

# **Setting up a Wave-Optics Model: Modeling a Gaussian Beam to a Focus**

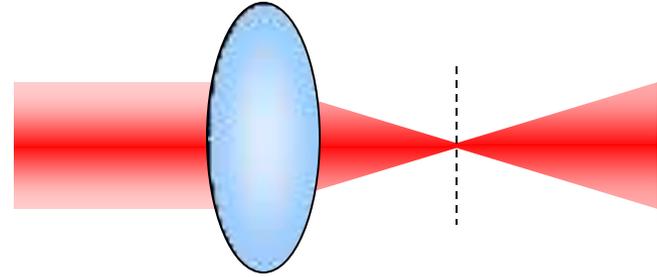
Justin Mansell, Ph.D.

# Outline

- The goal of this effort is to outline the procedure for setting up a basic model in WaveTrain.
- We'll use the case of propagating a Gaussian beam to a focus as an example because it is simple enough to have an analytical solution against which we can compare our results.

# Analytical Solution

- Setup:
  - 1-mm waist  
Gaussian
  - 1 m focal length  
lens
  - 1  $\mu\text{m}$  wavelength



For a circular aperture (for comparison):

$$D_{spot} = 2.44 \frac{f \lambda}{D_{beam}} = 2.44 \frac{(1m)(1\mu m)}{(2mm)} = 1.22mm$$

For a Gaussian:

$$w_{spot} = \frac{f \lambda}{\pi w_{beam}} = \frac{(1m)(1\mu)}{\pi (1mm)} = 0.318mm$$

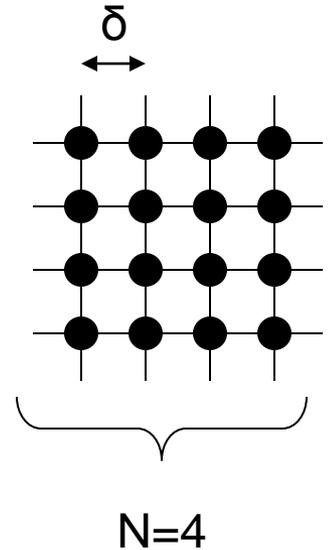
$$D_{spot} \approx 2w_{spot} = 0.637mm$$

# WaveTrain Model Setup

1. Determine the propagation mesh and method of propagation.
2. Setup the model.
3. Analyze the results.

# Propagation Introduction

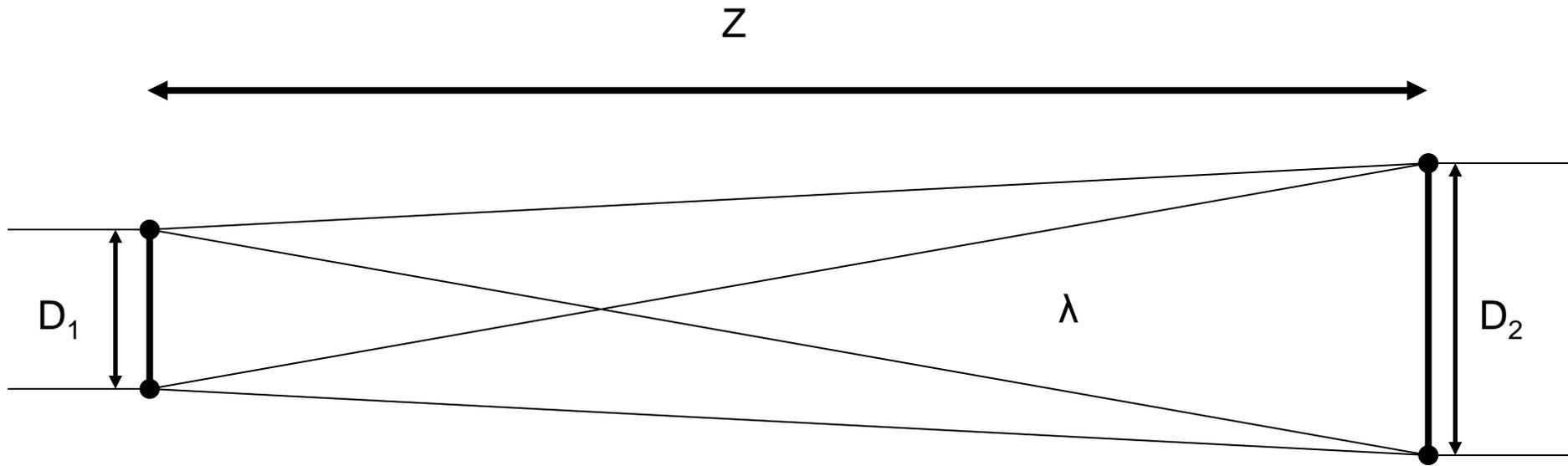
- Wave-optics models light as a 2D grid of complex numbers which represent magnitude and phase of a beam of light.
  - This grid is specified by the sample spacing ( $\delta$ ) and the number of samples in one dimension (N).
- WaveTrain uses a convolution propagator (see J. Goodman's Fourier Optics Ch. 4) which involves a 2D Fourier transform of the field, multiplication by a propagation kernel, and then a 2D inverse Fourier transform.
  - This is sometimes referred to as a 2-step propagator
  - It is done this way to control the mesh spacing in a way that a simple 1-step propagator does not allow



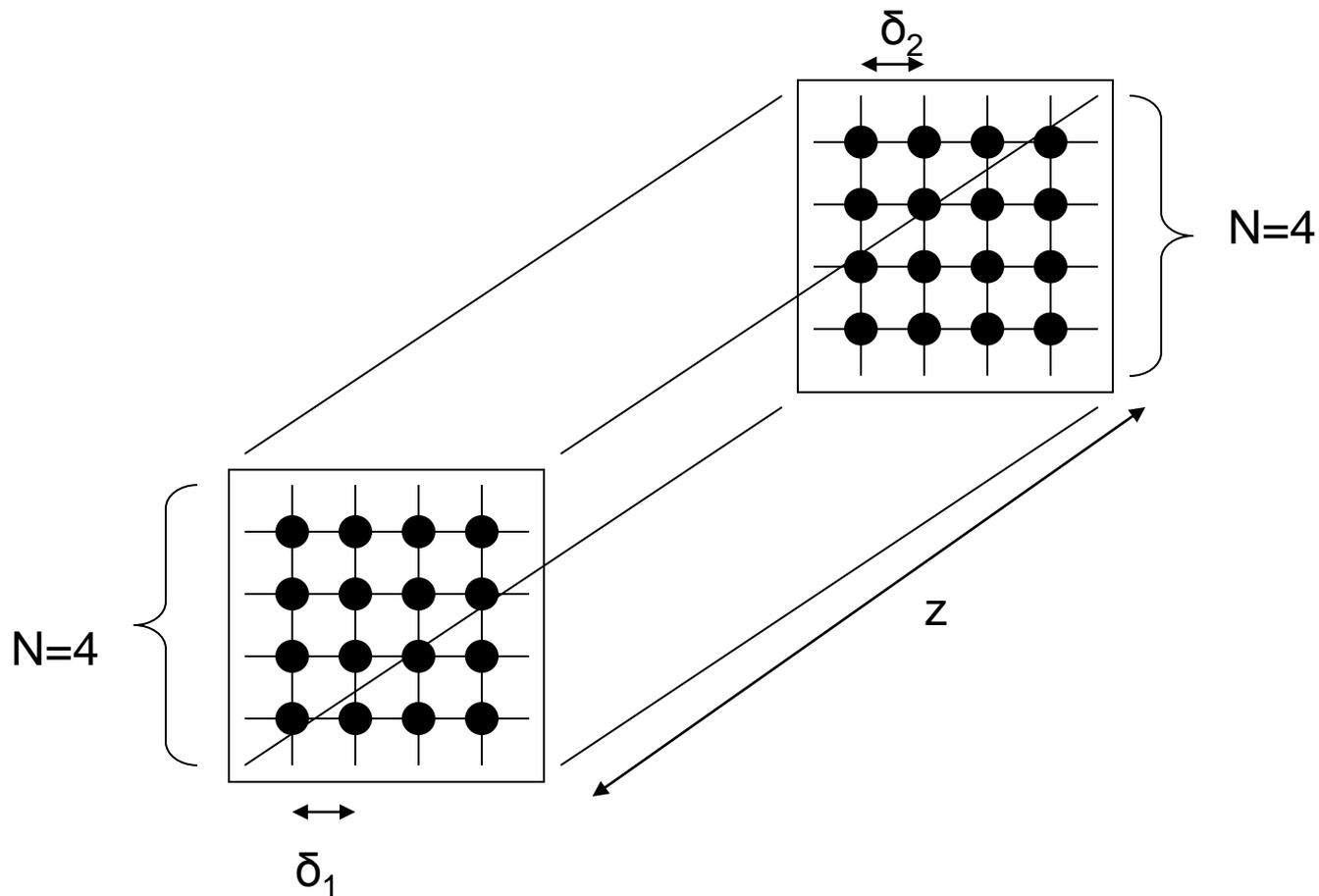
# Representing a Field on a Mesh

- When choosing mesh parameters for wave-optics modeling, Nyquist sampling theory needs to be applied to not only the amplitude, but the phase as well.
- We have derived equations for determining the mesh parameters based on the propagation setup.
  - The propagation setup requires specifying a diameter of interest in the first and final planes, a wavelength, and a distance between the two planes.
  - We have also included the capability of including the effects of atmospheric aberrations, but the problem we are addressing here does not require this capability, so it will not be substantially addressed here.
- These relationships are codified into an Excel worksheet that we will introduce later.

# Illustration of Propagation Specification



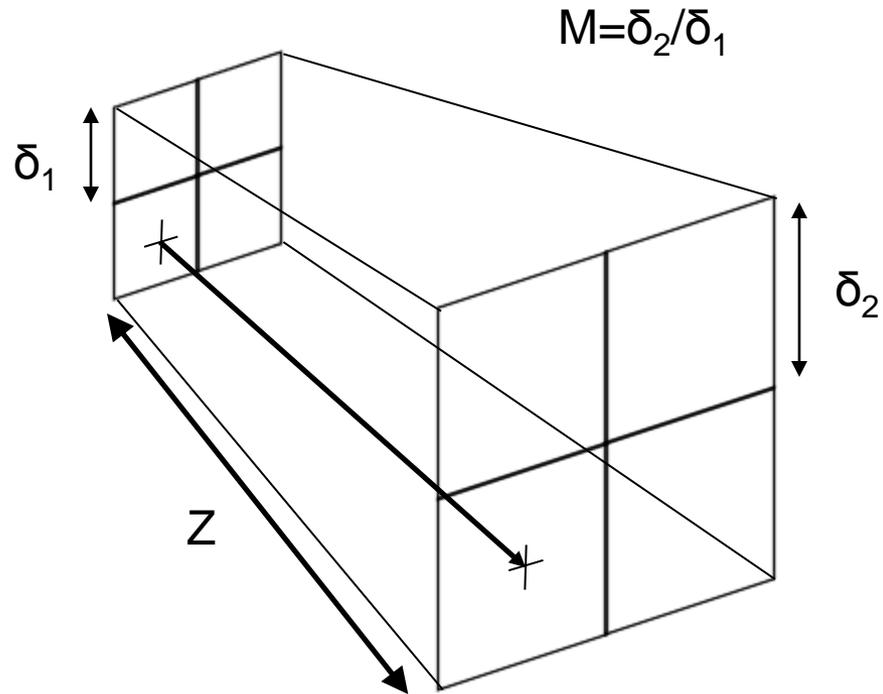
# Illustration of Propagation Parameters



# SWP vs PWP

- The 2-step Fourier propagation as described above is done relative to a planar wavefront reference, which enables the mesh spacing to be preserved.
  - This is called plane wave propagation (PWP).
- In some problems, like propagation to a focus, it makes sense to propagate relative to a reference curvature for two reasons:
  - This allows the mesh sample spacing to be changed during the propagation.
  - This reduces the number of mesh points required because the curvature is no longer required to be represented on the mesh (again Nyquist sampling requirements).
- This is called spherical wave propagation (SWP).

