WaveTrain

wave optics made ~~easy~~ easier

Version 2007B

Hands-On Workshop

Extended Version with the BLAT (V1.0) Model and Other Demonstrations

Bob Praus
Steve Coy
Keith Beardmore

MZA Associates Corporation
2021 Girard SE, Suite 150
Albuquerque, NM 87106
Voice: (505) 245-9970
Fax: (505) 245-9971
praus@mza.com

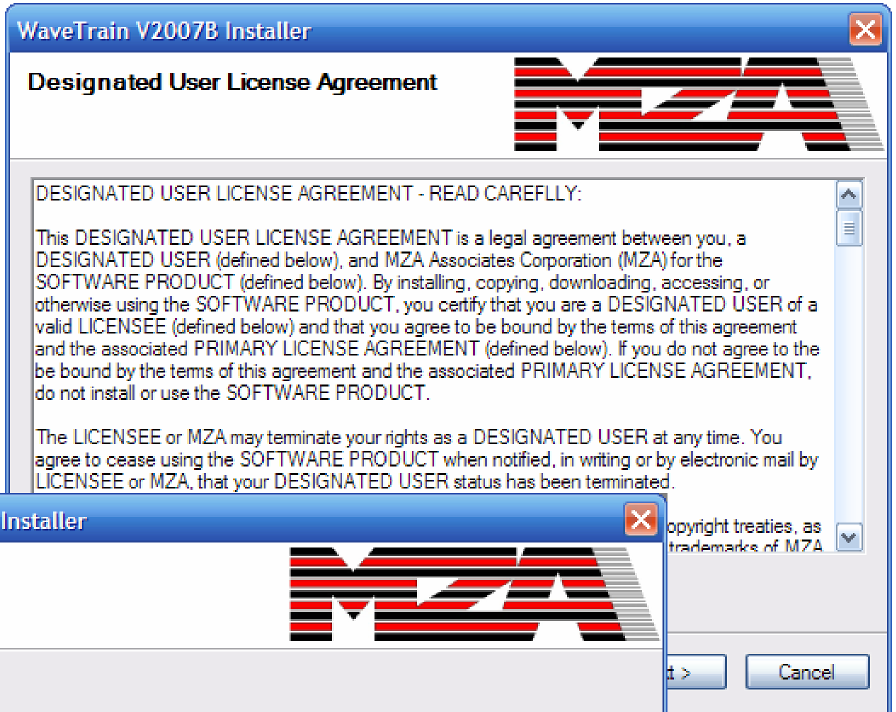
Workshop Program

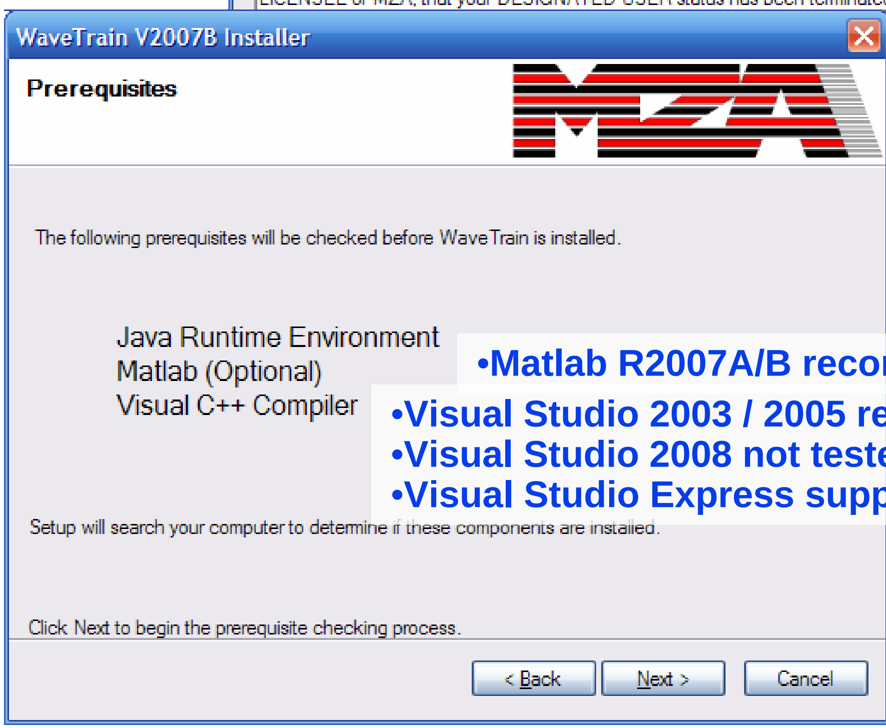
A Reasonable Schedule for a One Day Course

• Prologue	WaveTrain Installation	0800 - 0830
• Block 1	Introduction to WaveTrain (separate chart package)	0830 - 0930
	Break	0930 - 1000
• Block 2	Beginner's Workshop	1000 - 1200
	Break	1200 - 1300
• Block 3	Continuation of Beginner's Workshop	1300 - 1400
• Block 4	Introduction to Beam Control Simulation (separate chart package)	1400 - 1500
	Break	1500 - 1530
• Block 5	Demonstration of Beam Control Simulation & Independent Study	1530 - 1700

WaveTrain Installation

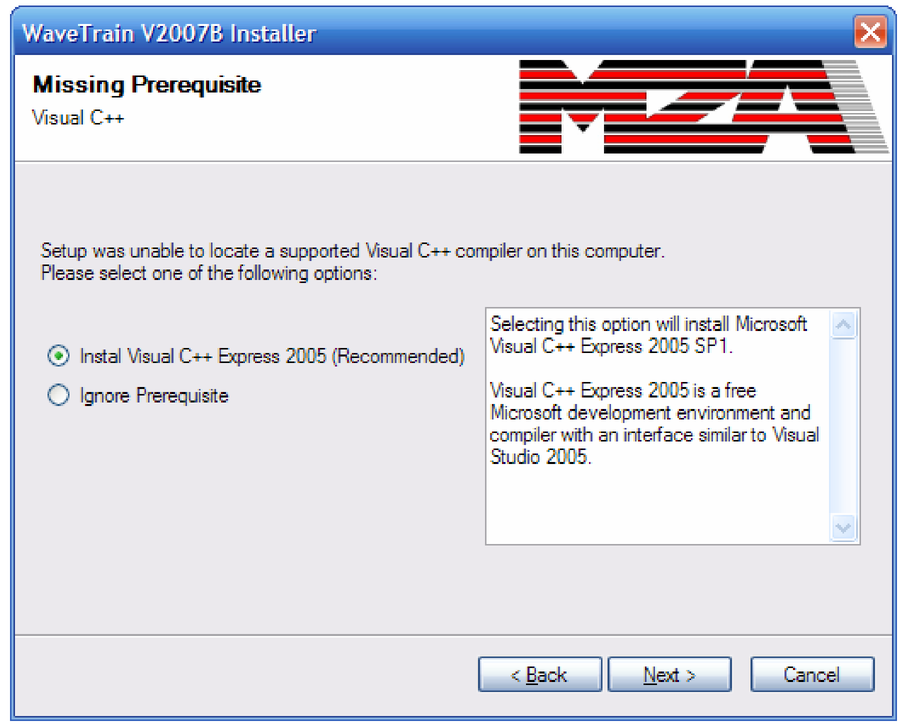
- Install Microsoft Visual C++ (optional)
- Install Matlab (optional)
- Install WaveTrain & tempus from DVD
- Visual Studio Express 2005 is installed if no C++ is found





•Matlab R2007A/B recommended

•Visual Studio 2003 / 2005 recommended
•Visual Studio 2008 not tested
•Visual Studio Express supported

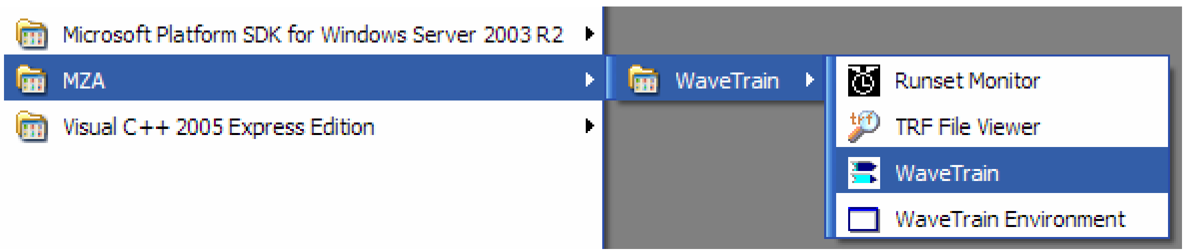


MZA Associates Corporation

WaveTrain Post-Installation

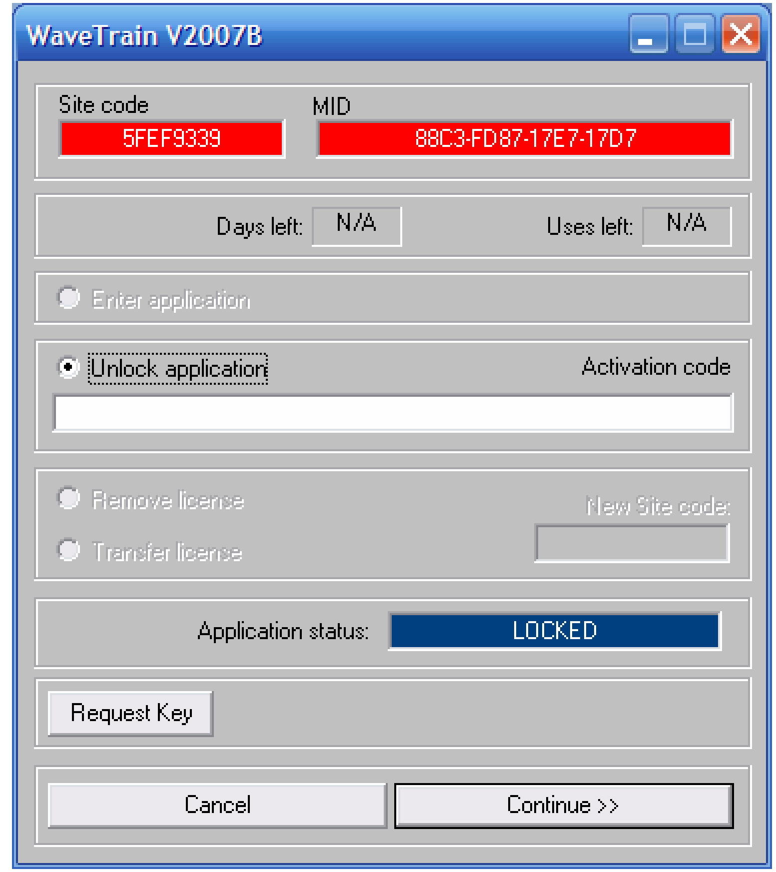
- You will have a desktop icon

- An MZA entry on the programs menu



- Other programs installed, such as Visual Studio, may also be listed on the programs menu

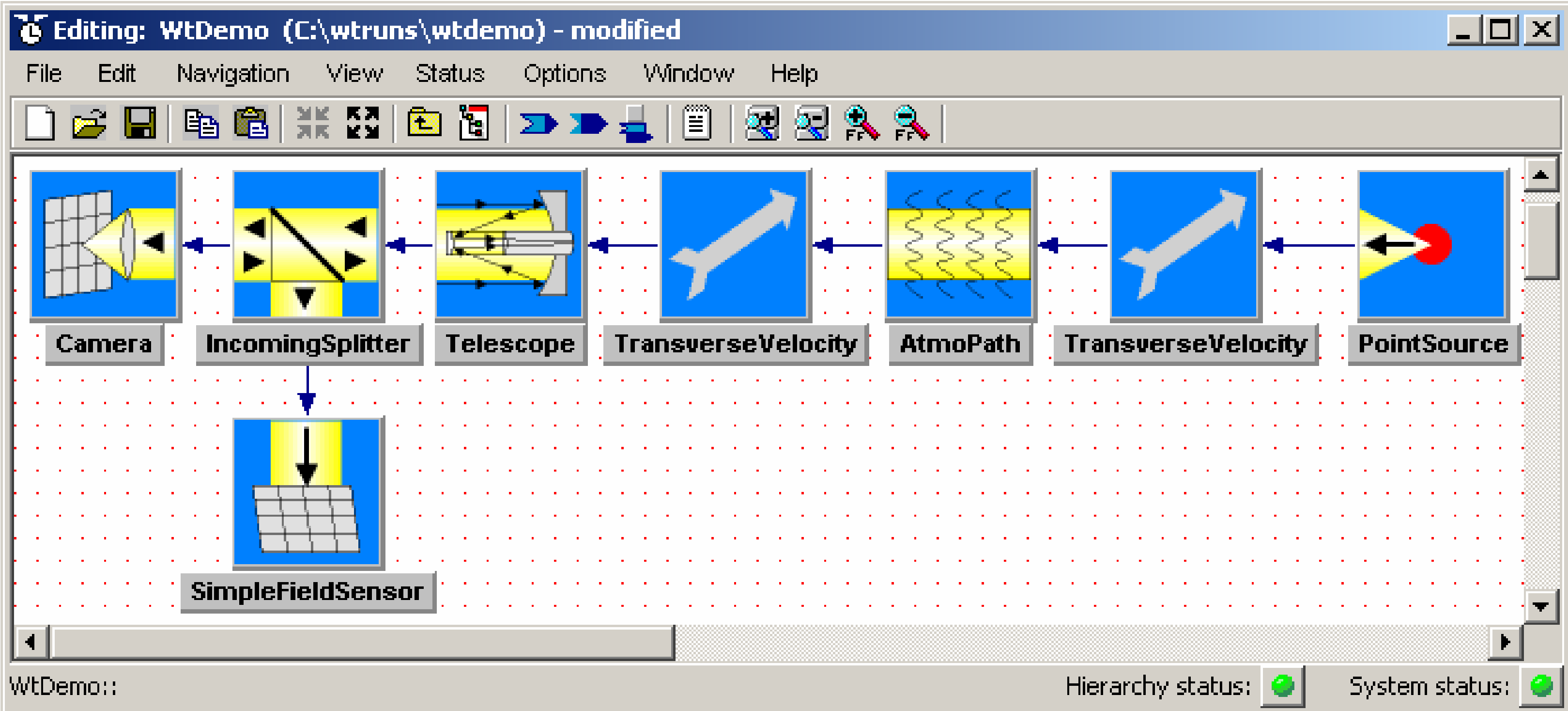
- If Matlab is present, the WaveTrain and tempus mfile paths will be added to your Matlab path



- On the first attempt to use WaveTrain, you will have to request a key to unlock WaveTrain (via email)

WaveTrain v2007B Beginner's Workshop

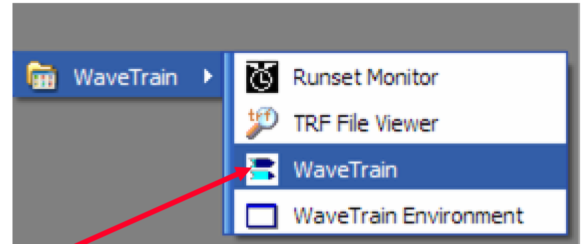
MZA Associates Corporation



- **In this workshop you will build a model of a telescope system imaging a point source through turbulence.**
- **You will then use the model to perform a simple parameter study, and look at the results.**
- **Model features:**
  - **Records amplitude and phase at the pupil plane, and intensity at the focal plane.**
  - **Models platform motion, source motion, and/or wind.**
  - **Uses standard turbulence models, e.g. Clear 1 or Hufnagel-Valley, and/or user-defined models.**
  - **All major system variables are parameterized, so they can be changed without changing the model itself.**

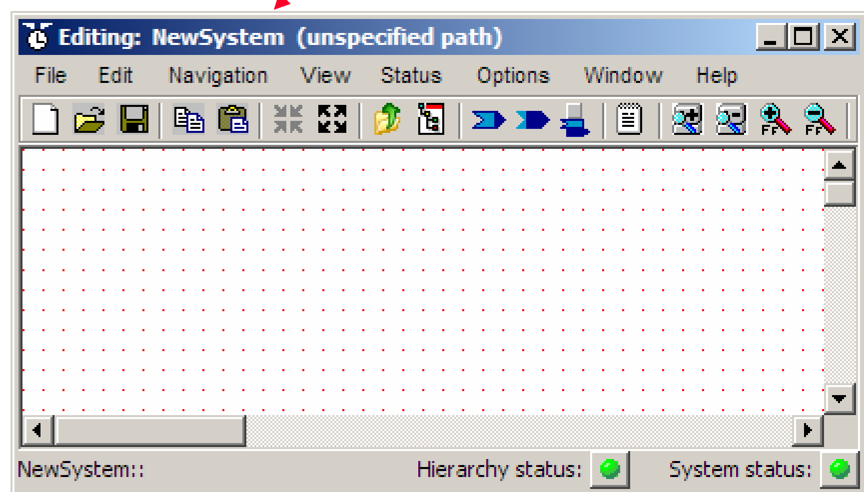
Create a New System Model

- WaveTrain is built atop tempus, a general-purpose simulation tool. In tempus, a system model is defined in terms of its interface (inputs, outputs, and parameters), its subsystems, and the connections between them. Each system model is mapped into a portable C++ class via automatic source code generation.



- To begin, start the GUI by selecting the WaveTrain desktop icon or MZA->WaveTrain under the Windows Start-Programs menu. This will bring up the tempus visual editor (TVE) top-level window.

- Click on  which will bring up the System Edit Window. When System Editor window comes up it already has a new system model, called "NewSystem", loaded by default.

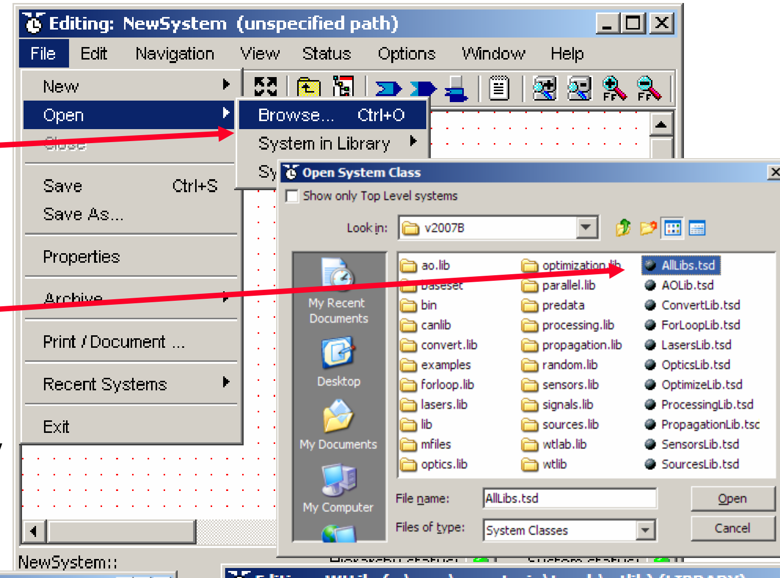


MZA Associates Corporation

Open the Component Library

- Go back to the top-level window, and click on  again, which will bring up a second System Editor Window.

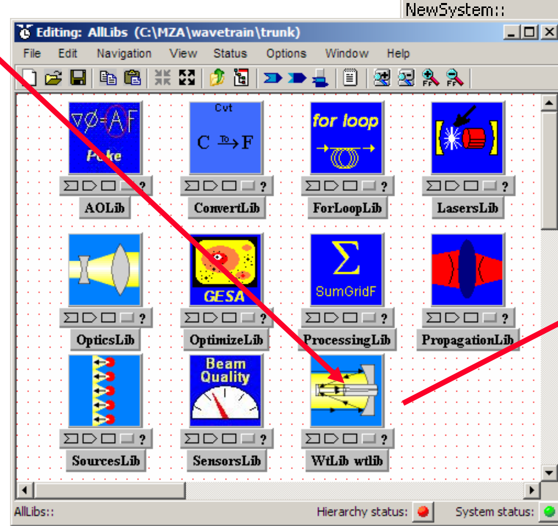
- Click on File->Open->Browse, which will bring up a file selection window.

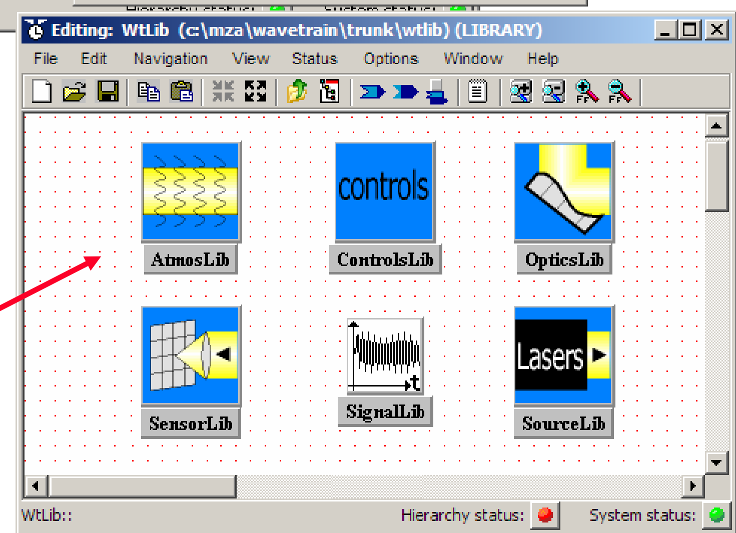


- Navigate to c:/Program Files/mza/wavetrain/v2007B and select AllLibs.tsd, the top level WaveTrain component library. Select it, then click "Open".

- Double click on WtLib, the primary component library

- You will see that WtLib contains six components, each of which is a more specialized sublibrary:
  - AtmosLib
  - ControlsLib
  - OpticsLib
  - SensorLib
  - SignalLib
  - SourceLib





MZA Associates Corporation

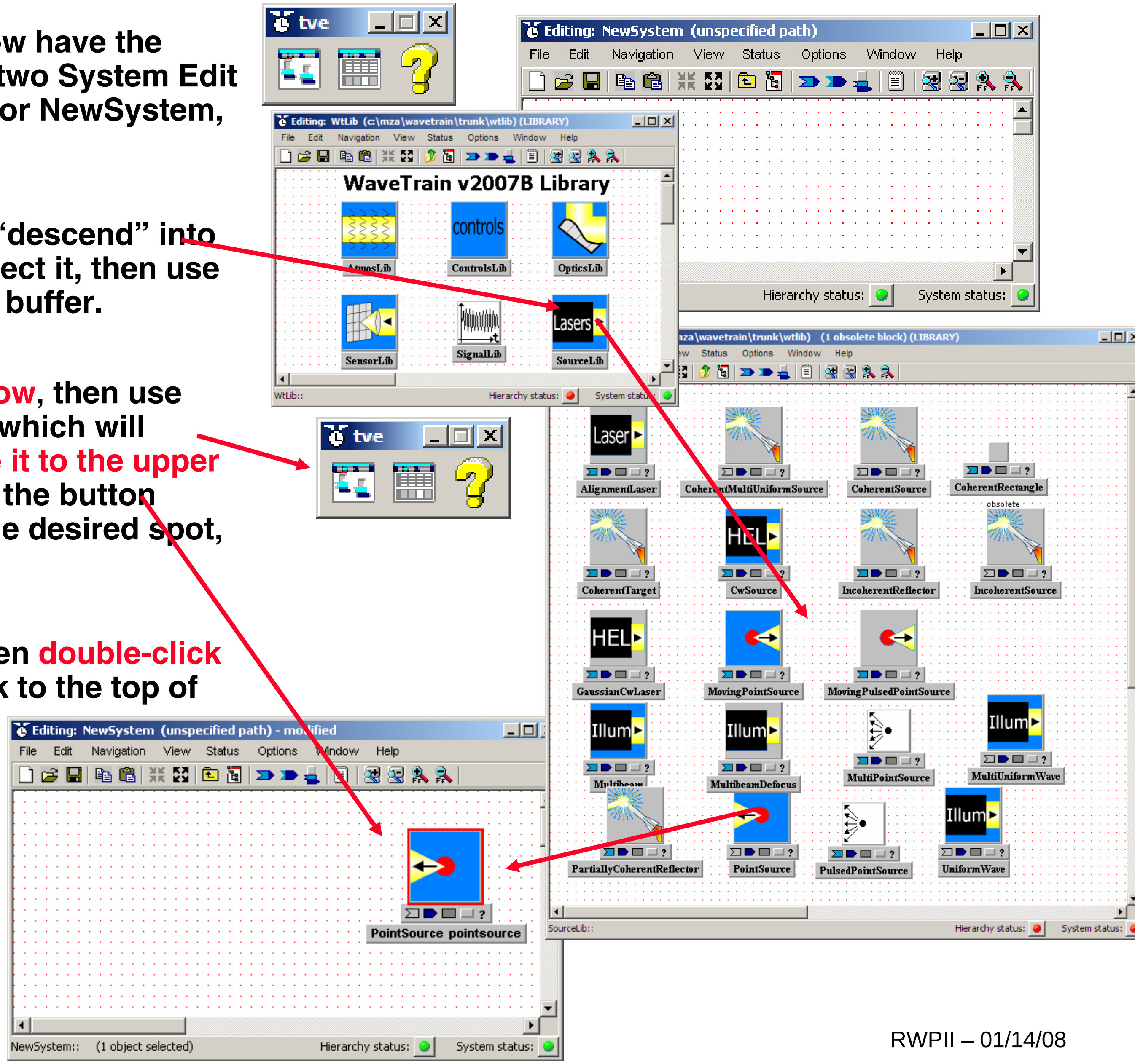
Copying a component from the library

- On your screen you should now have the tempus top-level window and two System Edit Windows, one for WtLib, one for NewSystem, as shown in the upper right.

- Double-click on SourceLib to “descend” into it. Click on PointSource to select it, then use Ctrl-C to copy it into the paste buffer.

- Click on the NewSystem window, then use Crtl-v to paste a PointSource, which will appear in the upper left. Move it to the upper right by clicking on it, holding the button down, moving the mouse to the desired spot, then releasing it.

