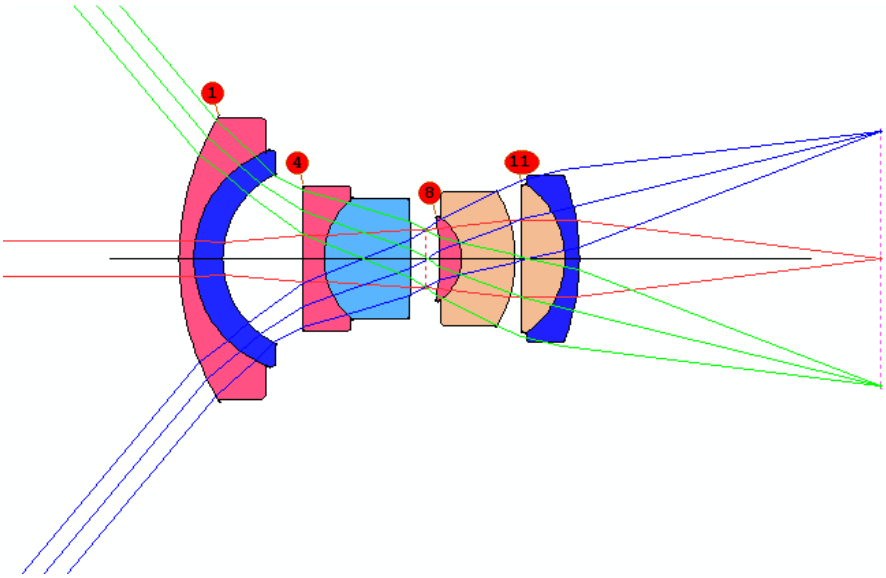
Lesson 18: What is a Good Pupil?

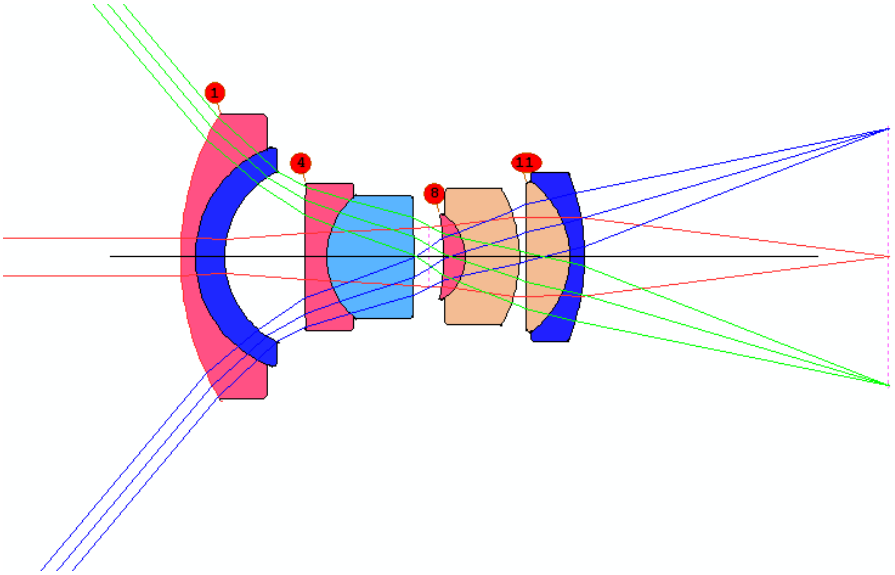
Users of other lens design codes already know about two common pupil definitions: a paraxial pupil that only applies to simple systems or where the stop is out in front, or, for more complex systems, “ray aiming”, which is intended to model a real stop somewhere inside the system.

For those not familiar with these concepts, let me explain. A lens often has a “stop” somewhere inside, as shown in the example below (which is found in X32.RLE). (For this picture, we have modified the lens to show a proper pupil definition. What you will see is the second one down.)



Rays from every field point fill the aperture of surface 7, which is declared the stop. When you tell the program to trace a ray, it first has to know where to aim that ray so it hits the stop at the desired point. For example, the ray at HBAR = 1 and YEN = 1 (the full-field marginal ray) should hit surface 7 at the edge of the aperture. How does it know where to aim? That is the whole issue with pupil definitions.