# Illustration of a Diverging SWP



# Propagation Setup

- MZA provides for free on its website a worksheet for calculating the mesh required for a given propagation.
- We will be working with a simplified version of this worksheet shown on the right.

## Wave-Optics Propagation Geometry Tool

### Basic Inputs

$\lambda$	1.03E-06
$\Delta z$	2,000
$D_1$	0.15
$D_2$	0.27

### Color Key

user input
advanced input
calculated

### Basic Turbulence Inputs

$c_{\text{turb}}$	3
$r_{01}$	0.05
$r_{02}$	0.05

### Advanced Turbulence Inputs

	Calculated	Calculated
$\theta_{1 \text{ blur}}$	0.0000618	6.18E-05
$\theta_{2 \text{ blur}}$	0.0000618	6.18E-05
$D_1'$	0.3972	0.2736
$D_2'$	0.3936	0.3936

### Two Step DFT w/o Filtering

#### Minimizing N

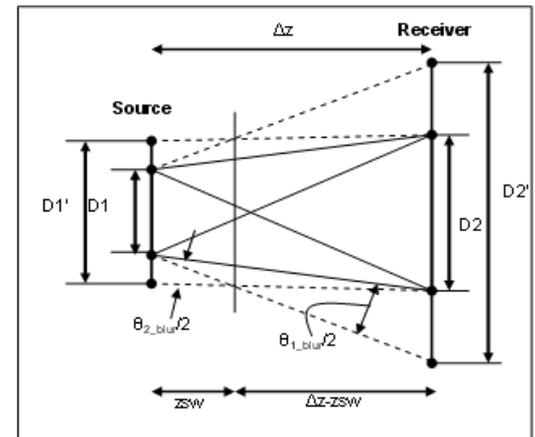
$\delta_1 =$	0.00261687
$\delta_2 =$	0.002593152
$N \geq$	303.5687767
$N_{\text{FFT}} =$	512

#### Plane Wave

$\delta_1 = \delta_2 <$	0.002604957
$\delta_1 = \delta_2 =$	0.002
$N \geq$	455.2
$N_{\text{FFT}} =$	512

#### General

$\delta_1 =$	0.001
$\delta_2 < \text{(dependent)}$	0.004195368
$\delta_2 =$	0.001
$\delta_1 < \text{(dependent)}$	0.004224593
$N \geq$	1425.4
$N_{\text{FFT}} =$	2048



# Worksheet Setup

## Wave-Optics Propagation Geometry Tool

### Basic Inputs

$\lambda$	1.00E-06
$\Delta z$	1
$D_1$	0.006
$D_2$	0.006

### Basic Turbulence Inputs

$c_{\text{turb}}$	0
$r_{01}$	0.05
$r_{02}$	0.05

Color Key
user input
advanced input
calculated

The top of the sheet has inputs in blue squares.

$c_{\text{turb}} = 0$   
because we have no turbulence in this case.

#### Definitions:

$\lambda$  = wavelength (m)

$\Delta z$  = propagation distance (m)

$D_1$  = input diameter of interest (m)

$D_2$  = output diameter of interest (m)

$c_{\text{turb}}$  = turbulence capture factor (0 to 4)

$r_{01}$  = Fried's coherence length as seen from the input aperture (m)

$r_{02}$  = Fried's coherence length as seen from the output aperture (m)

# Propagation Worksheet Results

- We use the fast Fourier transform (fft) to implement our propagator in WaveTrain, which requires that the number of points be a power of two.
  - The  $N_{\text{FFT}}$  parameter is the nearest power of two.
- There are three different propagation parameters given in the worksheet:
  - Minimizing N - choose these parameters to minimize the number of samples required.
  - Plane Wave - choose these parameters to get proper sampling for a planar reference wave propagation.
  - General - these parameters provide the user the ability to pick both the mesh sample spacing in the input and output planes, but checks them against the minimum sample spacing in each plane to ensure proper sampling. This is good for trying to have one propagation match the next propagations mesh.

## Two Step DFT w/o Filtering

### Minimizing N

$\delta_1 =$	8.33333E-05
$\delta_2 =$	8.33333E-05
$N \geq$	144
$N_{\text{FFT}} =$	256

### Plane Wave

$\delta_1 = \delta_2 <$	8.33333E-05
$\delta_1 = \delta_2 =$	8.30E-05
$N \geq$	144.8686312
$N_{\text{FFT}} =$	256

### General

$\delta_1 =$	8.00E-05
$\delta_2 < \text{(dependent)}$	8.66667E-05
$\delta_2 =$	8.00E-05
$\delta_1 < \text{(dependent)}$	8.66667E-05
$N \geq$	153.125
$N_{\text{FFT}} =$	256

# Propagation Decisions

- Because the beam does not significantly change size during the propagation, we chose to start with a plane wave propagation (PWP).
- Based on the worksheet results, we chose to use the parameters from the minimizing N, which were plane wave ( $\delta_1 = \delta_2$ )
  - $N = 256$
  - $\delta = 83\mu\text{m}$

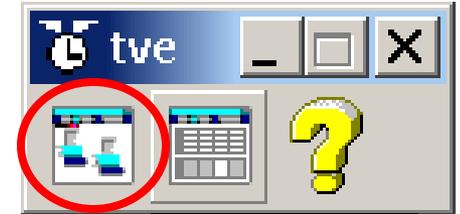
# Setting up the WaveTrain Model

# Model Setup Introduction

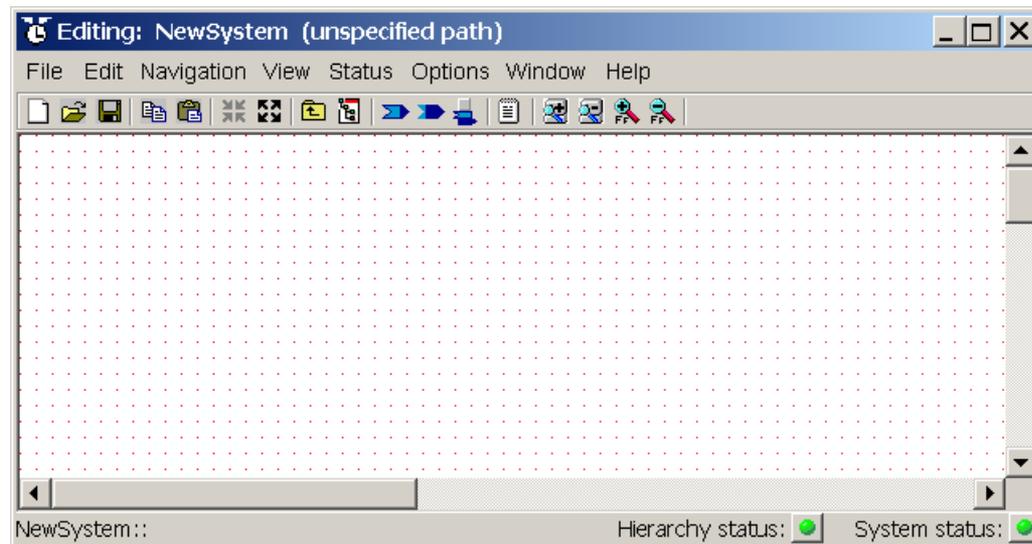
- We are presenting here one way of setting up this model.
- Other tutorials are available that describe other ways of doing this same type of model setup

# Open the System Editor

- Open the tve & click on the system editor icon.

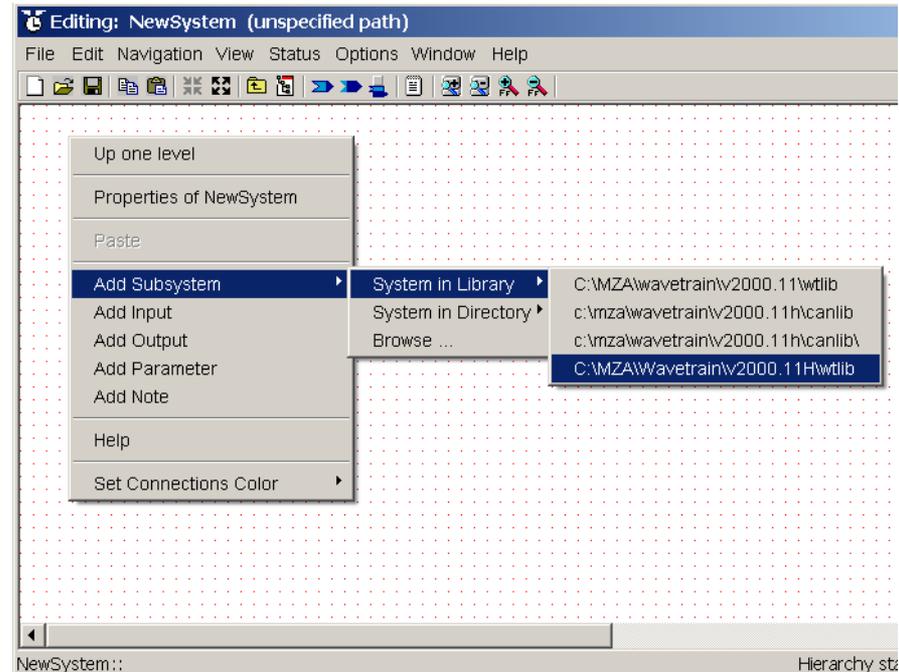


- This should open up the following window:



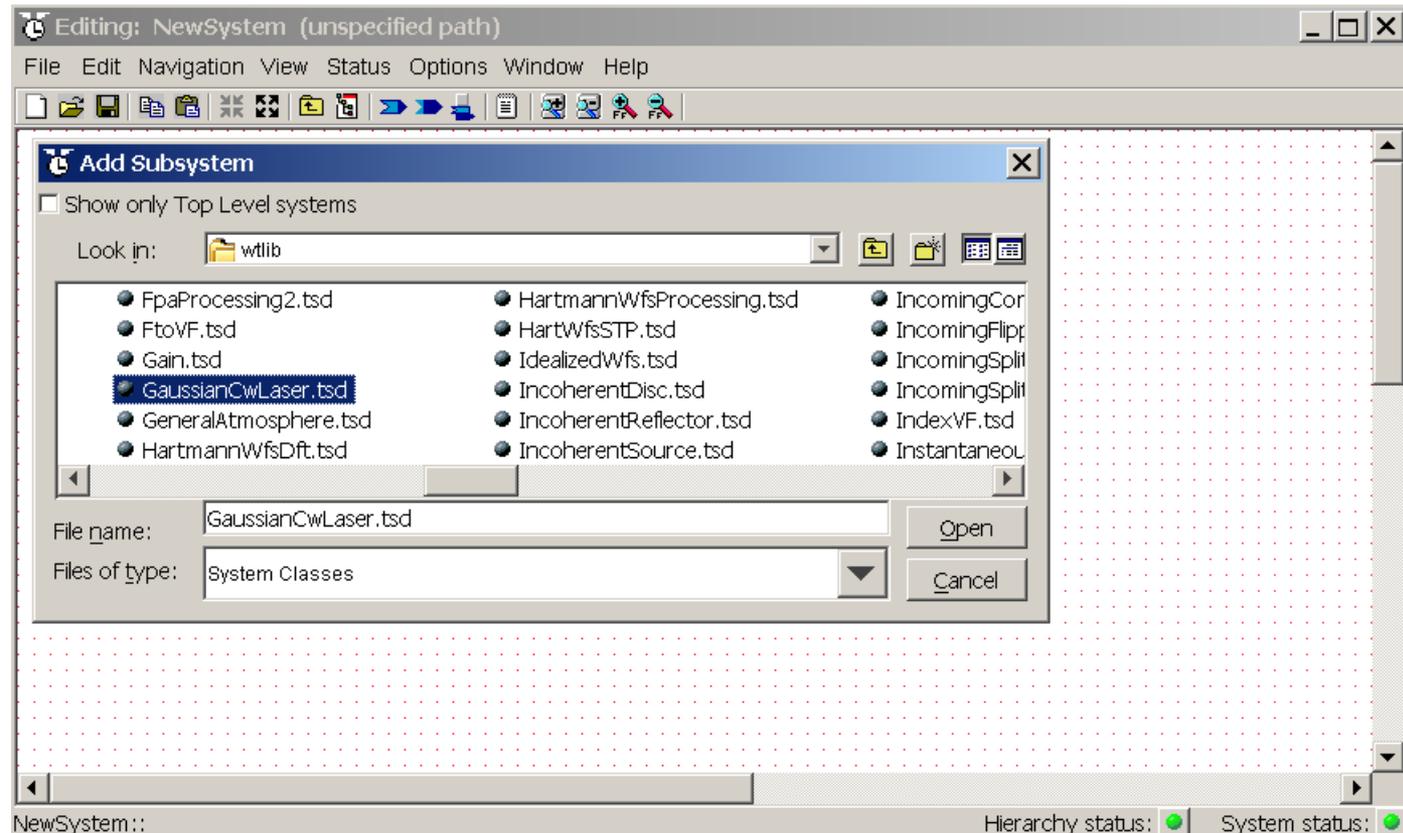
# Add a Source 1/3

- Right click in the window and choose “Add SubSystem”, “System in Library”, and then choose your wtlb directory.



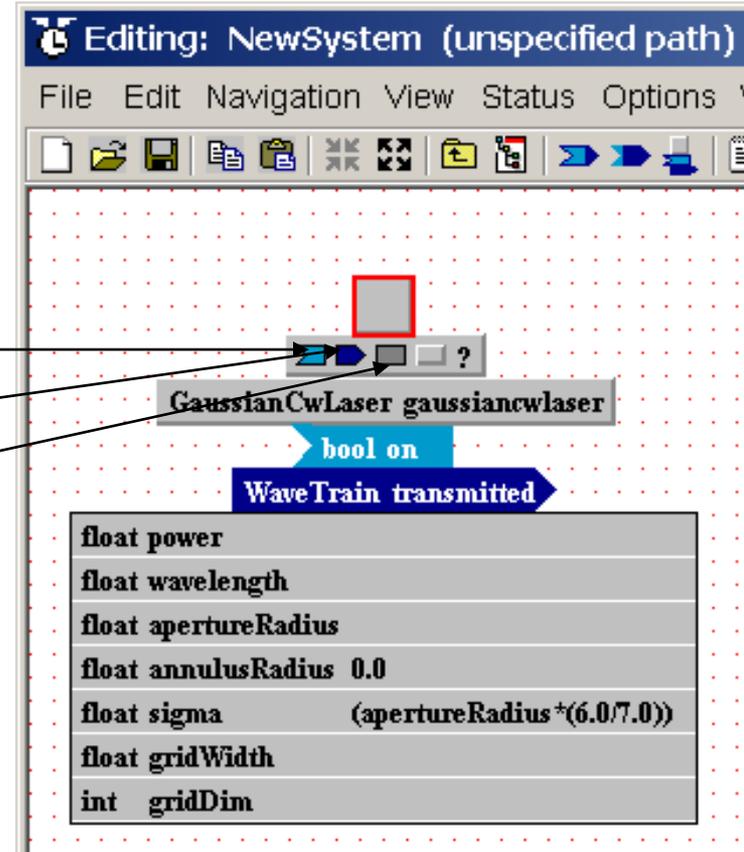
# Add a Source 2/3

- Choose “GaussianCwLaser” from the list of files.



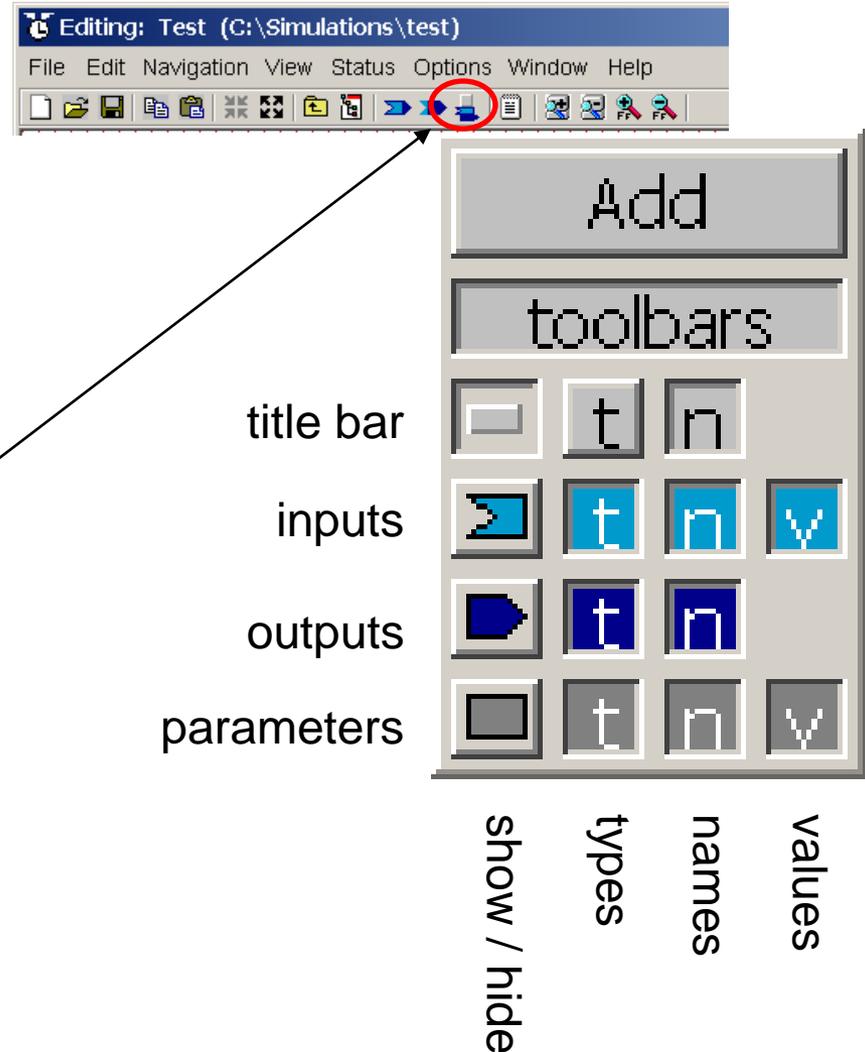
# Add Source 3/3

- Place the source on the left part of the screen.
- Show its
  - inputs,
  - outputs, and
  - parametersby clicking on the images below the grey box.



# Exposing types, names, and values

- Sometimes the software is left in a state where the data type, name, and/or value has been hidden from view.
- To expose these for inputs, outputs or parameters, click on the IO icons on the toolbar.
- A menu of icons will drop down. The rows correspond to the title bar, the inputs, the outputs, and the parameters respectively. The columns correspond to hiding and showing the list, types (t), the names (n), and values (v).



# Add the Propagation Controller

- Use the same technique to add a PropagationController.

The screenshot shows a software interface for editing a system. The main workspace contains two components: 'GaussianCwLaser gaussiancwlaser' and 'PropagationController propagationcontroller'. The PropagationController component is highlighted with a red box. Below each component is a table of parameters.

Parameter	Value
float power	
float wavelength	
float apertureRadius	
float annulusRadius	0.0
float sigma	(apertureRadius*(6.0/7.0))
float gridWidth	
int gridDim	

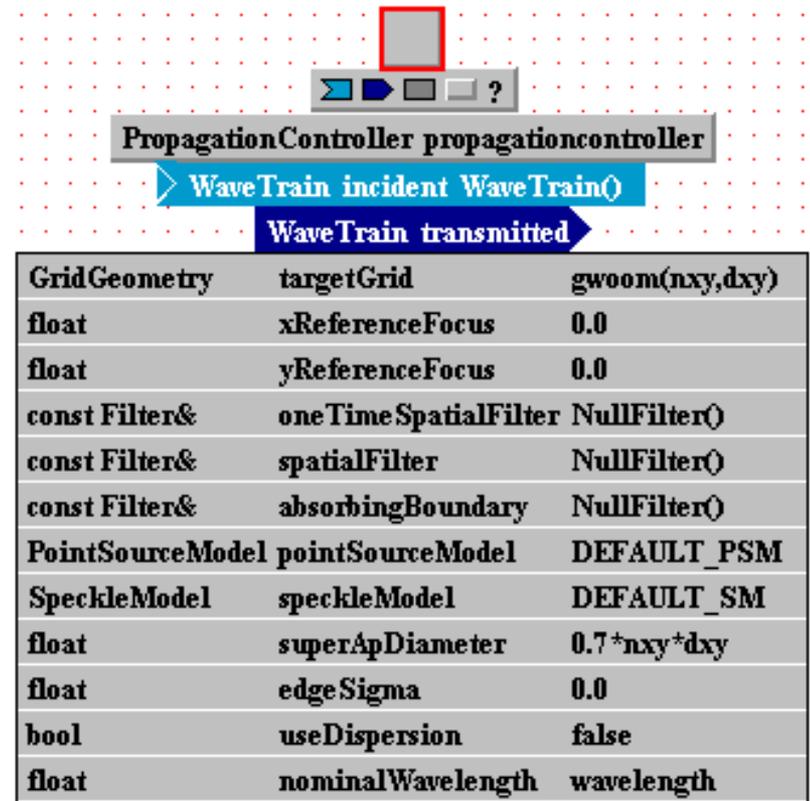
Parameter	Value
GridGeometry targetGrid	grid_with_origin_on_mesh(propnxy,propdy)
float xReferenceFocus	0.0
float yReferenceFocus	0.0
const Filter& oneTimeSpatialFilter	NullFilter()
const Filter& spatialFilter	NullFilter()
const Filter& absorbingBoundary	NullFilter()
PointSourceModel pointSourceModel	DEFAULT_PSM
SpeckleModel speckleModel	DEFAULT_SM
float superApDiameter	
float edgeSigma	0.0
bool useDispersion	false
float nominalWavelength	0.0

# Commentary on PropagationController

- The PropagationController allows you to specify various properties of the light model including mesh parameters.
- **Every source needs a PropagationController between it and a sensor.**
  - It is best to put the PropagationController just after the source.