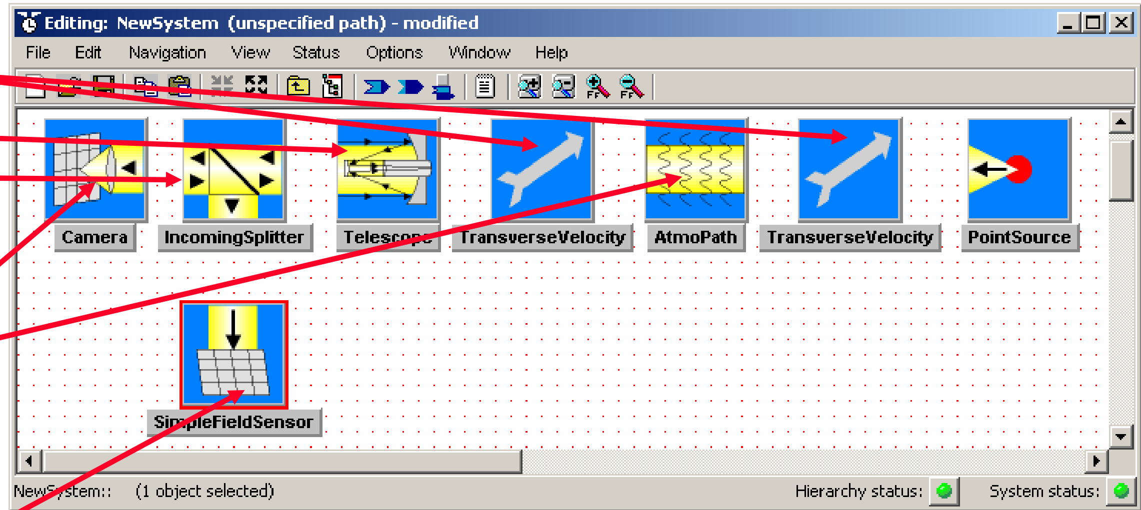
- Click on the WtLib window, then double-click on white space to ascend back to the top of the library.

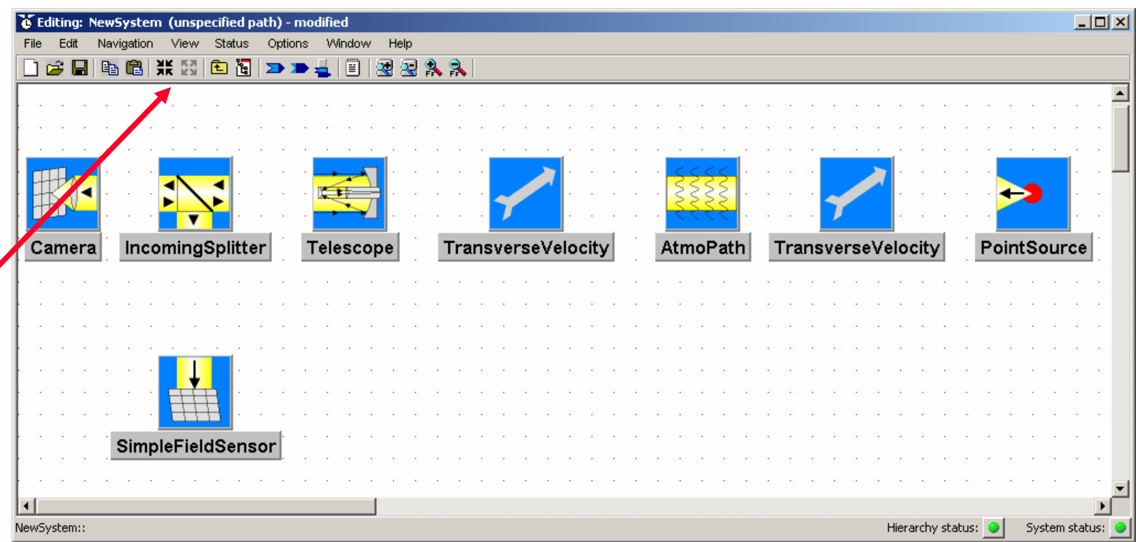


MZA Associates Corporation

Copy the rest of the components

- First, descend into OpticsLib, and get
  - two copies of TransverseVelocity
  - one Telescope
  - one IncomingSplitter
- Next, ascend back to the top of WtLib, then descend into AtmosLib, and get
  - one AtmoPath
- Finally, descend into SensorLib, and get
  - one Camera
  - one SimpleFieldSensor.
- Arrange the components as shown in the upper window (approximately).
- Click on the Expand button (four diverging arrows) at the top of the window, near the left; this will give you more room to make connections.

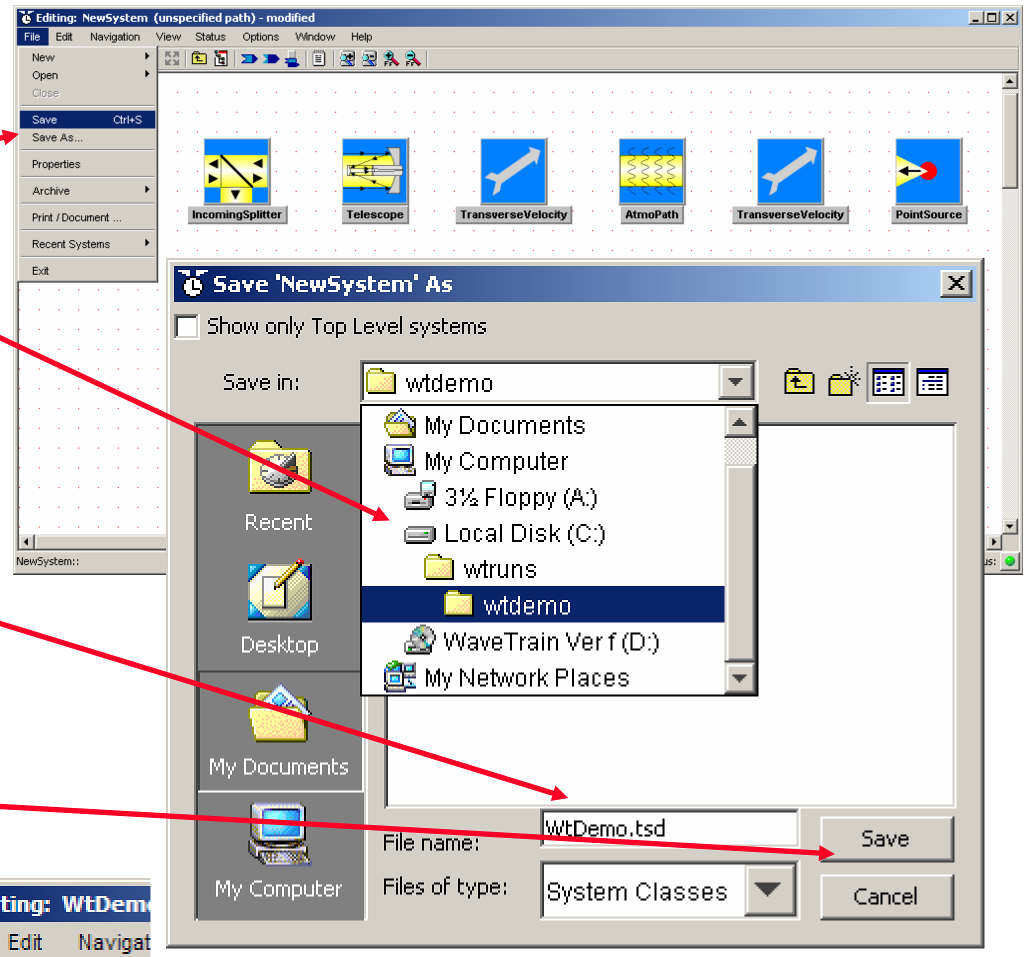


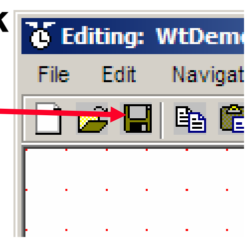


Save Your Work

As with all applications, it is a good idea to save your work on a regular basis so that if some sort of crash or mistake happens you can recall your work.

- **Click on File->Save As…, which will bring up the window shown at the bottom. Navigate to the directory c:\wtruns\wtdemo. The actual directory doesn't matter, but its better if you have a special directory for each WaveTrain model that you work with.**
- **Type in the filename WtDemo.tsd. The actual name doesn't matter, but we use a standard name to keep the tutorial the same for everyone.**
- **Click on Save.**
- **As you go along, you can save your work periodically by clicking on File->Save, or the disk icon.**



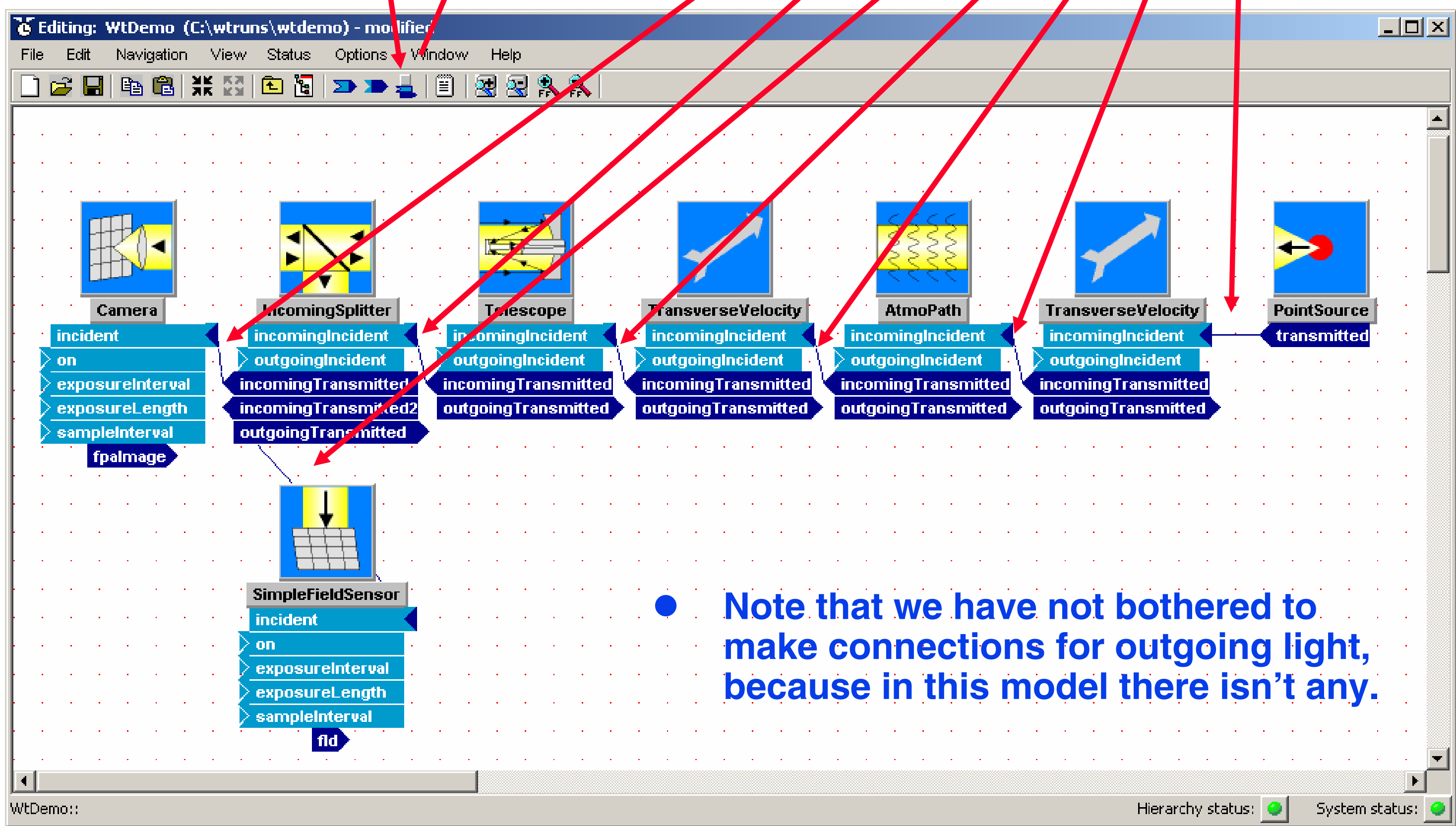


MZA Associates Corporation

Connect components

- **Click the toolbar button with image of the subsystem.** A small menu will pop up.
- **Select the button with a light blue "receptor" shape**, also shown depressed at right. This will display all subsystem inputs.
- **Select the button with a dark blue arrowhead**, shown depressed at right. This should cause all subsystem outputs (dark blue arrows attached to the bottom of each subsystem) to be displayed. If it does not work, click on white space and try again.
- You can reverse the orientation of an input/output by double clicking it

- **Connect outputs to inputs as shown**, by clicking on the pointed tip of each output, and dragging it to the receptor of the appropriate input.



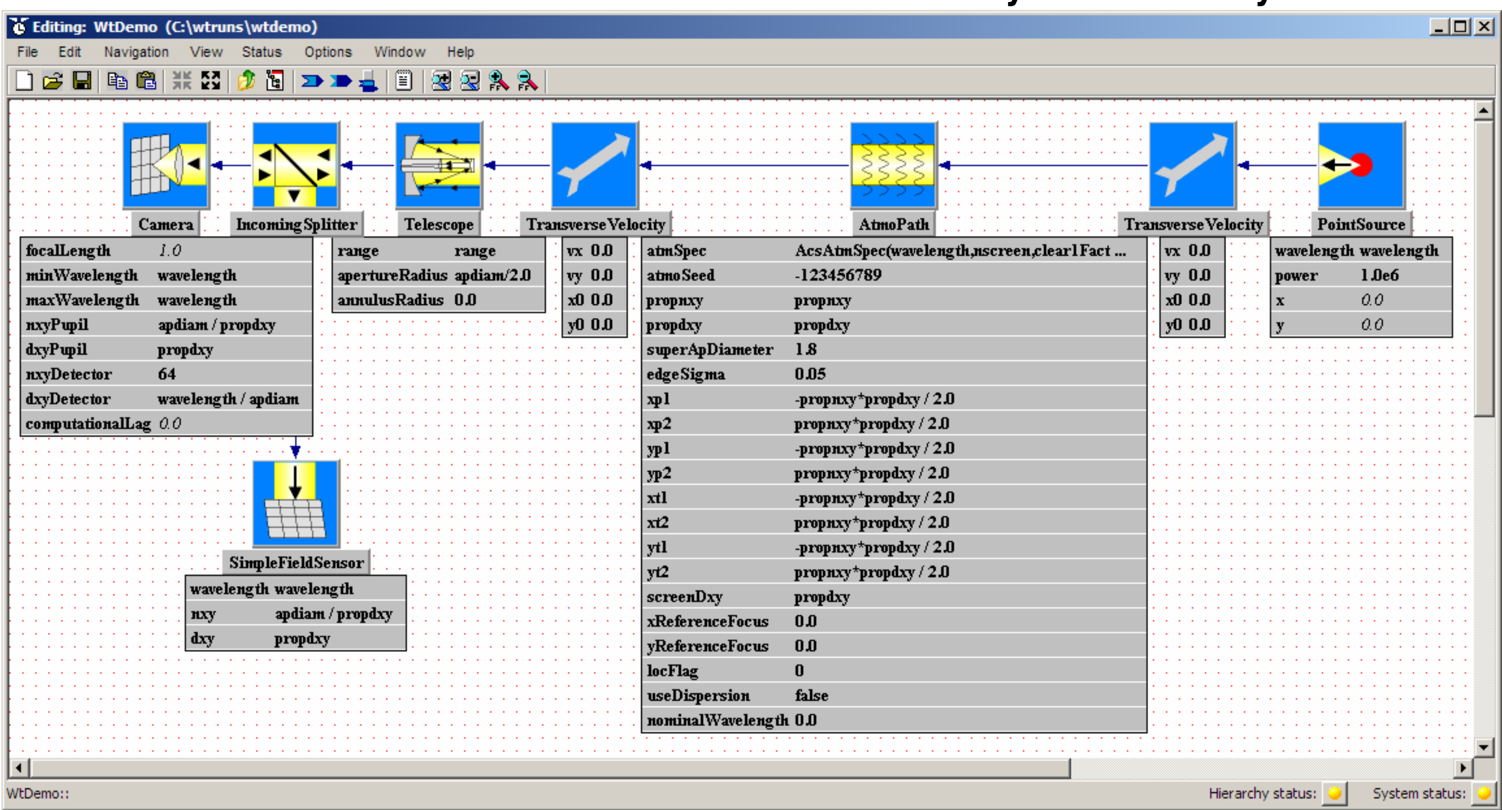
- Note that we have not bothered to make connections for outgoing light, because in this model there isn't any.

MZA Associates Corporation

Check Subsystem Parameters

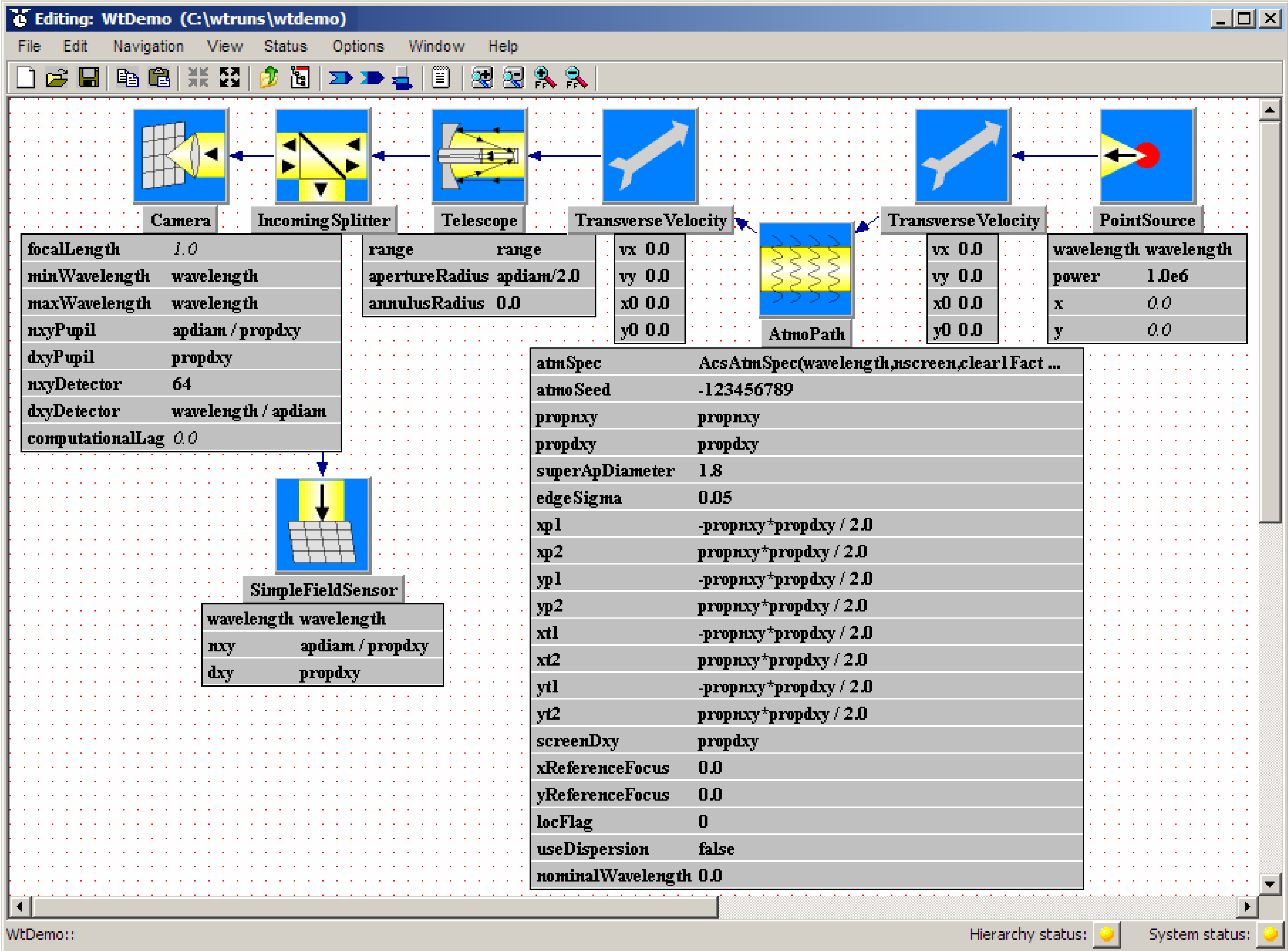
- **Undisplay** the subsystem inputs and outputs.
- **Click on the button with the medium gray rectangle** (lower left corner of the menu), which will display the subsystem parameters, as shown below.

- **For each parameter, the parameter name appears to the left, and its "setting expression" appears to the right, if any has been specified.**
- **Setting expressions are evaluated using the parameters of the containing system, but we have not yet defined any.**



Subsystem Parameter Values

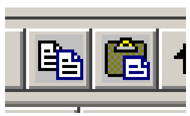
MZA Associates Corporation



MZA Associates Corporation

Add Needed System Parameters

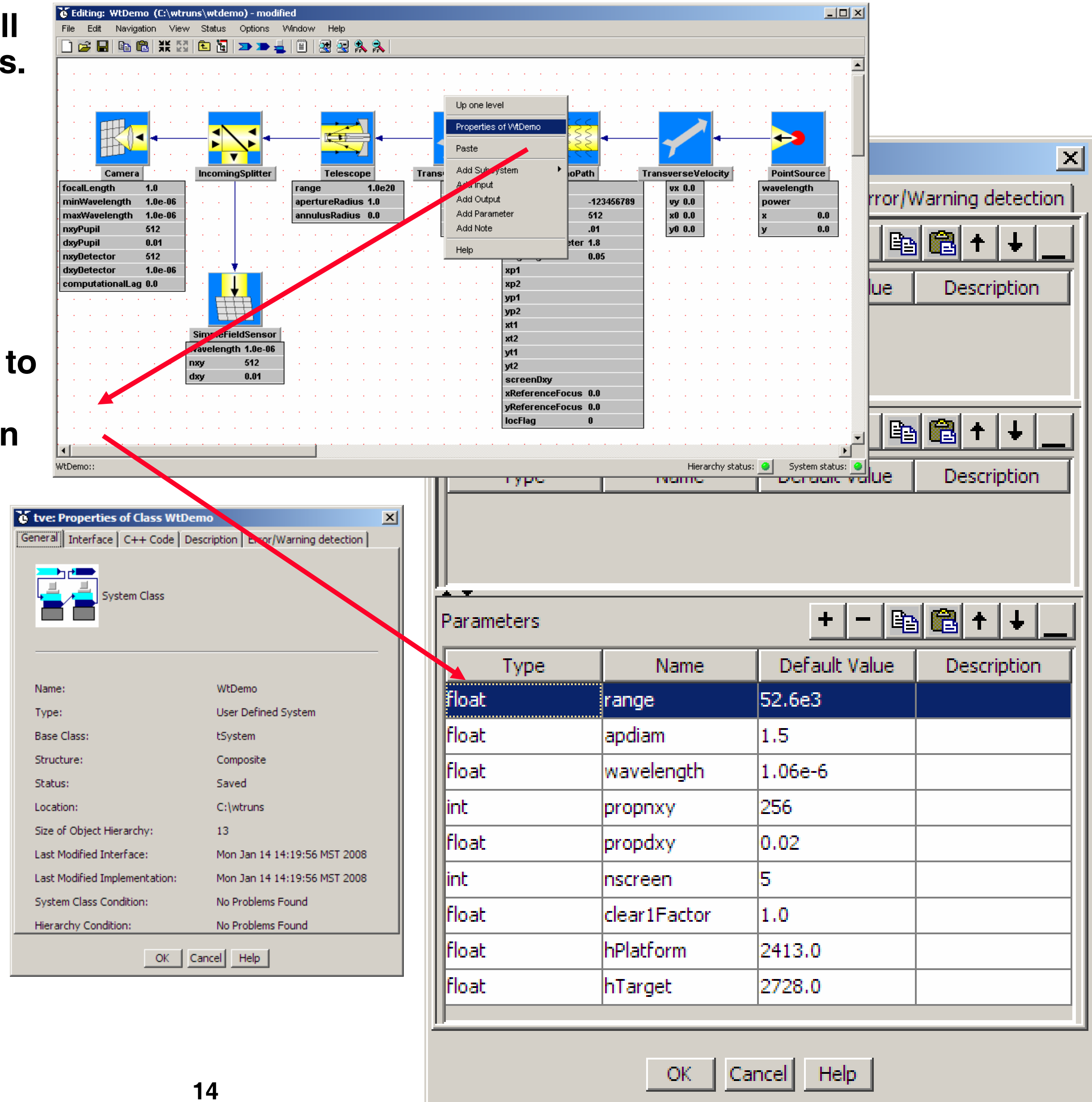
- **Right-click on white space, which will bring up a small window with options.**
- **Select the second one, Properties of WtDemo, which will bring up the window shown in the lower left.**
- **Click on the Interface tab. In the Parameters section, click on the "+" to create the first parameter & enter "float", "range", & "52.6e3" as shown at right.**
- **With the first parameter selected click the "copy" & "paste" buttons to create eight additional parameters, giving them the types, names, and default values shown.**



- **Click OK.**

WaveTrain and tempus names are case sensitive!