The two definitions most often used are the **paraxial** and **real** pupil. First, let us look at what we get using the simple paraxial pupil:

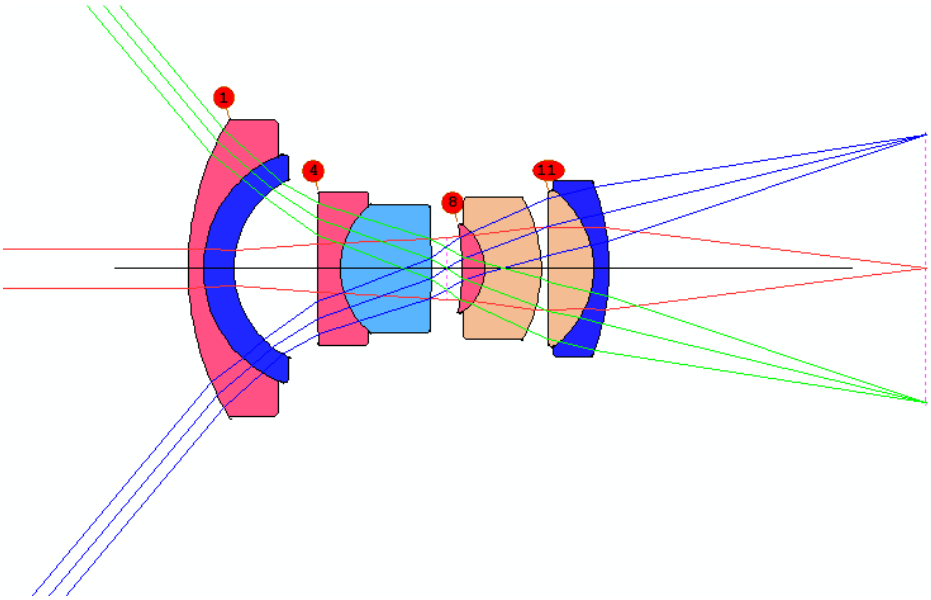


This pupil definition can be declared in the lens file with the input

```
RLE
…
APS  7
…
END
```

But you'll notice there are two problems with this definition: the chief ray does not go through the center of surface 7, and the marginal rays do not fill the aperture of that surface. Let us fix these problems in turn. First, we declare surface 7 to be a *real* stop, with

```
CHG
APS  –7
END
```

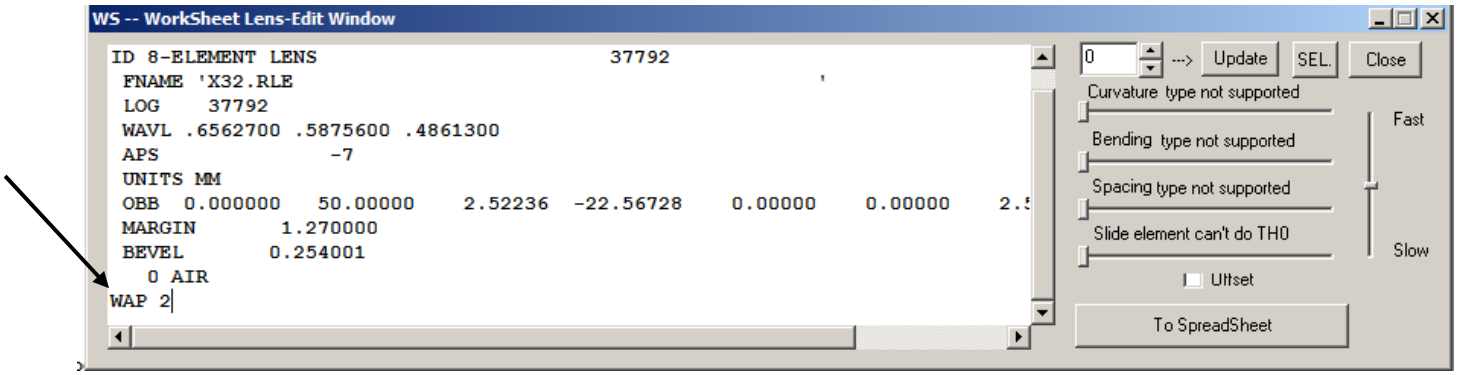


The minus sign indicates that this is a *real* stop, and the chief ray must be found by iteration. This activates ray aiming for the chief ray only.

