WaveTrain™
wave optics made easyier

User's Guide

Version 2025A

Jason A. Tellez, Ph.D

MZA Associates Corporation
1360 Technology Ct, Ste 200, Dayton, OH 45430-2211
http://www.mza.com
(937) 684-4100, WT-Support@mza.com

The revision date is August 26, 2026

Contents

List of Figures

List of Tables

What's New

Compatibility Considerations

This release of WaveTrain™ has been tested on Windows with Visual Studio 2019 & 2022 and Matlab R2023b and will be compatible with that version and newer. However, WaveTrain™ 2025A may still work with older versions of Visual Studio and Matlab. On Linux, WaveTrain™ has been tested on Red Hat Enterprise Linux (RHEL) 9.0 and Ubuntu 22.04LTS and Matlab R2022b.

New Features

MZA is always looking for better and more useful ways to take advantage of the capabilities that WaveTrain™ offers. In that spirit, some new capabilities added to WaveTrain™ includes

- A new set of components for incoherent imaging based on the previous Light Tunneling techniques
- Improved third party library integration to include specifying locations using environment variables, simplifying the build process and improving portability
- Added an option to build without Matlab integration
- Updated drop down menu feature for selecting the build type in the Tempus Runset Editor (TRE)
- New build selection for creating a dynamic library version of the runset for interaction with the RunsetAPI set of functions
- Conductor application for interacting with RunsetAPI builds (BETA)
- WaveTrain™ Concourse, a new way of building systems and running them without the need for a C++ compiler or Matlab

Bug Fixes

WaveTrain™ is continuously improving based on in-house use and customer feedback. As a result the following bug fixes have been address since the last release

- Fixed an issue where the gaussian random number generator may fail to work correctly
- Fixed an issue where conflicting libraries may interfere with WaveTrain™ execution from the TRE
- Fixed an issue where aogeom was not saving the subaperture locations correcly when the annulus is zero
- Fixed an issue where an unconnected polarizer component generates a false output wave
- Updated the memory allocation for the TSE to help prevent crashes when the TSE is open for a long time
- Fixed a bug where Tilt commands with NANs would cause the simulation to fail
- Added an AcsAtmSpec constructor that can be used to remove the "no altitudes defined" warning the the platform and target altitudes are available
- Fixed an issue where building a Mex function from the TRE would fail
- Fix a bug where thermal blooming (when available) was not being applied to non heating waves
- Fixed Matlab function tmxmkdll.m to use the new build process for creating MEX functions
- Fixed an issue where certain libraries may be left off the build link list for some users

1. Introduction

WaveTrain™ is a code suite that provides computer modeling of optical propagation through the turbulent atmosphere, and the modeling of associated optical imaging, beam control, and adaptive optics (AO) systems. WaveTrain™ provides a connect-the-blocks visual programming environment in which the user can assemble beam lines, control loops, and complete system models, include closed-loop AO systems. WaveTrain™ also provides graphical interfaces for setting up adaptive optics geometries and turbulence profiles through a handful of MATLAB® based utilities along with a spreadsheet-style interface for setting up and executing parameter studies. The generated output, comprising imagery, complex-field quantities, control signals, etc. can be inspected in a WaveTrain™ viewer or can be loaded into MATLAB® for further analysis and post processing.

This User Guide is designed to serve as a tutorial introduction to WaveTrain™ for new users, and a reference for more experienced users. The User Guide attempts to be as comprehensive as possible in documenting the scope, usage rules, and assumptions of the WaveTrain™ code. In doing so, we may occasionally digress into short tutorials on topics in optical propagation theory and AO but for detailed theory background we refer the reader to the 3.7.1. WaveTrain™ is a complex code, and its pieces have be constructed and assembled by numerous contributors at MZA Associates. Generation of documentation is an ongoing effort, consequently not all elements of WaveTrain™ are documented to the same level of details. When users have questions regarding WaveTrain™ we ask them to proceed as follows

- Search the present User Guide
- Check the MZA Website for updates
- Check the individual module documentation in the WaveTrain™ Components and Effects Library
- Depending on level of support contact MZA by email or phone
- If a suspected bug is found, feel free to contact MZA regardless of level of support (or lack thereof)

