Software for Opto-Mechanical Modeling

Bitmap Source Utility
User's Manual
Release 7.1

Lambda Research Corporation
25 Porter Road
Littleton, MA 01460
USA

Tel. +1-978-486-0766
support@lambdares.com
sales@lambdares.com
www.lambdares.com

COPYRIGHT AND TRADEMARK ACKNOWLEDGMENTS

COPYRIGHT

TRADEMARKS

TABLE OF CONTENTS

Introduction

The Bitmap Source Module provides a way for you to generate TracePro® source files from image files. The module can read popular image formats (Windows BMP, GIF, JPG, PNG, and MOV) and create a source file that models the bitmap. To create a source file, you simply open the file using the Bitmap Module and use the *Conversion*|*Source Wizard* dialog box to create the source file. Then, using TracePro, you can insert the source file in the usual way using *Define*|*File Source*.

Beware that to adequately sample a bitmap image file for this type of end-to-end simulation requires many rays for each pixel, perhaps 50 to 100 or more for each. Large image files may contain millions of pixels, so it easy to see that detailed sampling of large images through a complicated lens system may require many millions of rays. You are advised to restrict your modeling to small images until advances in computer hardware are sufficient to make the simulation of large images feasible.

This User's Manual assumes that you are proficient in the operation of TracePro. Please refer to the TracePro manual and tutorials for more information.

Example: Creating and ray-tracing a bitmap source file.

In this example we will open a small image file using the Bitmap Source module and trace rays. We will not have complete information regarding this particular bitmap, but we can estimate it by examining the image. The following sections provide step-by-step instructions for modeling the scene using this bitmap image.

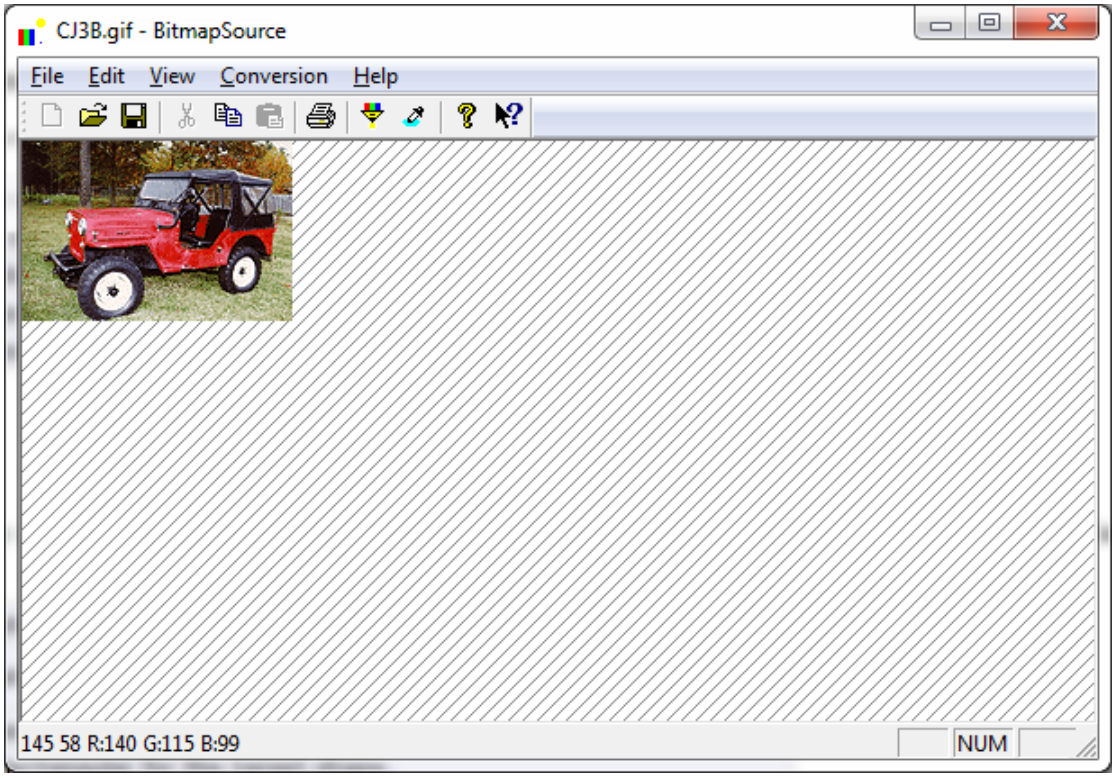
Prepare the lens system

Open the **lensdemo.oml** file in the **TracePro\Examples\Demos** folder. Though the units of the model are millimeters, the size of the lens indicates that it was created using inches. Scale all the objects in the model by a factor 25.4 to make the lens the correct size in millimeters. To do this easily, select *Edit*|*Select Object* and select all the objects in the model by dragging a "rubber band" rectangle around all the objects. Then select *Edit*|*Object>Scale* and enter a scale factor of 25.4. Be sure to select the option *Scale Position*, then click *Apply*. Save the scaled model with a new name using *File*|*Save As*.