Now the chief ray is okay, but the marginal rays are not. We need another common declaration, which will adjust the size of the pupil so the stop is nicely filled. This is the **WAP 2** option (there are three Wide-Angle-Pupil (WAP) possibilities, which you can read about in the User's Manual). It finds the shape of the entrance pupil by iterating a few rays at the edge of the stop. But this option requires a hard aperture on the stop surface so it knows where to aim. Let us assume for the moment that the aperture is not defined. You can do a CAP listing – to see the values of all current apertures – and then assign a "hard aperture" to surface 7. The value turns out to be 3.9937, so we could enter that value either in a CHG file or with the WorkSheet. Here's how to do it with a CHG file:

```
CHG
7 CAO 3.9937
END
```

An easier way is simply to type, in a CHG file or in the WorkSheet edit pane, **7 CFIX**. That fixes the current value, whatever is at the moment, so you don't need to type it yourself. Now change to WAP 2, again with the Worksheet…



and click the Update button. You get the lens as shown in the first picture, above. Now both the chief ray and the marginal rays go to the right place on surface 7. Here we have turned on ray aiming for a total of five rays.

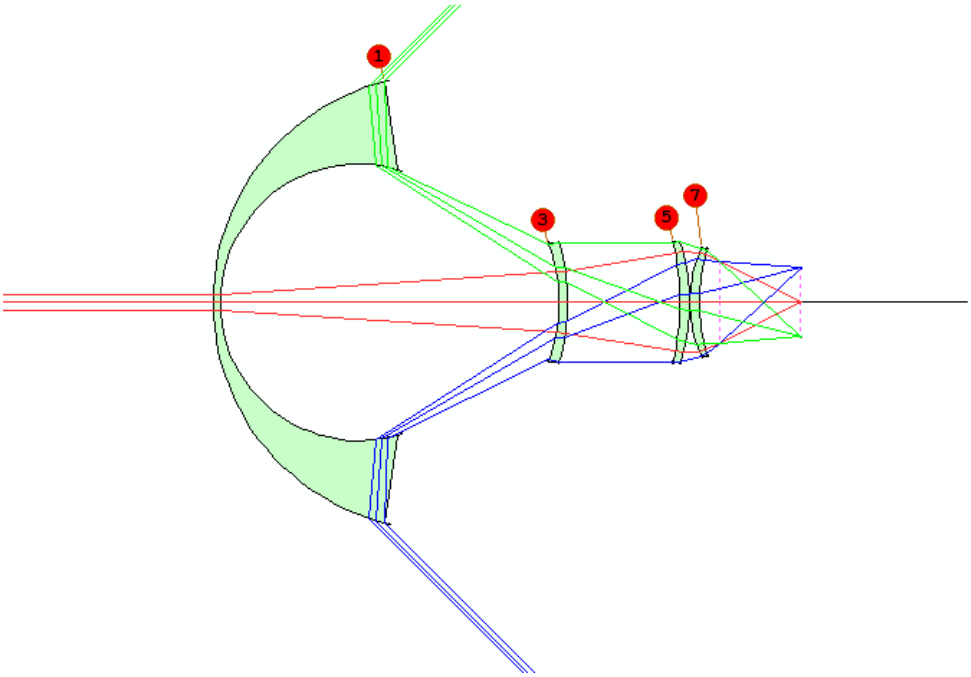
So far, this is not too complicated, and many users will not need anything else. But suppose you are optimizing the lens and the required aperture on surface 7 keeps changing. In that case, the hard aperture we assigned will be incorrect almost immediately.

No problem. We assign an option to recalculate that aperture every time the lens is changed. This is done by adding the directive **CSTOP** to the lens input file. Now the program will alter the CAO on 7 so it always equals the paraxial marginal ray height there.