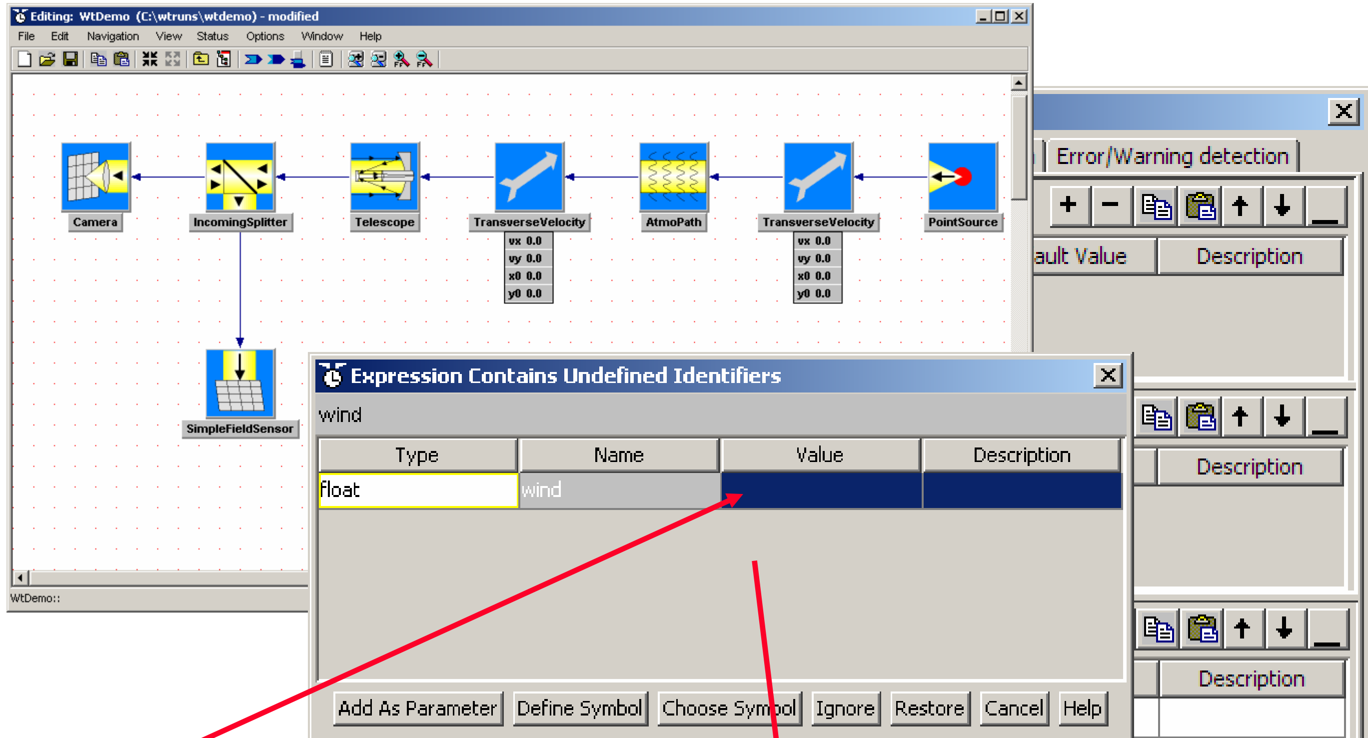
WaveTrain units are mks!



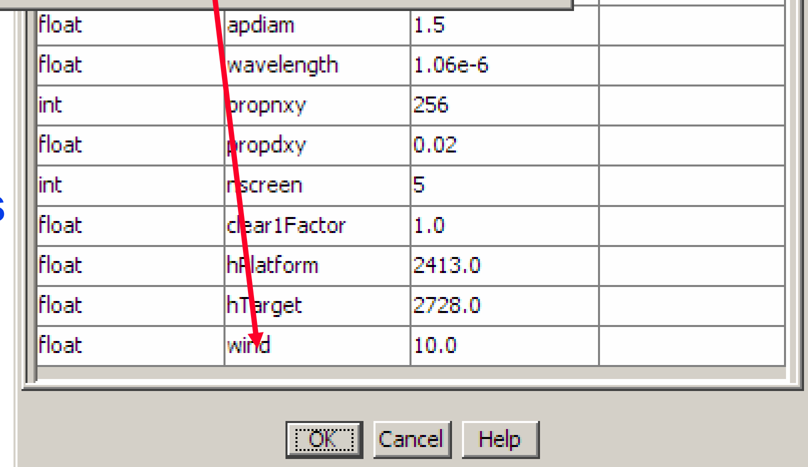
MZA Associates Corporation

Add More System Parameters

- **Click on the subsystem parameter button** again to undisplay them.
- **Click on either of the TransverseVelocity blocks, then Ctrl-click on the other, selecting both.**
- **Click on the subsystem parameter button once more**, which will display the parameters of only the TransverseVelocity blocks.
- **Click on one of the “vx” setting expressions**, and **enter “wind” and hit return.** This will bring up the window shown. **Enter “10.0” under “Value”** and hit return. **Click “Add As Parameter”.**
- Using the same approach, **set the “vx” for the other TransverseVelocity** system to **“-wind”.**

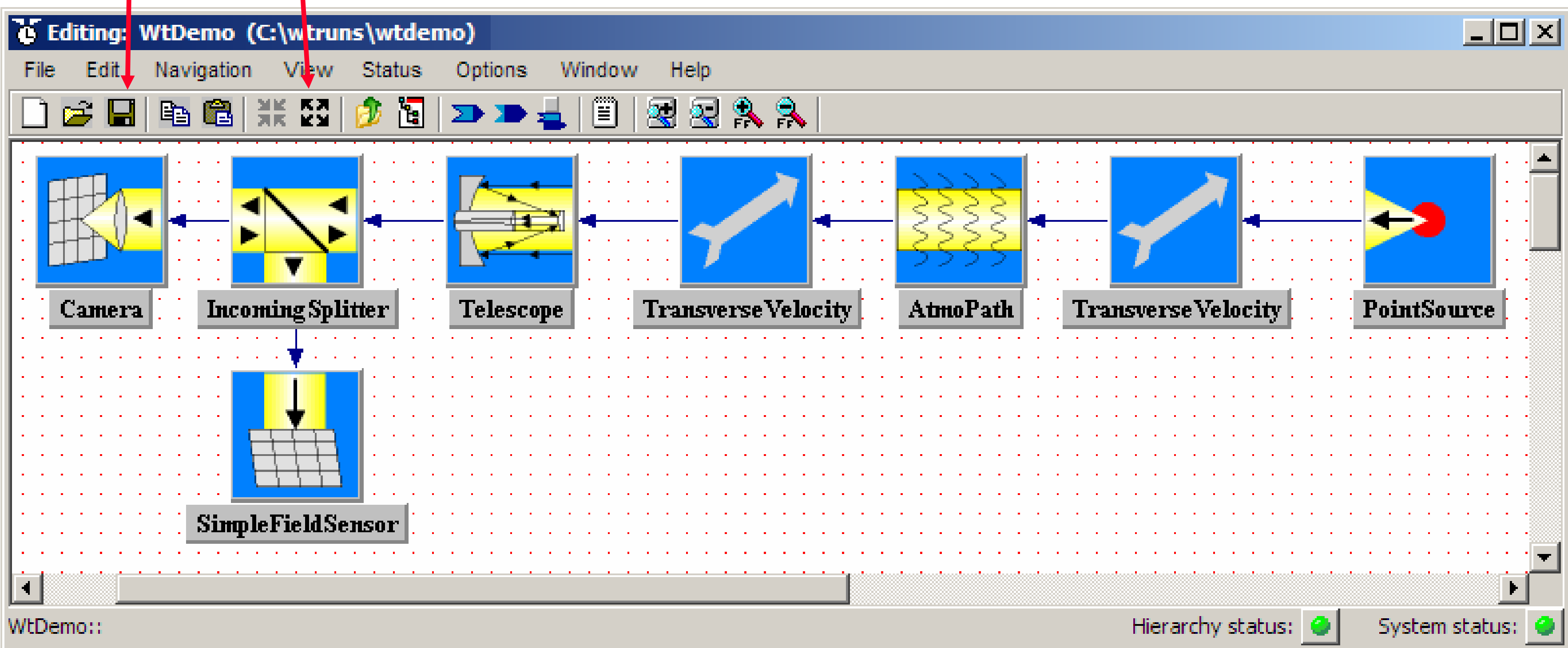


WaveTrain and tempus names are case sensitive!

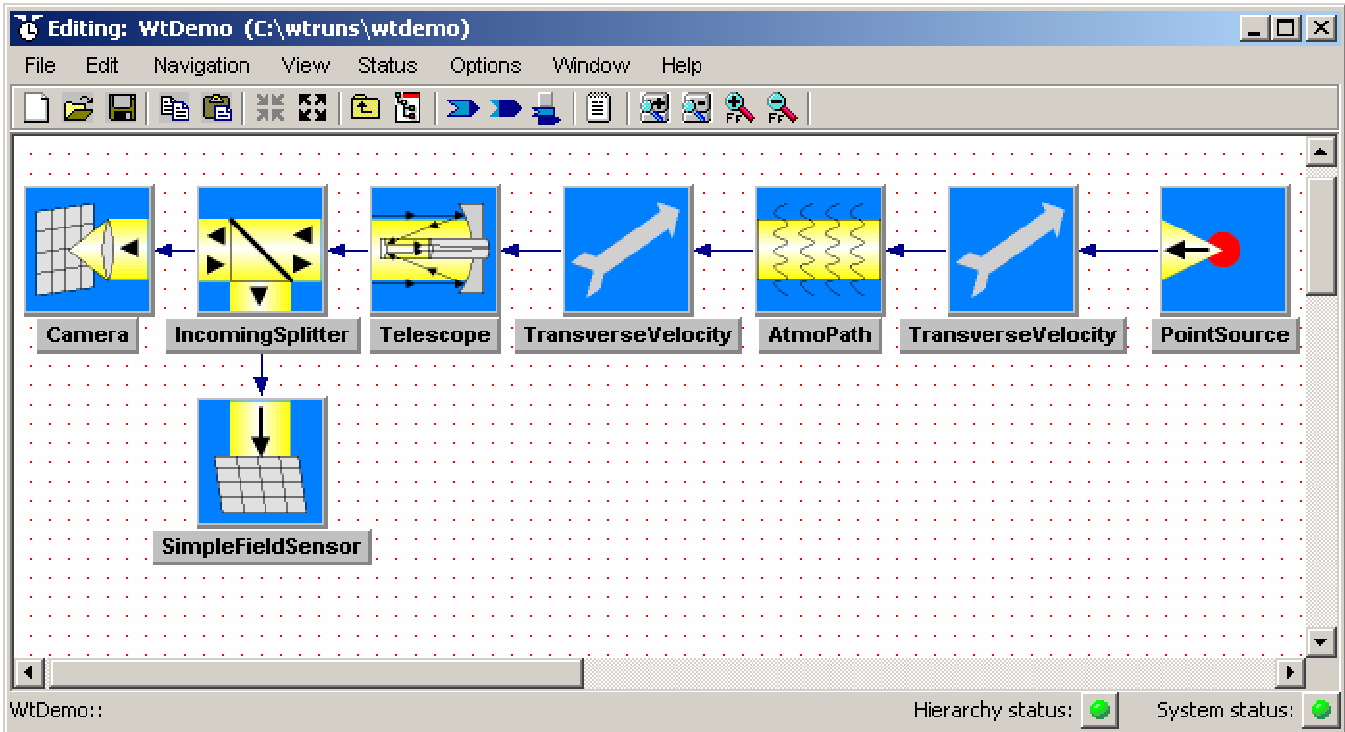


Finish the model and save

- Undisplay the TransverseVelocity parameters.
- Press the Contract button (four converging arrows) which will bring the blocks back close together.
- Depending on your esthetic preferences, you may wish to undisplay the subsystem labels and/or toolbars for a cleaner look; there are buttons for each.
- The system model is complete; now you will save the final version to disk. Click on File->Save or use the toolbar save button.



The Completed Model

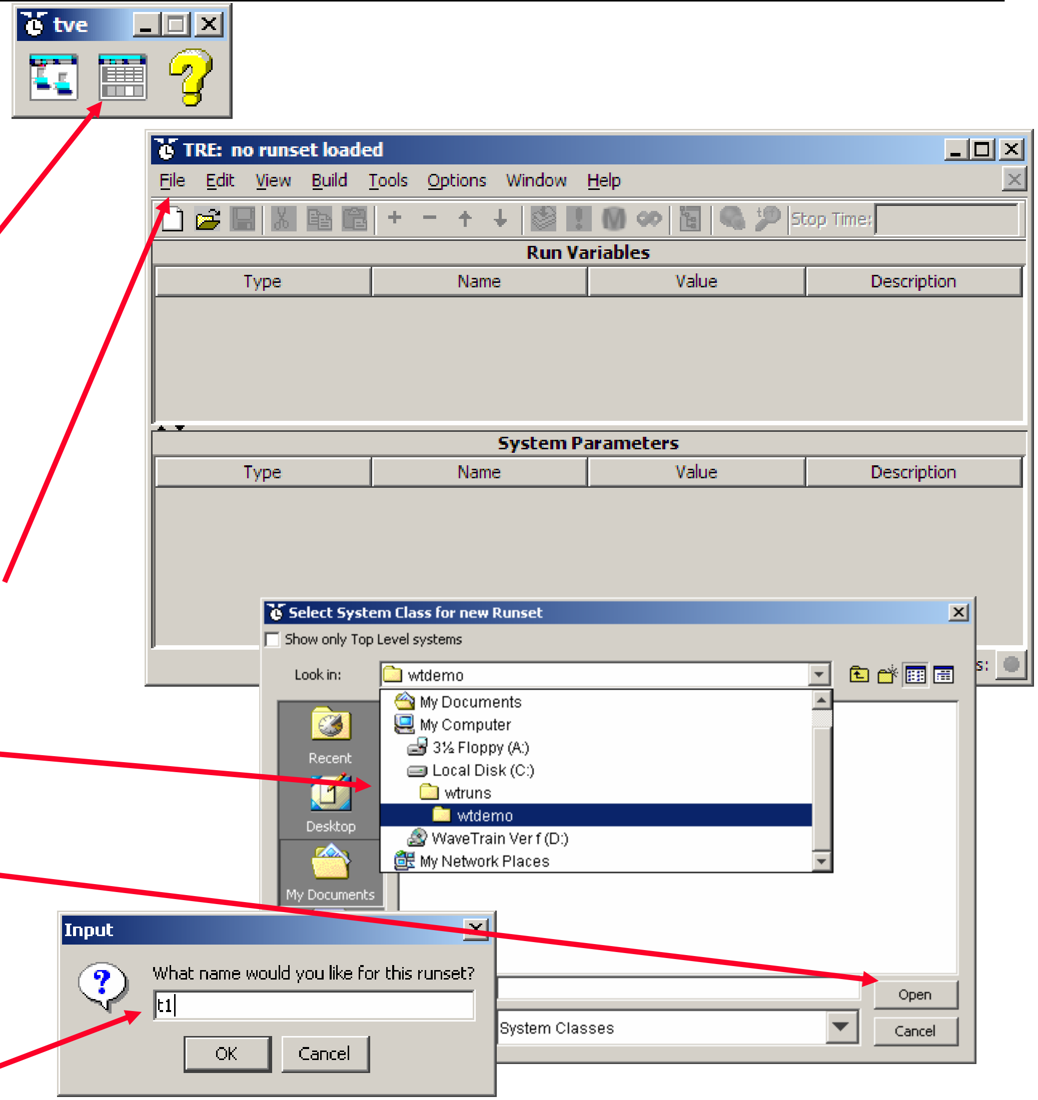


- You have built a complete model of a telescope system imaging a point source through turbulence, with the following features:
  - Records amplitude and phase at the pupil plane, and intensity at the focal plane.
  - Models platform motion, source motion, and/or wind.
  - Uses standard turbulence models, e.g. Clear 1 or Hufnagel-Valley, and/or user-defined models.
  - All major system variables are parameterized, so they can be changed without changing the model itself.
- Next, you will use the model to perform a parameter study.

MZA Associates Corporation

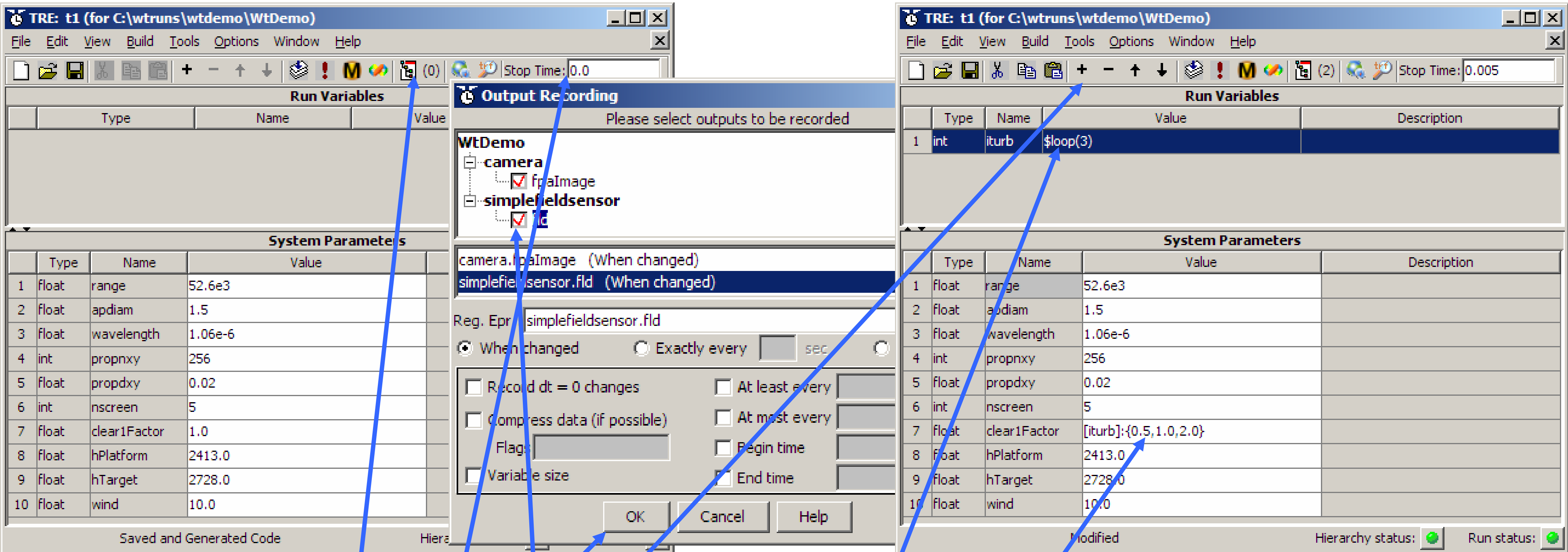
Create a new "Runset" for a parameter study

- A Runset describes a set of related simulation runs, in which any number of model parameters can be varied, either independently or in groups. Each Runset is mapped into a portable C++ main program via automatic source code generation.
- Go to the tempus toolbar window, and click on the middle button (tempus runset editor) which will bring up the "TRE" window, shown at right.
- Click on File->New->Runset … which will bring up the window shown at the bottom. Navigate to the c:/wtruns/wtdemo, and select WtDemo.tsd.
- Click Open, which will create a new Runset for the just-created system model.
- A dialog box will be displayed which asks you what to name the Runset. This identifies the particular group of settings with which you are going to run the simulation. Use a simple name for now like t1. Click OK.



MZA Associates Corporation

Specify the runs to be done, and the outputs to be recorded

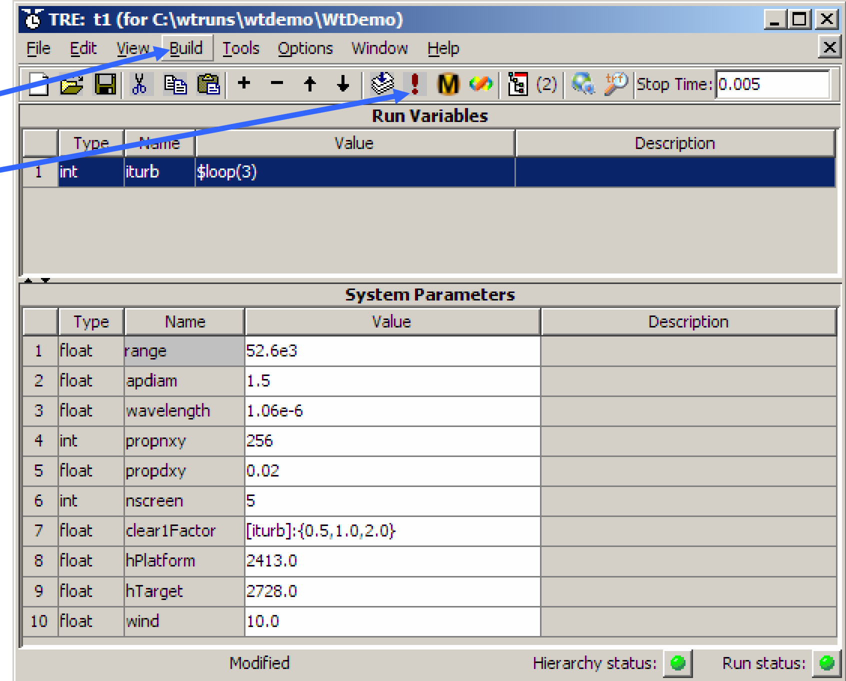


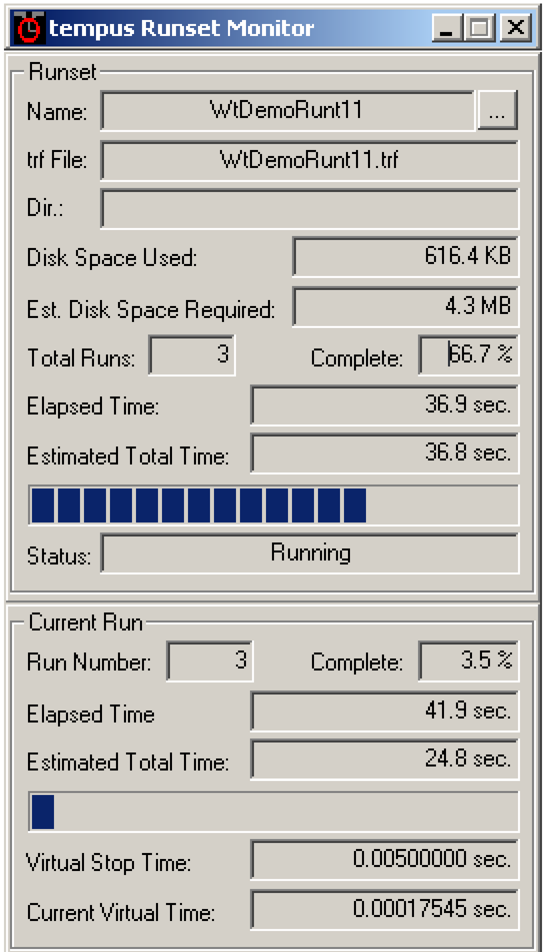
- Initially, the Runset will have all system parameters set to the defaults you specified when you built the system. The stop time for each run will be set to zero, and no outputs recording will be set up.
- Set the stop time to 0.005
- Click the button (Recorded Outputs) to display a window for specifying output recording. Click on checkboxes next to each of the two outputs. Click "OK".
- Click the button "+" to create space for one run variable, then enter "int" "iturb" "$loop(3)"; this will create a for-loop, resulting in three separate simulation runs.
- Set clear1Factor to "[iturb]:{0.5,1.0,2.0}"; so its value will change with each loop iteration.

WaveTrain and tempus names are case sensitive!

Execute the Runset

- Click on Build->Execute. This will automatically save the Runset Information to disk, generate the C++ main program, compile it, link it, and execute it.
- You could use the toolbar button to run instead.





- Shortly after execution begins, a "tempus Runset Monitor" will appear. This provides information such as elapsed time, disk space used, etc. When execution is complete, it will appear as shown.

- After a run is complete you should close the Command Line window and tempus Runset Monitor that was opened during execution.

MZA Associates Corporation

Load the results into Matlab

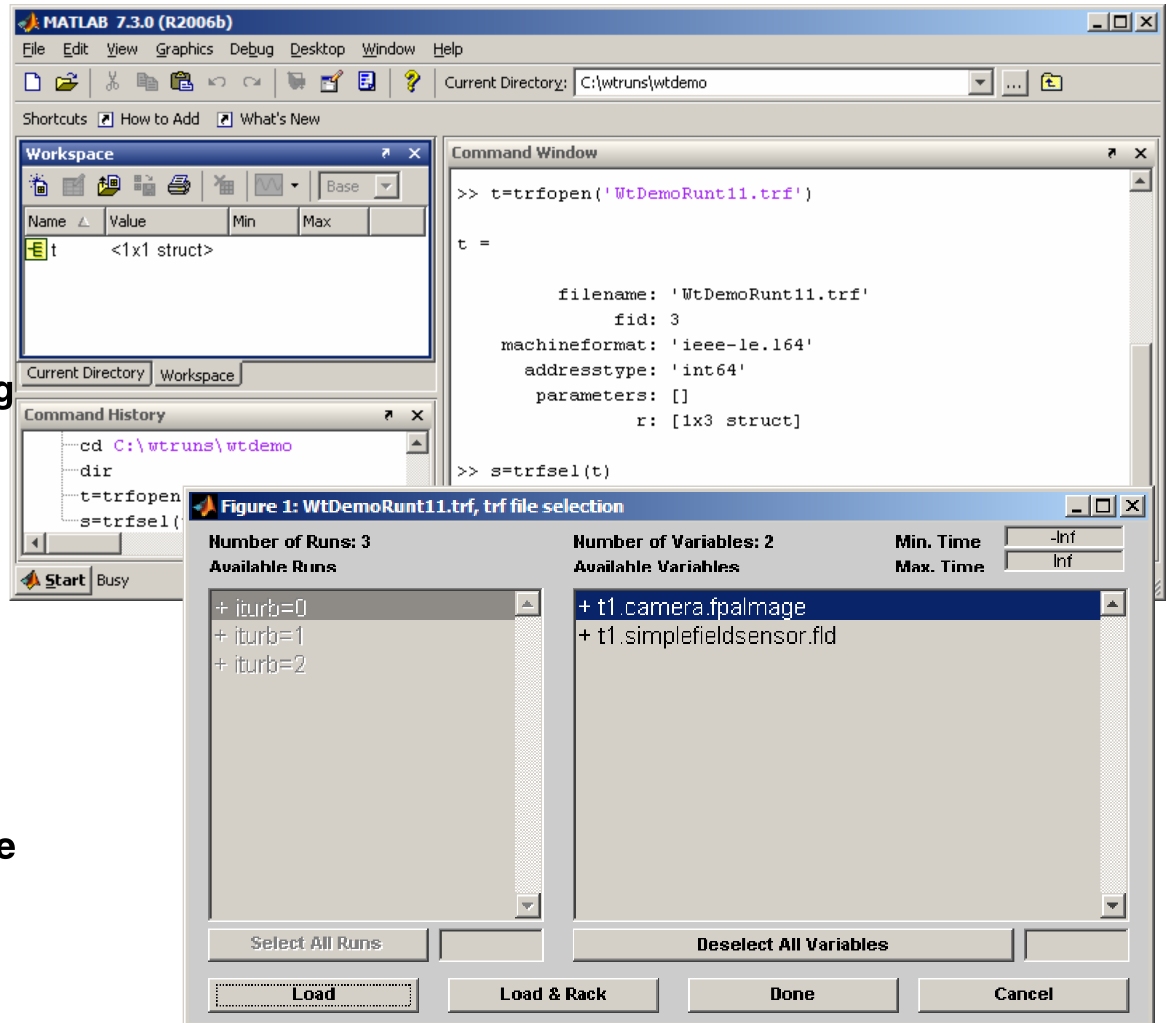
- tempus simulation outputs are stored in specially formatted random access files called “trf” files which preserve the structured character of the data, and support interactive browsing without having to load the entire file.