Observe that the scaled model has a front aperture diameter of about 100mm and the front of the lens barrel is located at approximately (0,0,8).

Open the image using the Bitmap Source Module

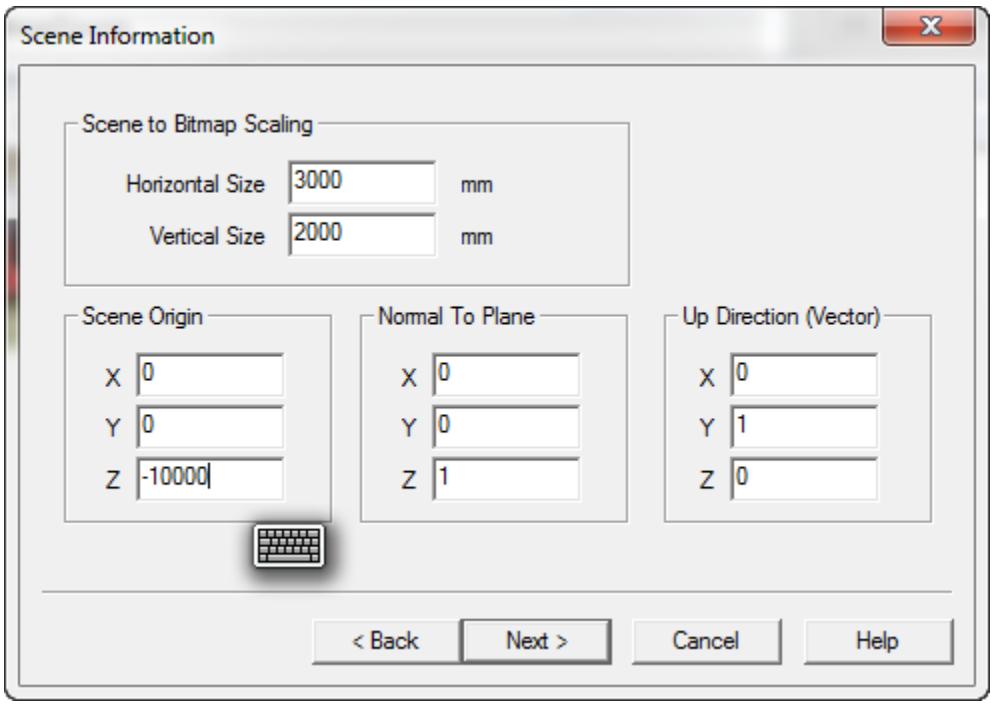
Start the Bitmap Source Module and open the file **CJ3B.gif**. Once the file is open, it should appear as below, with the bitmap displayed in the upper left corner of the window. You can view the specifications of the image by selecting *View*|*Bitmap Info*. The dimensions of the image in pixels are 100 x 150 (height x width). We have chosen a small image for this example so that the memory and ray-trace time requirements are modest.



Generate a test ray file

To generate a test file in the Bitmap Source Module, select *Conversion*|*Source Wizard*. The test image contains a vehicle, so we can estimate that the physical dimensions of the bitmap should be about 2m x 3m (vertical x horizontal). It is important to keep the aspect ratio of the image (2:3) the same as that of the pixel dimensions (100:150) so that the pixels are square. If you know that the pixels are not square in your image, you should set the physical dimensions to achieve the desired scene aspect ratio. We estimate that the scene was about 10m from the camera.

The first pane of the Source Wizard gives you some helpful description of the geometry for setting up the source. Click the *Next* button to go to the next pane. Enter the scene dimensions into the Source Wizard dialog as shown below.