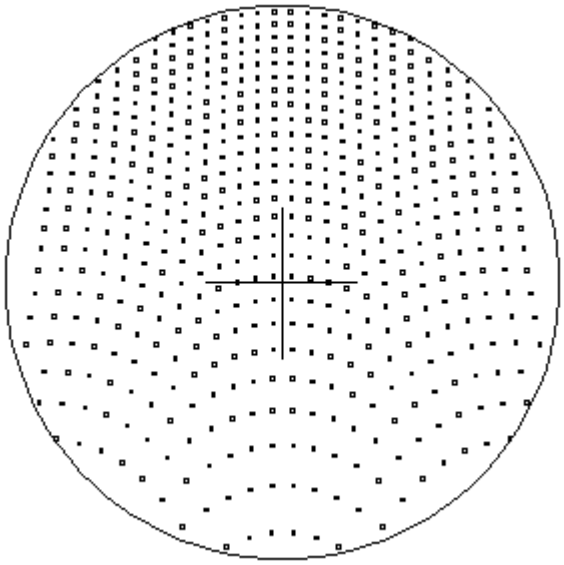
And if the lens has pupil aberrations so large that a *real* axial marginal ray requires a different aperture than the *paraxial* ray, change this to **CSTOP REAL**. You can even specify *which* real ray is to define this aperture, explained in the UM.

But what is the point of all these options? Isn't it easier just to deal with the kind of "ray aiming", that the other codes do?

Yes, it's easier – but it is much slower. As usually implemented, when those programs trace a grid of rays for any kind of image analysis, they create a *square grid* at the stop and then iterate *every ray* so it goes through that grid point. All that iteration takes time. Lots of time. Here is an example of a very wide-angle design.

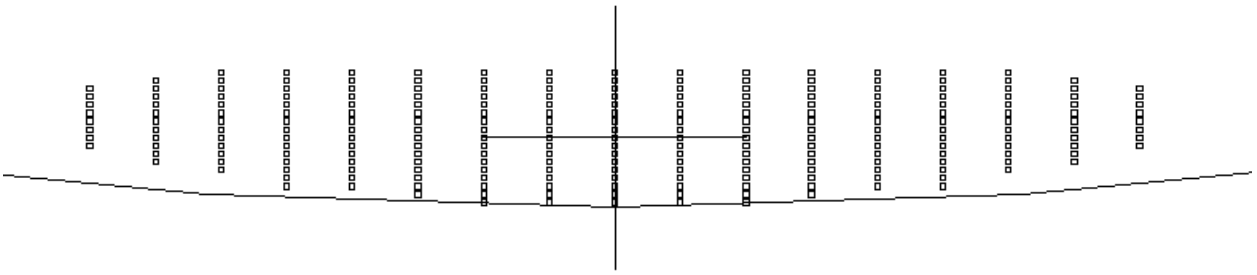


The stop is on surface 9, and it is nicely filled by the WAP 2 option. Let's look at a footprint on that surface that shows rays from the full-field point:

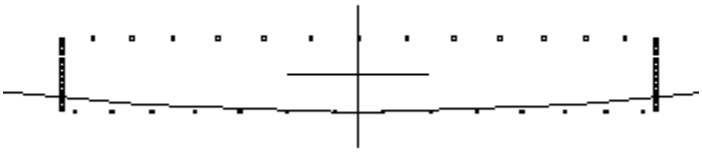


Wow! This is certainly *not* a uniform square grid! Those programs that employ "ray aiming" fill this aperture with the wrong distribution, and then, to account for this wrong distribution, alter the effective energy of each ray according to the actual ray density at that point. While this can indeed produce a correct evaluation of the image, one has to ask why they spend so much time with all the ray iterations in the first place.

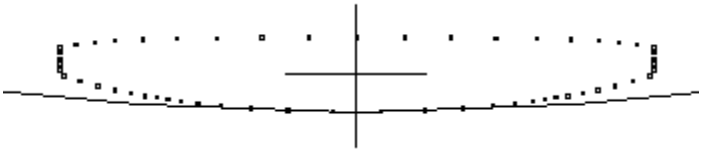
Instead, SYNOPSYS finds the size and shape of the *entrance pupil*, and then fills *that* with a uniform grid. Here is that pupil, on surface 1, for the above lens. It is regular, as it should be:



The pupil option in SYNOPSYS models the *outline* of this distribution, so a regular grid can fill it as it should. It is not necessary to iterate each ray, so it is much faster, and the distribution at the stop is correctly modeled. For this extreme example, a simple outline is not quite good enough, however (normally it is modeled by an elliptical shape). A better pupil, in this case, is found by declaring RPUPIL in the lens file. Now it starts with a rectangle that encloses that ellipse, and deletes any rays that fall outside the stop aperture. Here is that shape, as it enters the lens:



And here is what gets through:



We like this better than the slow “ray aiming” used in other codes.

