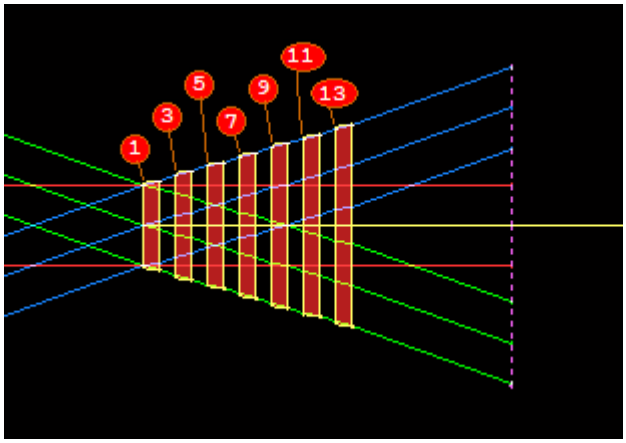
Lesson 7: The Optimization Challenge

This lesson explains in depth what SYNOPSYS™ does when running the Optimization Challenge. If you want to run it yourself, you will find the input in aMACro called S-Z_TEST.MAC. (You can load this into the MACro editor by typing MWM (Menu Window MACros) and selecting that name from the list). It starts with this very bad design:



and contains the following lines:

RLE ! Read Lens; the start of the input lens file.

ID TEST PSD III ! Optional identification.

OBB 0 20 12.7 ! One of several object selections: infinite conjugate, 20-degrees semi-field, 12.7 mm semi-aperture.

WAVL CDF ! Visible wavelengths: C, D, and f Fraunhofer lines.

```
UNITS MM                          ! Lens is in millimeters.
1 TH 5 GLM 1.6 50                 ! Surface 1 is flat (since no radius or curvature is
entered), has a thickness of 5 mm and a glass model in the middle of the glass
map.
2 TH 5                            ! Surface 2 is also flat, with an
airspace of 5 mm.
3 TH 5 GLM 1.6 50                 ! And so on, for 14 surfaces.
4 TH 5
5 TH 5 GLM 1.6 50
6 TH 5
7 TH 5 GLM 1.6 50
8 TH 5
9 TH 5 GLM 1.6 50
10 TH 5
11 TH 5 GLM 1.6 50
12 TH 5
13 TH 5 GLM 1.6 50
14 TH 50
15                                ! The image plane is surface 15.
APS 1                             ! A paraxial stop is assigned to
surface 1.
END                               ! This ends the lens input file.
```

```
PAD/U                                   ! I put this in so you could see this
starting lens on the screen.

! PRESS THE "ENTER" KEY TO OPTIMIZE THIS LENS          ! This is a
comment that shows up on the monitor.
NEW
                                        ! This pauses the MACro so you can examine
the lens picture; pressing the <Enter> key resumes it.
PROJECT                                 ! This command starts a timer so you
can see how long the run took.

QUIET                                   ! If run in quiet mode, there is
nothing scrolling on the screen and it executes a little faster.

PANT
                    ! Start of parameter definition.
VY                                                              1
YP1
          ! The paraxial chief-ray height on surface 1 is a variable.  This finds the
best stop position.
VLIST RAD 1 2 3 4 5 6 7 8 9 10 11 12 13 14          ! All radii are
```

variables except for the image plane.

VLIST TH ALL EXCEPT 14 !

All thicknesses (and airspaces) are variables.

VLIST GLM

ALL ! All

glass types are variables.

END

! This ends the parameter definitions.

AANT ! This starts the aberration list,

which defines the merit function.

AEC ! Watch edges of all lenses whose

thickness is a variable; default target is 1 mm minimum edge thickness.

ACC ! Watch all varying center

thicknesses so they don't get over 1 inch.

M 33 2 A GIHT ! Target the Gaussian image height to a

value of 33 with a weight of 2.