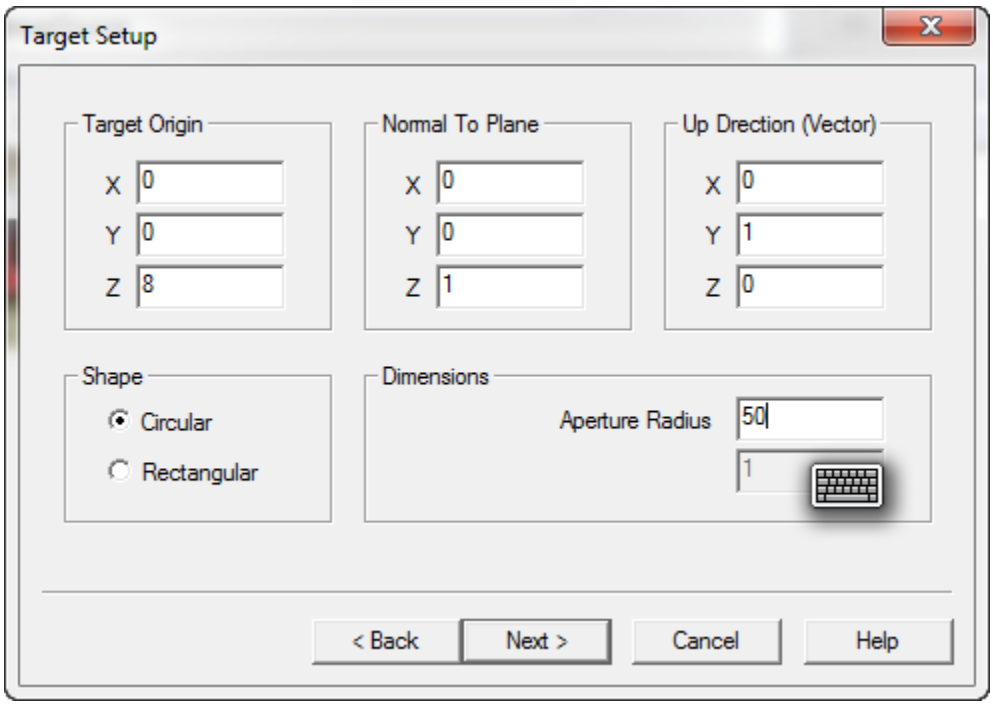


The lens in **lensdemo.oml** is near the origin of coordinates and has its optical axis along the z axis, so we put the origin of the scene 10m away (at z = –10,000mm) and on the z axis (x = 0, y = 0). We orient the scene so that it is perpendicular to the z axis (*Normal to Plane* = 0,0,1) and has the y axis pointing up in the scene (*Up Direction* = 0,1,0). Click *Next* to go to the next pane.

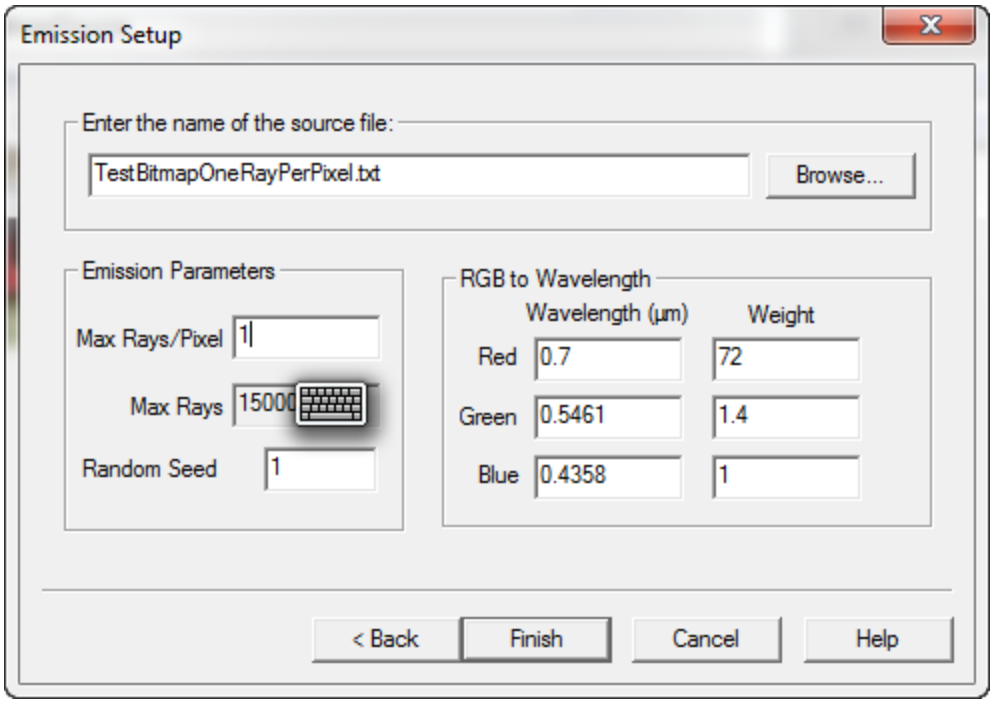
When rays are emitted from the bitmap, we want them to go toward the optical system. To ensure that this happens, we must provide a target to the Bitmap Source Module to use for aiming rays. This is similar to importance sampling in TracePro. Switch back to TracePro to verify that the front of the lens is at about z = 8mm and has a radius of about 50mm. Imagine a circular plane target at the front of the lens, with radius equal to 50mm and located z = 8mm. This is the plane from which rays in the source file will be launched. *The Bitmap Source Module will trace rays from the bitmap to this target plane, and store the directions and locations of the rays in the source file at this plane*. Therefore it is important that the target plane not be embedded in any geometry. This is in contrast to importance sampling in TracePro, in which importance sampling targets are often embedded in the model.

To accomplish this, we use the Bitmap Source Module to create a source file with target at the front of the optical system. Then when we insert the source file into the TracePro model, we insert it at the origin of coordinates.