Don’t forget to check out the dialogs **MPW** (Menu, Pupil Wizard) and **MOW** (Menu, Object Wizard), where you can define the kind of pupil you need by checking boxes and selecting from various options. Both of these dialogs do much the same thing, but they are organized differently; you can choose which you like best. We’ve made all this power as easy to use as we can.

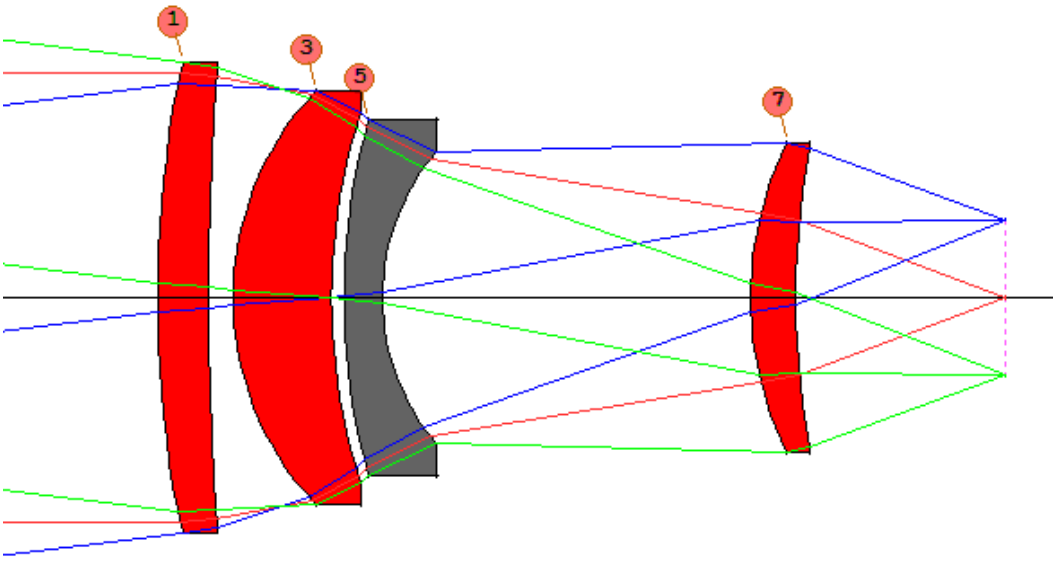
Which Way is Up?

The unique pupil definition in SYNOPSYS offers an interesting possibility – which is handy but takes some getting used to.