# PropagationController Parameters

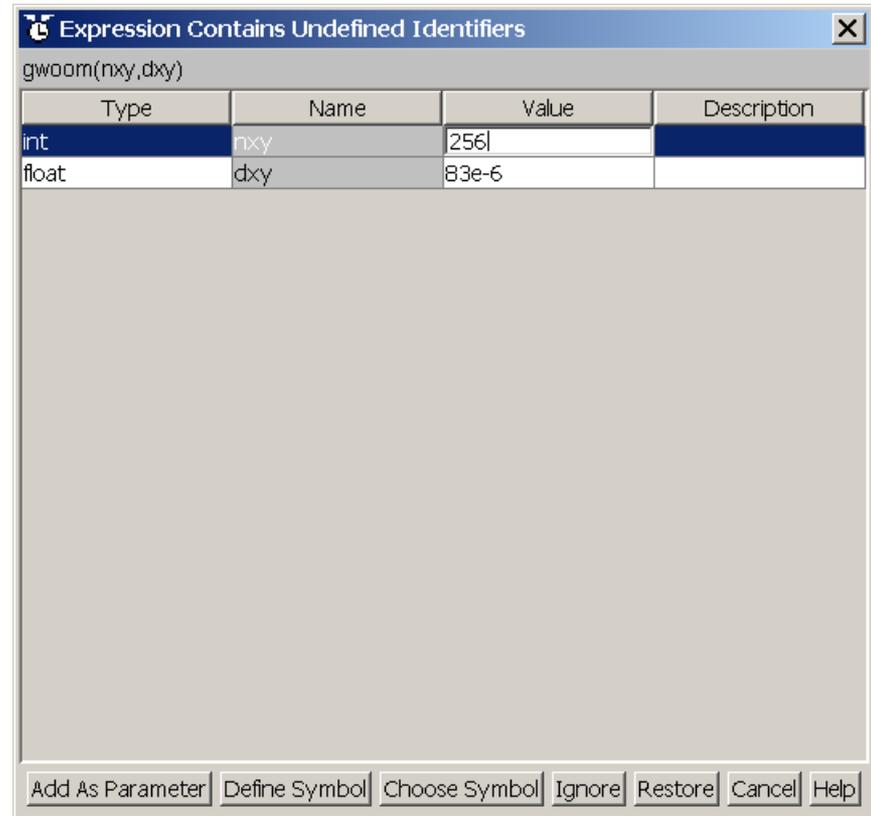
- The parameters list for each component specifies the type of parameter, the parameter name, and the parameter value.
- Click on the area to the right of the parameter name (the parameter value) to get a dialog box for editing it.
- Start by changing the targetGrid parameter value to “gwoom(nxy,dxy)”.
- Upon pushing enter you will see the screen on the next slide pop-up...



GridGeometry	targetGrid	gwoom(nxy,dxy)
float	xReferenceFocus	0.0
float	yReferenceFocus	0.0
const Filter&	oneTimeSpatialFilter	NullFilter()
const Filter&	spatialFilter	NullFilter()
const Filter&	absorbingBoundary	NullFilter()
PointSourceModel	pointSourceModel	DEFAULT_PSM
SpeckleModel	speckleModel	DEFAULT_SM
float	superApDiameter	0.7*nxy*dxy
float	edgeSigma	0.0
bool	useDispersion	false
float	nominalWavelength	wavelength

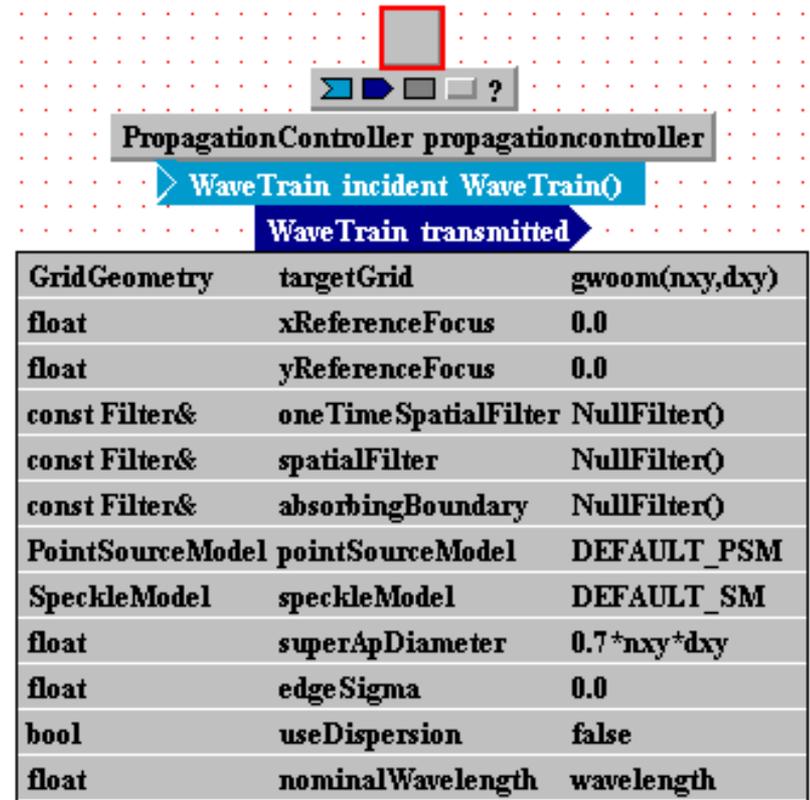
# Parameter Addition

- Change the nxy type to int from float.
- Enter a value for nxy of 256.
- Enter a value for dxy of 83e-6.
- Push enter after each enter or the values will not be registered.
- Click on the “Add add Parameter” button twice (once for each of them) to add them as system parameters.



# PropagationController Parameters

- Using the same procedure, change the superApDiameter to  $0.7 * nxy * dxy$ .
- Change the nominalWavelength to a new parameter called wavelength, and set its initial value to  $1.0e-6$ .



PropagationController propagationcontroller		
WaveTrain incident WaveTrain()		
WaveTrain transmitted		
GridGeometry	targetGrid	gwoom(nxy,dxy)
float	xReferenceFocus	0.0
float	yReferenceFocus	0.0
const Filter&	oneTimeSpatialFilter	NullFilter()
const Filter&	spatialFilter	NullFilter()
const Filter&	absorbingBoundary	NullFilter()
PointSourceModel	pointSourceModel	DEFAULT_PSM
SpeckleModel	speckleModel	DEFAULT_SM
float	superApDiameter	$0.7 * nxy * dxy$
float	edgeSigma	0.0
bool	useDispersion	false
float	nominalWavelength	wavelength

# GaussianCWLaser Parameters

- Set the power to 1.0.
- Right click on the wavelength parameter and choose “Elevate”.
  - This will link the wavelength of this component to the same as the value specified for the PropagationController.
- Set the remaining values to those specified on the right.

The screenshot shows a software interface for configuring a `GaussianCWLaser` component. A red box highlights the `wavelength` parameter in the parameter list. A context menu is open over this parameter, with the `Elevate` option selected. Below the main interface, two tables show the parameter values before and after the `Elevate` action.

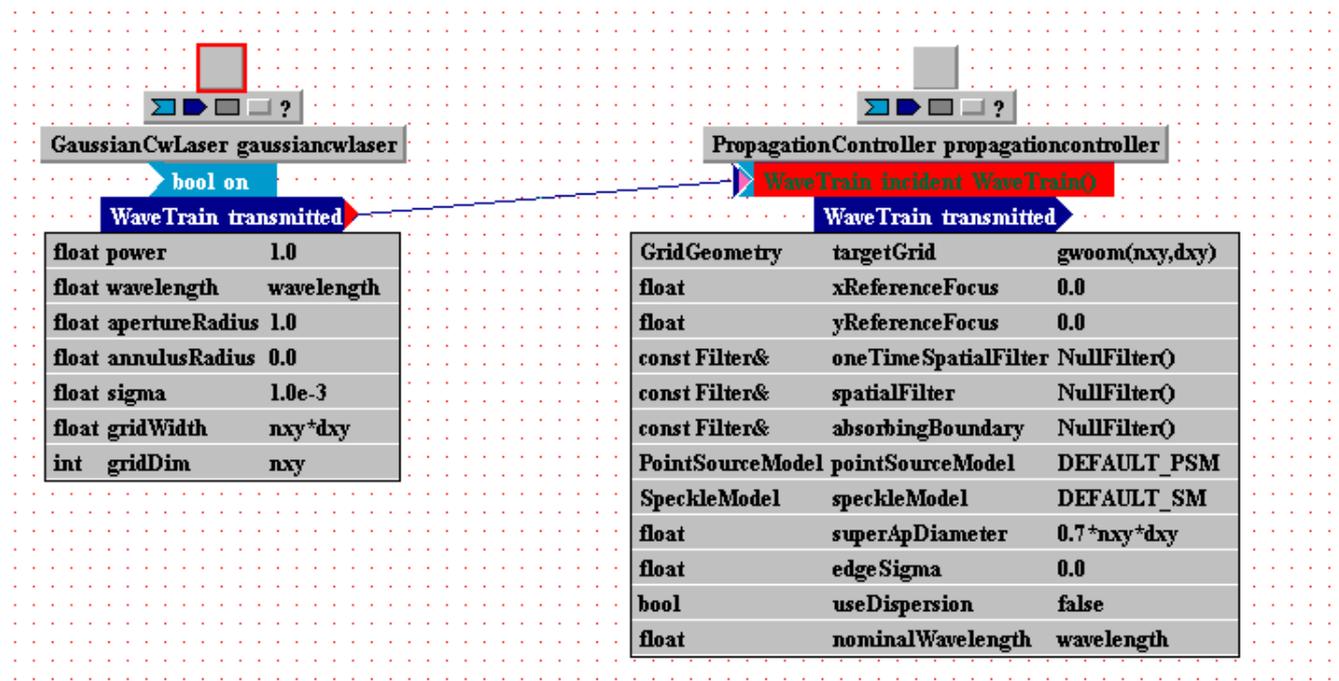
Parameter	Value
<code>float power</code>	1.0
<code>float wavelength</code>	
<code>float aperture</code>	
<code>float annulus</code>	
<code>float sigma</code>	$(\text{apertureRadius} * (6.0/7.0))$
<code>float gridWidth</code>	
<code>int gridDim</code>	

Parameter	Value
<code>float power</code>	1.0
<code>float wavelength</code>	wavelength
<code>float apertureRadius</code>	1.0
<code>float annulusRadius</code>	0.0
<code>float sigma</code>	1.0e-3
<code>float gridWidth</code>	$nxy * dxy$
<code>int gridDim</code>	nxy