You can choose either a rectangular or circular target plane. For a circular target plane, the up direction is unimportant. For a rectangular target plane, the Aperture Radius plane entry changes to Aperture Height and an additional entry is available for Aperture Width. The figure below shows what values to enter into the dialog for this example. After entering the values, press Next to go to the next pane.



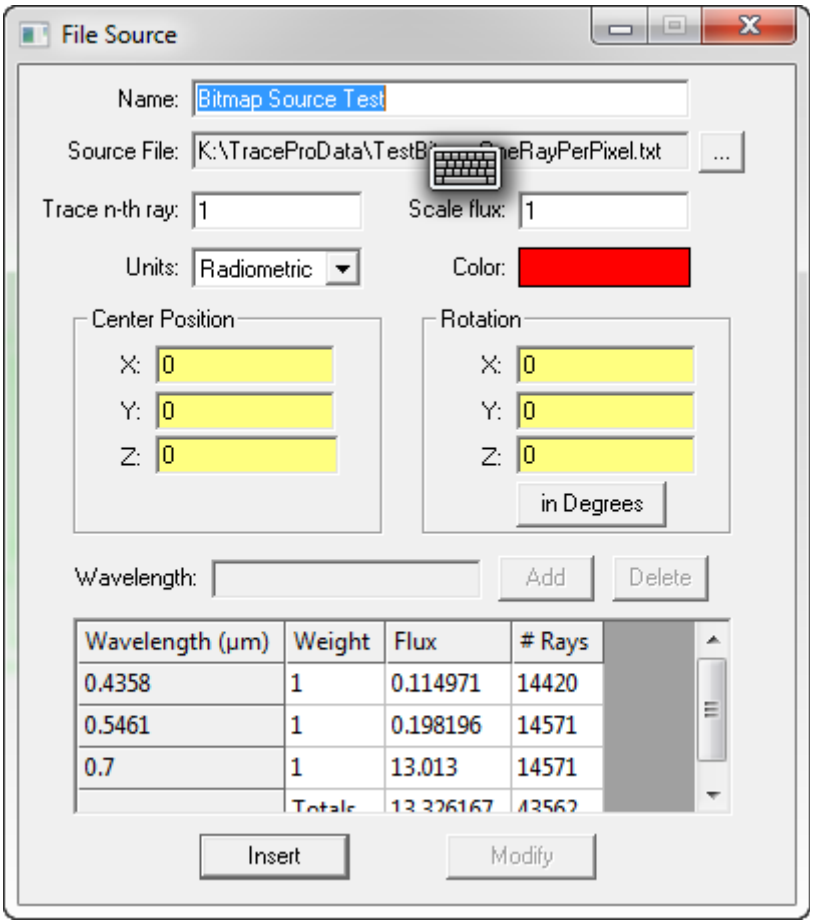
This pane allows you to specify the name of the source file, the number of rays per pixel to trace, and the representative wavelengths for red, green, and blue content of the bitmap file. The number of rays to trace is a matter of judgment, but you should trace as many as can be done in a reasonable amount of ray-trace time to get the best possible sampling of the optical system and the bitmap. At first we will create a test file to verify the correctness of our source file. Specify *Rays per Pixel* = 1 and use the file name **TestBitmapOneRayPerPixel.txt**. Leave the red, green, and blue wavelengths and weights as they are. Click *Finish* to actually create the source file.



The actual number of rays generated depends on the Red, Green Blue value of each pixel. The Bitmap Source Module will adjust the number of rays for each pixel for each color so that the flux for each ray is approximately the same. Pixels that are black will generate no rays.

Define the file source and trace rays

Now switch back to TracePro. Select *Define|File Source* and browse to the source file you just created. As discussed above, insert the source at (0,0,0) with rotation angles (0,0,0). Click Insert to to insert the source file.



Now you can begin the ray-trace in the usual way. The source file that we created, with only one ray per pixel, creates a maximum of (100 x 150 pixels) x (1 ray/pixel) x (3 wavelengths) = 45,000 rays. Any pixel that has a 0 value for red, green, or blue will not have a ray for that wavelength, so the actual number of rays generated will probably be less than 15,000 for each wavelength. Also, the target we chose has a radius of 50mm to completely fill the front of the lens system, while the entrance pupil has a radius of only 8.6mm. This means that only about $(8.6/50)^2 = 0.03$ of the rays contribute to the image. This condition is necessary, for two reasons: 1) the projection of the entrance pupil onto the target plane is in a different location for each pixel, and 2) we must sample the entire target to generate scattered and ghost rays. They are real and cause degradation of the image.

When the ray-trace is finished, select the image surface and display an irradiance map. Select *Analysis*|*Irradiance/Illuminance Options*, and in the *Quantities to Plot* field select *True Color*. The image is very grainy because only 3% of the rays we started (or about 1/30) have contributed to the image. Also notice that the image is a small rectangle in the center of the image. If you wish, you can create a new solid object that is about the size of the actual image, then delete the original circular disk image. Insert a block that is 16 x 16 x 2 and position it so that the surface nearest the lens is at the same z location as the original image plane. You can find out the original z location by selecting the original image surface and displaying an Incident Ray Table (*Analysis*|*Incident Ray Table*). Use $z_{original}$ + 1 as the z-center of the 16x16x2 block. Apply the Perfect Absorber surface property to the new image surface and repeat the ray-trace to verify that the image is correctly positioned.