- tempus provides a rich set of mechanisms for accessing and operating upon trf files, including many designed for use from within Matlab, either at the command line, or from within m files.

- To look at the results from the just-completed Runset, open a Matlab session, and cd to the appropriate directory.

- Open the file, then bring up an interactive browser using the following commands:
  - » t=trfopen('WtDemoRunt11.trf')
  - » s=trfsel(t)

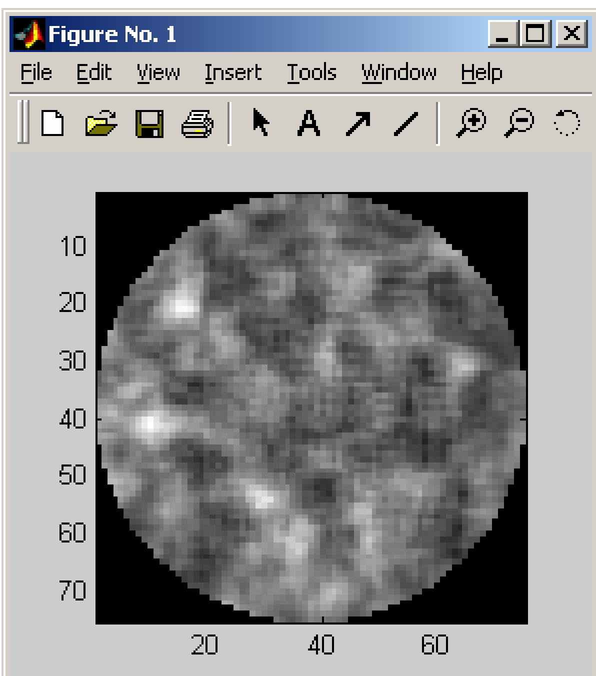
- Click on Select All Runs, then Select All Variables, then Load, which will load all the recorded data.

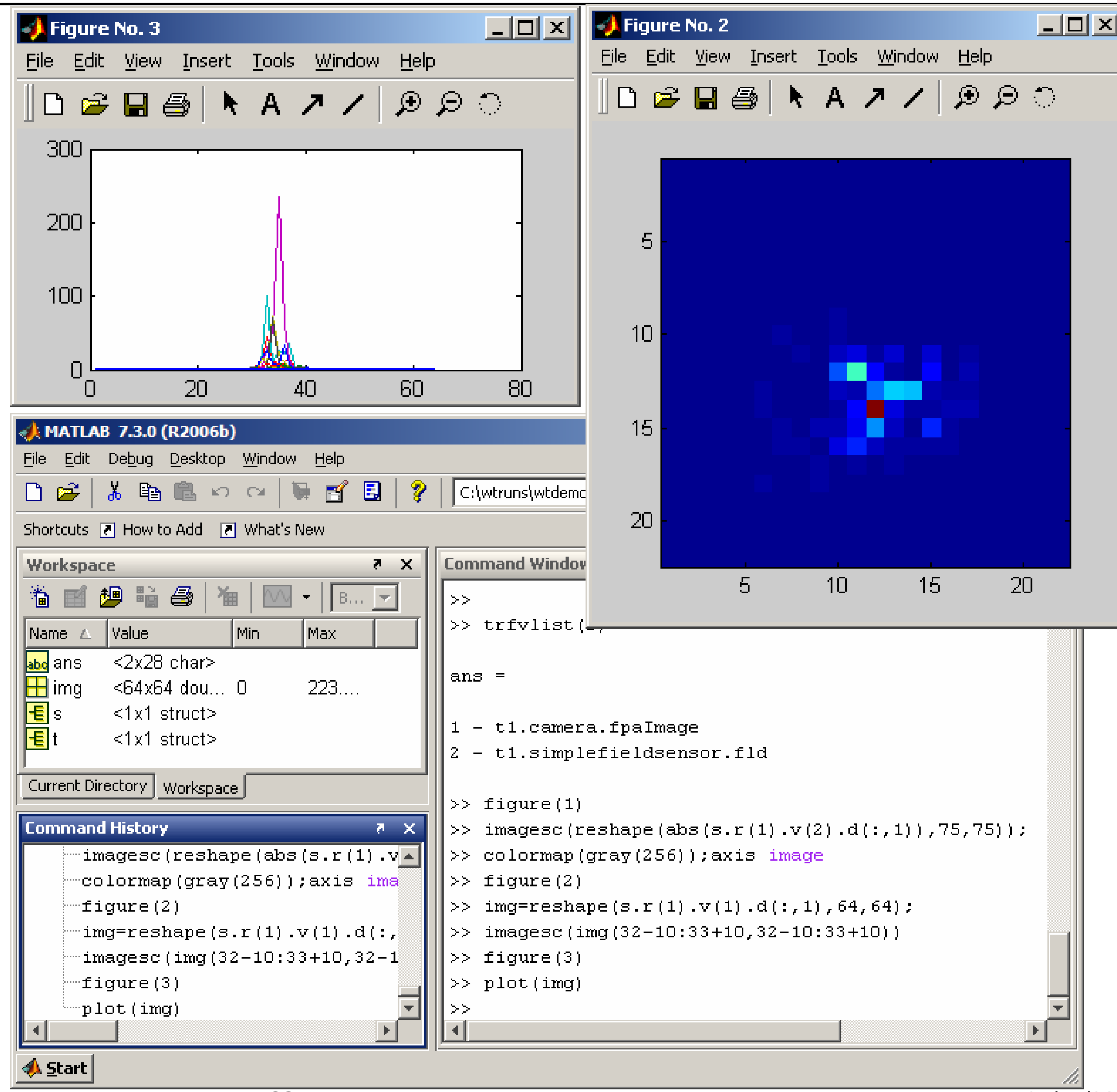


- You must have the WaveTrain and tempus mfile paths in your Matlab path before you can use the Matlab functions.

Look at the results in Matlab

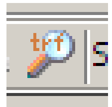
- Data can be loaded into Matlab in various forms; in this example we have loaded it into a structure.
- Once the data has been loaded, all the functionality of Matlab is available - analysis, plotting, movies, etc.

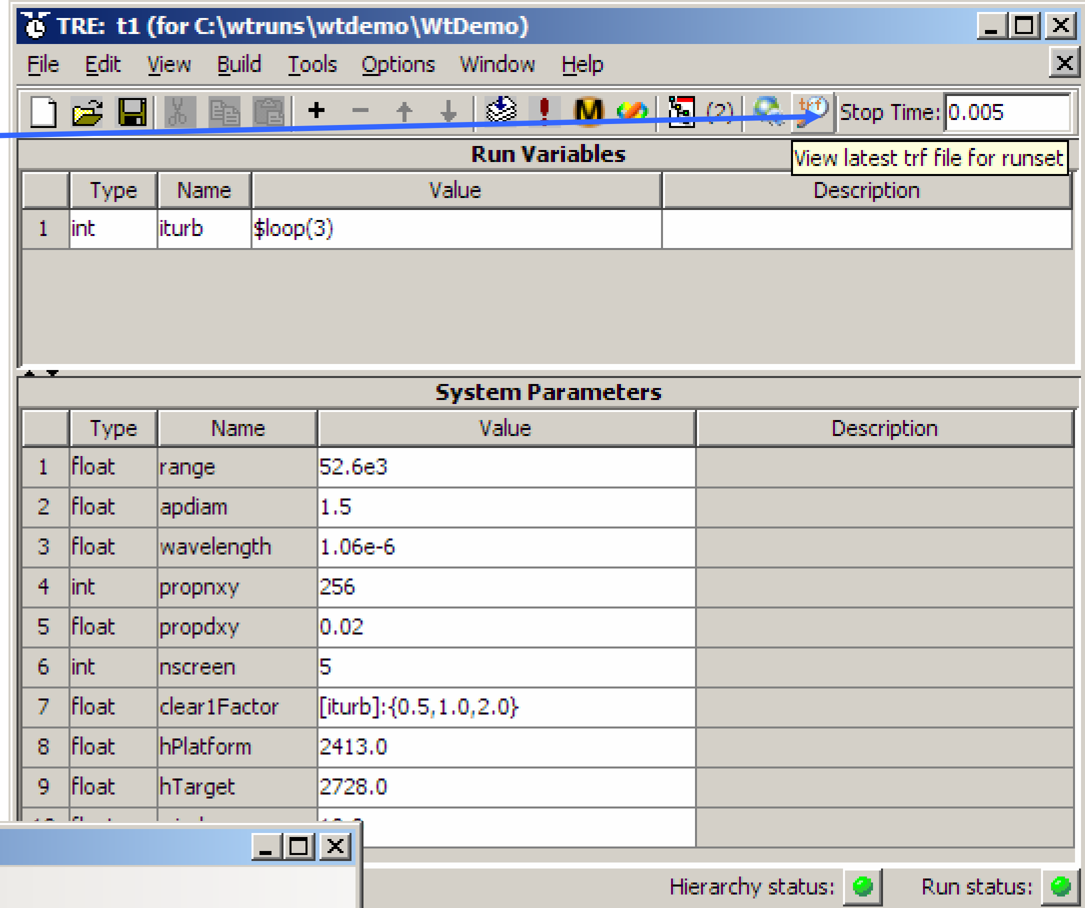




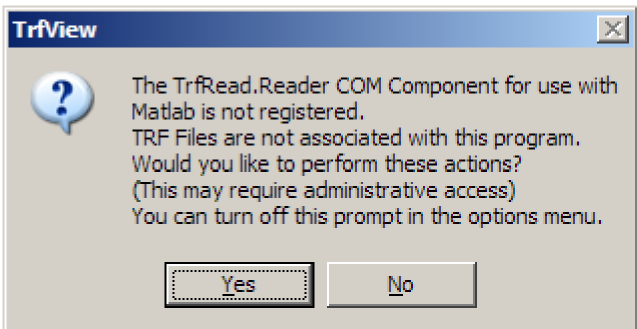
Alternatively, look at results in TrfView

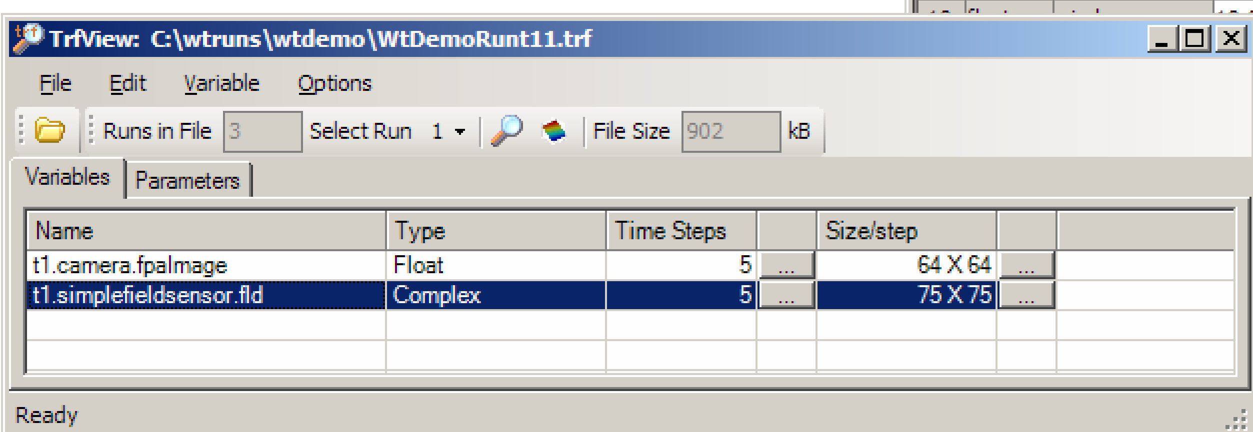
- TrfView is a recent addition to WaveTrain that enables basic plotting directly from TRE.





- On first use, you will be prompted to associate .trf files (you can also do this later from TrfView Options)

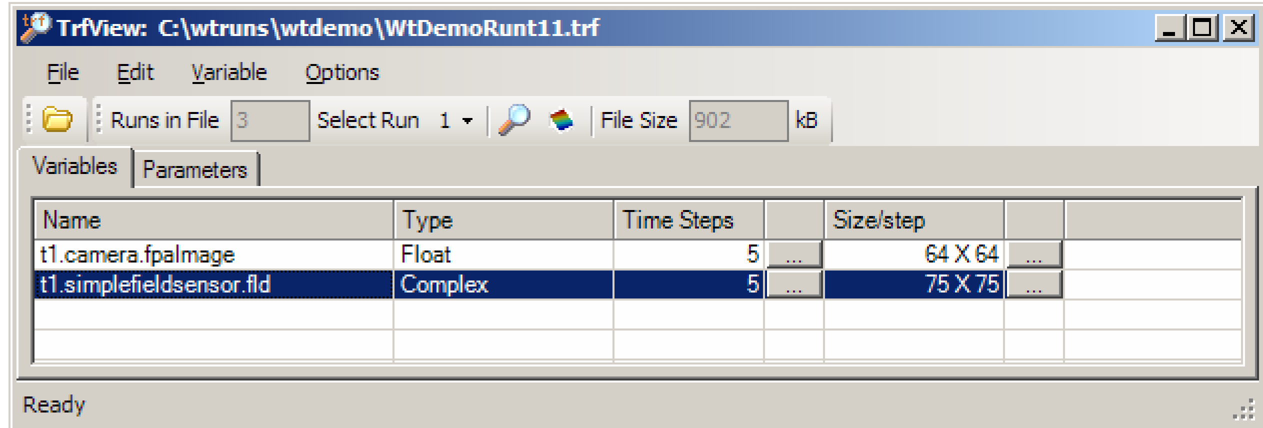


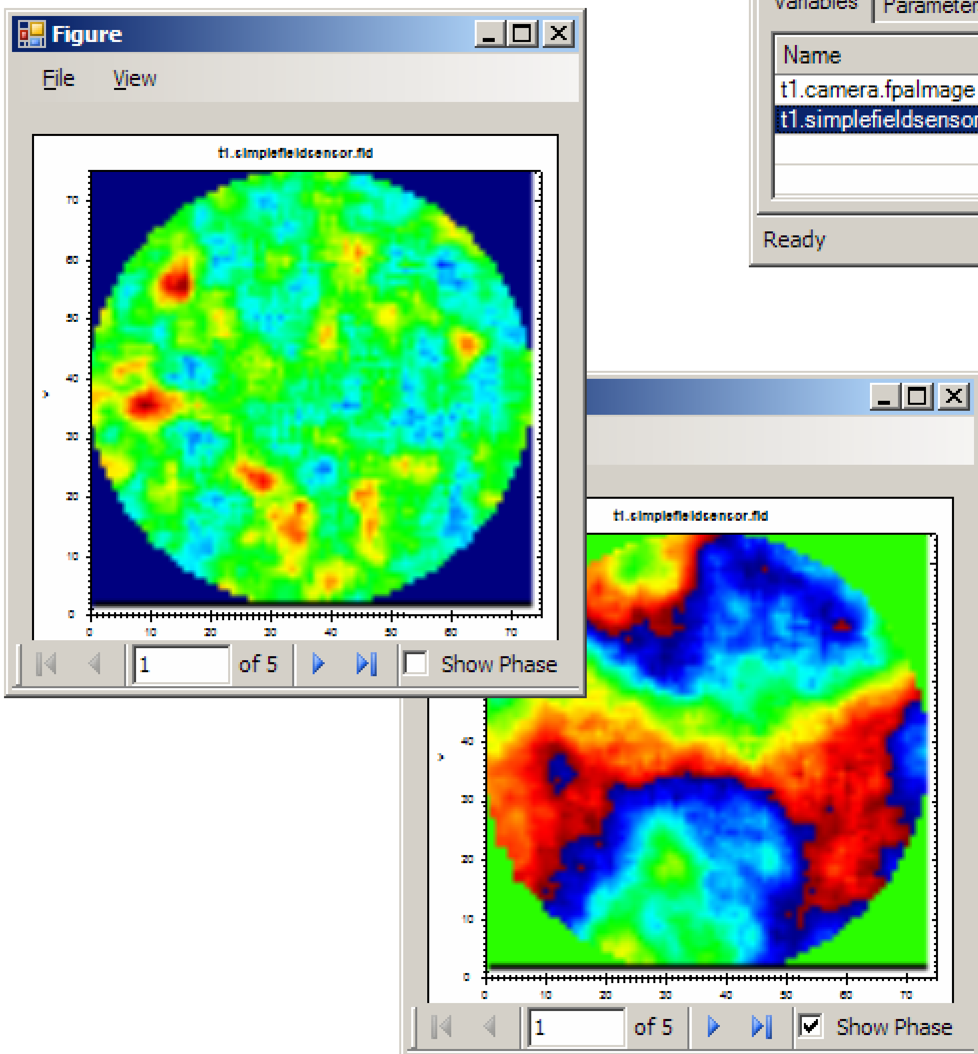


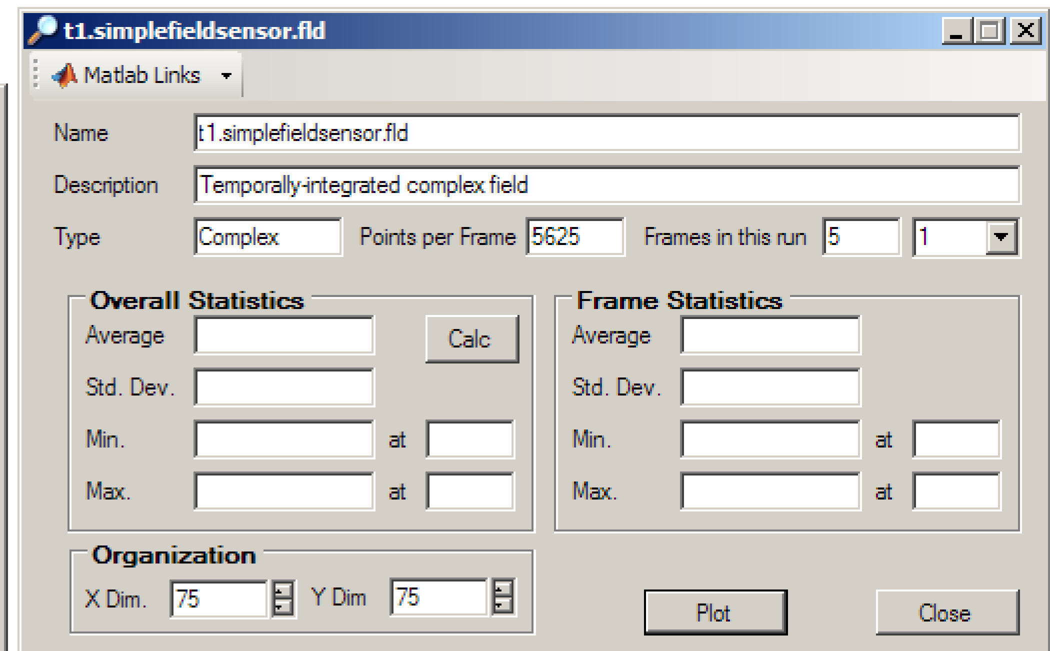
MZA Associates Corporation

Look at results in TrfView

- Right-click on variable name & select “Show” or “plot”
- E.g. Field amplitude & phase



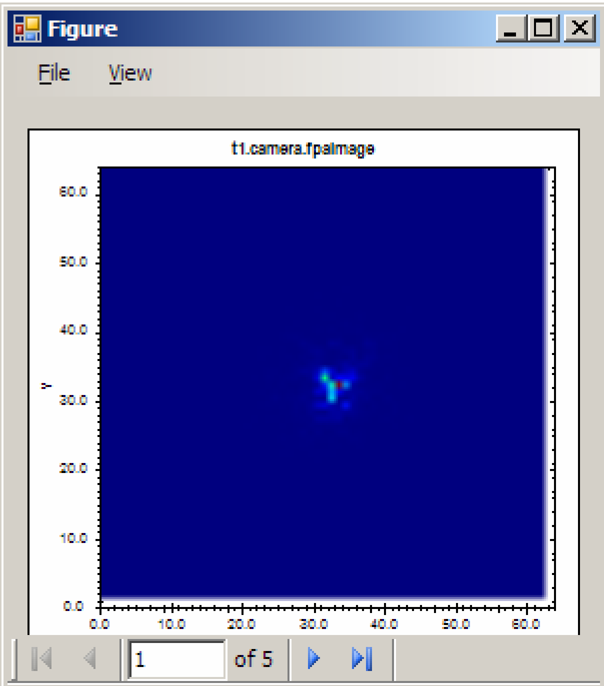


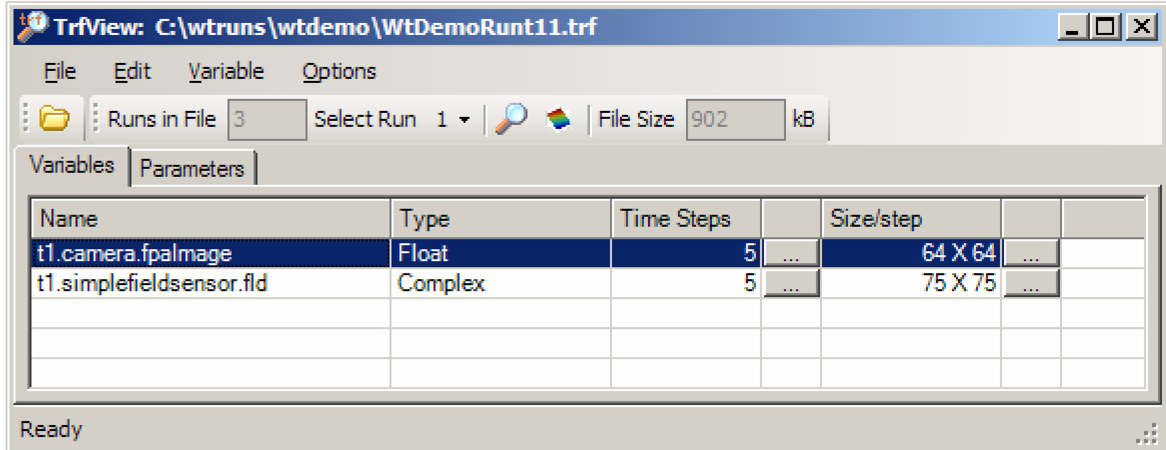


MZA Associates Corporation

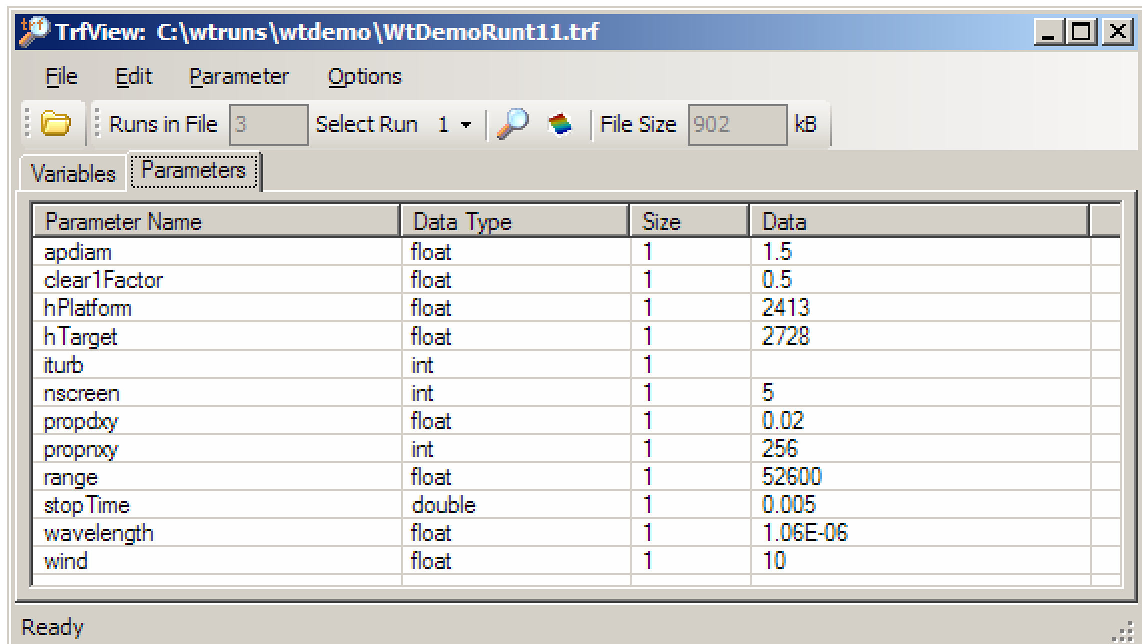
Look at results in TrfView

- E.g. Camera image





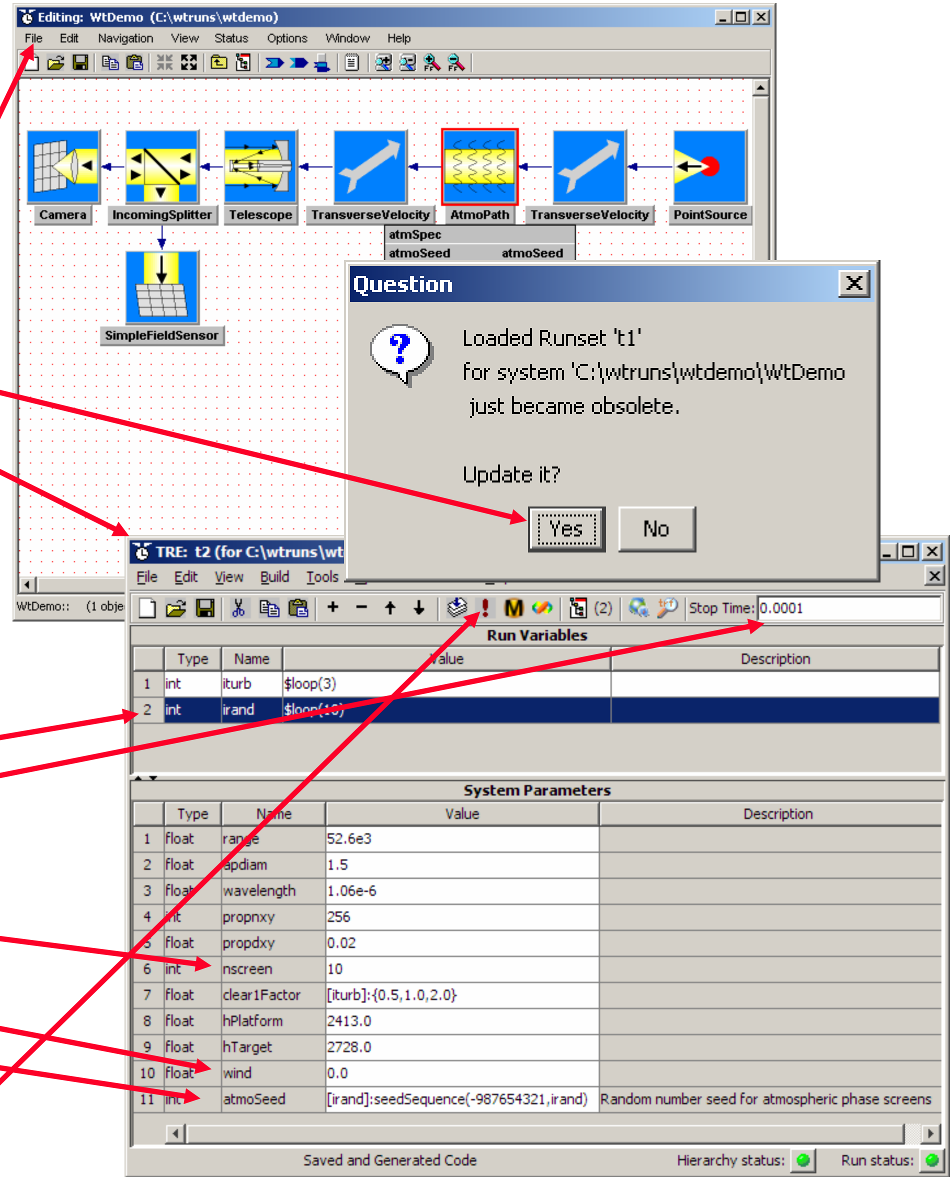
- Simulation Parameter values are also viewable