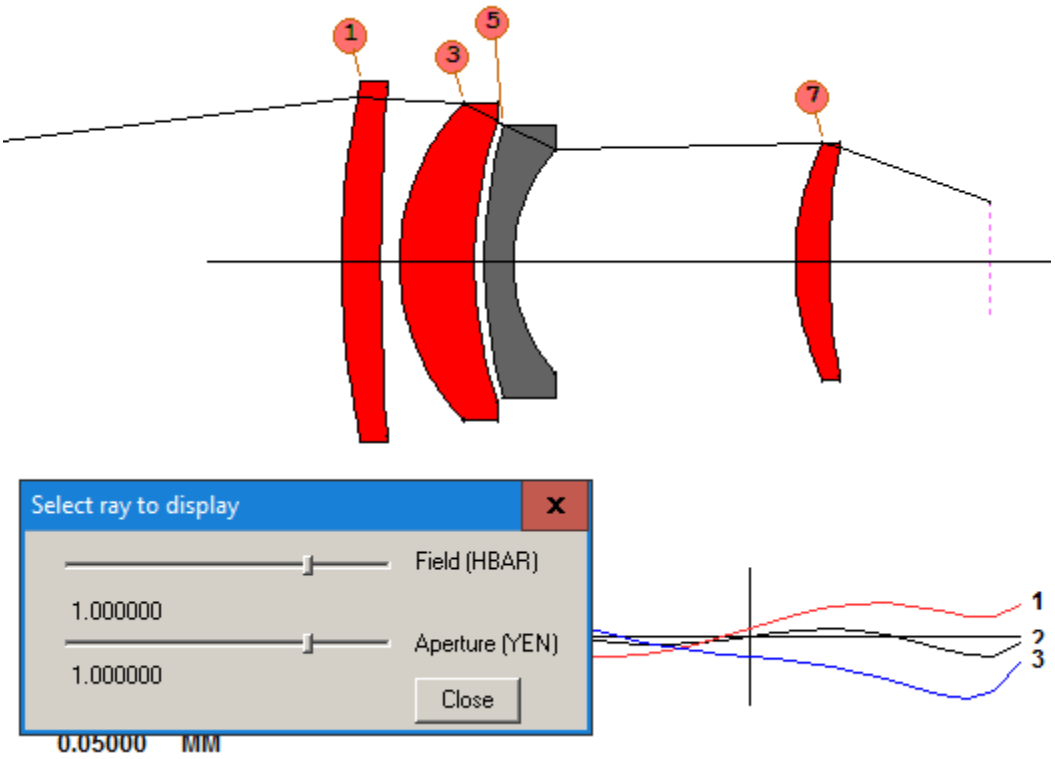
Let us illustrate. First we will show you some rays that don’t go where you expect them too, and then we’ll describe a simple way to keep things straight. Fetch the lens bundled as 1.RLE.

```
FETCH 1
```

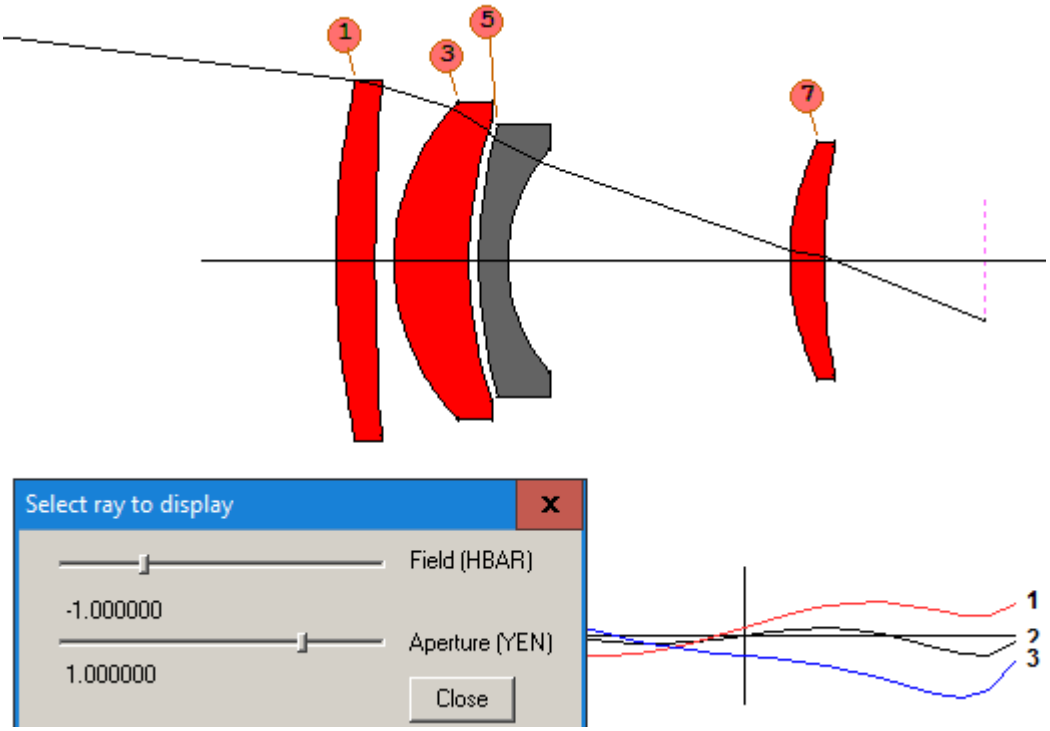
Now look at it in PAD



The lens at the moment has been assigned a *paraxial* stop on surface 4. In PAD, click the PAD Top button  , and select the option to draw a single ray. Click OK, and a small box opens where you can select which ray to draw, with two sliders. Move the top slider to full field (HBAR = 1) and the bottom to full aperture (YEN = 1). This object has been defined with a positive angle coming in, which means that the “full field” rays start from an object *below* the axis.