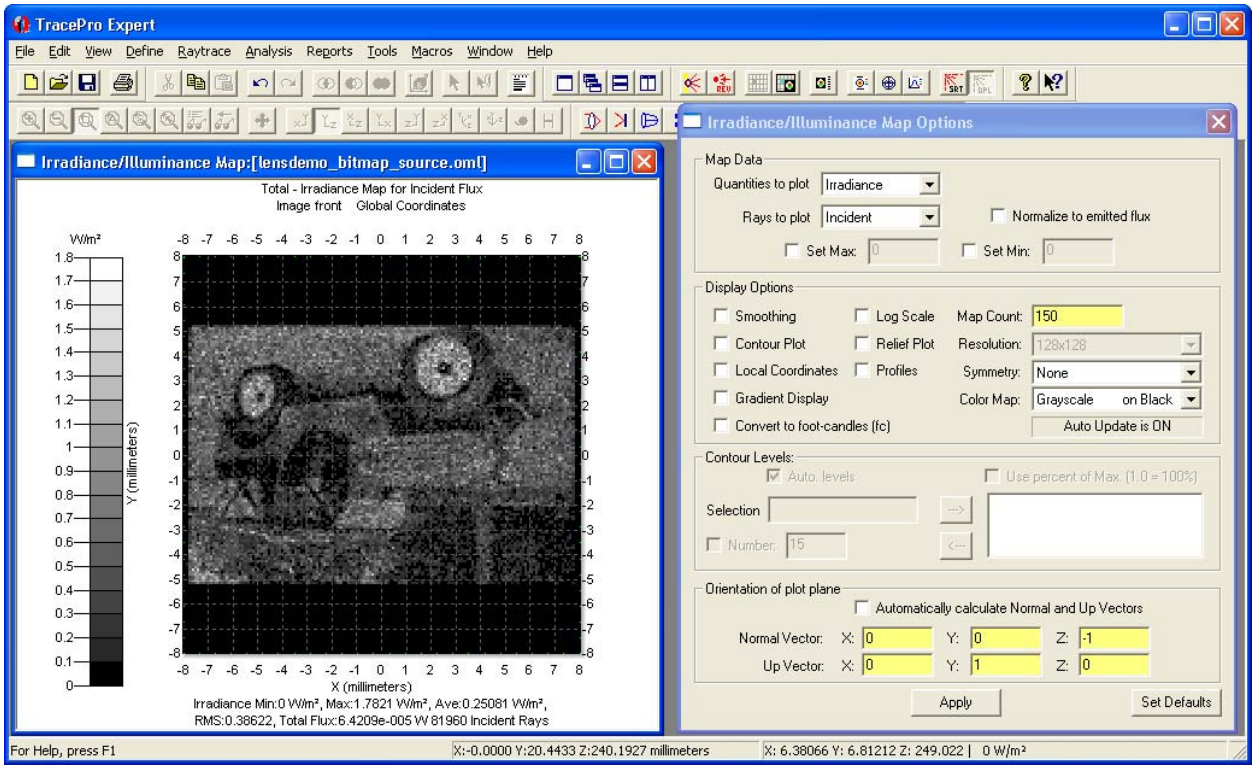
Switch back to the Bitmap Source Module. Using the *Conversion*|*Source Wizard*, create another source file, the same as the first one except more rays per pixel. If we suppose that we want 5 rays per pixel per wavelength to be image-forming rays, then we need 30x5 = 150 rays per pixel in the source file. Enter all the same data as before, except on the last pane enter *Rays per Pixel* = 150. This will create a very large ray file (approximately 170 MB) and will take several minutes to complete. Click *Finish* to create the source file.