MZA Associates Corporation

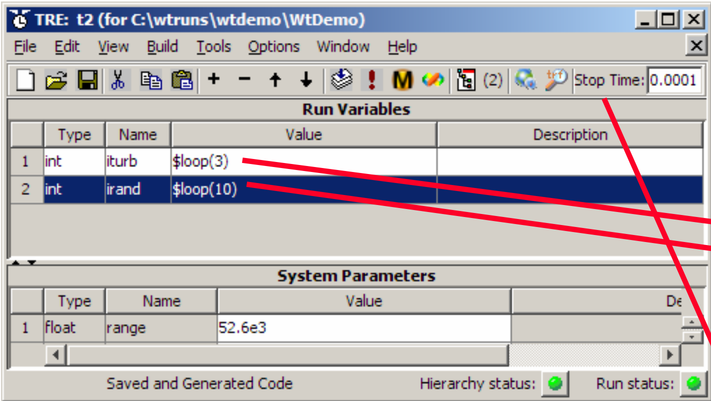
Extended Analysis: Uncorrelated Data

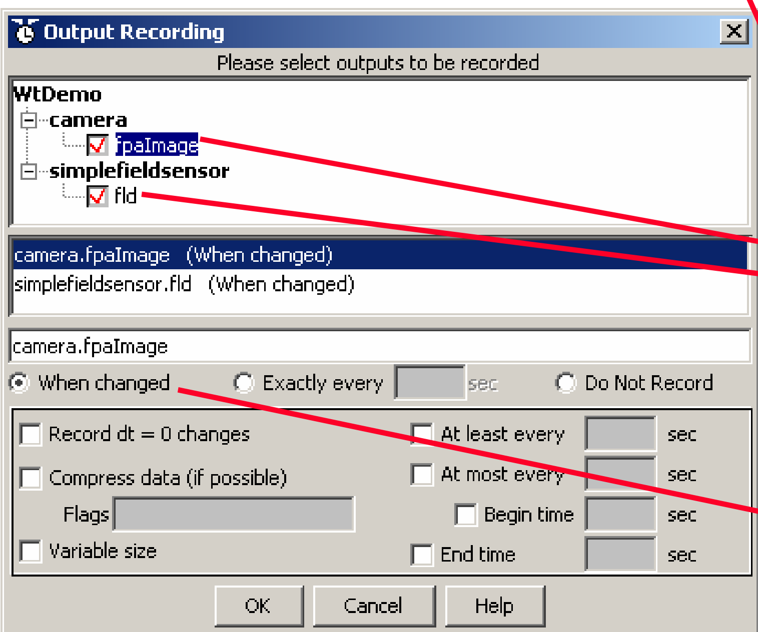
- Go to the System Editor for WtDemo. Display the parameters of the AtmoPath and elevate the atmoSeed parameter, by right–clicking on it and selecting Elevate in the small window that pops up.
- Select File->Save. When it asks if you want to update the Runset, click Yes.
- Go to the TRE.
- Choose File->Save As…, and save a new Runset t2.



- Add a run variable called irand and set it to $loop(10) (copy/paste iturb & edit).
- Change Stop Time to 0.0001.
- Change nscreen to 10.
- Change wind to 0.0.
- Set the newly-created atmoSeed parameter to [irand]:seedSequence(-987654321, irand)
- Build->Execute. This will take a minute or so…

Anatomy of a trf File





- **trf files also contain the run variable and parameter settings.**

- **Each trf file is like a database; organized into runs and variables.**
  - **The number of runs is equal to the product of the value of all loop variables.**
  - **For this case:**
    - **nr = iturb × irand = 10 × 3 = 30.**
  - **The number of variables per run is less than or equal to the number of variables selected by the user for recording.**
    - **It can be less than the number selected because it is possible that a variable which was selected for recording does not get computed during execution.**
  - **For this run 2 variables were recorded.**
  - **A time history of each variable is stored. The precise times and the number of times that a variables data is stored is dependent on:**
    - **The amount of simulation time per run.**
    - **User recording settings.**
    - **Simulation execution logic.**
  - **Each type of data is stored in a fairly simple stream format.**

Anatomy of a trf Handle

- In Matlab, trf files are incrementally loaded into a structure of the following form:
  - t.r(nr).v(nv)
  - The jth variable for the ith run is stored in a structure at t.r(i).v(j).
- When a variables' data is read from disk, its is stored as a time history:
  - t.r(i).v(j).t contains the virtual time at which the data was recorded.
  - t.r(i).v(j).d contains the data. It is always two dimensional, nd x nt, where nd is the number of elements required to store the data and nt is the number of times the data was recorded.
- A scalar quantity is stored as:
  - t.r(i).v(j).d(1:1,1:nt)
- A two-vector is stored as:
  - t.r(i).v(j).d(1:2,1:nt)
- A 64x64 grid is stored as:
  - t.r(i).v(j).d(1:4096,1:nt)

- trf handles also contain run variable and parameter settings.
- You need not load an entire file. Data is loaded incrementally.
- trf handles contain a lot of ancillary information.

- The present example has 1 time-step for 2 variables for 30 runs
  - s2.r(1:30).v(1).d(1:5625,1:1) is a complex array representing the light hitting the receiving aperture.
  - s2.r(1:30).v(1).d(1:4096,1:1) is real array representing the image of the distant point source.

MZA Associates Corporation

Process Uncorrelated Data

shops1.m and shops2.m

- **Add the workshop scripts to your path**
  - path('C:\Program Files\MZA\wavetrain\v2007B\examples\wtdemo\scripts',path);
- **Load the data**

  >> t2=trfopen('WtdemoRunt21.trf');

  >> s2=trfload(t2); % trfload is simpler than trfsel and is used more often.
- **Review and run the script in shops1.m to calculate the following quantities from the complex field.**

  >> edit shops1.m <F5>

  >> disp(niv)
  - Normalized irradiance variance, $\sigma_I^2 = (< I^2 > / <I>^2) - 1$
  - Rytov number (log-amplitude variance) is approximately $\sigma_I^2/4$.

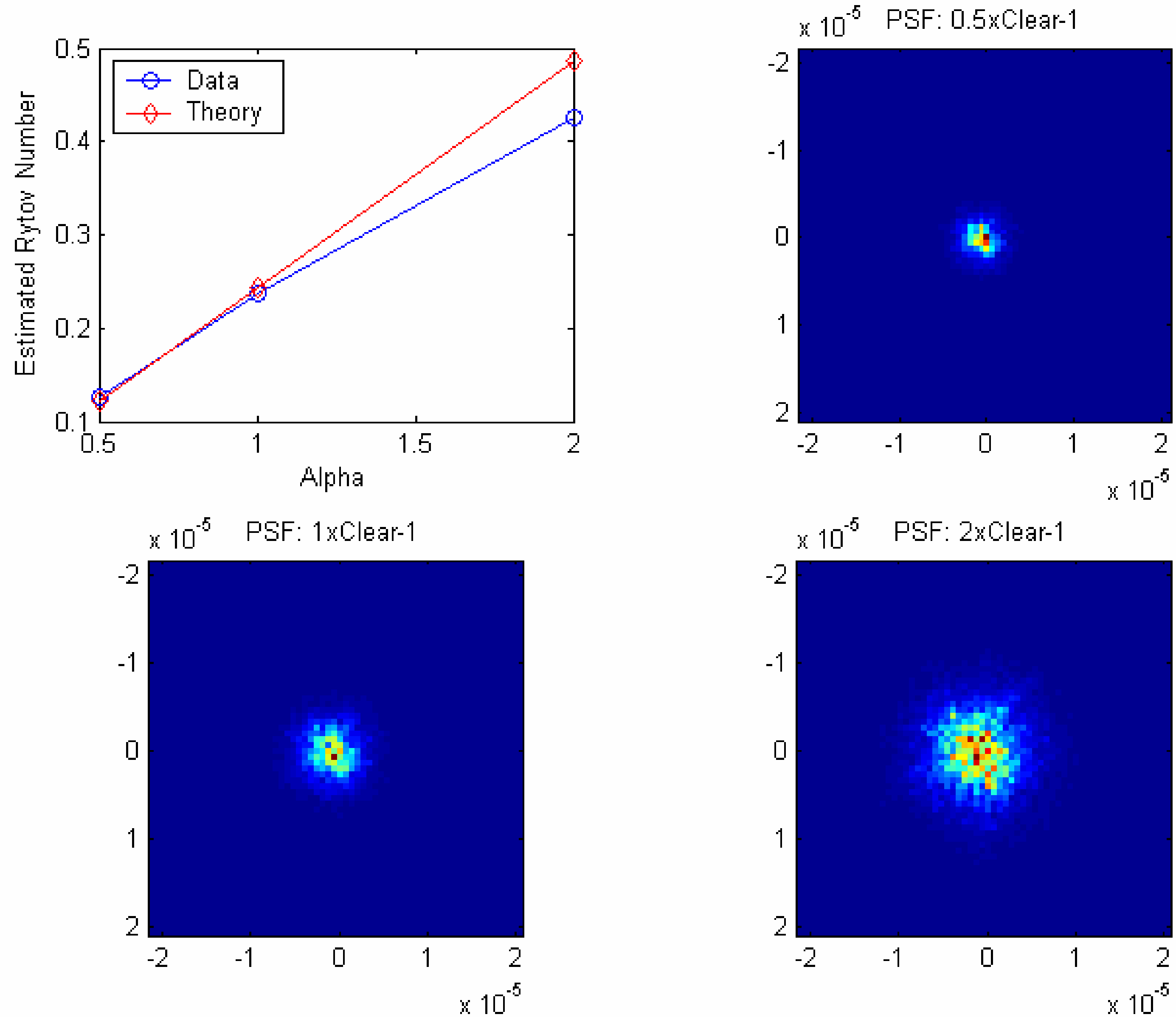
  >> disp(pcstrehl)
  - Phase corrected Strehl, $I_{rel} = <<A>^2/<I>>$
- **Review and run the script in shops2.m to calculate the following quantity from the point source image.**

  >> edit shops2.m <F5>
  - Time-averaged point spread function (PSF)
- **Plot the data with shops12p.m**

  >> edit shops12p.m <F5>

Processed Results

shops12p.m

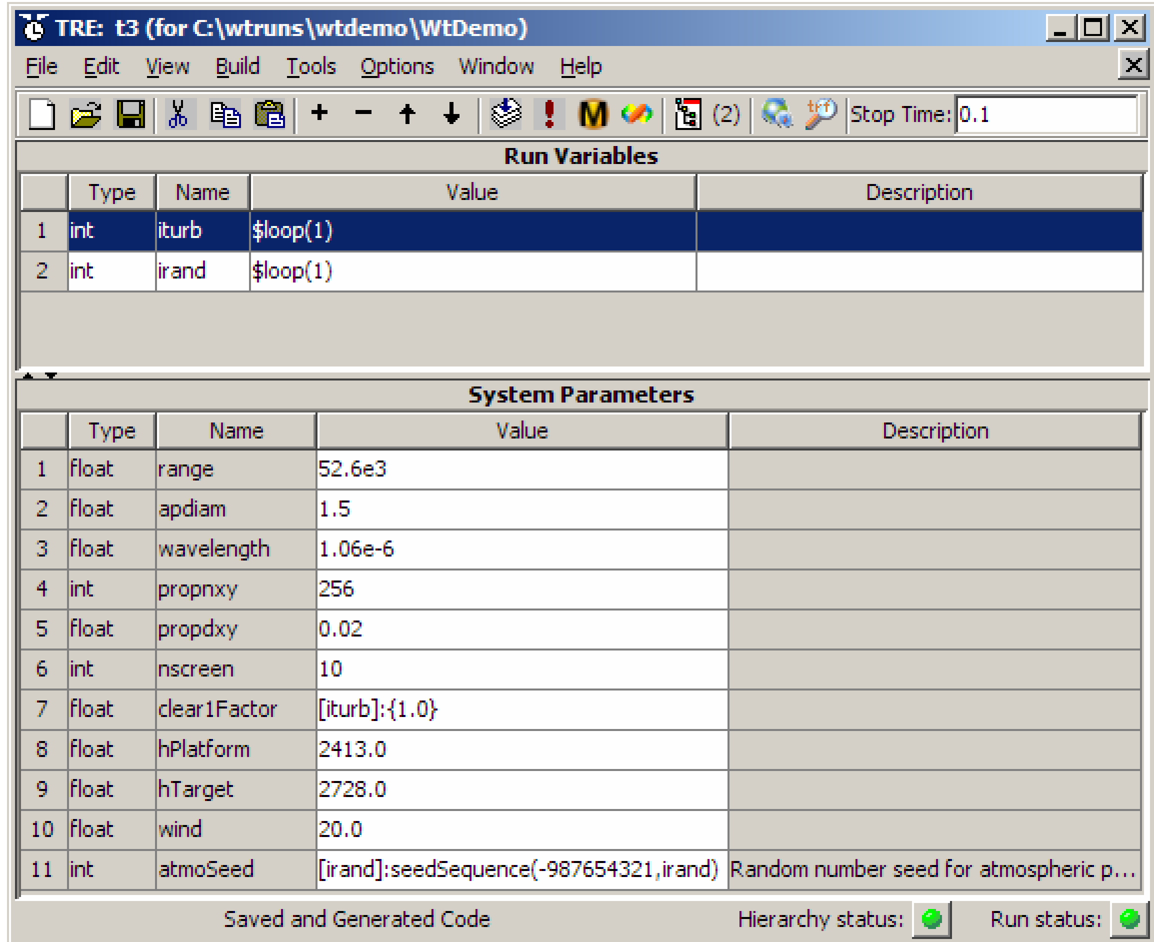


MZA Associates Corporation

Extended Analysis: Correlated Data

- **Go to the Runset Editor.**
- **Open runset t2.**
- **Choose File->Save As…, and name the new Runset t3.**

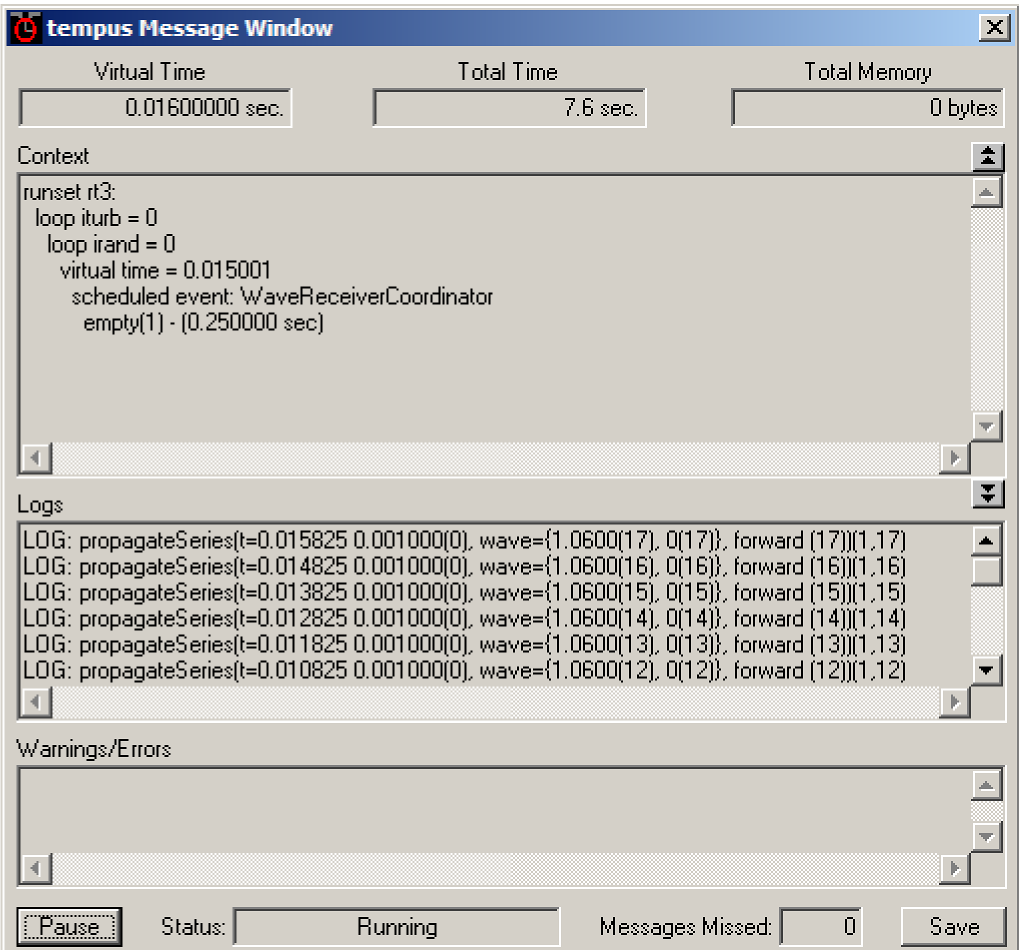
- **Change iturb to $loop(1).**
- **Change irand to $loop(1).**
- **Change clear1Factor to a single value (e.g., 1.0).**
- **Change wind to 20.0.**
- **Change Stop Time to 0.1.**
- **Build->Execute. This will take about four minutes…**

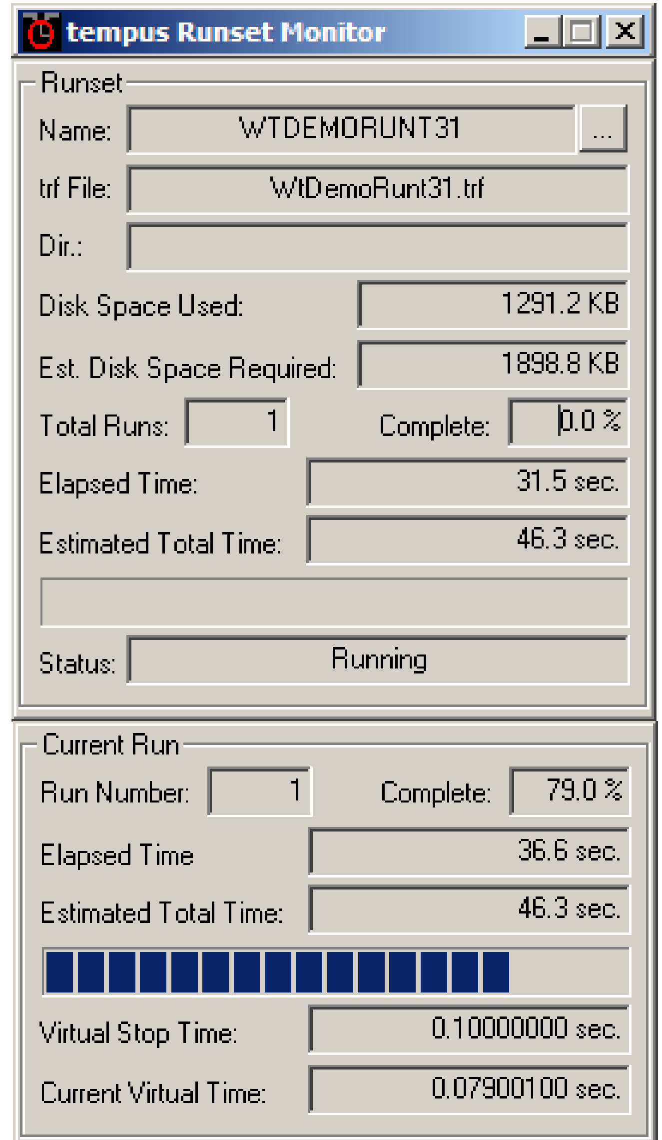


MZA Associates Corporation

Monitor the Simulation in the trm

- While the simulation is running, right click on the tempus Runset Monitor (trm) and choose Messages.
- Here you can view detailed messages which track the execution status.

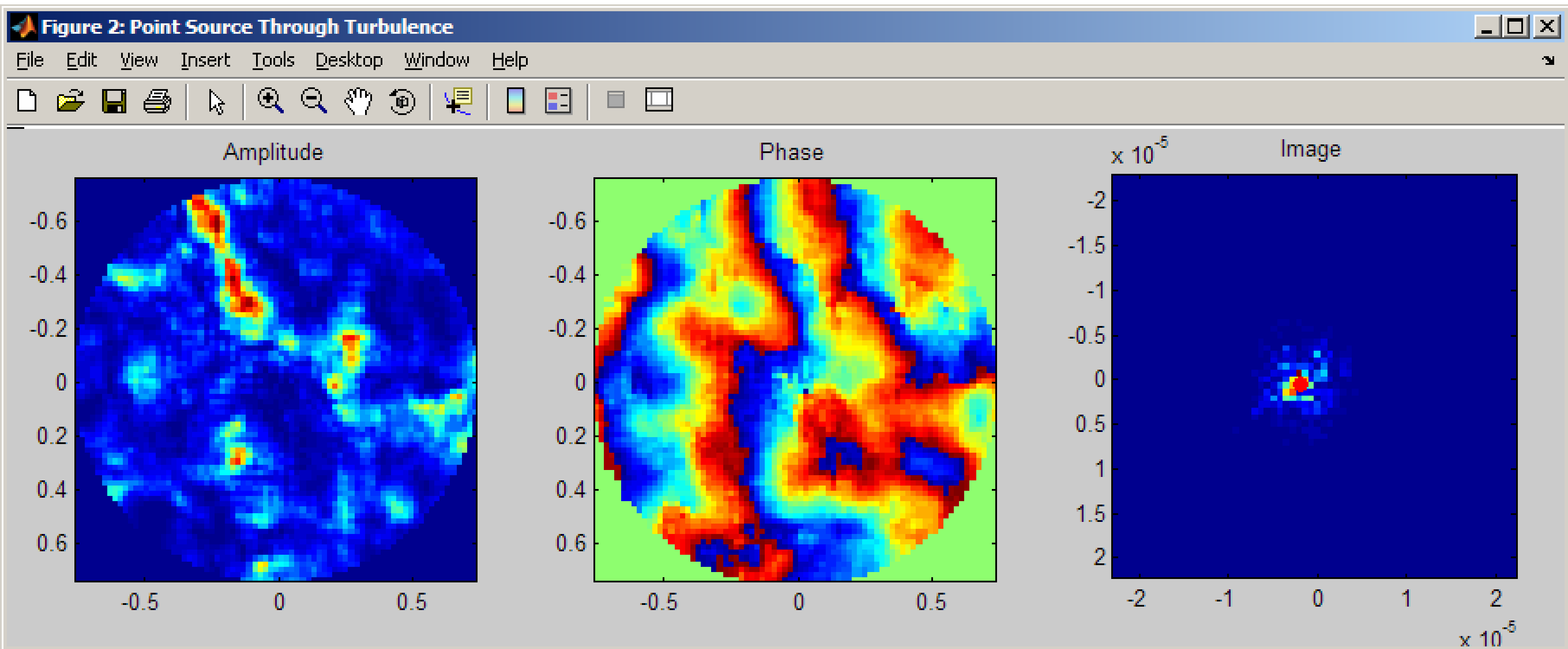




Process Correlated Data

shops3.m

- **Load the data**
  - \>> t3=trfopen('WtdemoRunt31.trf');
  - \>> s3=trfload(t3); % trfload is simpler than trfsel and is used more often.
- **Review and run the script in shops3.m to create a movie of the point source propagation data.**
- **To repeat the movie use movie(mb).**