WaveTrain™ is built on tempus, a general-purpose simulation tool also created by MZA Associates. tempus provides the visual programming environment, the software architecture, and all the basic mechanisms that support model-building and simulation in the time domain. WaveTrain™ then adds a component library specifically for modeling optical systems and effects along with a handful of application specific graphical user interface (GUI) components. The later includes a tool for setting up AO system components (deformable mirror (DM) and wavefront sensor (WFS) via aogeom) and a tools for specifying the turbulence distribution along the optical propagation path (via PropConfig from ATMTools). These two MATLAB® based tools include additional utilities useful (and in some cases necessary) for additional functionality. For instance, the AO utilities include routines for computing the reconstructor matrices for the AO systems while ATMTools contains a variety of function for computing turbulence statistics over the atmospheric path and the mesh parameters for the numerical propagation.

1.1 Guide to the User Guide

To get a quick idea of what WaveTrain™ can do and how one can use it, start with our Quick Tour of WaveTrain™ (this should take less than 10 minutes). For a more detailed introduction (a few hours), work through the step-by-step tutorial in 3. After these introductory exercises, there will undoubtedly still be many details that are fuzzy to the new user. We suggest that the best procedure at that point is:

- Skim the rest of this User Guide to get a sense of the organization and contents
- Attempt construction of original simple systems, based on what you have learned from the "Assembling" section and auxiliary tutorials
- As conceptual and procedural questions then arise, dive into the remaining detailed documentation chapters of this User Guide as needed

By way of preview, we mention the following details chapters:

- For physics-oriented issues and usage details regarding important specific WaveTrain™ library systems, users should consult the chapter **??**
- For details regarding data entry in the two editor windows, users should consult the chapter **??**.
- For details regarding TrfView, trf files (recorded outputs) and the extraction of trf data, users should consult the chapter **??**
- For details regarding the construction of user-defined WaveTrain™ subsystems, users should consult the chapter **??**

2. A Quick Tour of WaveTrain™

Modeling an optical propagation system in WaveTrain™ is essentially a four-step process. The steps are:

1. Assemble the WaveTrain™ model, by copying optical and processing components from the WaveTrain™ component libraries, and connecting the inputs and outputs of the components
2. Set the numerical parameters of all components, and the simulation timing parameters
3. Run the simulation
4. Inspect the WaveTrain™ simulation outputs, and post-process as needed

Steps (1)-(3), and parts of step (4), are carried out within WaveTrain™ 's visual programming environment. Arbitrary post-processing in step (4) requires the use of an auxiliary visualization and computation environment, such as MATLAB® .

2.1 Step 1 - Assembling and connecting the WaveTrain™ components

For example, suppose we want to look at the optical effects of propagation through atmospheric turbulence. We might assemble a basic model like shown in Fig. 2.1 of a simple point source imaging system. The window in Fig. 2.1 is called the TSE window (equivalently, the Block Diagram Editor or just System Editor), and is used for assembling the optical model from individual components. The displayed model consists of eight components (also called subsystems, or modules) connected together. All of the subsystems shown can be found in the WaveTrain™ component library, and all of the connections in this particular model represent optical interfaces. (Connections that correspond to electronic or general data signals are also available). Starting at the far right, we have a Point Source, a TransverseVelocity, an AtmoPath, another TransverseVelocity, a Telescope, and an IncomingSplitter, which splits the incoming light and sends part to a Camera and part to a SimpleFieldSensor. The PointSource represents an idealized point source, radiating uniformly in all directions. The two TransverseVelocities can be used to model source and detector transverse motion, and/or a true wind velocity. The AtmosphericPath module is a complex module that represents two processes: (a) the optical effects of turbulence using multiple discrete phase screens distributed along the propagation path, and (b) the diffractive propagation of light along the propagation path. By assigning screen motions, AtmosphericPath can also model transverse wind velocities that vary spatially along the propagation path. In between the phase screens the optical wavefronts are propagated using a two-step FFT propagator. The Telescope applies an aperture and a focus adjustment to the incoming light. The IncomingSplitter duplicates the incident beam and sends it out into two branches. The SimpleFieldSensor records the complex field - amplitude and phase - at the aperture plane. The Camera brings the light to a focal plane (using a Discrete Fourier Transform) and records the intensity pattern in that focal plane.