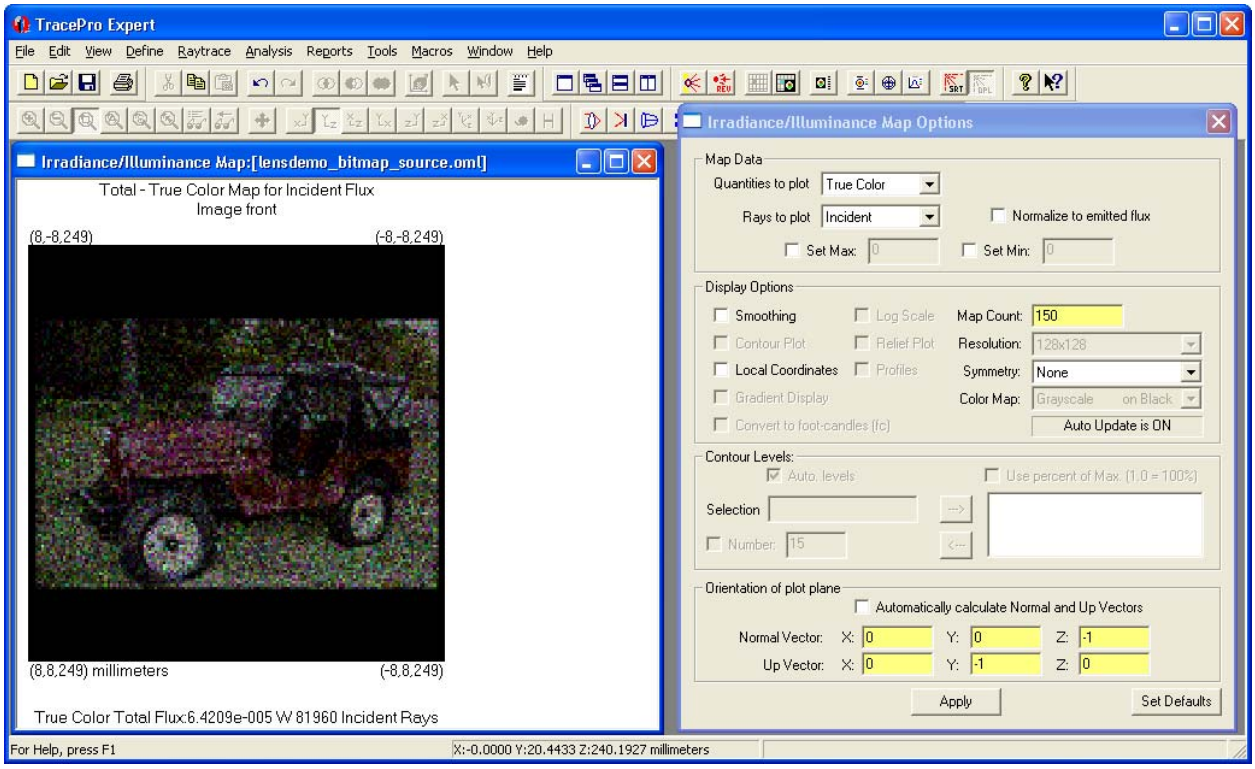
Switch back to TracePro. Select the Source tab of the System Tree, find the file source and double-click on it to open the File Source dialog box. Browse to the new source file and click Modify. This will take several minutes because of the large size of the file. Due to the large number of rays in this source file (approximately 2.7 million rays), an Analysis Mode ray-trace will consume too much memory to complete.

To avoid using too much memory, switch to Simulation Mode. Select the image surface and apply the *Exit Surface* property to it using the *Apply Properties* dialog box. Select *Raytrace*|*Simulation Mode* to switch to simulation mode. Save the model and begin the ray-trace.

When the ray-trace is completed, examine the Irradiance Map to see the output bitmap. Note that with the Normal and Up vectors set to their default values ((0,0,1) and (0,1,0), respectively), the image is inverted and reversed left to right. The bitmap is displayed as seen from the lens side of the image, with the y axis in the up direction. You can uninvert the image by setting the Up Direction to (0,-1,0) and you can unreverse the image by setting the Normal direction to (0,0,-1).