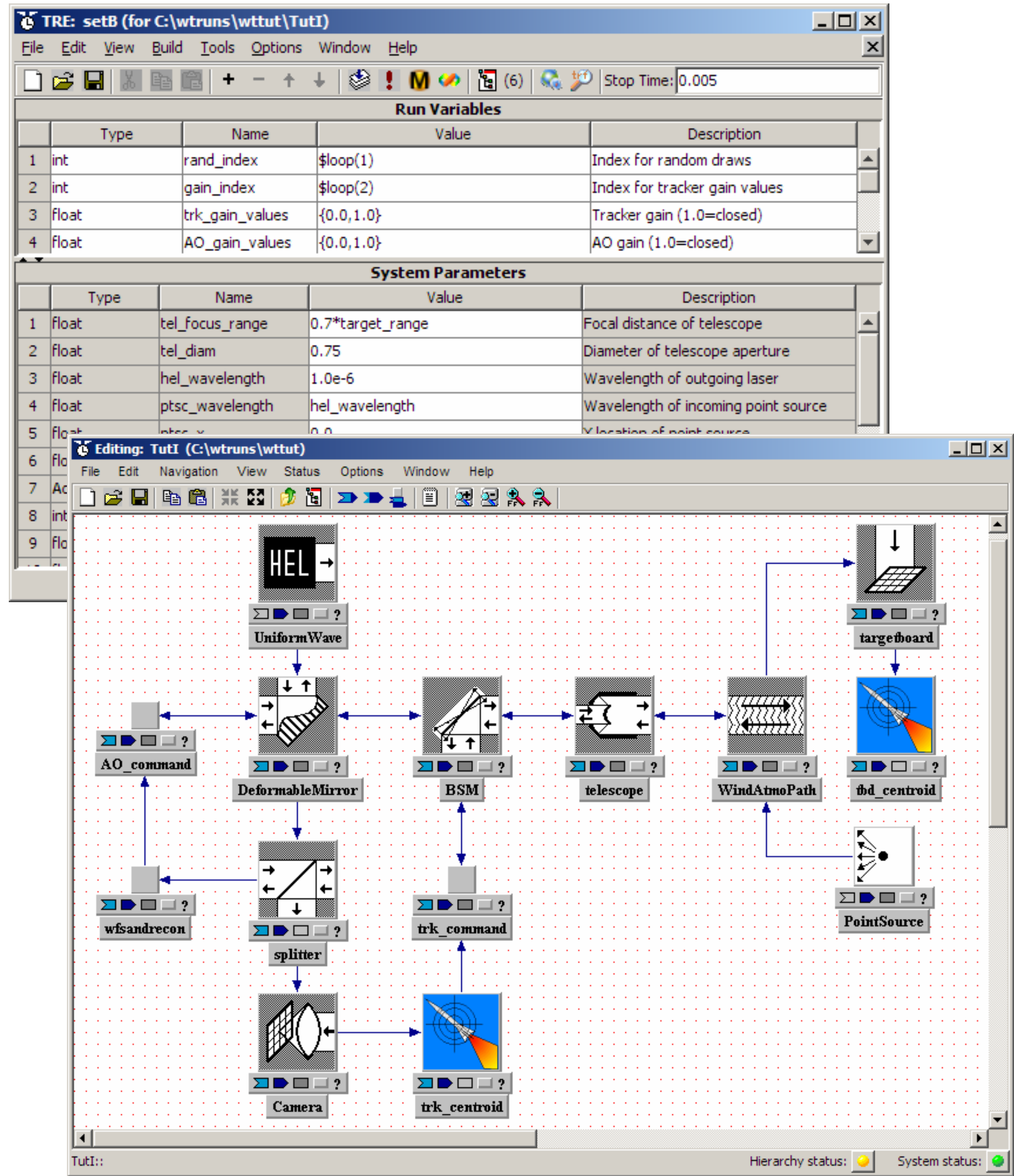


MZA Associates Corporation

The Whiteley Tutorial

Closed-Loop AO Example

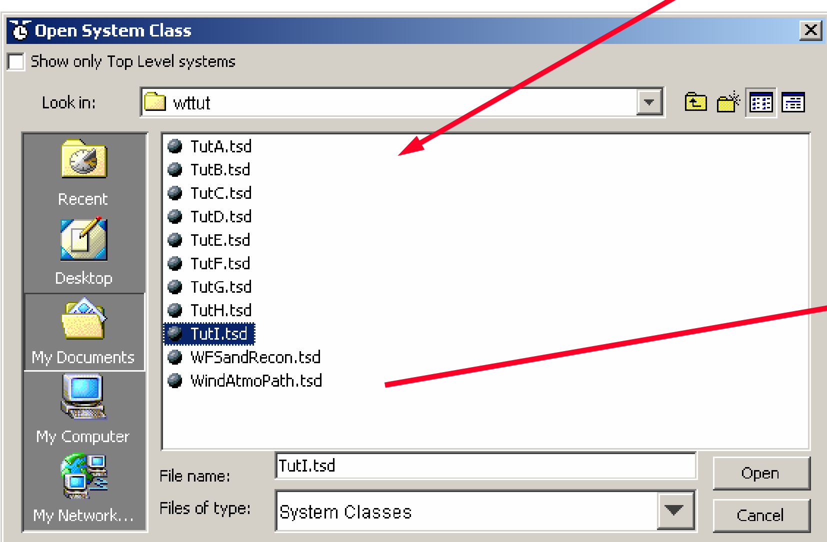
- **Matt Whiteley (then of AFRL/DEBA, now with MRC) created a three-day WaveTrain tutorial workshop in which users incrementally build up a closed-loop adaptive optics system.**
  - **The workshop also serves as an introduction to fundamental wave optics simulation concepts.**
  - **The tutorial materials are on the Workshop disk in the directory "whiteleyTutorial".**
- **Since we don't have three days to go through all of the steps of building the model, we will concentrate on working with the complete model.**
- **The tutorial will now proceed a bit more quickly, assuming that you are starting to get the feel for how things work in the GUI.**
  - **Instructions are less explicit.**
  - **Emphasis will be placed on the model, rather than the mechanics.**

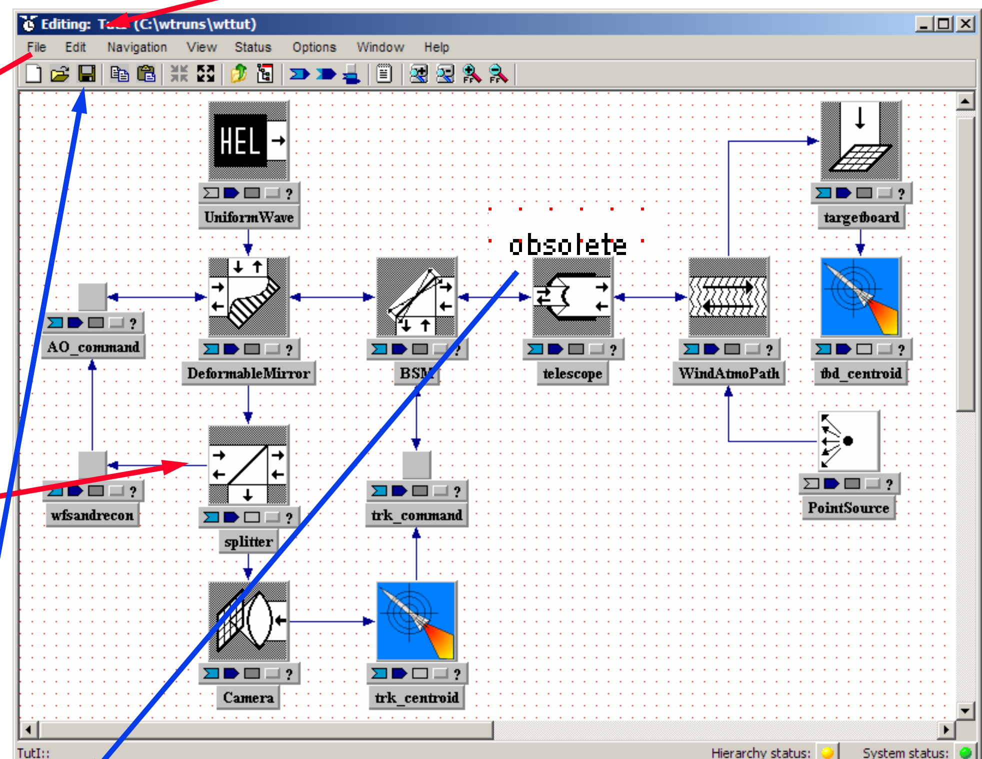


MZA Associates Corporation

Closed-Loop AO: Get Ready to Run

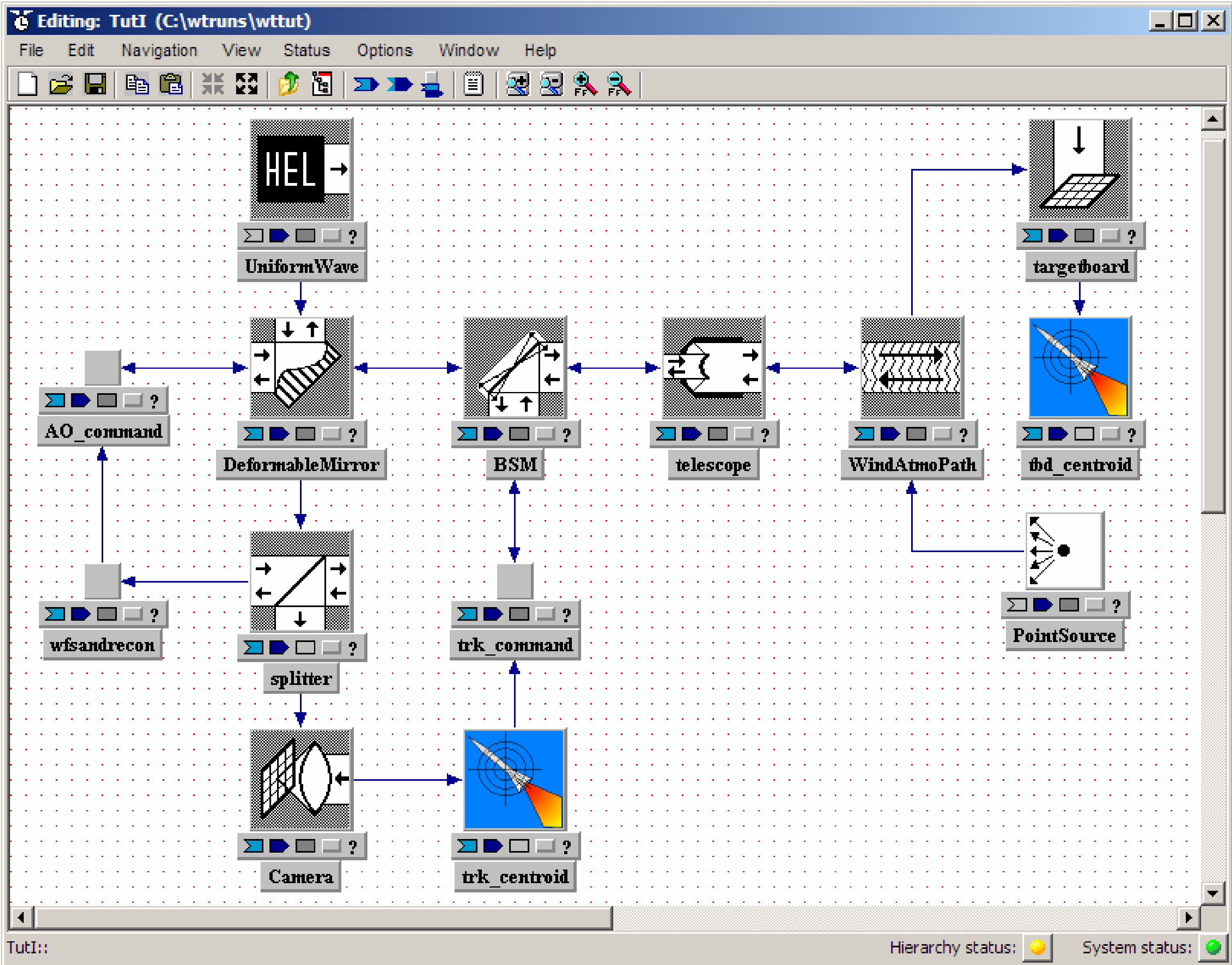
- **Copy the directory whiteleyTutorial to the c:\wtruns directory.**
  (Available from: http://www.mza.com/doc/PPT/whiteleytutorial.zip)
- **Rename the directory to wttut.**
- **Display the directory properties. Uncheck read-only. Click OK. When it asks, tell it to propagate the change to subdirectories.**
- **Close the tve and restart it.**
- **Open the System Editor window.**
- **File->Open->Browse…, traverse to c:\wtruns\wttut and select the system TutI.**





- **If any subsystems are marked 'obsolete', save the system to update them**

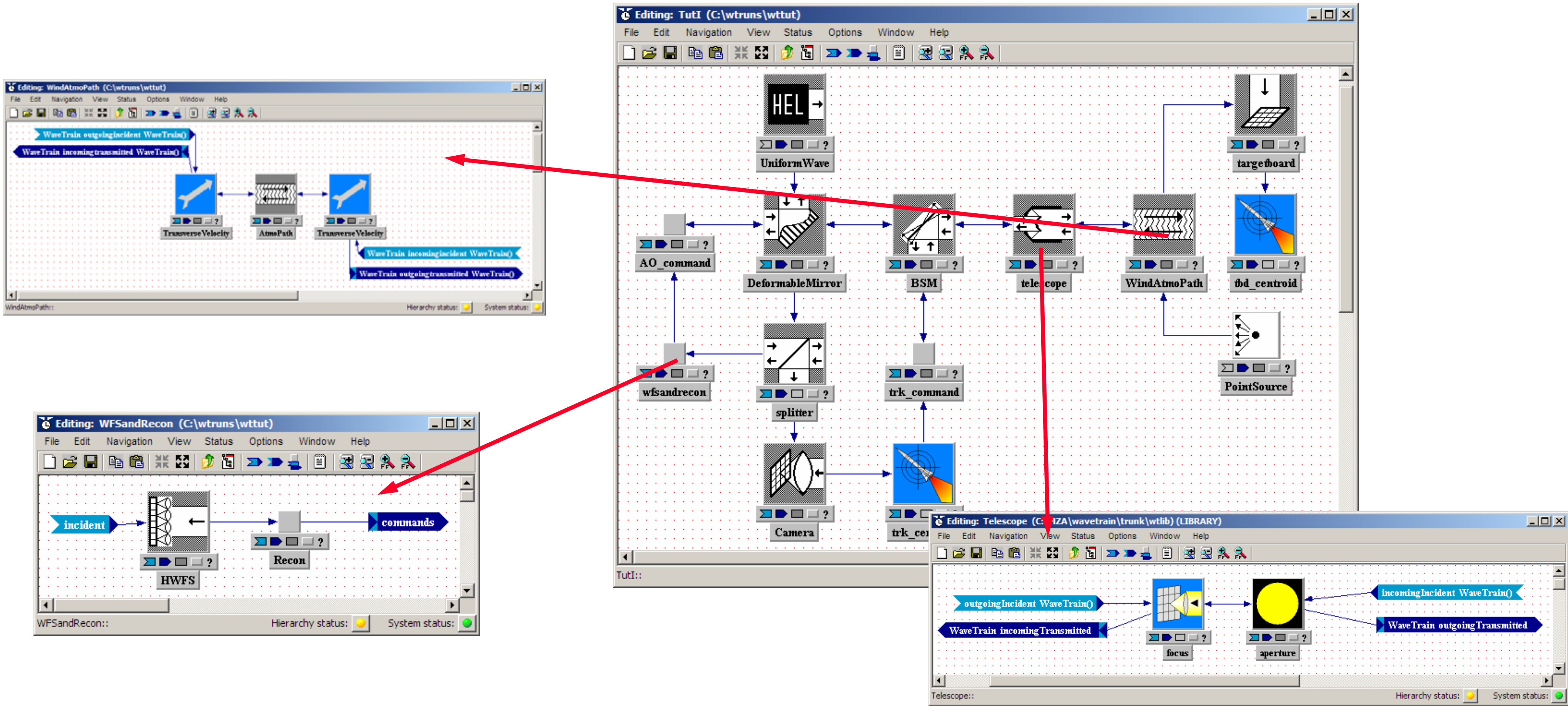
The Block Diagram



MZA Associates Corporation

Poke Around

- **Navigate** into **wfsandrecon**. Go back up.
- **Navigate** into **atmosphericpath (WindAtmoPath)**. Go back up.
- **Navigate** into **telescope** (it's a library system). Go back up.
- **Display** various **inputs**, **outputs**, and **parameters** to get a feel for the model.

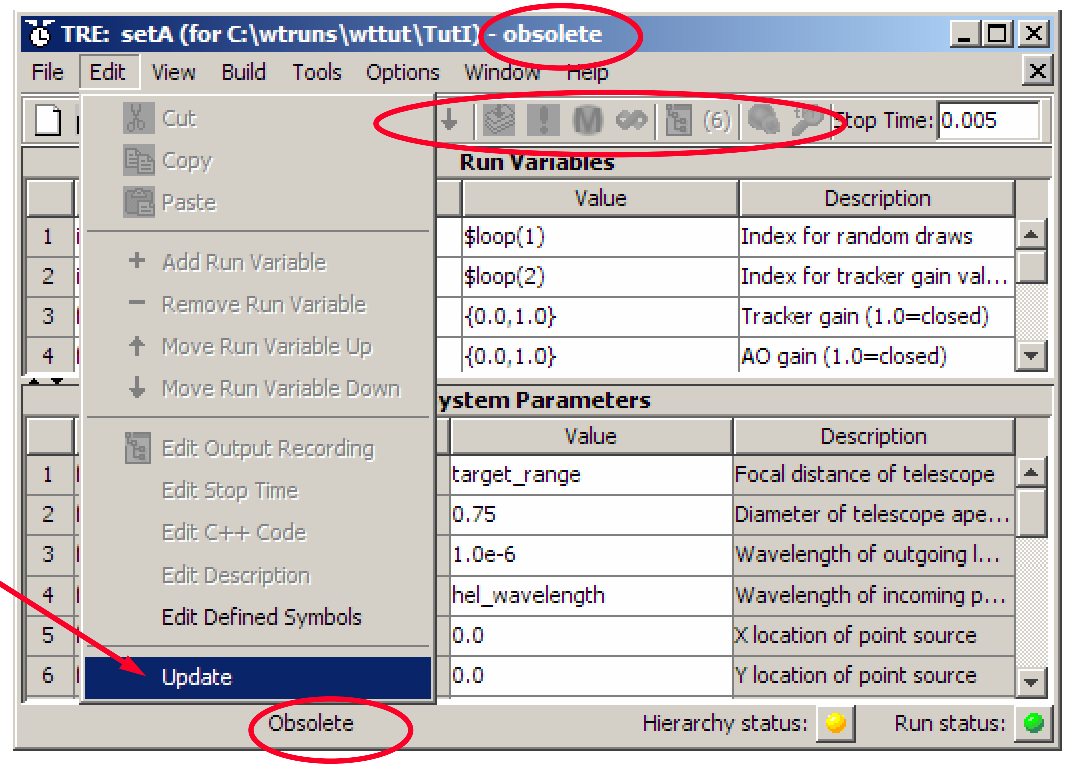


MZA Associates Corporation

Make a Run

- **Start the TRE (Runset Editor)**
- **File->Open…->Tutl->SetA**

- **If the system was modified more recently than the runset, the runset will be marked "obsolete" & the toolbar is grayed out. Update the runset from the Edit menu**



- **Inspect the Runset.**
  - **This Runset has two runs, looping only over gain_index.**
  - **gain_index is used to subscript trk_gain_values and AO_gain_values.**
  - **The first run is open-loop because trk_gain_values[0] and AO_gain_values[0] are zero.**
  - **The second run is closed-loop because trk_gain_values[1] and AO_gain_values[1] are one.**

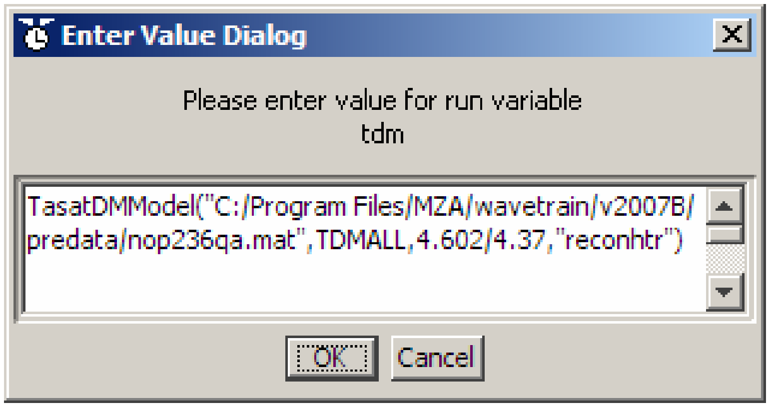
- **C++ and the tve use zero-based arrays.**
- **Matlab uses one-based arrays.**

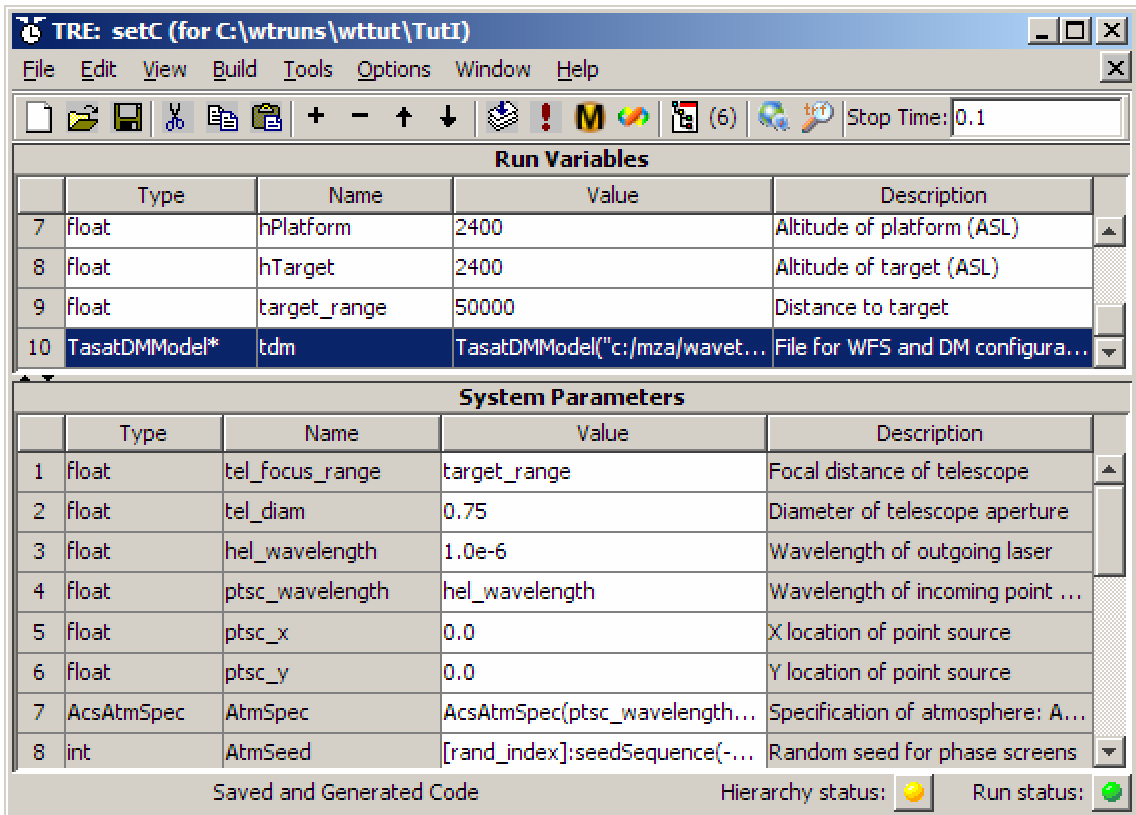
MZA Associates Corporation

Make a Run

- Create a new runset: **File->Save As…, SetC.**

- **Change Stop Time** to **0.1.**
- **Change** setting for tdm to load the file "C:/Program Files/MZA/wavetrain/v2007B/predata/nop236qa.mat "



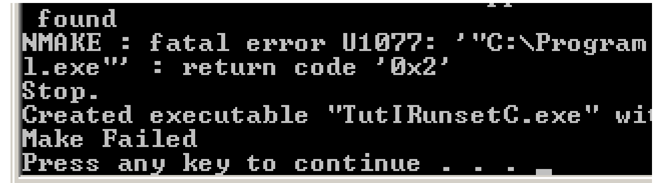


- **Replace the first three digits of the number in the AtmSeed setting to your favorite three digit number.**
- **Build->Execute**.
- The run will take about ten minutes.