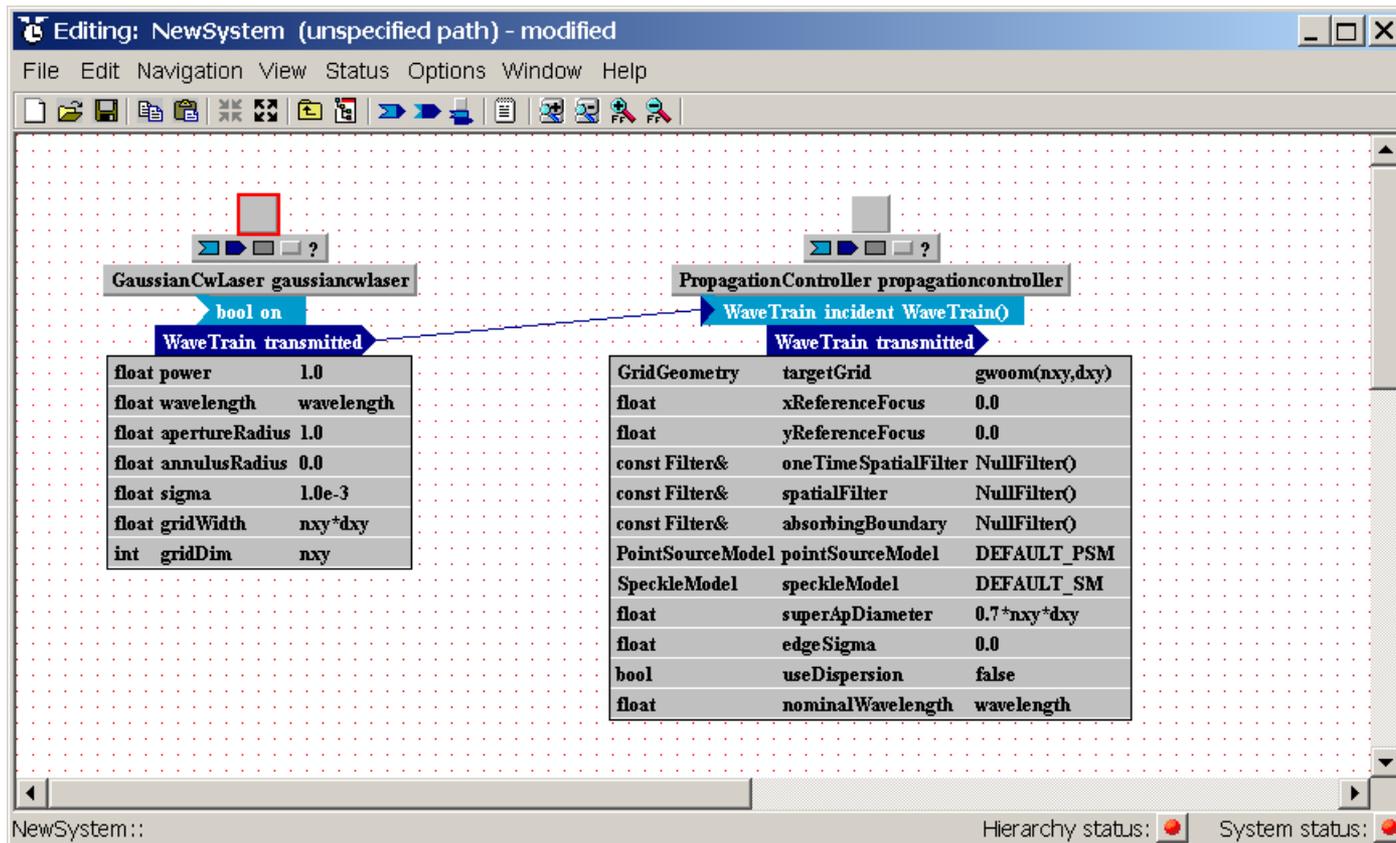
# Connecting the Source and the PC

- Drag the tip of the “transmitted” output from the GaussianCW Laser to the “incident” input on the PropagationController.



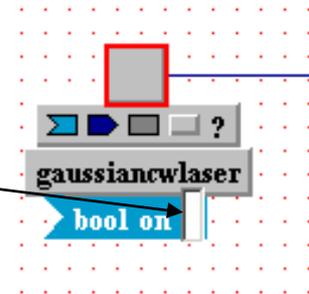
# Status Check

Your system should now look like this:



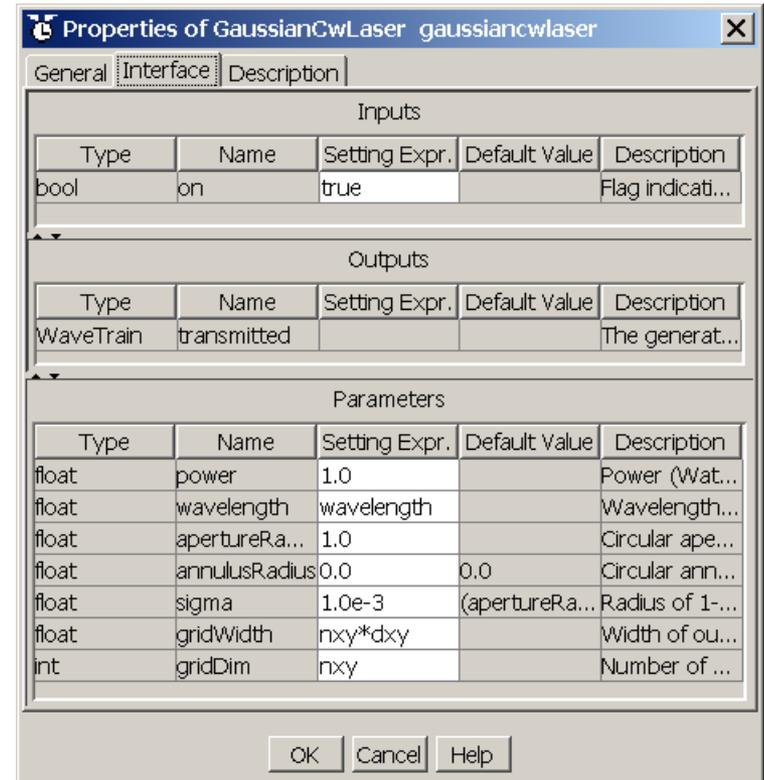
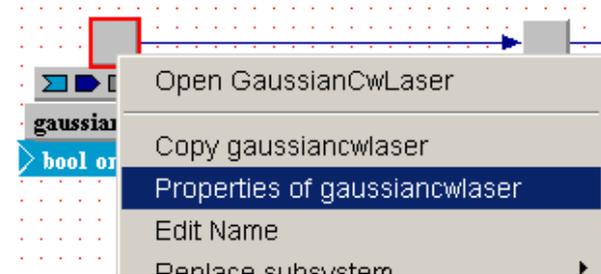
# Setting an Input to a System Parameter

- Inputs can be set to constant values or as system parameters.
- If an input has no default value, it will appear like this.
- To add a value to this input, click on the area to the right of the name and an edit box will appear.
- Begin typing in this box even though the text will not appear until you press enter.
  - This is a bug in the JAVA interface code.



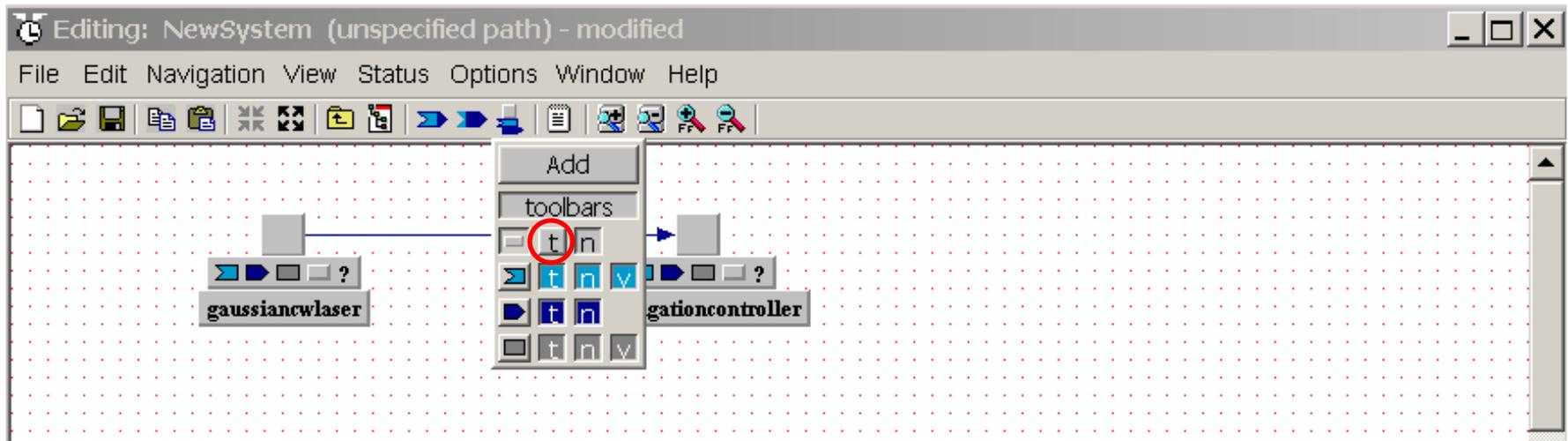
# Alternative Method of Setting Inputs, Outputs, and Parameters

- Right click on the system and choose “Properties of gaussiancwlaser”.
- A dialog box will then appear that provides a tabular input on the “Interface” tab.



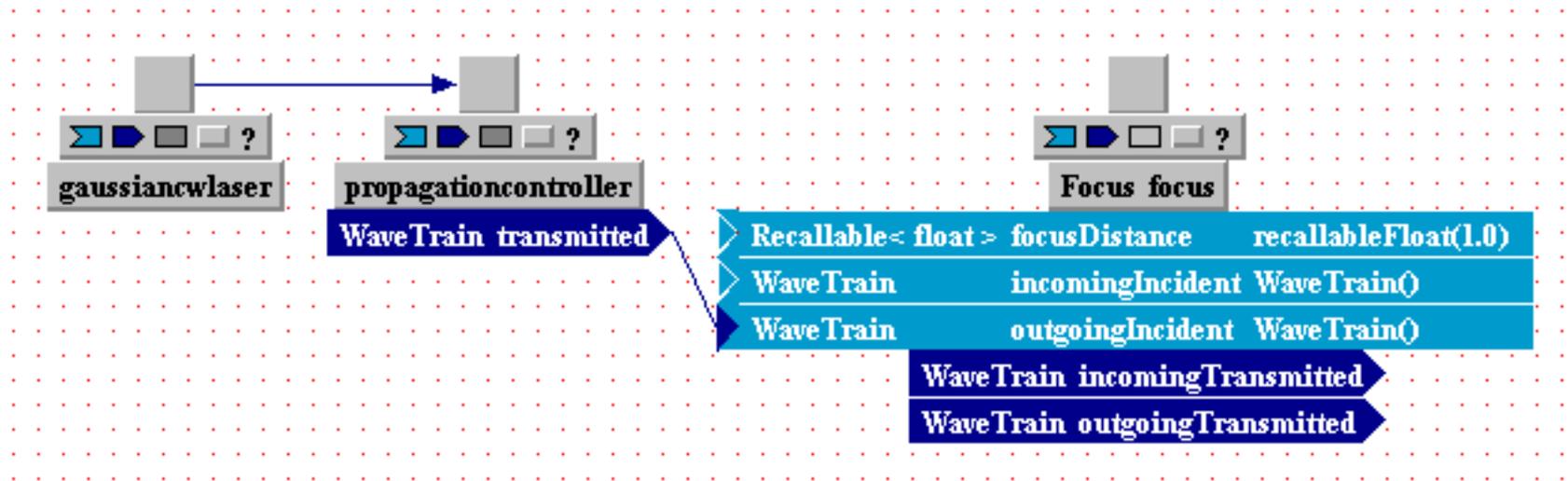
# Give yourself more room

- Give yourself more working room by turning off the component types using the icon menu icon shown below and choosing the grey “t”.
- Hide the inputs, outputs, and parameters of the components.



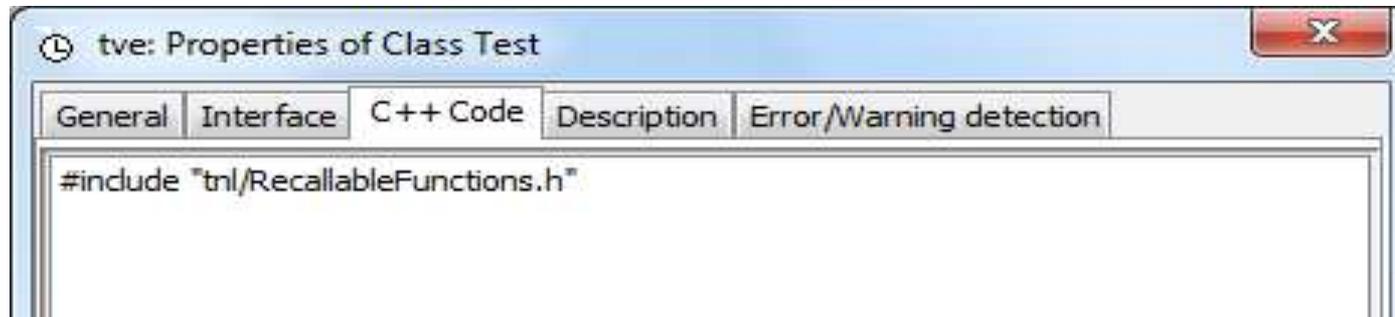
# Adding a Focus

- Add a “Focus” from the wtlib library.
- Connect the PropagationController output to the focus “outgoingIncident” input.
- Change the focus distance input value to read “recallableFloat(1.0)”.