This relatively simple model can be assembled by a novice user in an hour or so - see our step by step tutorial - and it is not a toy; it can do serious calculations.

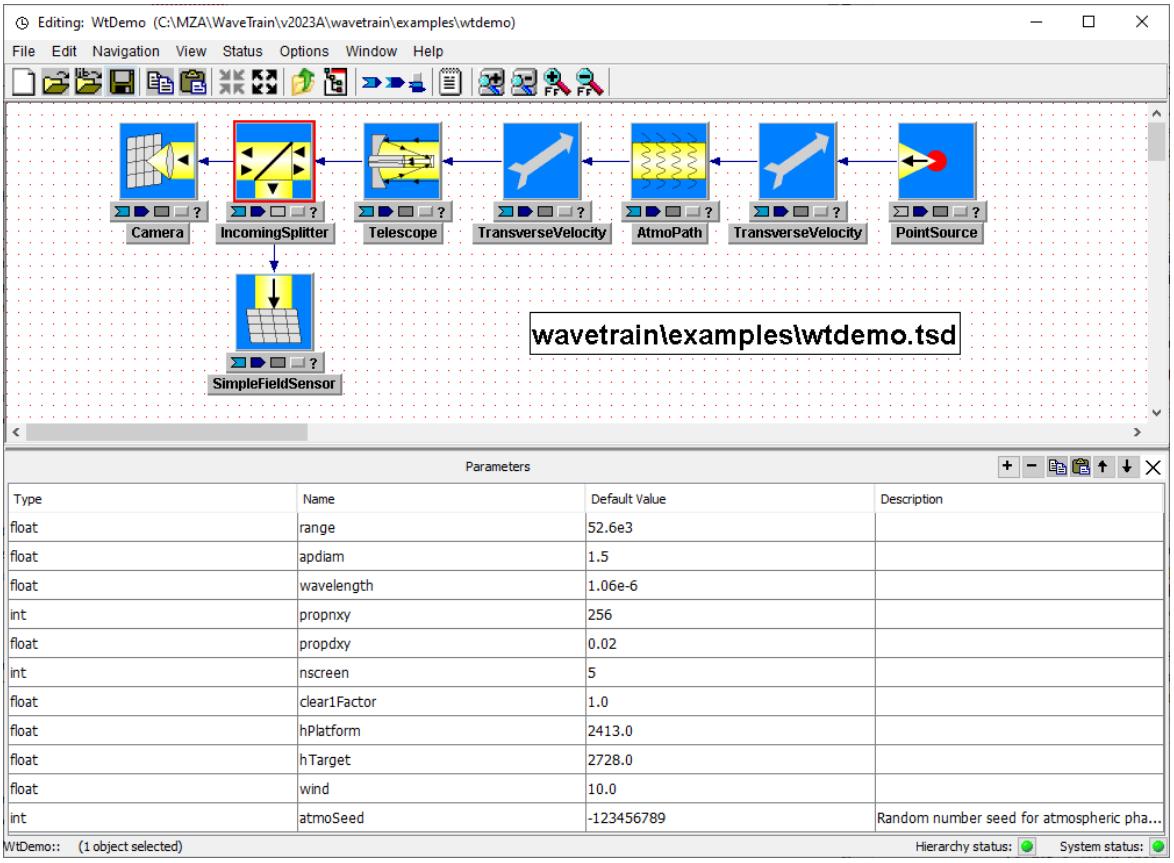

Figure 2.1: WaveTrain™ 's TSE, showing the WtDemo propagation model

2.2 Step 2 - Setting the numerical parameters of all components

After assembling and connecting the system components, the user must assign all the necessary parameter values. These include source powers, aperture diameters, propagation distances, propagation mesh dimensions, turbulence phase screen parameters, sensor spatial and timing parameters, etc. Setting all these parameter values is done partly in the TSE shown in Fig. 2.1 and partly in a second editor window, called the TSE. A sample runset for the WtDemo system is shown in Fig. 2.2.