You see the full-field marginal ray, as expected. Now move the top slider to the bottom of the field (HBAR = -1).



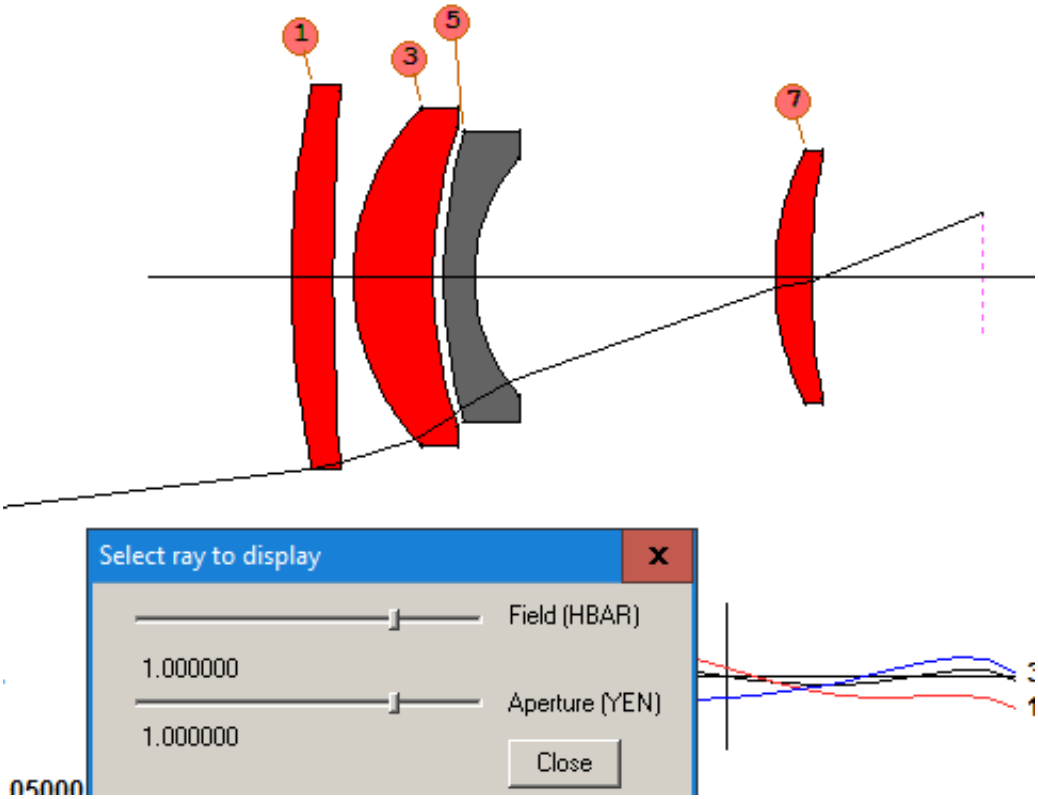
Again, the ray enters at the top of the pupil. This is the basic idea of the paraxial pupil. Simple.

But not always adequate. Close the ray display dialog and change the stop designation to