GSR 0 10 5 P 0 ! Create a fan of sagittal rays, weight 10, 5

rays per SFAN, primary color (the D line).

GSR 0 10 5 1 0 ! Do the same in color 1 (the C line).

GSR 0 10 5 3 0 ! And again in color 3 (the f line).

GNR 0 2 3 P .7 ! Generate a grid of rays at 0.7 field in color P,

from a 6x6 grid (+/- 3 rays).

GNR 0 2 3 1 .7 ! The same, color 1.

GNR 0 2 3 3 .7 ! And again, color 3.

GNR 0 2 3 P 1 ! More rays, this time at full field.

GNR 0 2 3 1 1

GNR 0 2 3 3 1

END ! Okay, that's the entire merit function! Short and simple, isn't it?

SNAP 100 ! Watch the lens change – but only every 100 iterations, so it runs faster. Change to a smaller number to see more updates.

SYNOPSYS 300 ! And run for 300 iterations. Yes, it's a huge number of passes, but this is a horrible starting point!

LOUD ! Now turn on the monitor output again.

RMS M 0 600 ! Here we use the Artificial Intelligence feature to calculate the score; find the RMS spot size on axis.

Z1 = FILE 1 ! Save that value in variable Z1 (There are 9 Z-variables for use by AI).

RMS M .5 600 ! Get the RMS at 0.5 field.

Z2 = FILE 1 ! Save that in Z2.

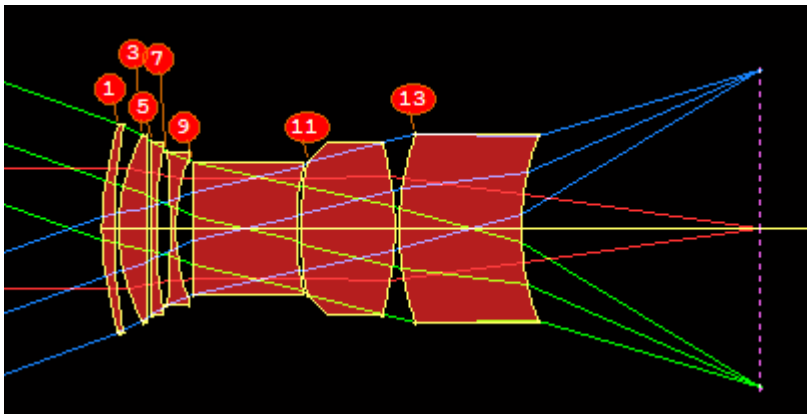
RMS M 1 600 ! And now do it at full field.

Z3 = FILE 1 ! Save in Z3.

= (Z1 + Z2 + Z3)/3.0 ! Here we do some math, finding the average RMS spot size at three field points in three colors. That's our score.

PROJECT ! End up with another PROJECT command, to show the elapsed time.

So that's the input. When we run this MACro, we get the following lens after 3.434 seconds:




```
...
--- = (Z1 + Z2 + Z3)/3.0


 The composite value is      0.00677814
 ---
 --- PROJECT
 FINISH: 163706.499 20121206
```