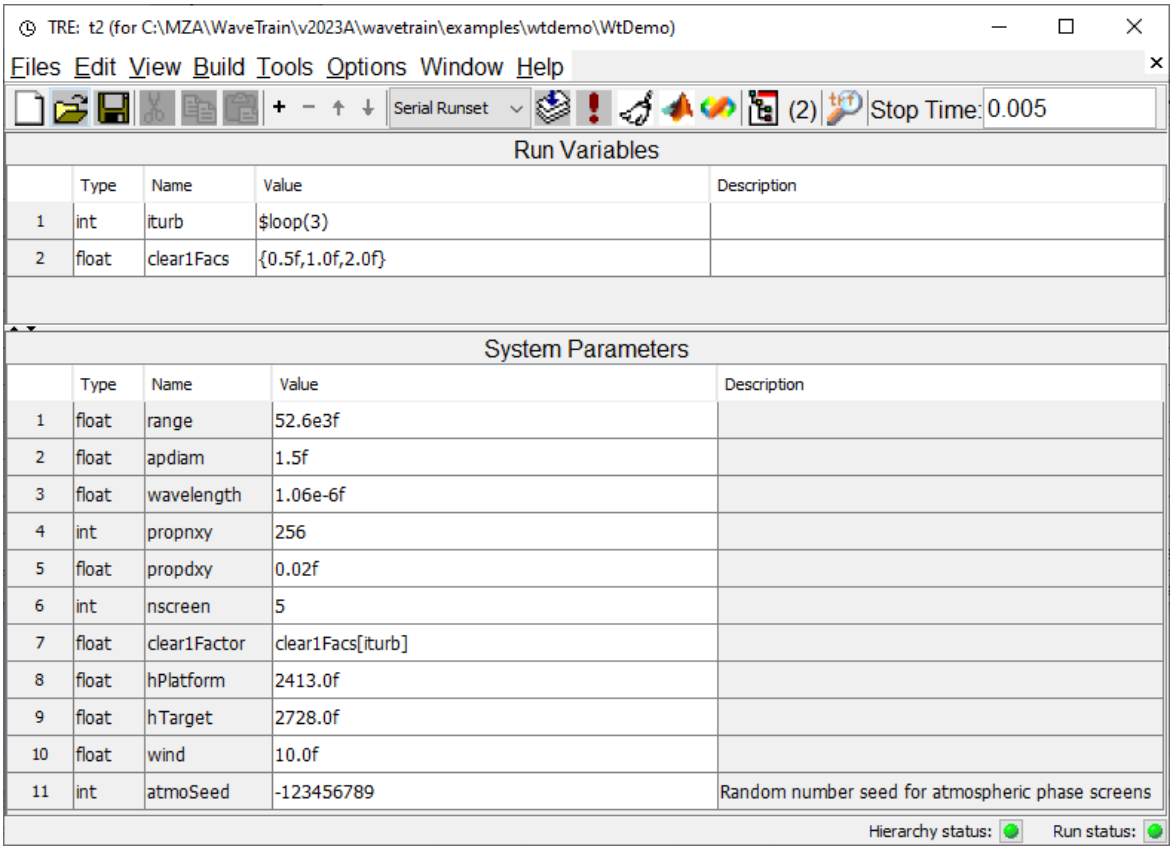

Figure 2.2: WaveTrain™ 's TSE, showing a runset for the WtDemo propagation model

2.3 Step 3 - Running the simulation

After specifying all the parameter values, the user initiates execution of the simulation with a few button clicks in the TSE. Upon completion of the execution, WaveTrain™ 's output - a combination of sensor images, complex field data, and/or electrical signal data - is stored to data files in a form that can be conveniently accessed for visualization and post-processing.

2.4 Step 4 - Inspecting and post-processing the WaveTrain™ outputs

Visualization and limited post-processing of outputs can be accomplished using the TrfView viewer that is part of WaveTrain™ suite. Typical outputs might be the instantaneous wave amplitude and phase maps in the "telescope1" pupil plane of the above WtDemo model. These particular outputs are based on the complex optical field sensed by the SimpleFieldSensor component in the model. Sample outputs as displayed in TrfView are shown in Fig. 2.3.