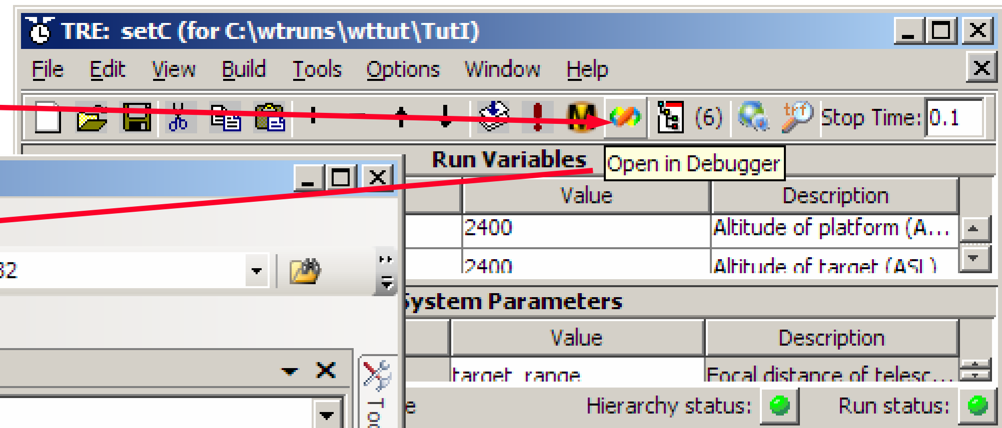
MZA Associates Corporation

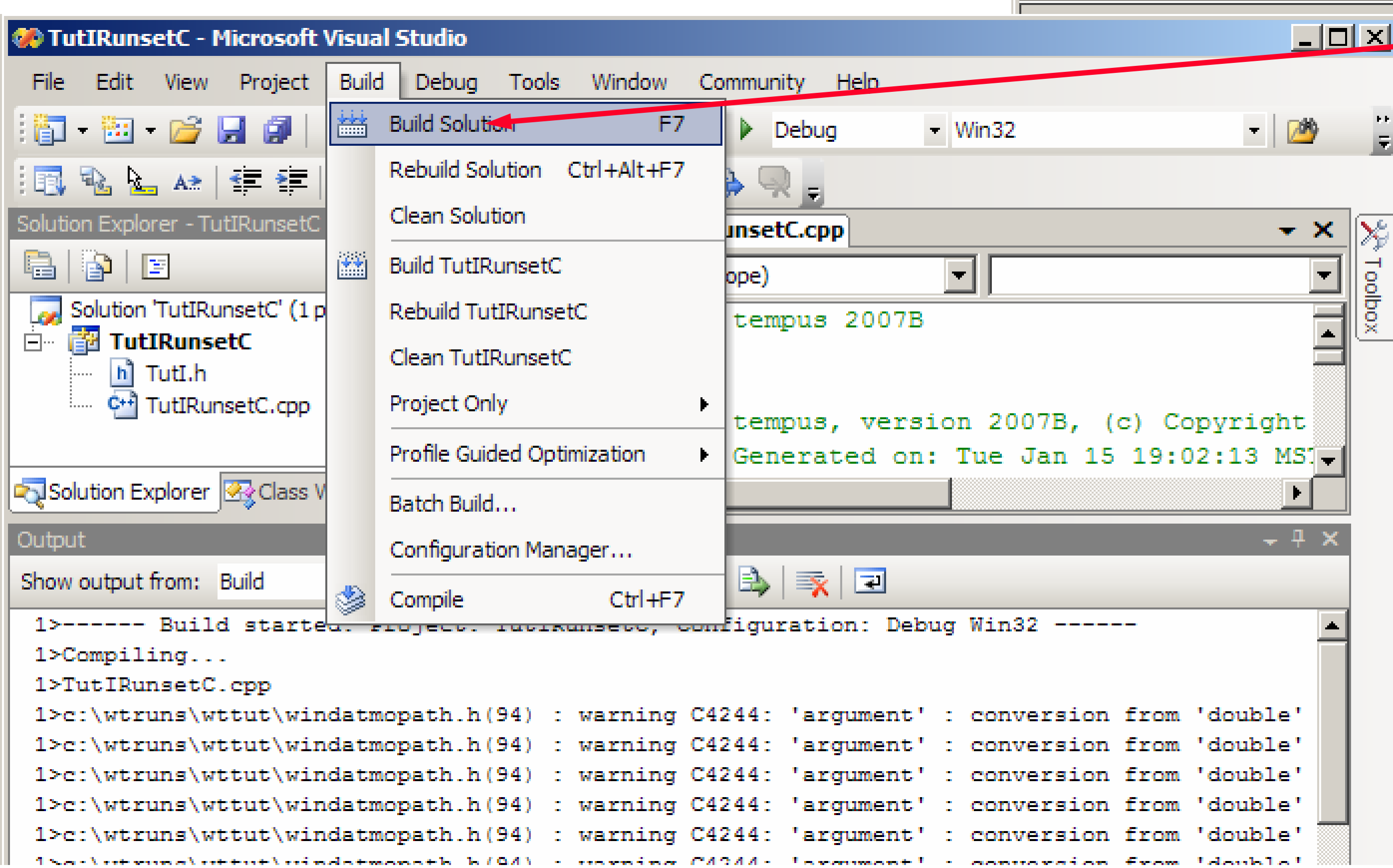
… Debug …

- **The system and runset will not compile as-is, due to updates to WaveTrain since they were written.**



- **This gives us a chance to show how to launch the debugger …**



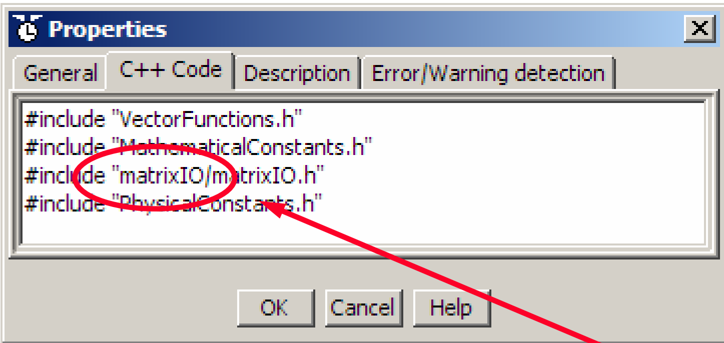


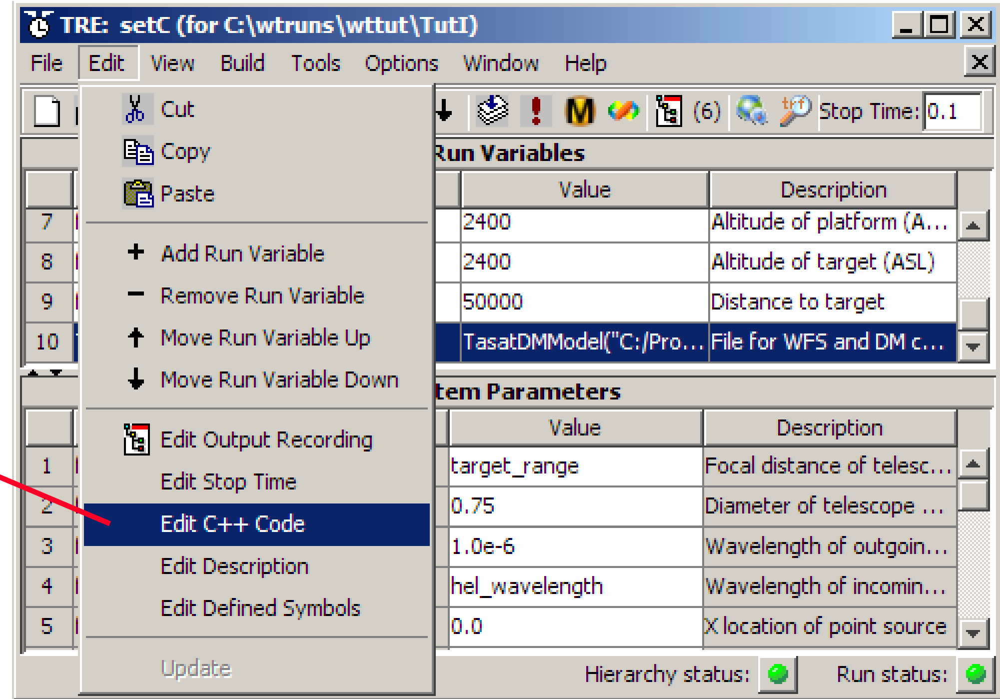
- **Find problems in C++ via Visual Studio**
- **Fix in system / runset**

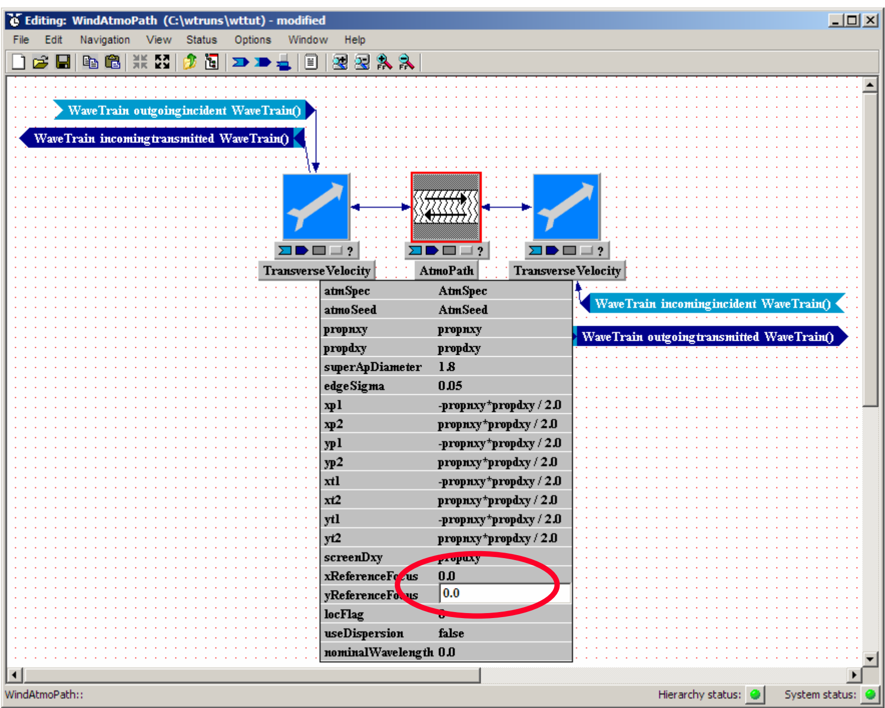
MZA Associates Corporation

Fix & Run

- **Fix runset**







- **Fix system WindAtmoPath**

Create a Movie

shops4.m

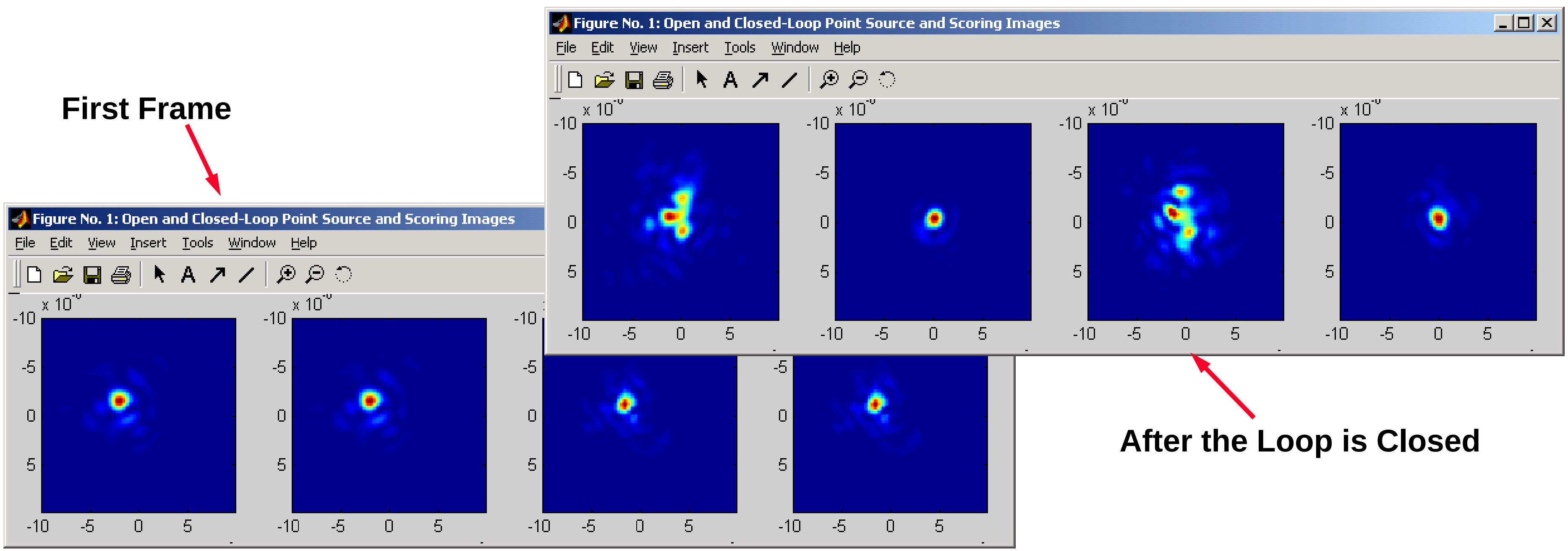
- **Start Matlab.**

>> cd c:/wtruns/wttut

>> tia=trfopen('TutlRunsetC1.trf');sia=trfload(tia);trfvlist(sia)

- **Review and run the script in shops4.m to display a movie showing open-loop and closed loop runs side-by-side.**
- **To repeat the movie use movie(mb).**
- **Verify that the open-loop and closed-loop runs begin with the same conditions with movie(mb(1)).**

First Frame

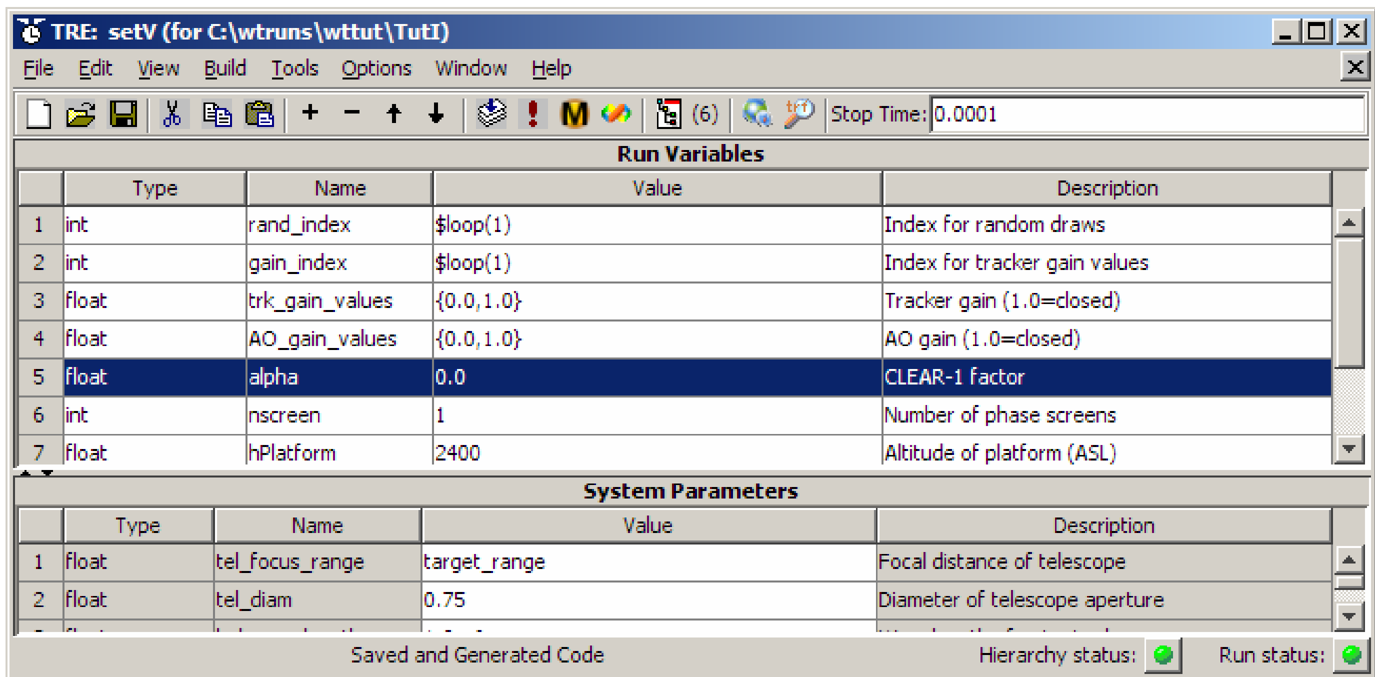


After the Loop is Closed

MZA Associates Corporation

Run a Vacuum Case

- To calculate Strehl, we need a propagate through a vacuum.
- Go to the TRE (Runset Editor).
- File->Save As…, SetV.
- Change Stop Time to 0.0001.
- Change gain_index to $loop(1).
- Change nscreen to 1.
- Change alpha to 0.0.
- Build->Execute.
- This will run fast.



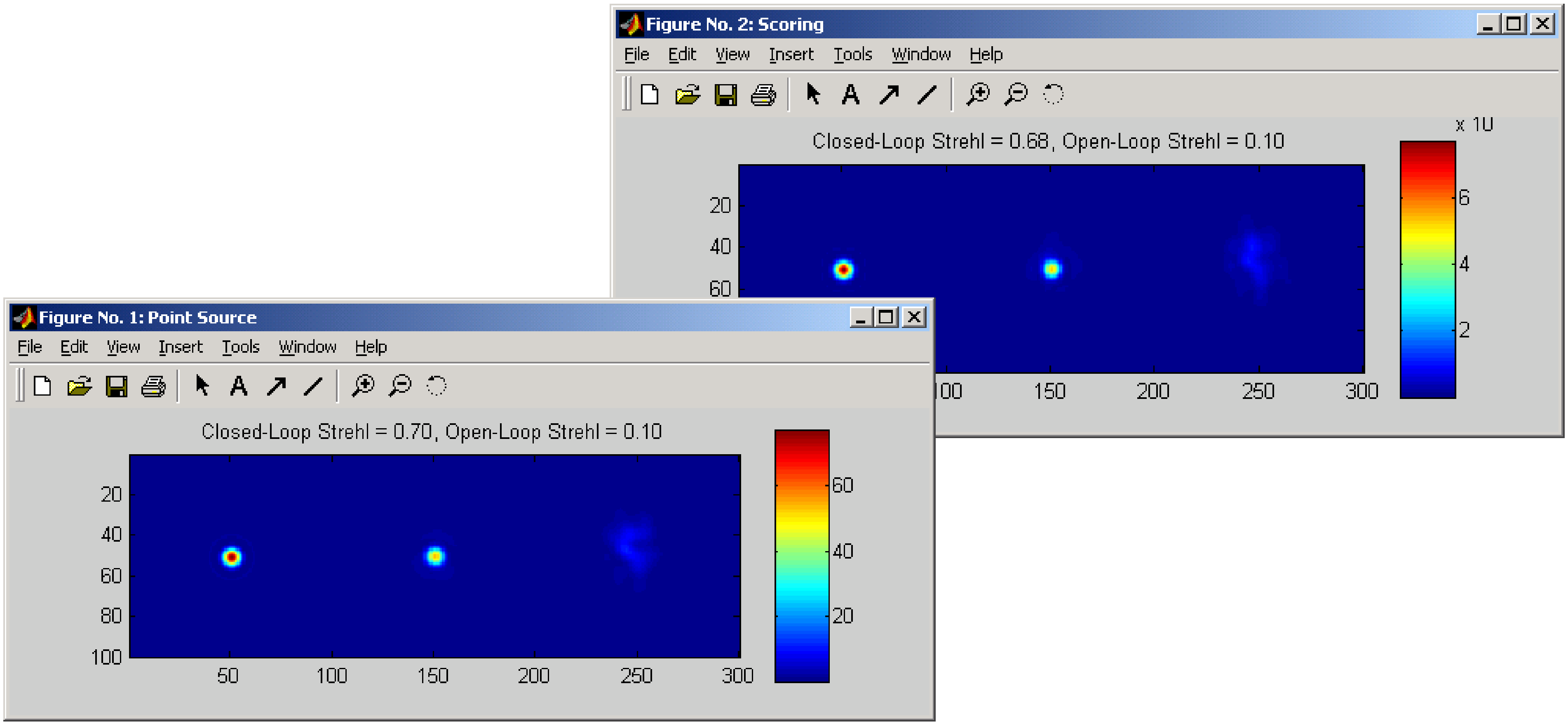
Compute Strehl

shops5.m

MZA Associates Corporation

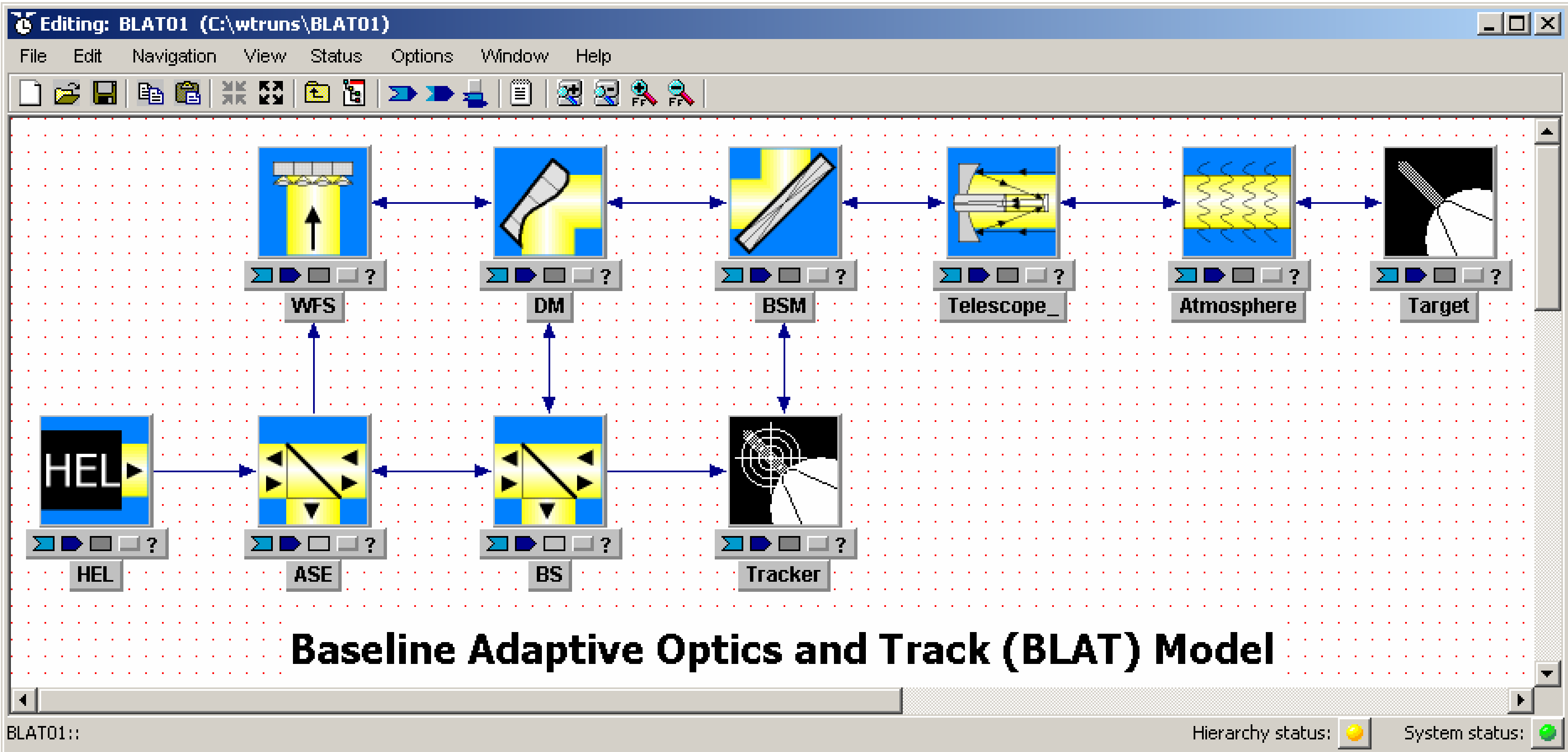
>> tiv=trfopen('TutlRunsetV1.trf');siv=trfload(tiv);trfvlist(siv)

- Review and run the script in shops5.m to which computes the time-averaged open and closed-loop Strehl.
  - Be sure to look at the use of trfavg in the script.
- Also try figure;mesh([dlimg,climg,olimg]) and figure;mesh([dltbd,cltbd,oltbd]).



Baseline Adaptive Optics and Track (BLAT) Model

MZA Associates Corporation

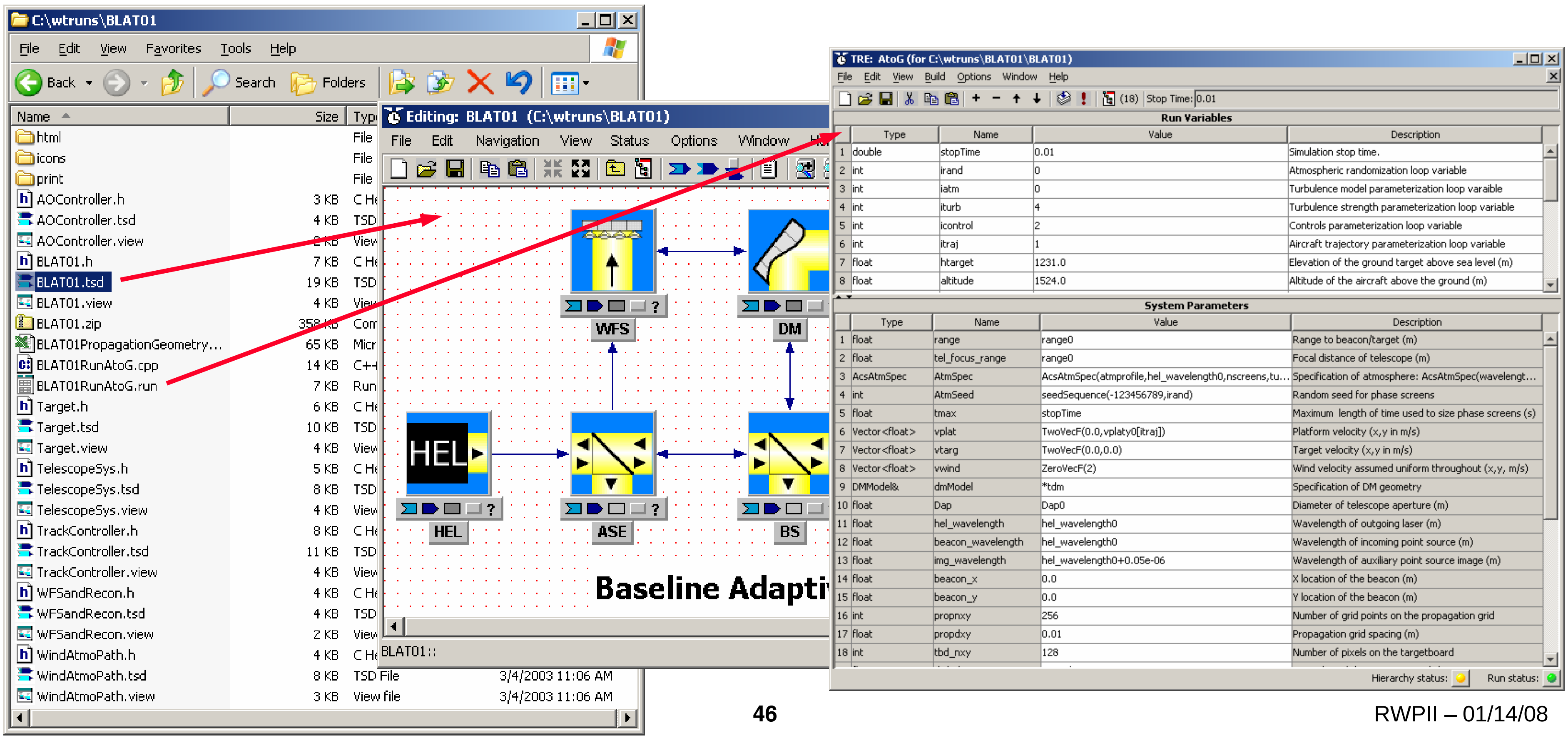


A closed-loop AO and track system using a standard tip-tilt centroid tracker and a tilt-removed least-squares reconstructor on a Shack-Hartmann wavefront sensor.