ELAPSED TIME: 3.434 SECONDS

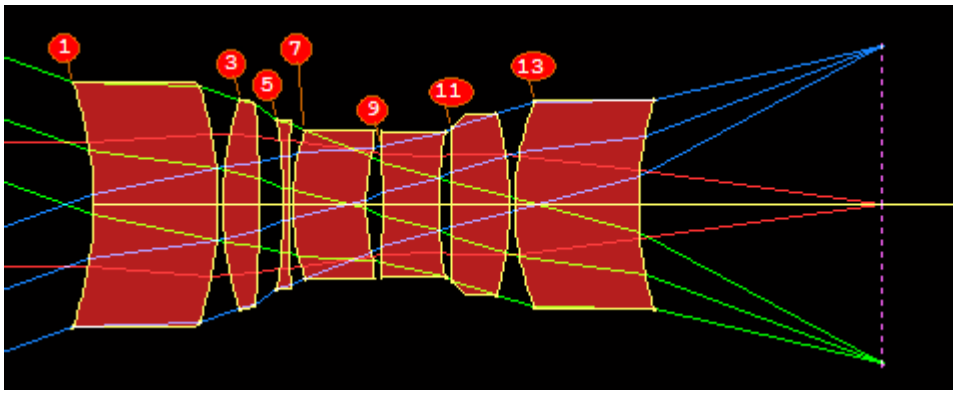
This was run on a 3.1 GH PC, and according to the challenge, the time limit was 6.45 seconds (20/CPU speed). So we beat that limit very nicely. The average blur size is only 6.78 microns. We got that result in less than four seconds, starting with flat surfaces! Now let's try a few things. The second parameter on the ray-generation directives currently is 0. This applies an aperture-dependent weighting on each ray, and if we increase the value to, say 0.5, then rays near the center of the pupil will be weighted more heavily than rays at the edge. Edit the MACro so the merit function looks like this:

```
AANT
AEC
ACC
M 33 2 A GIHT
GSR .5 10 5 P 0
GSR .5 10 5 1 0
GSR .5 10 5 3 0
GNR .5 2 3 P .7
GNR .5 2 3 1 .7
GNR .5 2 3 3 .7
GNR .5 2 3 P 1
GNR .5 2 3 1 1
```

GNR .5 2 3 3 1

END

If you then run it again, you get a very different lens:



and the score is slightly higher, at 0.007509 mm. It's still a very good lens, but this exercise shows an important insight: When you start with flat surfaces, the lens can go anywhere – and a very slight change in the starting point or requirements can send it down a different path. This is the principle behind the design search feature, DSEARCH, which you can read about in the User's Manual. (Type HELP DSEARCH in the Command Window to open that section.) You can find many great configurations with this tool.

Let us point out some of the ways we believe SYNOPSYS™ handles this kind of job better than any other optics code on the market:

1. The lens file is just a lens file. The merit function goes into a separate file, which we like much better.

2. The variables are also declared in the separate file. This is neat and clean.

3. The merit function definition is short and sweet. It creates a number of rays – but you don't have to know about them, see them, or edit them yourself. This is very friendly. Why look at 32000 ray definitions? That is a blunder of Zemax, in our opinion.

4. The program can find the best stop location simply by using the variable called YP1. Doing this with another program is more complicated.

5. The glass model is a very effective variable. This is a huge advantage over Zemax, since SYNOPSYSS™ can move glass parameters anywhere on the glass map until they hit a boundary somewhere. Zemax does not employ glass boundaries, and varying glass properties does not work very well. That is a big reason why it does so poorly on this challenge.

6. SYNOPSYS™ has no trouble starting out with all-flat lenses. No other code can do this as well, mostly because we use the PSD III algorithm, while other codes use a variety of DLS optimization. This is another huge difference.

When you combine different kinds of variables, such as index and curvatures, the metric characteristics are very different. PSD III treats this situation very well; DLS cannot and never will. Unless you start out close to a good design, DLS takes a very long time because it moves so slowly. That is the main reason why no other code can do this simple problem. We are waiting for them to report their scores, so we can compare them to ours. We're still waiting.

Just for fun, change the SNAPSHOT parameter from 100 to 1, and run the job again. Now you see the lens changing, looking much like some kind of life-form whose shape morphs, grows, and shrinks. And all of this is shown on the PAD display with a sophisticated rendering algorithm that looks like a very smooth movie, without the breaks and stuttering you find in Zemax. That stuttering really bothers us, when we witness it, and after you've watched SYNOPSYS™ optimize a lens you'll know why. The smooth rendering is so much nicer to watch.

So that's a taste of how SYNOPSYS does the Optimization Challenge. No other lens design code in the world can work this problem this well. If you do not agree, please send us the timing and score of your current program. We're waiting for your response. Still waiting. Still waiting….