Completely general visualization and post-processing can be accomplished using WaveTrain™ 's MATLAB® interface. WaveTrain™ can be used with or without MATLAB® , but WaveTrain™ provides a full-featured MATLAB® interface. Not only can WaveTrain™ output data be conveniently imported into MATLAB® , but also complete WaveTrain™ system models

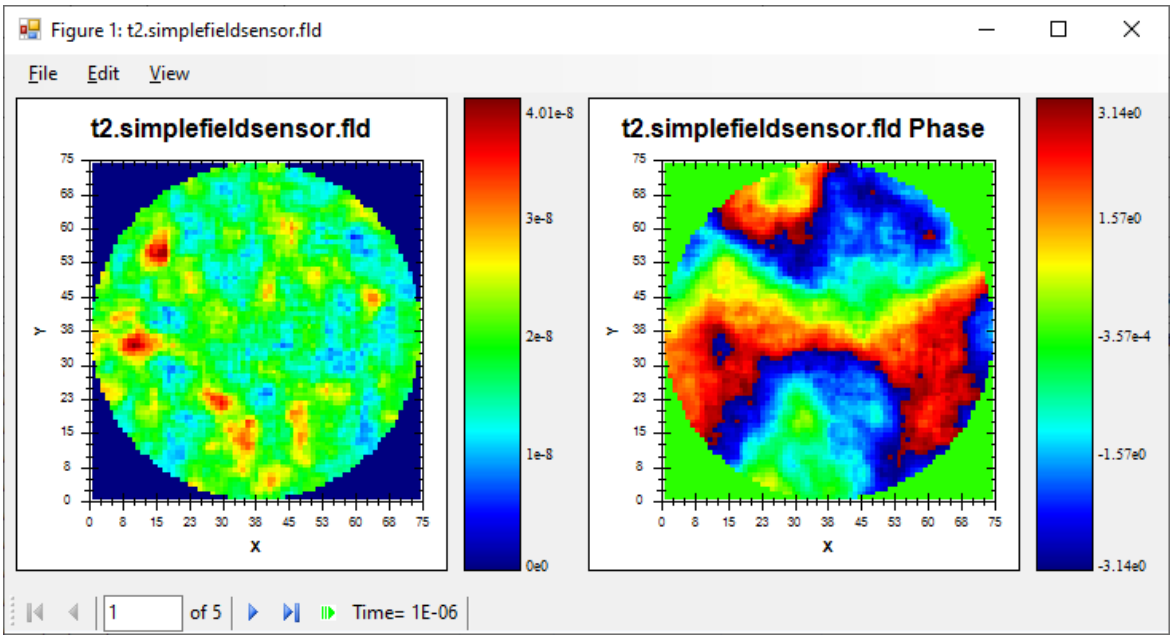

Figure 2.3: WaveTrain™ sample outputs - amplitude (left), and phase (right) maps in the telescope pupil of the WtDemo system, as displayed in WaveTrain™ 's TrfView viewer

can be created and executed from within MATLAB® , or converted into "S-functions" for use within Simulink, which is MATLAB® 's general purpose simulation environment. Additionally, MATLAB® m-files can be automatically converted into WaveTrain™ components, so that MATLAB® users have a simple way of creating custom WaveTrain™ components to supplement the WaveTrain™ library.

By varying the parameters of the WtDemo model illustrated in the Fig. 2.1 and Fig. 2.2 the user can investigate a wide variety of issues fundamental to the problem of optical propagation through turbulence. These issues include the statistics characterizing the spatial and temporal variation of the amplitude and phase effects, in both pupil and image planes of a receiver, and the way those statistics changes when the distribution of turbulence along the path is changed.

2.5 Auxiliary tools for defining input parameter specifications

In addition to the fundamental TSE and TRE, the WaveTrain™ suite includes several other GUIs. The purpose of these auxiliary tools is to assist in setting the numerical inputs of some of the WaveTrain™ components, particularly the more complicated modules that deal with atmospheric turbulence and AO components. For example, there is a MATLAB® GUI for setting up deformable mirror and wavefront sensor geometries. Also, there is a MATLAB® GUI that facilitates turbulence strength and atmospheric specifications and allows the evaluation of certain analytical formulas that estimate key integrated-turbulence quantities (such as scintillation variance and Fried's r0).