APS -4

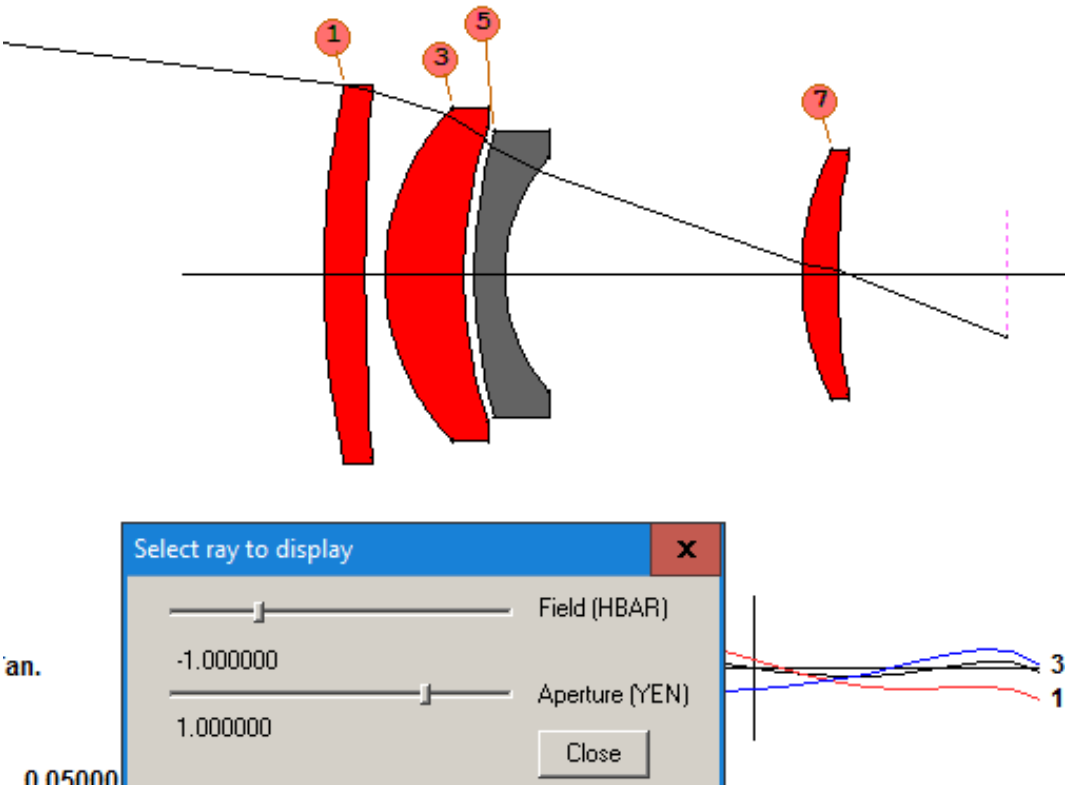
in the WorkSheet. Remember, the full-field object is located at a *negative* Y-coordinate, far to the left of the lens.

Okay, it's time to get interesting. Now open the single-ray dialog again, and set it to full aperture and full field once more.



Wow! What happened? The "full aperture" ray is now at the *bottom* of the pupil. What's going on?

Simple. This feature is designed to make it easy to correct feathered edges, no matter where in the field you are looking. In the picture above, you would correct along the "upper" rim ray (the ray shown) if feathering were a problem. Now go to the lower field point, HBAR = -1.



Aha! The ray to correct is *still* the upper rim ray! The program *rotates* the entire entrance pupil according to the orientation of the field point from which you are tracing. If you traced a point in the skew field, the "upper marginal ray" would turn out to be the extreme *skew* ray, so again you can easily control feathered edges. If we left the definition of the upper and lower marginal rays the same for all field points (as it is with a paraxial pupil), it would not be so easy to do; you'd have to figure out which skew ray to fix and then create an aberration for it. This is much simpler – once you get used to it.

So how can you easily figure out which ray to examine or correct for feathered edges? Simple. While the PAD display is open, press the **F7** key. Only the "lower" rim ray shows up. The **F8** key shows only the "upper". Presto! You can tell which ray is where with a single keystroke!

There is another advantage to this kind of pupil definition: The entrance pupil is usually modeled as an ellipse, as illustrated in the first part of this lesson, and it turns out that the ellipse also rotates with field point. So it can model the vignetted pupil at all points in the field. Neat.

Take a look at Section 2.6.2 in the User's Manual for an example of a rotated pupil.

Oh, and one more thing: The program decides which ray to call the "upper" ray based on the sign of the height of the full-field object. Since that was negative in this example, it flipped the marginal rays for positive HBAR. At negative HBAR, the object comes from a positive Y-coordinate, and the rays are not flipped.

And at HBAR = 0? Well, to avoid confusion from being neither positive or negative, the program displays a very small but nonzero field point there.

Confused? Try the F7 and F8 keys. Simple.