axis with their centers at $X = 0$, $Y = 0$, and none of them is tilted. The position of each surface is described by the $Z$-coordinate of the point where it crosses the axis. The surface figure is written as a mathematical function describing how deviation from a plane surface depends on the distance $r$ from the axis of symmetry.

$$\text{Deviation from a plane} = \frac{Cr^2}{1 + \sqrt{1 - SC^2r^2}}.$$

If $S = 0$, the surface is a paraboloid. It curves toward the $+Z$ or $-Z$-direction, depending on whether the curvature $C$ is positive or negative. The shape factor $S$ allows the same formula to describe other kinds of surfaces too. Negative values of $S$ generate hyperboloids, while positive values of $S$ describe ellipsoids (prolate for $S < 1$ and oblate for $S > 1$). The special value $S = 1$ represents a spherical surface, which is by far the most common in the design of lenses because spherical shapes are the most easily manufactured.

```
1000 REM  RAY TRACING PROGRAM
1010   DEFINT I-N: CLS : X0=1: Y0=0: Z0=0: U0=0: V0=0: W0=1
2000 REM SPECIFICATION OF THE OPTICS
2010   IF EN(1) < 1 THEN 2050
2020   PRINT : INPUT "CHANGE OPTICS?  Y/N/QUIT:", X$
2030   IF X$ = "q" THEN END
2040   IF X$ <> "y" THEN 3000
2050   INPUT "HOW MANY SURFACES?  ", N: IF N < 1 THEN END
2060   PRINT "IndexOfRefract, Z, Curv, 0=parab/1=sph, transmit/reflect"
2070   FOR I = 1 TO N
2080     INPUT EN(I), ZV(I), C(I), S(I), T$(I)
2090     IF EN(I) < 1 THEN EN(I) = 1
2100   NEXT I
3000 REM specification of the ray start
3010   PRINT : INPUT "New ray start? y/n/quit:", X$
3020   IF X$ = "q" THEN END
3030   IF X$ <> "y" THEN 4000
3040   PRINT " x0, y0, z0, u0, v0": INPUT X0, Y0, Z0, U0, V0
3050   IF U0*U0+V0*V0 > 1 THEN PRINT "Error: u0 v0 too big": GOTO 3040
3060   W0 = SQR(1-U0*U0-V0*V0)
4000 REM run the ray
4010   X=X0: Y=Y0: Z=Z0: U=U0: V=V0: W=W0
4020   PRINT "    x          y          z          u          v          w"
4030   PRINT USING "####.######"; X,Y,Z,U,V,W
4040   FOR I = 1 TO N                  : REM for each surface,
4050     Z = Z - ZV(I)                 : REM go to its frame
4060     GOSUB 5000                    : REM find ray intercept
4070     IF I = N THEN 4100            : REM all done
4080     IF T$(I) = "r" THEN GOSUB 6000: REM reflect it
4090     IF T$(I) = "t" THEN GOSUB 7000: REM refract it
4100     Z = Z + ZV(I)                 : REM return to lab frame
4110     PRINT USING "####.######"; X,Y,Z,U,V,W
4120   NEXT I
4130   GOTO 2000
5000 REM intercept subroutine finds new X,Y,Z
5010   XI=X: YI=Y: ZI=Z: OLDF=1000000  : REM save starting point
5020   IF ABS(W)>.7 THEN X=X-U*Z/W: Y=Y-V*Z/W: Z=0
5030   F = Z-.5*C(I)*(X*X+Y*Y+S(I)*Z*Z): REM f=0 at quadratic surface
5040   IF ABS(F) < .000001 THEN 5100   : REM converged
5050   IF ABS(F/OLDF) >= 1 THEN PRINT "ray missed surface": GOTO 2000
5060   OLDF=F: DFDS = -C(I)*X*U-C(I)*Y*V+(1-S(I)*C(I)*Z)*W
5070   DS=.1: IF ABS(DFDS) >.001 THEN DS=-F/DFDS
5080   X=X+U*DS: Y=Y+V*DS: Z=Z+W*DS    : REM take the step
5090   GOTO 5030
5100   DIR = (X-XI)*U+(Y-YI)*V+(Z-ZI)*W: IF DIR < 0 THEN PRINT "backwards"
5110   RETURN
6000 REM reflection subroutine finds new U,V,W
6010   GOSUB 8000
6020   U=U-2*DOT*P: V=V-2*DOT*Q: W=W-2*DOT*R: RETURN
7000 REM refraction subroutine finds new U,V,W
7010   GOSUB 8000
7020   D=EN(I+1)/EN(I): REM refraction ratio
7030   DOT2=1-(1-DOT*DOT)/(D*D)        : REM (emergent ray dot normal)^2
7040   IF DOT2 < 0 THEN PRINT "error: internal reflection": GOTO 2000
7050   GAMA = SQR(DOT2)*SGN(DOT)-DOT/D
7060   U=U/D+P*GAMA: V=V/D+Q*GAMA: W=W/D+R*GAMA: RETURN
8000 REM surface normal subroutine finds new P,Q,R,DOT
8010   P=-C(I)*X: Q=-C(I)*Y: R=1-S(I)*C(I)*Z: A=SQR(P*P+Q*Q+R*R)
8020   P=P/A: Q=Q/A: R=R/A: DOT=U*P+V*Q+W*R: RETURN
```

■ FIGURE E5.1
Ray tracing program




■ FIGURE E5.2
Two parallel rays traced through a bi-convex glass lens.