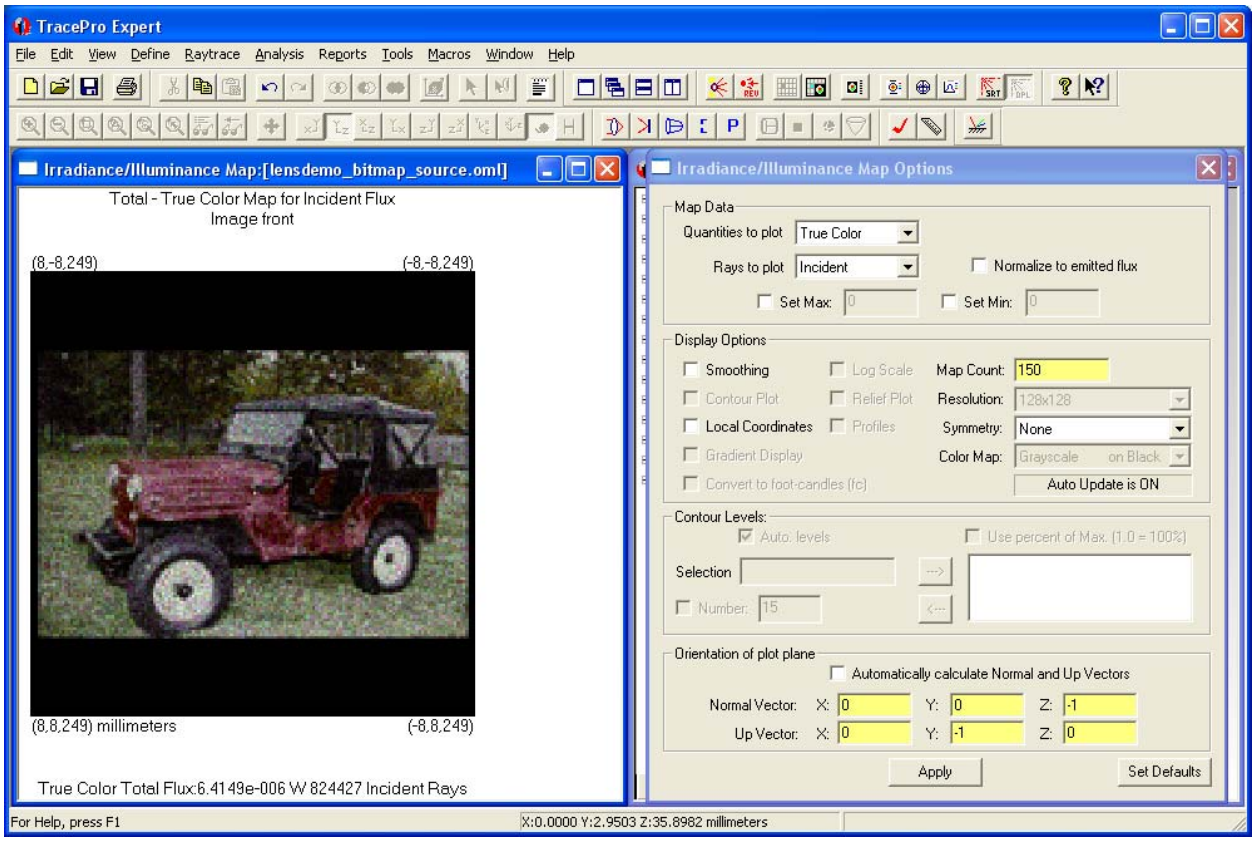
A grayscale display of the image is shown below as well as a True Color image with Up Vector changed to (0 -1 0) to show the image upright.





Clearly, many more rays are needed to obtain an image that has tolerable signal-to-noise ratio. We have repeated the exercise with ten times as many rays. The Bitmap Source module was run

again with the Maximum Rays per Pixel set to 1500. This new source file was inserted into the model and run. The result is shown below.



Operation of the Bitmap Source Converter

The converter works by starting one or more rays from each pixel in the image, and aiming the rays at the optical system. You supply data specifying the size, location, and orientation of the image, as well as the size, location, and orientation of the optical system. Finally, you specify how many rays per pixel you would like to model, and three wavelengths for modeling the red, green, and blue content of the image. After you enter a file name for the source file, the module creates a source file containing the rays that model the bitmap. When you insert the source file into TracePro, it traces the rays through the optical system as usual, and displays the irradiance in true color.

Color information in bitmap files is encoded as red, green, and blue values (additive color). The colors take on values from 0 to 255 for 24-bit color (8 bits per color), this being the preferred color format. Other color depths are also supported.

To generate the TracePro source file, rays are started from each pixel in the bitmap. The user may specify many ray sample points per pixel for good sampling, and there are three rays for each sample point, one ray for each color. The rays are assigned a flux and wavelength attribute corresponding to the brightness and color of the pixel and aimed toward the entrance surface of the optical system. Ray sample points are selected in a uniform random distribution over the pixel, and their directions are selected to give a random distribution over the entrance aperture of the optical system. The information written to the ray file contains the starting points and directions of

rays in the entrance aperture of the optical system, i.e., it comprises a virtual source representing the bitmap scene.

The scene luminance is specified by

- The global coordinates of the center of the scene.
- The x and y dimensions of a pixel at the center of the scene.
- The direction cosines of the local x and y (horizontal and vertical) axes of the scene, expressed in global coordinates.
- The red, green, and blue (RGB) values for each pixel in the scene.
- A mapping of the RGB values into one or more wavelengths.

The default mapping of RGB into wavelength is into three wavelengths in order to produce a true color image.

In general, the luminance mapping for 8-bit RGB values is:

$$L_\lambda(C_\lambda) = k\left(\frac{C_\lambda}{255}\right)^\gamma + L_0$$

Where L_0 is a DC offset, k is a proportionality constant to convert from digital color values to luminance, and C_λ is the digital color value (red, green, or blue), ranging from 0 to 255 for 24-bit color. γ corresponds to the contrast of the scene, and allows compression or expansion of the dynamic range of the original scene luminance values. Rays with zero flux are not traced. For the Bitmap Source Module, the k and γ values are set to one, and L_0 is set to zero.

This means that for a typical scene, e.g. 640x480 pixels, if we are to emit 50 rays from each pixel for each of three wavelengths to adequately sample the entrance aperture, up to 50 million rays would be generated. Be prepared for long ray-trace times, depending on the complexity of your optical system. In most cases, examining an enlarged small portion of a large image is preferred over examining the full image. This avoids having the resolution of the display affect the perception of the simulation results.