# Commentary on recallableFloat()

- tempus (the backbone modeling package for WaveTrain) sometimes needs to look back in time to determine a value of a variable or input, so a type of inputs was created called “recallables”.
- Most systems do not need this functionality, so a conversion routine was developed called “recallableFloat()” that converts a float to a recallable<float>
- To use this routine, you need to include the file “RecallableFunctions.h” in the “C++ Code” tab of the system properties.
  - From the main menu, select File-Properties
  - Click on the C++ Code tab
  - Enter: #include “tnl/RecallableFunctions.h”



# Commentary on outgoing Incident

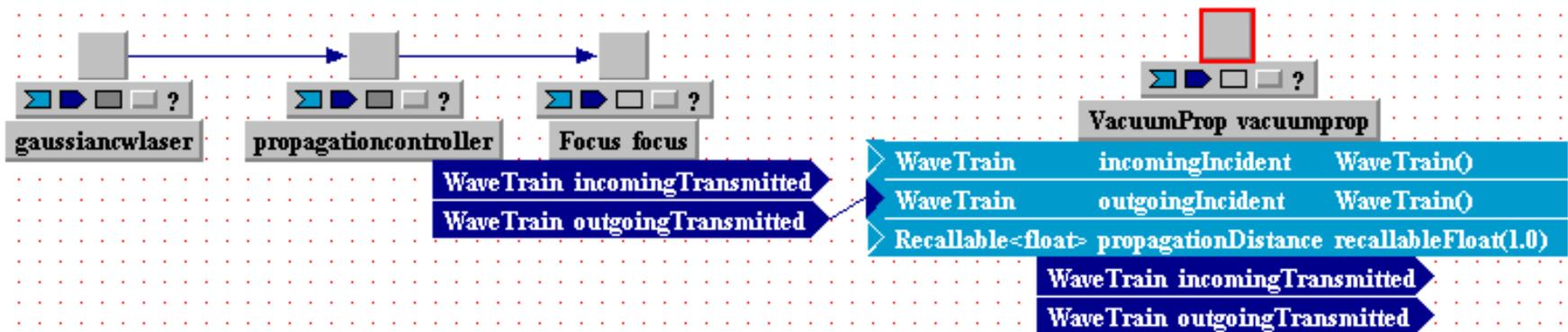
- “Outgoing” and “incoming” inputs and outputs were designed to make optics bidirectional.
- The present system uses only “outgoing” waves. (Internally, “outgoing” is associated with the Cartesian +z direction).
- For more complex WaveTrain systems that involve light propagating in both directions, it is typical to use both “outgoing” and “incoming” waves processed by the same optical and propagation components.

# Commentary on Sources and Sensors

- Light in WaveTrain always travels from a source to a sensor.
  - Every model needs at least one source and one sensor.
- Sensors initiate the modeling by requesting light from all sources attached to a given sensor.

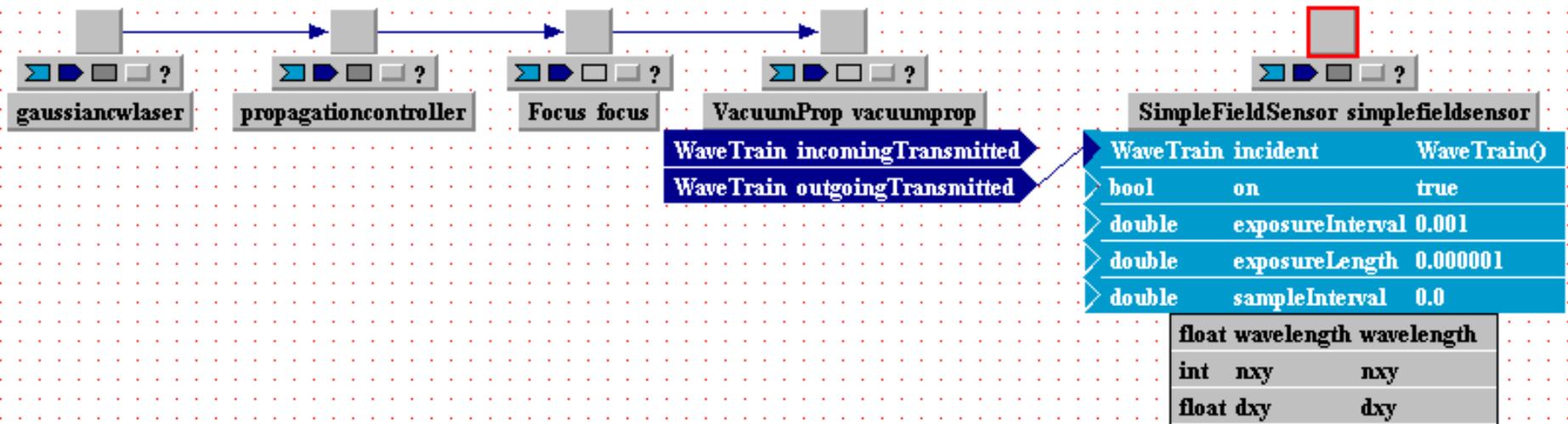
# Add a VacuumProp

- Add a vacuumProp and connect it to the outgoingTransmitted output of the focus.
- Change the propagationDistance input to “recallableFloat(1.0)”.



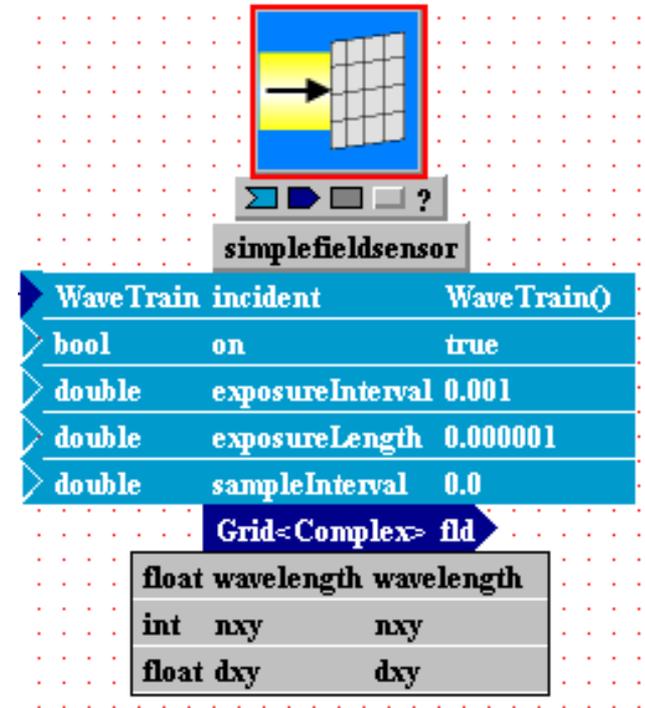
# Add a SimpleFieldSensor

- Add a simpleFieldSensor to the system and connect it to the outgoingTransmitted output of the vacuumProp component.
- Set the parameters as shown below.



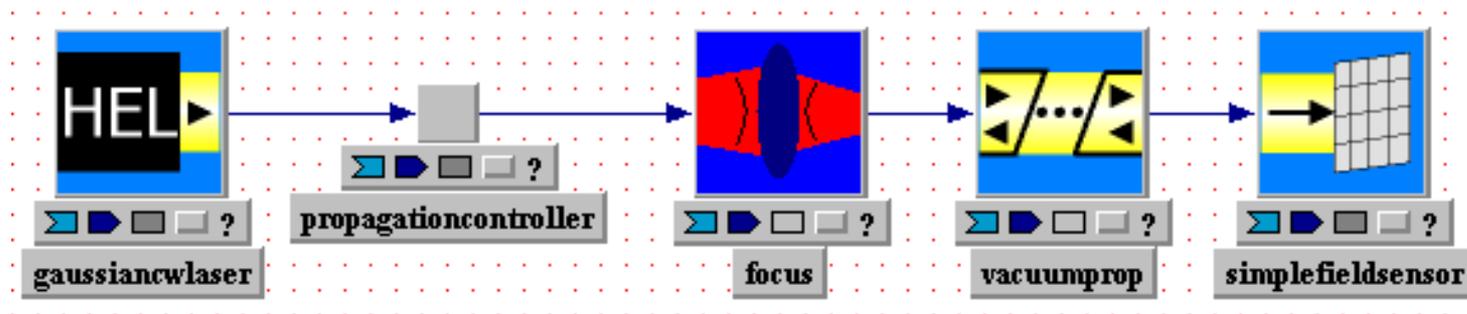
# Commentary on the SimpleFieldSensor

- The simple field sensor has a variety of inputs corresponding to the timing of the sensor.
- The `exposureInterval` is the time between exposures.
- The `exposureLength` is the amount of time that the sensor is exposed to light (shutter open time).
- The sample interval is the time between samples that are averaged on the sensor
  - 0.0 means that one sample is taken at the center of the `exposureLength`.
- The boolean `on` input can be driven externally in some situations to delay the on-time in order to stage the exposure of various sensors.



# Inserting Icons

- Icons can make the system more easy to understand at a glance.
- To add an icon to a component, right click on that component and choose “Edit Icon”.
- WaveTrain comes with a variety of icons for making a system more easily readable.



# Parameter Viewing

- To view the system parameters at a glance at the bottom of the screen, go to View-Parameters.

The screenshot shows a simulation software window titled "Editing: Test (C:\Simulations\test)". The main workspace contains a block diagram with the following components: gaussianw laser, propagationcontroller, focus, vacuumprop, and simplefieldsensor. Below the diagram is a "Parameters" table.

Type	Name	Default Value	Description
int	nxy	256	
float	dxy	83e-6	
float	wavelength	1.0e-6	

Test:: Hierarchy status:  System status:

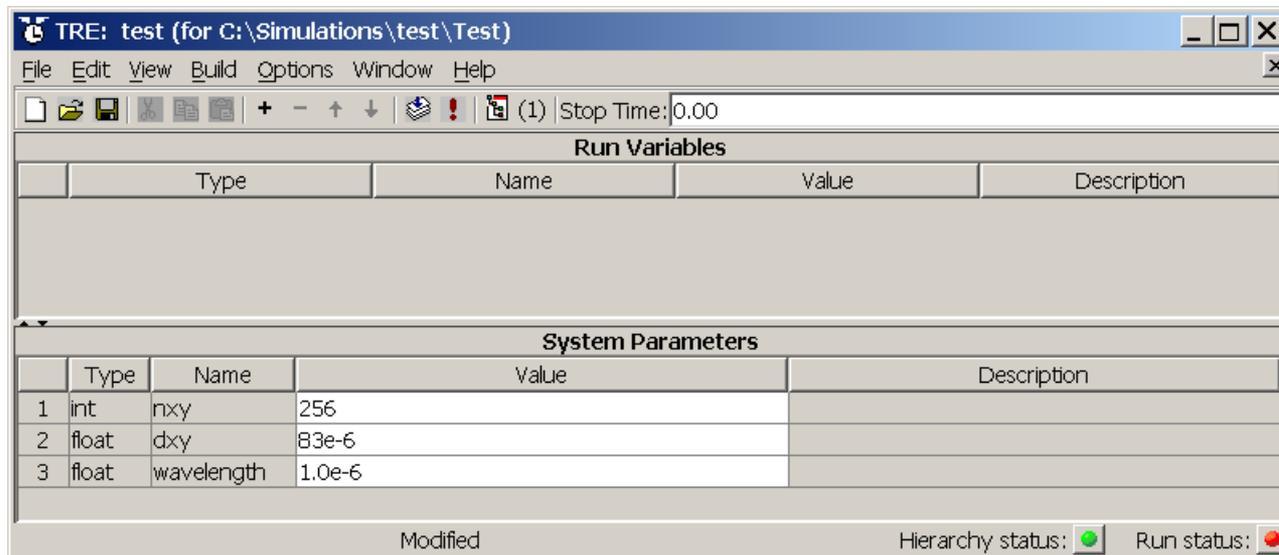
# Saving the System

- Once the system has been created, select File-Save or Save As to save the system.
- Save the system in a path without spaces in the directory structure or the file name.
- Do not put spaces in the file name!
- It is recommended to save the systems in a “c:\Simulations” directory.
  - For reference, we saved this as “c:\simulations\test\test.tsd”
  - \*.tsd are tempus system description files

# Creating a Runset

# Runset Creation

- Now you are ready to create a runset.
- From the system editor, go to File-New-Runset for test.
- Enter a name for the runset.
  - we used “test” as the name
- The tempus runset editor (TRE) will pop-up.