MZA Associates Corporation

Open the BLAT01 Model

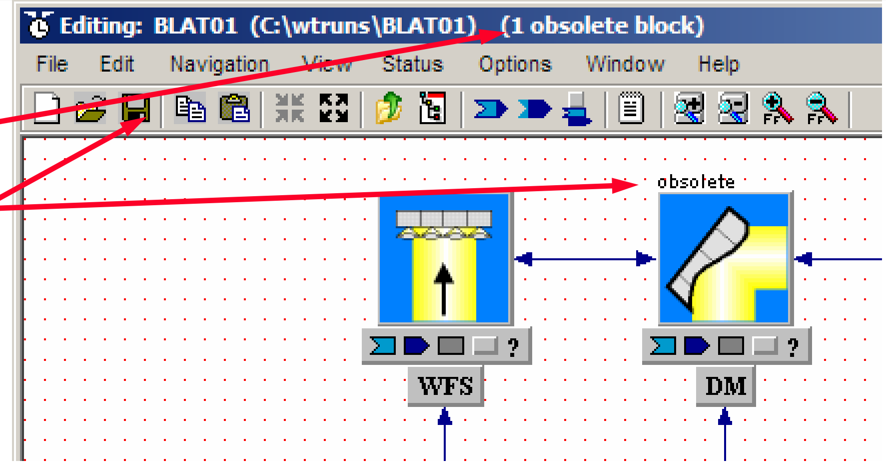
- **Copy the directory BLAT01 from C:\Program Files\MZA\wavetrain\v2007B\examples to the c:\wtruns directory.**
- **Display the directory properties. Uncheck read-only. Click OK. When it asks, tell it to propagate the change to subdirectories.**
- **In a file browser window Navigate to the c:\wtruns\BLAT01 directory.**
- **Double-click on BLAT01.tsd and then on BLAT01AtoG.run.**

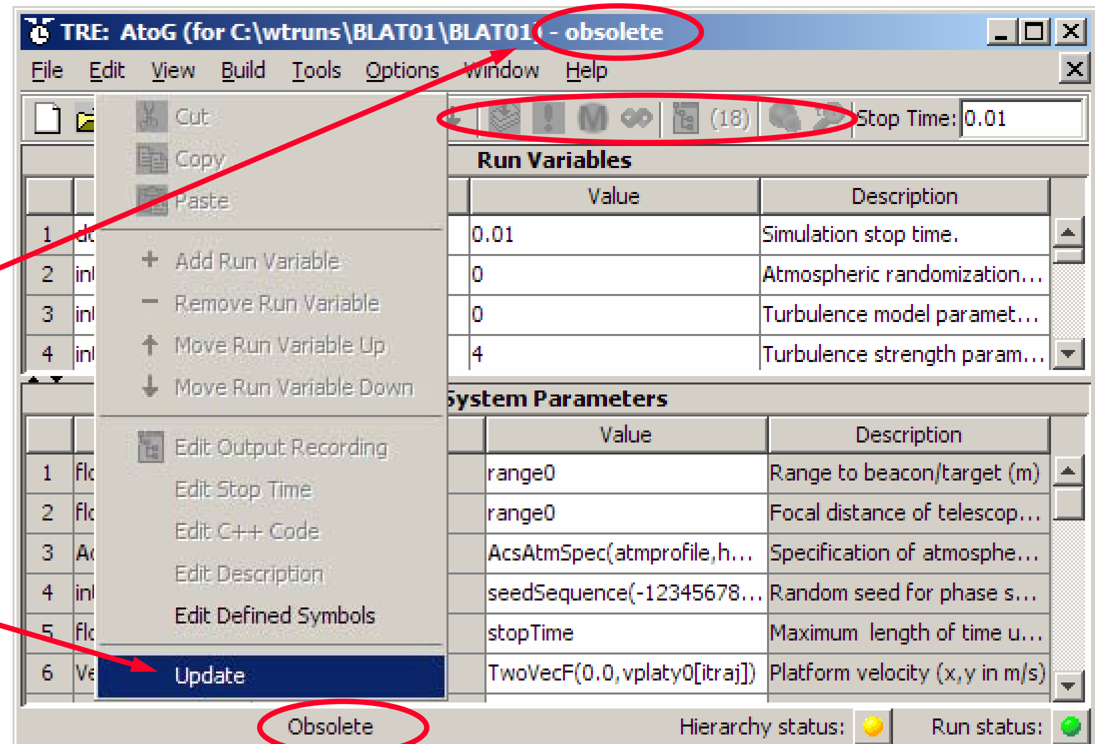


Aside: ‘Obsolete’ systems & runsets

MZA Associates Corporation

- **If any library components have been updated since the system was last saved they will be marked ‘obsolete’**
- **Simply save the system to update any obsolete subsystems**





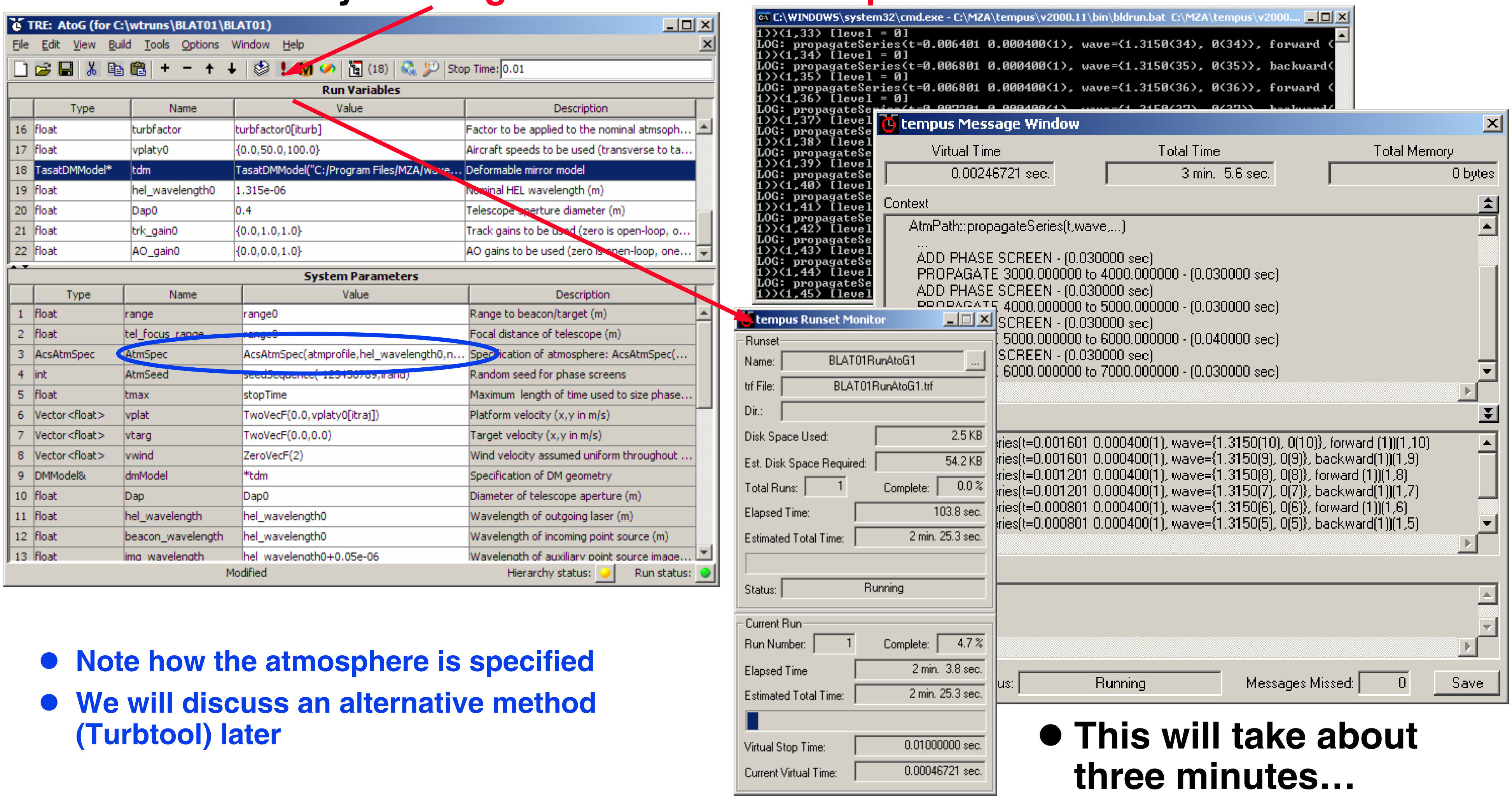
- **If the system was modified more recently than the runset, the runset will be marked “obsolete” & the toolbar is grayed out.**
- **Update the runset from the Edit menu**

MZA Associates Corporation

Run the BLAT01 Model

- **Edit the setting for tdm so that the file loaded is "C:/Program Files/MZA/wavetrain/2007B/predata/fdf.mat"**

Run the model by clicking on the exclamation point.



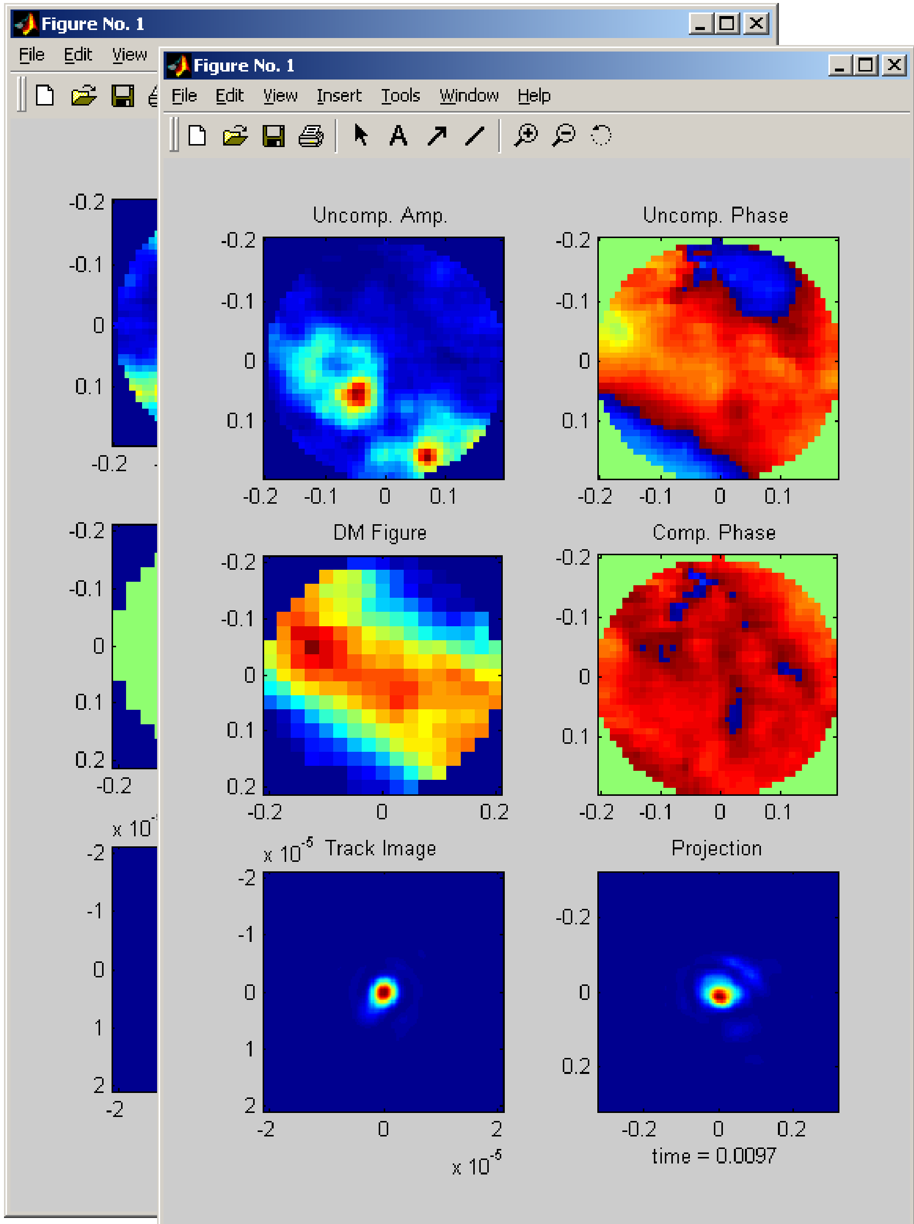
- **Note how the atmosphere is specified**
- **We will discuss an alternative method (Turbtool) later**

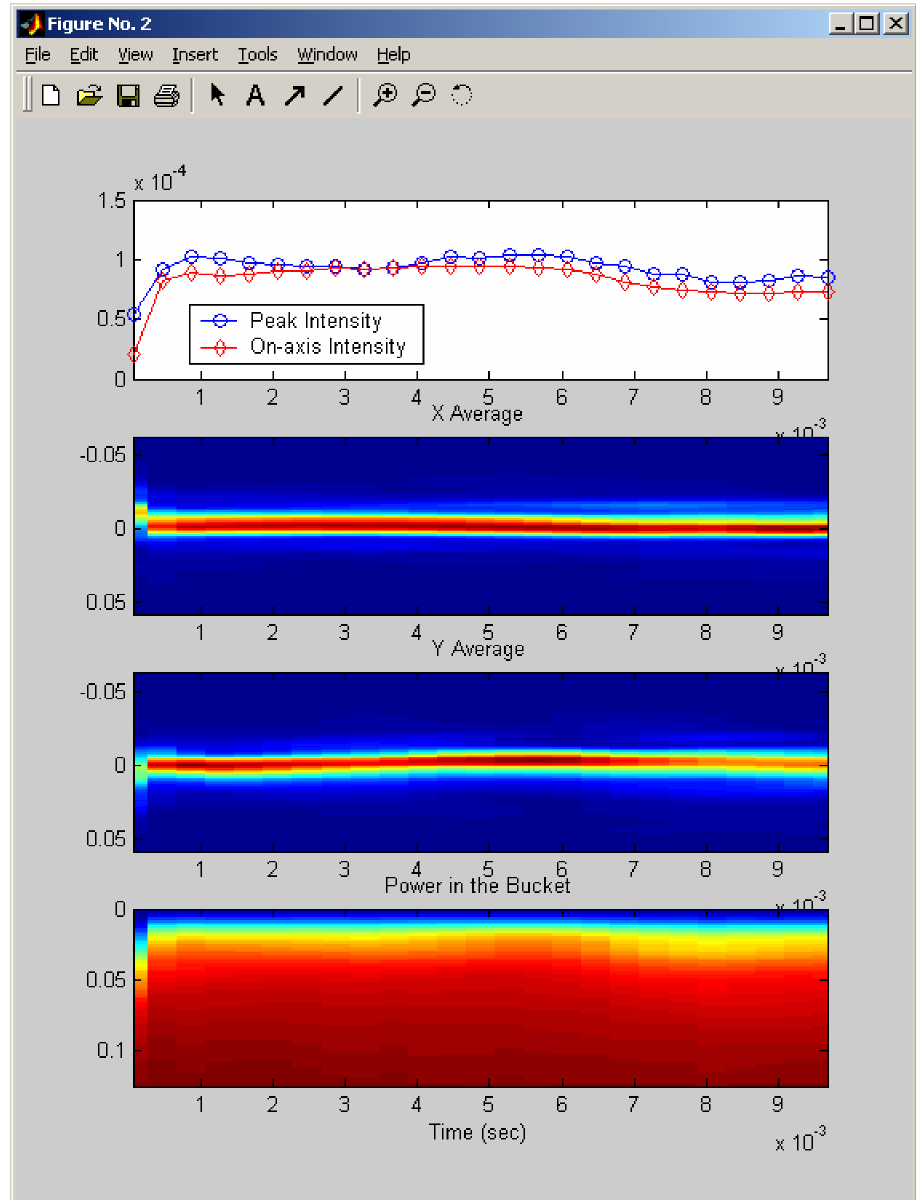
- **This will take about three minutes…**

MZA Associates Corporation

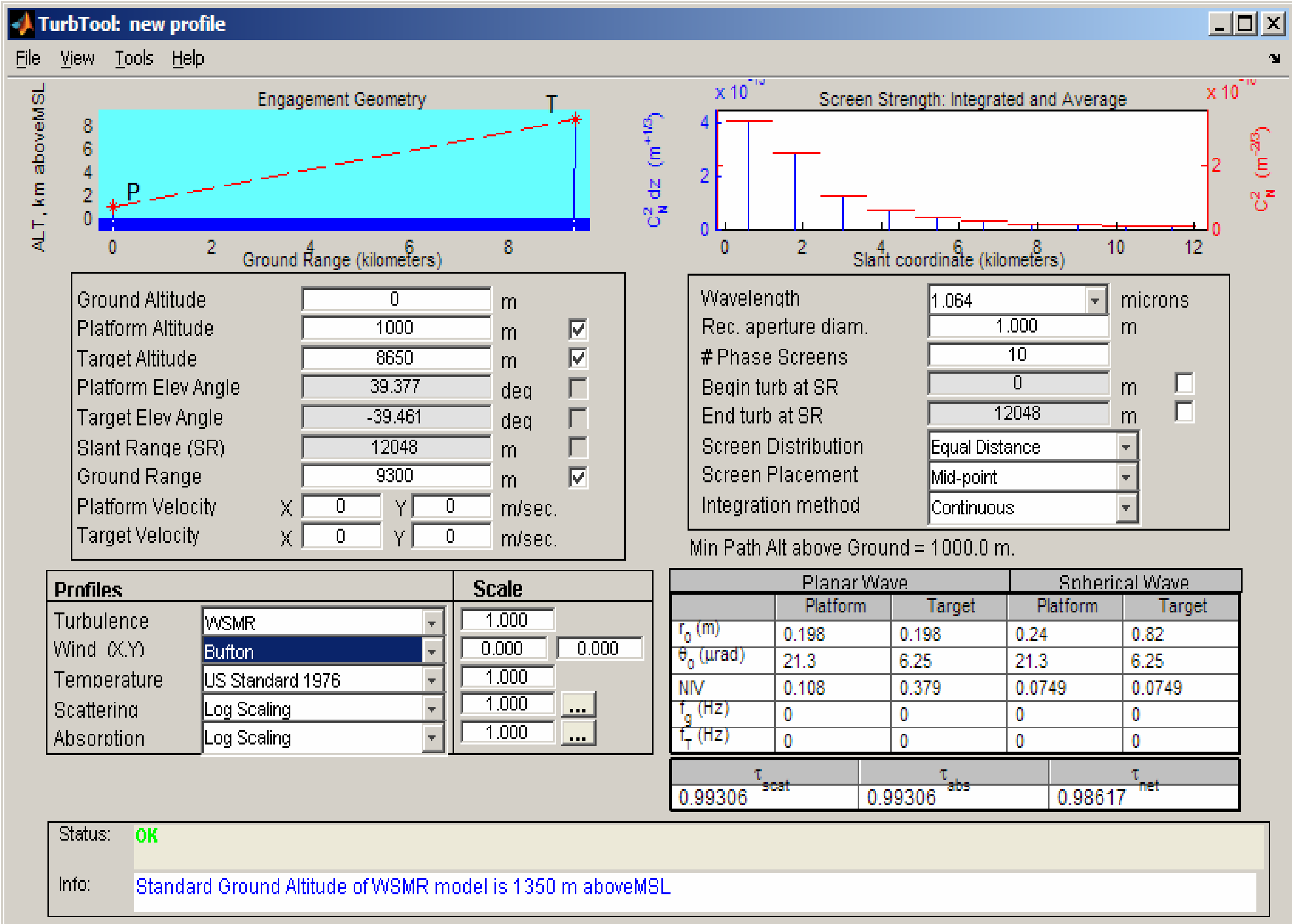
Process and Plot the Data

BLAT01_01.m (in c:\wtruns\ BLAT01\)





Turbtool: Atmosphere Specification



WaveTrain Object Orientation

- **Using the OOP model, each of the blocks in the block diagram are objects of type** `System`**.**
  - **Systems have static** `Parameter`**s, dynamic** `Input`**s, and generate dynamic** `Output`**s.**
  - `System`**s must respond to requests for information following the requirements of the tempus** `System` **interface specification.**
  - **This is done by implementing "virtual methods" (C++ polymorphism).**
- **The key to the wave optics aspect of the code is the object type** `WaveTrain` **which implements the relationship between light source and receiver necessary to compute physical propagation quantities.**
  - `WaveTrain` **is used as both an** `Input` **and an** `Output` **type.**
  - **A** `Wave` **represents coherent light and travels through** `WaveTrain` **connections.**
  - **A** `WaveSource` **is a** `System` **that generates** `Wave`**s.**
  - **A** `WaveMap` **is a** `System` **that modifies an incident** `Wave` **to create a transmitted** `Wave`**.**
  - **A** `WaveReceiver` **is a** `System` **that accepts and integrates incident** `Wave`**s to compute a measurement (usually a sensor** `Output`**).**

WaveTrain Code

WaveTrain has "scripts" too

MZA Associates Corporation

- **Each composite system declares and initializes its subsystems:**

```
pointsource(this, "pointsource", wavelength, 1.0e6, 0.0, 0.0),
transversevelocity1(this, "transversevelocity1", -wind, 0.0, 0.0, 0.0),
transversevelocity3(this, "transversevelocity3", wind, 0.0, 0.0, 0.0),
atmosphericpath1(this, "atmosphericpath1",
           AcsAtmSpec(wavelength,nscreen,clear1Factor,hPlatform,hTarget,range),
           atmoSeed, propnxy, propdxy, 1.8, 0.05,
           -propnxy*propdxy/2.0, propnxy*propdxy/2.0, -propnxy*propdxy/2.0, propnxy*propdxy/2.0,
           -propnxy*propdxy/2.0, propnxy*propdxy/2.0, -propnxy*propdxy/2.0, propnxy*propdxy/2.0,
           propdxy, 0.0, 0.0, 0),
camera1(this, "camera1", 1.0, wavelength, wavelength, apdiam/propdxy,
           propdxy, 64, wavelength/apdiam, 0.0),
simplefieldsensor1(this, "simplefieldsensor1", wavelength, apdiam/propdxy, propdxy),
telescope1(this, "telescope1", range, apdiam/2.0, 0.0),
incomingsplitter1(this, "incomingsplitter1"),
```

- **Then the subsystems inputs and outputs are connected:**

```
simplefieldsensor1.incident <<= incomingsplitter1.incomingTransmitted2;
camera1.incident <<= incomingsplitter1.incomingTransmitted;
incomingsplitter1.incomingIncident <<= telescope1.incomingTransmitted;
telescope1.incomingIncident <<= transversevelocity3.incomingTransmitted;
transversevelocity3.incomingIncident <<= atmosphericpath1.incomingTransmitted;
atmosphericpath1.incomingIncident <<= transversevelocity1.incomingTransmitted;
transversevelocity1.incomingIncident <<= pointsource.transmitted;
```

- **Then the simulation is run:**

```
advanceTime(stopTime);
```

blue names are systems
green names are inputs
red names are outputs
cyan names are regular variables

MZA Associates Corporation

A Complete WaveTrain Run

```
#include "tempus.h“
#include "Recorders.h“
#include "FileSys.h"

#include "PointSource.h"
#include "AtmoPath.h"
#include "Telescope.h"
#include "Camera.h"

#ifndef NO_TEMPUS_SMF_MONITOR
  #include "TempusStatusSMF.h"
#endif

main(int argc, char* argv[])
{
//
// Decoration related to monitoring the system during the run.
//
#ifndef NO_TEMPUS_SMF_MONITOR
  double stopTime = 0.0050;
  char *___outfile = "WtDemoRunHand.trf";
  char *___trfname;
  char *___smfname;
  parseName(argc, argv, ___outfile, &___smfname, &___trfname, stopTime);
  TempusStatusSMFWriter ___smfWriter(___smfname, ___trfname, "", 1);
  setCurrentSMF(&___smfWriter);
#endif
Universe ut1("Hand");
//
// Construction of all the systems. Variables could be used in the parameters
// below rather than the constants.
//
  PointSource pointsource(NULL, “ps", 1.0e-06, 1.0e+06, 0.0, 0.0);
  AtmoPath atmosphericpath(NULL, “ap",
                  AcsAtmSpec(1.0e-06,10,2.0,2413.0,2728.0,52600.0),
                  -765432189, 256, 0.02, 1.8, 0.05,
                  -256*0.02/2.0, 256*0.02/2.0, -256*0.02/2.0, 256*0.02/2.0,
                  -256*0.02/2.0, 256*0.02/2.0, -256*0.02/2.0, 256*0.02/2.0,
                  0.02, 0.0, 0.0, 0);
  Telescope telescope(NULL, “tel", 52600.0, 1.5/2.0, 0.0);
  Camera camera(NULL, “cam", 1.0, 1.0e-06, 1.0e-06, 1.5/0.02, 0.02, 64,
                 1.0e-06/1.5, 0.0);
//
// Connection of the systems.
//
  atmosphericpath.incomingIncident <<= pointsource.transmitted;
  telescope.incomingIncident <<= atmosphericpath.incomingTransmitted;
  camera.incident <<= telescope.incomingTransmitted;
//
// Construction and connection of non-connected inputs.
//
  Output<bool> camera_on(&camera, "cam_on", true);
  Output<double> camera_ei(&camera, "cam_ei", 1.0e-3);
  Output<double> camera_el(&camera, "cam_el", 1.0e-6);
  Output<double> camera_si(&camera, "cam_si", -1.0);
  camera.on <<= camera_on;
  camera.exposureInterval <<= camera_ei;
  camera.exposureLength <<= camera_el;
  camera.sampleInterval <<= camera_si;
//
// Decoration related to recording the outputs.
//
  ParamSet pst1;
  RecorderFile rft1(NULL, "rft1", ___trfname, ParamSet_stringify(pst1),
       pst1);
  GridRecorder<float> rft11(NULL, "rft11", "camera.fpaImage",
       "Grid<float>", "image", true, (float)0.0, 0.0);
  rft11.dr <<= rft1.dr;
  rft11.i <<= camera.fpaImage;
//
// Run the simulation.
//
  advanceTime(stopTime);
}

// Black code is always the same.
// Blue code is dependent on the problem.
// Green code is administrative in nature.
// Gray code supports optional functionality.

// To run:
//      setupwt
//      mktr WtDemoRunHand
//      WtDemoRunHand
```

Code Generation Strategy

- Simple systems are built by the GUI as System templates.
  - The programmer is expected to implement virtual methods which define the system's behavior.
  - Because many systems have common features, inheritance and polymorphism is used a lot.
- Composite systems are coded as complete Systems
  - Parameters are constructor arguments.
  - External inputs and outputs are member objects.
  - Subsystems are declared and initialized using expressions involving the parameters of the system.
  - Subsystems are connected using the simple overloaded operator <=.
  - Miscellaneous code handles default unconnected inputs.
- Runsets are coded as the main program.
  - The code contains explicit loops for loop variable.
  - The run variables and top-level system parameters are declared and set. Run variables and system parameters which are dependent on loop variables inside the appropriate loops.
  - The top-level system is constructed using the system parameters.
  - Recording systems are constructed and connected.
  - Each run is executed with a call to advanceTime(…).
  - There is miscellaneous code which takes care of runset monitoring and setting up the output trf file.