2.6 Scope and development of WaveTrain™

WaveTrain™ is designed to be useful to scientists, engineers, teachers, and students engaged in the design and development of advanced optical systems, or in the study of propagation through turbulence and associated AO systems. It is equally well-suited for simple models, like the example shown above, for intermediate complexity models such as are typically used in concept exploration, and for the kind of detailed engineering models necessary for accurate performance prediction, design refinement, and trouble-shooting. We are continually adding new components and features, and making improvements, often in response to customer feedback. If you need or want a feature that we don't yet offer, please let us know. Alternatively, WaveTrain™ is designed to be extensible, so you can create your own components to supplement the WaveTrain™ library, as described in the chapter on creating your own WaveTrain™ components.

3. Step by Step Tutorial

This tutorial is a compromise between constructing the simplest possible first system yet will be interesting to most WaveTrain™ users. A new user should be able to work through the tutorial in a few hours. The tutorial has the following goals:

- To lead a new user step by step through the construction and execution of a WaveTrain™ simulation.
- To introduce a new user to several of the key WaveTrain™ components that are needed for most WaveTrain™ systems.
- To introduce the user to some of the specialized nomenclature that is used in the WaveTrain™ program.
- To introduce the user to TrfView, which is WaveTrain™ 's utility for quick inspection and plotting of simulation results.

There are also various tutorials, workshop slides, and various other materials that can be found at the online WaveTrain™ documentation site. Slight drawbacks of those online documents may be that:

1. They may use illustrations or procedures from previous version of WaveTrain™
2. Certain core procedures may have changed, particularly some of the MATLAB® tools
3. Some breifing material was meant for use in an instructor led training environment so may be missing some explanations

Despite these warnings, users may still find the extra tutorials useful, particularly in describing the underlying wave-optics techniques used.

WaveTrain™ also comes packaged with several examples. Each example system includes runsets and any required input files. The purpose of the examples is to provide working systems which involve application of different WaveTrain™ concepts and to give users a start on building more complex systems. The example systems are distributed with standard WaveTrain™ installations and may be found in the sub directory "wavetrain
examples" of the WaveTrain™ installation directory and depending on your installation type in your default Documents folder. It is recommended to make a copy of the example system before modifying it in order to preserve the orginal.

3.1 Create a new WaveTrain™ System Model

To start the WaveTrain™ user interface, double-click the WaveTrain™ desktop shortcut, or use the Windows Start menu looking for MZA WaveTrain 2025A → WaveTrain. The desktop shortcut and start group should have been generated during the WaveTrain™ installation process. Starting WaveTrain™ brings up a small toolbar with the title "TVE". The TVE is the overall graphical interface program through which the user interacts with WaveTrain™. A picture of the TVE toolbar is shown in Fig. 3.1.

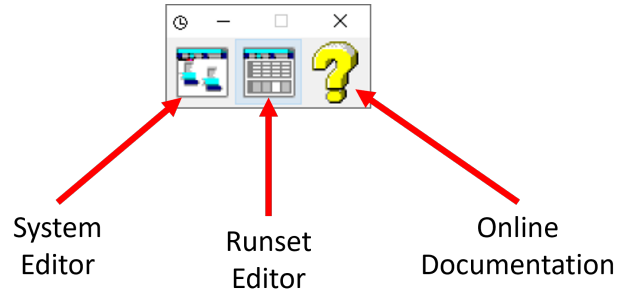

Figure 3.1: The WaveTrain™ TVE

To begin the setup process for a new system, go to the TVE toolbar and left-click on the System Editor button. This brings up a blank TSE window, as shown in Fig. 3.2. A new system is created either by selecting **File → New → New System**, clicking on the far left icon in the toolbar, or using the keyboard shortcut **Ctrl-N**. This changes the window title to "NewSystem" instead of "no system loaded". The TSE is now ready to accept the insertion of new components to build a system.