To use this program, first draw a diagram of the optical system you want to study. Choose a $Z = 0$ point at the object if it is at a finite distance, or at some convenient place near the front of the instrument if the object is at infinity. For geometrical ray tracing, only the relative dimensions matter, so any convenient unit of length can be used. In optical work, lengths are often specified in inches or millimeters. Whatever unit you choose, be consistent. Surface curvatures are the reciprocals of the radii of curvature, with each radius in the same units you have used for the rest of the system. Number the surfaces in the sequence that light will arrive at them. Each surface needs a curvature and shape value. Flat surfaces have $C = 0$ and $S =$ anything. Mark the $Z$-axis values of the location of each surface—that is, the point where it crosses the $Z$-axis.

For our first example we shall trace two parallel rays of light through a thick biconvex glass lens (■ Figure E5.2). The rays come to an approximate focus on a flat screen near the focal plane. The optical media involved are air, whose refractive index is 1.0, and glass, assumed here to have a refractive index of 1.60. The media and surfaces are specified in the same sequence that light reaches them.

Run the program by typing GWBASIC TRACE or BASICA TRACE. The first thing the program asks is:

```
How many surfaces?
```

Our first example has three surfaces: the front of the lens, the back of the lens, and the screen. So, respond with a 3. The program then asks you to specify the three surfaces. Each surface is described by a list of five items separated by commas. The five things are the refractive index of the medium approaching the surface, the surface's location along the $Z$-axis, the curvature of the surface, the shape of the surface, and the letter "t" or "r" indicating whether the transmitted or reflected ray is to be followed (use "t" for lenses and "r" for mirrors). Press the Enter key after typing in each row of data. For our example, the three surfaces are specified as follows:

```
1,      1,      0.2,     1,     t
1.6,    2,     −0.2,     1,     t
1,      6,      0,       1,     t
```

The first row specifies a surface approached in air ($n = 1$), with the surface located 1 unit along the $Z$-axis. It is positively curved; that is, it curves towards positive $Z$. Since the medium on the left of the surface is air and the glass will be to the right, this makes the surface convex. Curvature is the reciprocal of the radius of curvature. With a curvature value of +0.2, its radius of curvature is 5 units, curved towards larger $Z$. The surface shape value of 1 indicates a sphere.

The second row starts off with a refractive index of 1.6, so the ray will be traveling through a dense glass medium as it travels from the first surface to the second. The second surface in our example is located at $Z = 2$ units. Unlike the first surface, this second surface is curved

towards the negative $Z$-direction. Since the glass lies to the left of this surface and air is to the right, this second surface is also convex. Again it is spherical: shape = 1. Again, we follow the transmitted ray: "t".

The third row specifies that the ray will travel through air ($n = 1$) until it gets to the third surface at $Z = 6$. The third surface has zero curvature, which means it is flat. It is the image screen.

When you have entered all the surface information, the program then asks if you want to set up a ray to launch at the system:

```
New ray start? y/n /quit:
```

The program has a built-in default ray start that we can use. This default ray is one unit above the $Z$-axis and parallel to it. For this example we will use this default ray. Enter "n" to use this existing ray start.

Now the program has all the information needed to compute the ray trace. It works through the list of surfaces, displaying the ray position $X$, $Y$, and $Z$, and the ray direction components $U$, $V$, and $W$. The initial line of output shows the ray starting position and direction. The subsequent lines of output show where the ray intercepted the first, second, and third surfaces:

```
   x          y          z          u          v          w
 1.000000   0.000000   0.000000   0.000000   0.000000   1.000000
 1.000000   0.000000   1.101021  -0.075957   0.000000   0.997111
 0.938285   0.000000   1.911173  -0.240905   0.000000   0.970549
-0.076626   0.000000   6.000000  -0.240905   0.000000   0.970549
```

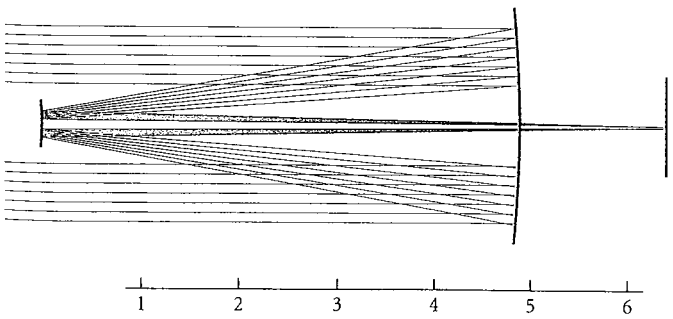
This completes an entire cycle of data input and output. The program then loops back to see whether you have any changes to make in the optical system and then asks if you want to change the ray start data. At either point, a "q" response will quit the program.