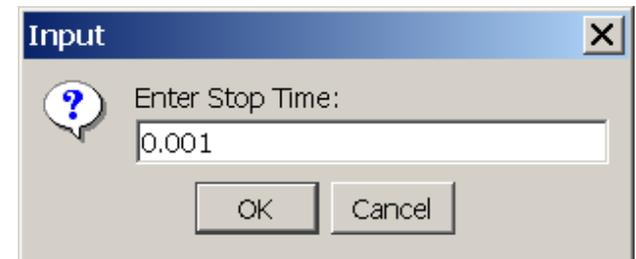
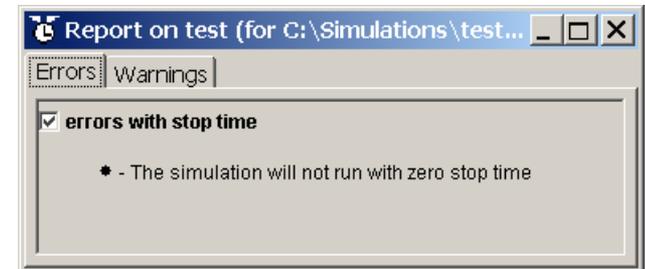
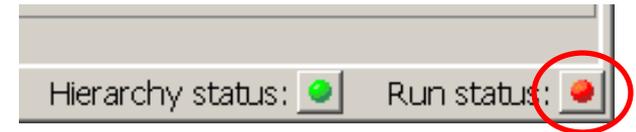


# Commentary on Runset Names

- A tempus results file (trf) is created each time a tempus run is made.
- The code generator will automatically terminate the trf file name with an incrementing number so that the old data is not overwritten.
  - Example: TestRunTest1.trf
- Therefore, it is highly recommended that the runset name not end in a number because it can make interpreting the trf file names difficult.
  - Example: “TestRunTest11.trf” could be the first run for a runset named Test1 or the 11<sup>th</sup> run of a runset named Test.

# Stop Time

- Upon entering the runset editor, the run status indicator will be red
  - see the bottom right part of the screen
- Click on the indicator to see the list of errors.
- The first error is usually that the stop time is zero.
- Close the error list and select Edit-Edit Stop Time from the main menu of the TRE.
- Enter 0.001 in order to get one field from the simpleFieldSensor (SFS)
  - the SFS's exposureInterval is 1 ms, so it will record a frame every 1ms.



# Selection Outputs to Record

- The runset has no outputs selected for recording upon creation.
- To select the outputs for recording, click on the output recording button or go to Edit-Edit Output Recording from the main menu.
- To select an output for recording, check the box to the left of the output.
- Click OK when all the desired outputs are selected.



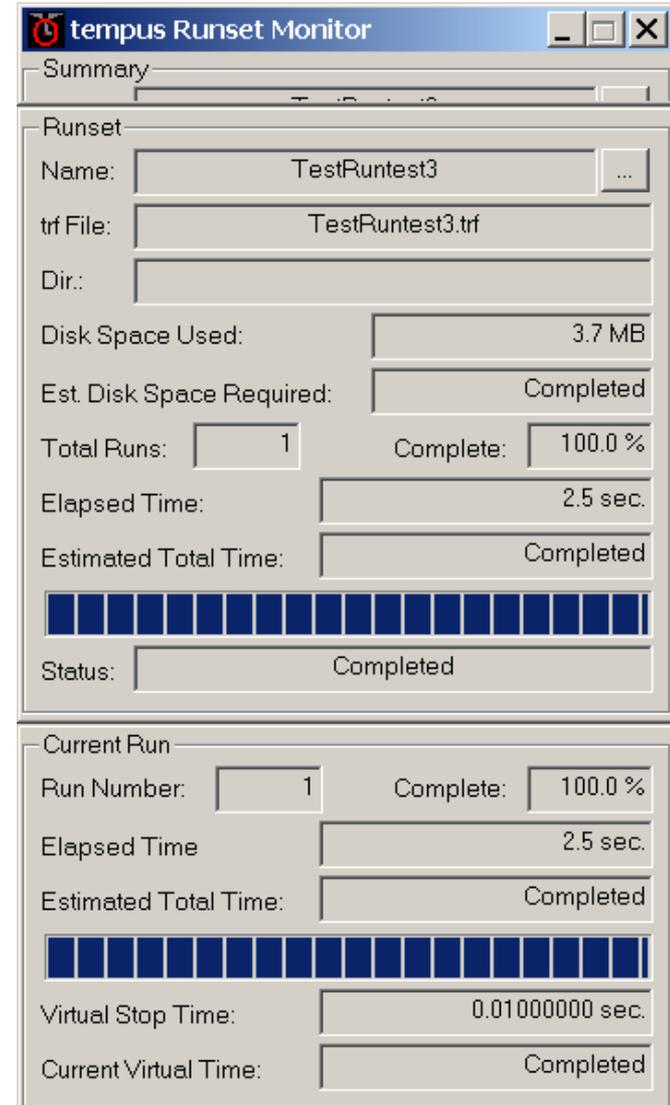


# Linking / Compiling Problems

- It is not uncommon that the runsets do not properly compile and link.
- The console window will list the problems with the compilation or linking and give you some indication as to where the problem is and sometimes how to fix it.
- With persistent issues in this regard, call MZA's technical support.

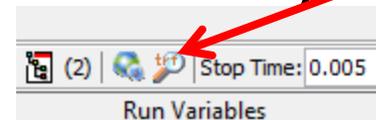
# Executing the Simulation

- To execute the simulation press the execute button (red exclamation point) or go to Build-Execute on the main menu.
- The model will compile, link, then execute.
- When execution begins, the tempus runset monitor will pop-up to show the progress.
  - For simple systems the runset monitor may not have time to execute before the system completes, so the monitor will report an error.



# Results - .trf files

- As soon as the model execution starts, a .trf file is created.
- As the model runs this file is filled up with model data.
- You can click on the TrfView button in the runset editor to view the most recently created .trf file for the runset.
- You can also visualize the .trf file results with Matlab.



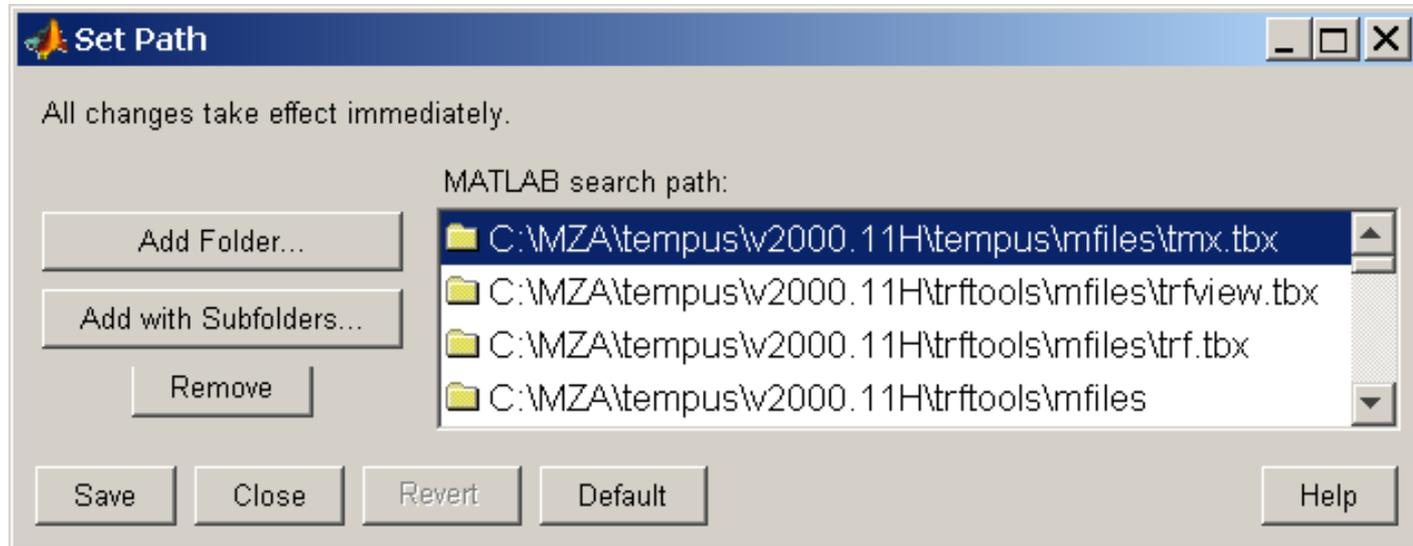
# Visualizing the Results

# trf Routines in Matlab

- MZA provides a whole suite of routines in Matlab for reading in data from .trf files, including:
  - trfopen
  - trfload
  - tmxparams
- The most commonly used tool is a GUI package called “trfview”.
  - Run it by typing “trfview” at the command line and pressing enter.

# Matlab Path

- If none of the trf tools work in Matlab, make sure your Matlab path has the directories listed below in it.
  - Go to File-Set Path for the following dialog box:



# trfview

- Open your trf file using File-Open from the main menu.
- A list of variables and parameters should appear.

The screenshot shows the TRF Viewer application window. The title bar reads "TRF Viewer". The menu bar includes "File", "Variable", and "Parameter". The "Filename" field contains "C:\Simulations\test\TestRuntest4.trf". Below this, there is a "Runs:" field with the value "1", a "Select" dropdown menu, and a checked "View All Runs" checkbox.