To generate a ray from within a particular pixel, we first pick a uniformly distributed random point on the entrance aperture. A unit vector pointing from the pixel coordinates to this point is the ray direction.

To determine the flux of the ray, we use a normalized luminance for the particular primary wavelength. The flux given to a particular ray of wavelength λ is

$$\Phi_\lambda = \frac{L_\lambda(C_\lambda)\int_\Omega \frac{\cos\theta}{\pi} d\Omega}{N}$$

where θ is the angle between the ray direction and the normal to the bitmap, $\cos\theta/\pi$ is the Lambertian angular dependence, Ω is the solid angle subtended by the entrance aperture as seen from the pixel, and N is the number of rays to be generated from the current pixel. The number of rays to be generated from the current pixel is equal (ColorValue/255)*M, where M is the maximum number of rays per pixel. If the color value is positive, at least one ray will be generated.

Following the tradition of the Monte Carlo method, the integral in the above expression can be approximated by

$$\int_{\Omega} \frac{\cos\theta}{\pi} d\Omega \approx \frac{\cos\theta}{\pi} \Omega$$

and Ω can be approximated (small angle approximation) by

$$\Omega \approx \frac{\hat{r} \cdot \hat{n}_a A_a}{r^2}$$

where unit vector r is the ray direction, unit vector n_a is the normal to the entrance aperture, A_a is the area of the entrance aperture, and r is the distance from the ray starting point on the pixel to the chosen point in the aperture.

Limitations and other considerations

Adequately sampling the image and optical system

TracePro performs a Monte Carlo ray-trace to simulate the performance of optical systems. For the Bitmap Source Module, creating a ray file for use by TracePro means that the starting locations and directions of rays are chosen randomly, but within the entrance aperture and the given pixel. Each ray started can create zero or one imaging ray and zero or more non-imaging rays. In the example outlined in this manual, only 1/30 of the starting rays can be expected to be imaging rays. In such a case, you must start 30 times more rays than you might think, in order to produce enough image-forming rays.

Any Monte Carlo simulation is subject to noise. This noise is analogous to that in an actual measurement. Just as in an optical measurement we need to increase the exposure or integration time to increase the signal-to-noise ratio, we must increase the number of samples in a Monte Carlo simulation. The number of samples required to give an adequate simulation depends on your particular optical system and image. For example, if you are simulating the effects of aberrations, vignetting, or aperture diffraction on the image, then the flux threshold can be set relatively high while still obtaining good results. If you are simulating the effects of stray light, you must set the flux threshold much lower, and the ray-tracing time will be much longer. As you have seen if you followed the example in the previous chapter, getting five rays per pixel per color to contribute to the image causes objectionable graininess. It is analogous to getting 5 photoelectrons per color in an image obtained with a photovoltaic detector.

Color fidelity and wavelengths

Using an RGB-encoded digital image as a source in a sophisticated ray-tracing program like TracePro is an approximation at best. The original scene from which the bitmap was obtained contained a continuous spectrum and continuous spatial content. The spectrum of the scene is collapsed or encoded into three primary colors approximating the color discrimination ability of the human eye. The colors are then digitized using typically 8 bits per color, and the spatial content is sampled by averaging over discrete pixels. A true simulation of the scene would require knowing the continuous spectrum of every point in the scene and continuous spatial information.

In practice, the issue of spatial sampling is not so severe, so long as you can obtain images that have more resolution than the optical system you are simulating. The caveat here is that the lens in the camera used to create the image may have had optical aberrations that further degrade the bitmap. The collapsing of the spectrum into three colors, however, is an insurmountable limitation.

The TracePro Bitmap Source Module allows a reasonable approximation for simulating the response of an optical system to a scene. It allows you to choose three wavelengths to represent the three primary colors. A ray-trace from a bitmap will produce a reasonably correct-looking output for many wavelength selections. A ray-trace from a bitmap will ***not*** accurately simulate the true performance of the optical system, especially if there is spectral discrimination present. Examples of spectral discrimination are filters and dichroic beamsplitters. In the presence of dispersive elements (e.g. a prism or a lens with chromatic aberration) TracePro will produce three distinct wavelength behaviors rather than a continuous spectrum. All of this is due to the fact that it is impossible to extract the original spectral content of the scene from an RGB bitmap file.

The science of color and the color encoding of images is a field of study unto itself and is beyond the scope of this User's Manual. The process of encoding an image into RGB throws away a great deal of information about the scene, as discussed above. Indeed, there are an infinite number of spectra that will produce the same color response in the human eye. The eye is an image sensor, not a spectrophotometer. The information contained in a RGB image is just sufficient to reproduce color. (Actually it is not quite sufficient, since no color gamut based on three primary colors can represent all the colors the eye can discriminate, but this shortcoming is not noticeable to the untrained eye.)