What can a ray trace tell us about our example lens? If the system were sharply focused on an object on the $Z$-axis at infinity, then this entering ray would have landed at the focal point on the axis. The final value of $X$ would have been zero. But it is not zero! It is negative, showing that it crossed the axis on its way to the final image screen. For this ray, the screen is too far away from the lens. Should we trace again, moving the screen closer to the lens? Perhaps not. Let's trace again, this time using a ray that starts much closer to the axis, at $X = 0.1$, more closely imitating a paraxial ray:

```
x          y          z          u          v          w
0.100000   0.000000   0.000000   0.000000   0.000000   1.000000
0.100000   0.000000   1.001000  -0.007501   0.000000   0.999972
0.092513   0.000000   1.999144  -0.023109   0.000000   0.999733
0.000032   0.000000   6.000000  -0.023109   0.000000   0.999733
```

This time, the ray lands extremely close to the desired $X = 0$ point in the image, 2000 times closer, in fact. What is different? Real lenses, even when made from mathematically perfect spherical surfaces, have a variety of optical aberrations. We have just seen the effect of spherical aberration. The ray near the axis is very well focused, but the initial ray, starting ten times farther from the axis, landed badly out of focus.

Ray tracing is used in advanced work too. NASA's premier space observatory, Hubble Space Telescope, shown in ■ Figure E5.3, is a two-mirror instrument that was designed to




■ FIGURE E5.3
Hubble's two-mirror Optical Telescope Assembly.

produce images just a few microns in size, while employing a large 2.4-m aperture to gather useful amounts of light from extremely distant galaxies. The primary mirror is hyperbolic (but very nearly a paraboloid) and the secondary mirror is hyperbolic. Unfortunately, a manufacturing flaw caused the primary mirror to be made with a shape factor that is too negative. We can explore the consequences of this error with our ray tracer.

First, Hubble as it was designed (units are meters):

```
Number of surfaces: 3
  1,      4.906071,      -.09057971,      -.0023,       r
  1,      0,             -.736377,        -.49686,      r
  1,      6.4062,          0,               0,          t
```

Using a typical ray 1 m from the axis (the default ray serves nicely), we get:

```
   x          y          z          u          v          w
 1.000000   0.000000   0.000000   0.000000   0.000000   1.000000
 1.000000   0.000000   4.860781  -0.179684   0.000000  -0.983725
 0.111315   0.000000  -0.004558  -0.017362   0.000000   0.999850
-0.000005   0.000000   6.406200  -0.017362   0.000000   0.999850
```

Notice the superb focus: for this ray, or any ray starting within 1.2 m of the axis, the final ray coordinate lies within a few microns of the axis. Moreover, rays making known small angles to the axis arrive at focal points that measure those incoming angles precisely, as you can verify by ray tracing. The image quality is superb.

Hubble's primary mirror, as built, had too negative a shape factor:

```
Number of surfaces: 3
  1,      4.906071,      -0.09057971,      -0.0120,      r
  1,      0,             -0.736377,        -0.49686,     r
  1,      6.4062,          0,                0,          t

 x          y          z          u          v          w
1.000000   0.000000   0.000000   0.000000   0.000000   1.000000
1.000000   0.000000   4.860783  -0.179676   0.000000  -0.983726
0.111350   0.000000  -0.004561  -0.017303   0.000000   0.999851
0.000409   0.000000   6.406200  -0.017303   0.000000   0.999851
```

Notice the final ray $X$-coordinate has increased enormously. This situation indicates a negative spherical aberration, the opposite of what we discovered in the simple glass lens example.

The Hubble Space Telescope can be focused in orbit by moving its secondary mirror slightly towards or away from the primary mirror. If the secondary mirror is shifted 200 microns farther away from the primary mirror, the edge rays are better focused while the rays nearer the axis are more poorly focused:

```
Number of surfaces: 3
  1,       4.906071,      -0.09057971,      -0.0120,      r
  1,      -0.000200,      -0.736377,        -0.49686,     r
  1,       6.4062,          0,                0,          t

 x          y          z          u          v          w
1.000000   0.000000   0.000000   0.000000   0.000000   1.000000
1.000000   0.000000   4.860783  -0.179676   0.000000  -0.983726
0.111314   0.000000  -0.004758  -0.017355   0.000000   0.999849
0.000034   0.000000   6.406200  -0.017355   0.000000   0.999849
```

Between 1990 and 1993, Hubble was operated in a number of different configurations, including this one. In December 1993, astronauts visited Hubble and installed an optical corrector that introduces a tiny amount of positive spherical aberration, eliminating most of the negative spherical aberration caused by the shape parameter $S$ being excessively negative.