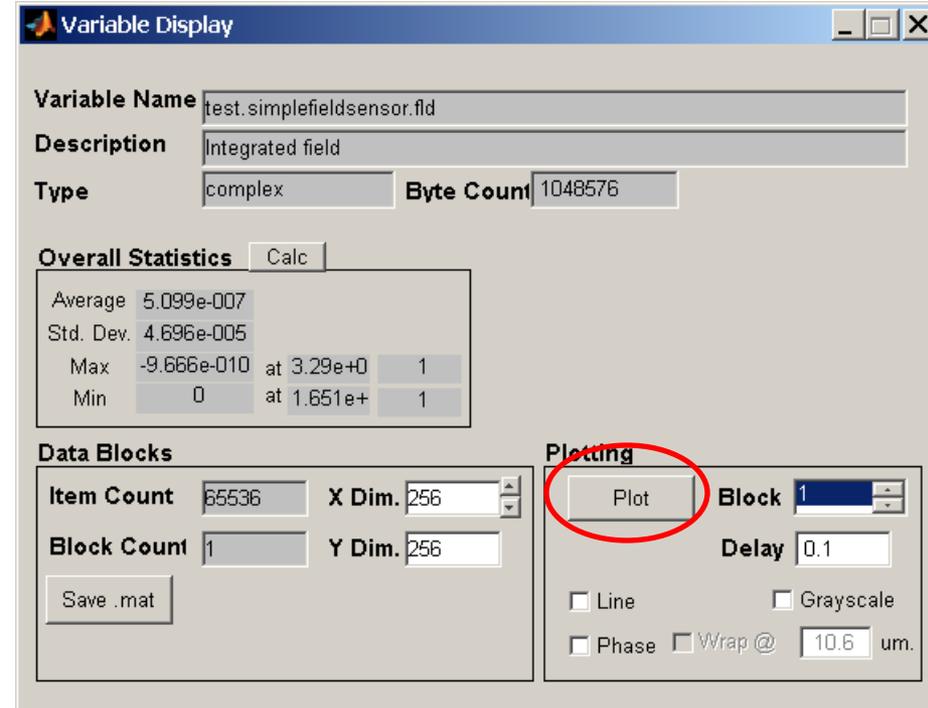
Variable Name	Data Type	Count
test.simplefieldsensor.fld	complex	65536

Parameter Name	Data Type	Value
dxy	float	8.3e-005
nxy	int	256
stopTime	double	0.001
wavelength	float	1e-006

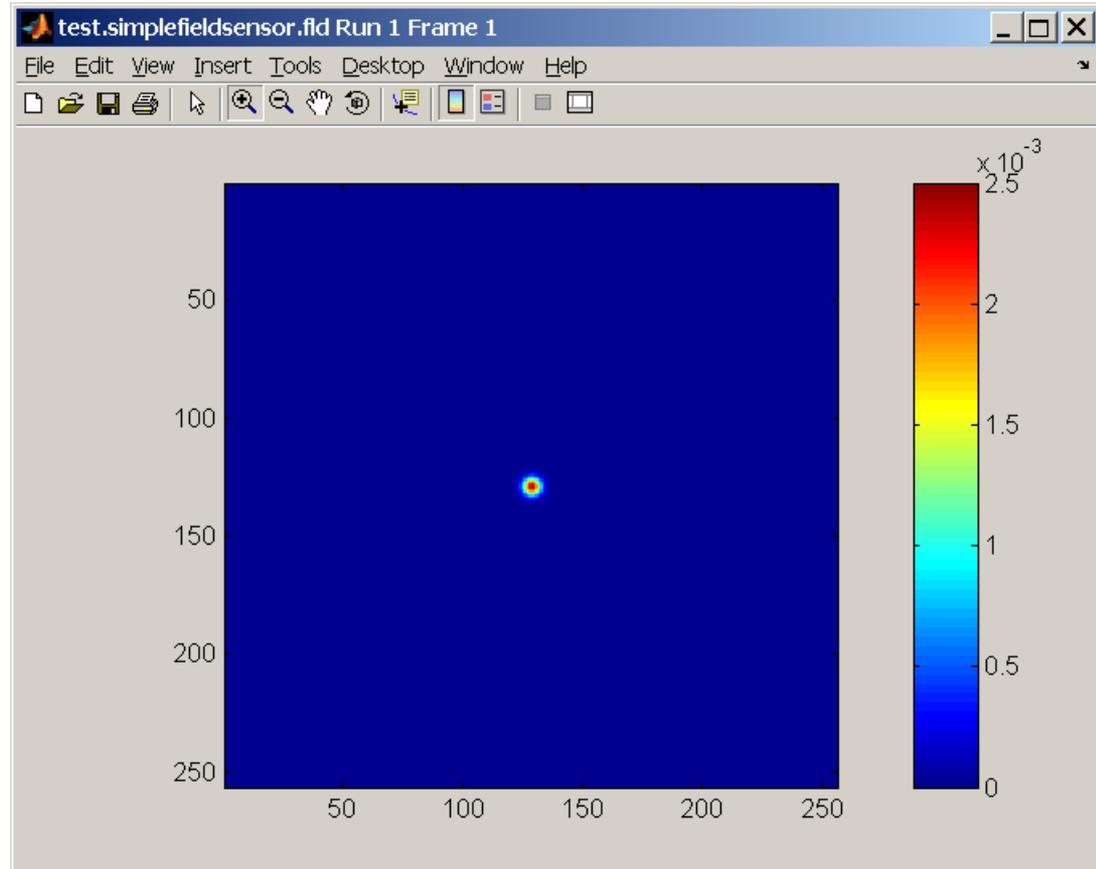
# Viewing Output Data

- To view output data, double click on the named variable and a variable display box should appear.
- To plot that data, click on the plot button in the bottom right of the screen.



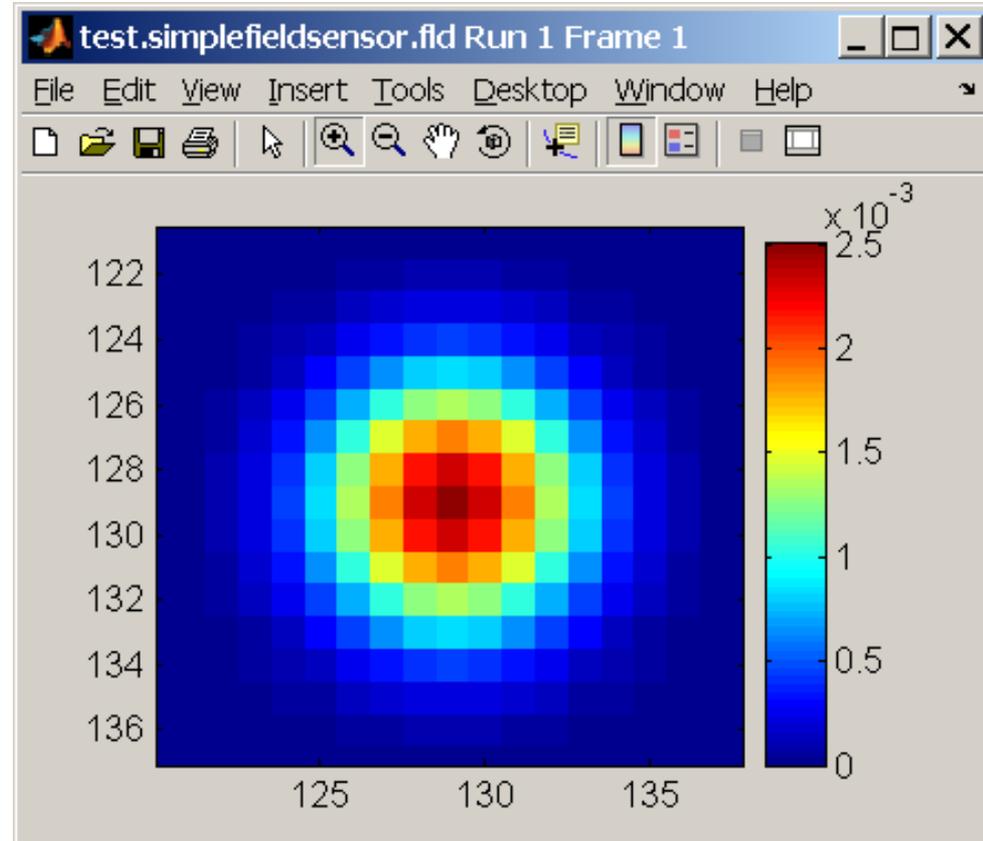
# False-Color Plot

- This window should appear showing the field magnitude.



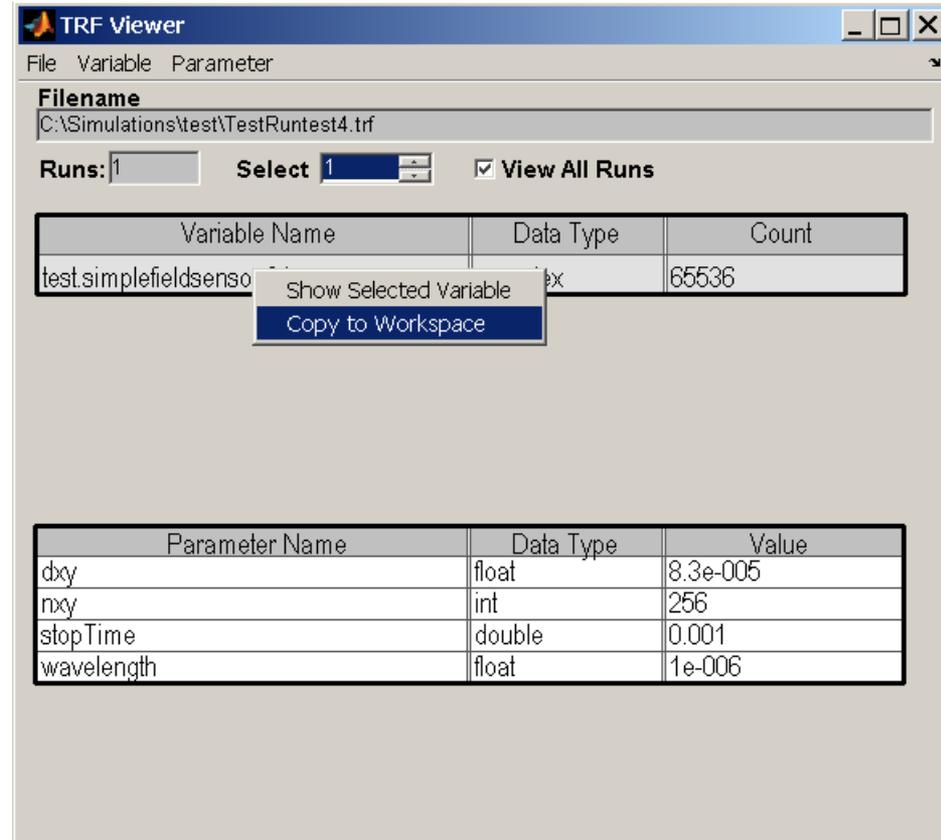
# Estimation of Data Correctness

- If we zoom in on this plot, we can approximately count the number of pixels across the Gaussian beam is at the focus.
- In this example, there are about 10 pixels and the mesh spacing is  $83\ \mu\text{m}$ , so the beam diameter is about  $830\ \mu\text{m}$ .
- This corresponds reasonably well to the expected  $640\text{-}\mu\text{m}$  diameter calculated from theory, but it is not very scientific.



# Extracting the Data to Matlab

- The data can be extracted to Matlab by right clicking on the variable name in trfview and then selecting “Copy to Workspace”.
- The data is then available as the variable name (with the periods).



# Data Interpretation

- The data in the workspace is in vector form with each time sample as a new column.
- A single time-step can be converted into a 2D field with the following command:
  - `E=reshape(test.simplefieldsensor.fld(:,1),256,256);`
- The field can be converted to an intensity profile with:
  - `I = E .* conj(E);`

# Data Analysis

- Matlab can be used to calculate the second moments of the intensity profile as follows:
  - $n=256$ ;  $dx=83e-6$ ;  $x = (-n/2:1:n/2-1)*dx$ ;
  - $[xx,yy]=meshgrid(x,x)$ ;
  - $lx2 = \text{sum}(\text{sum}(xx.^2 .* I))$ ;
  - $ly2 = \text{sum}(\text{sum}(yy.^2 .* I))$ ;
  - $lsum = \text{sum}(\text{sum}(I))$ ;
  - $x2 = \text{sqrt}(lx2/lsum)$ ;  $y2 = \text{sqrt}(ly2/lsum)$ ;
- 4 times the Intensity second moment is the beam diameter.
  - For our data the beam diameter is 637  $\mu\text{m}$ , which corresponds exactly with our initial theoretical prediction.

# Conclusions

- In this example, we showed how to start modeling with WaveTrain.
- We used the example of propagating a Gaussian beam to a focus.
- Our WaveTrain results corresponded exactly with our theoretical prediction.