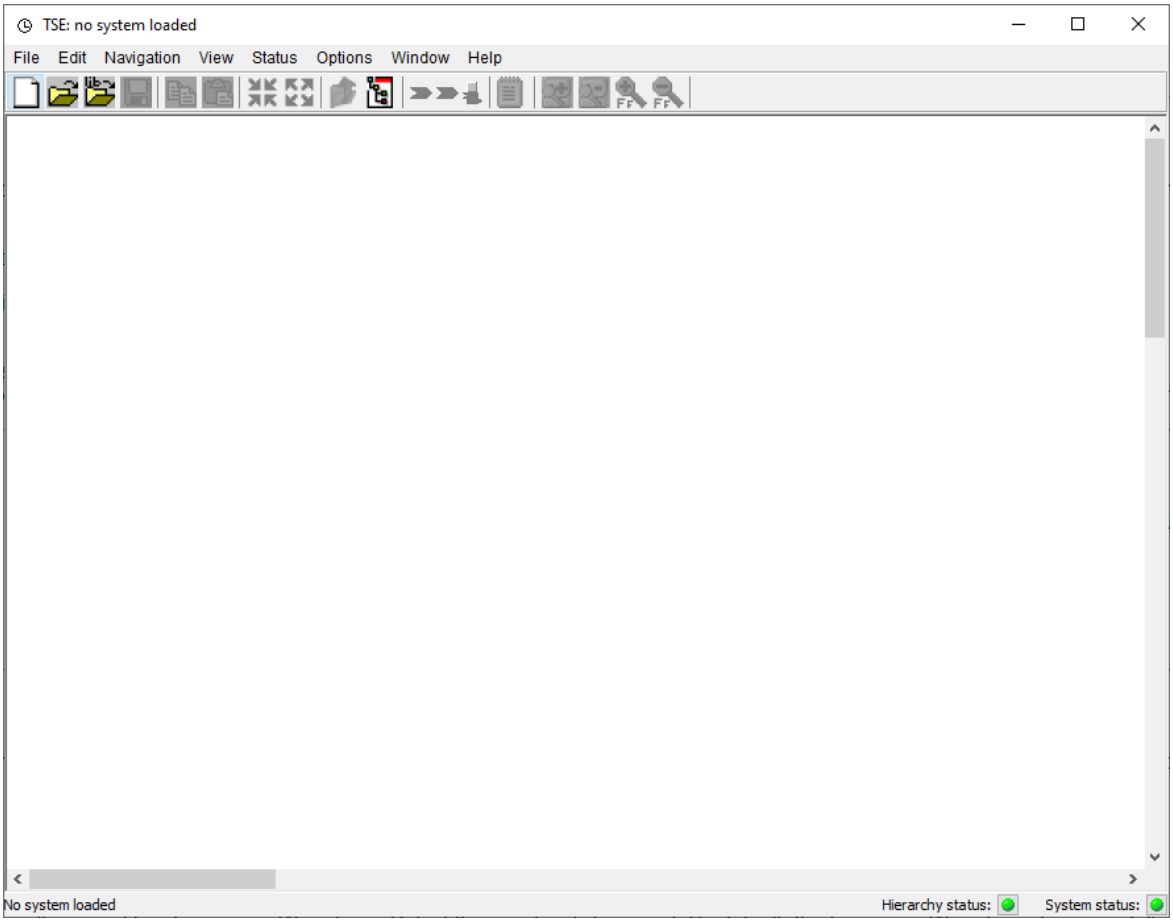

Figure 3.2: A blank TSE window after opening

At this point new components can be copied from the WaveTrain™ libraries into the "NewSystem" window. There are several methods for inserting library components into a new system:

- Open a second TSE window containing the WaveTrain™ library using the "lib" icon in the toolbar (third icon from the left) then copy and paste individual components into the new system
- Open a previously built system and copy components over to the new system
- Insert components directly by right clicking in the TSE window and selecting "Add Sybsystem", selecting the appropriate library, and navigating to select the desired component

3.2 The WaveTrain™ Component Libraries

The Master Component library is show in Fig. 3.3. This contains several libraries for different optical components, sensors, signals, effects, among other things that WaveTrain™ can simulate. However, the main library that users should start with is the *WtLib*, the top system in the Master Library, shown in Fig. 3.4. This library contains the key components

that can be used to simulate the most common systems. While the other libraries have many useful components, the ones in *WtLib* have the benefit of many years of use and standardization in terms of their implementation and documentation.

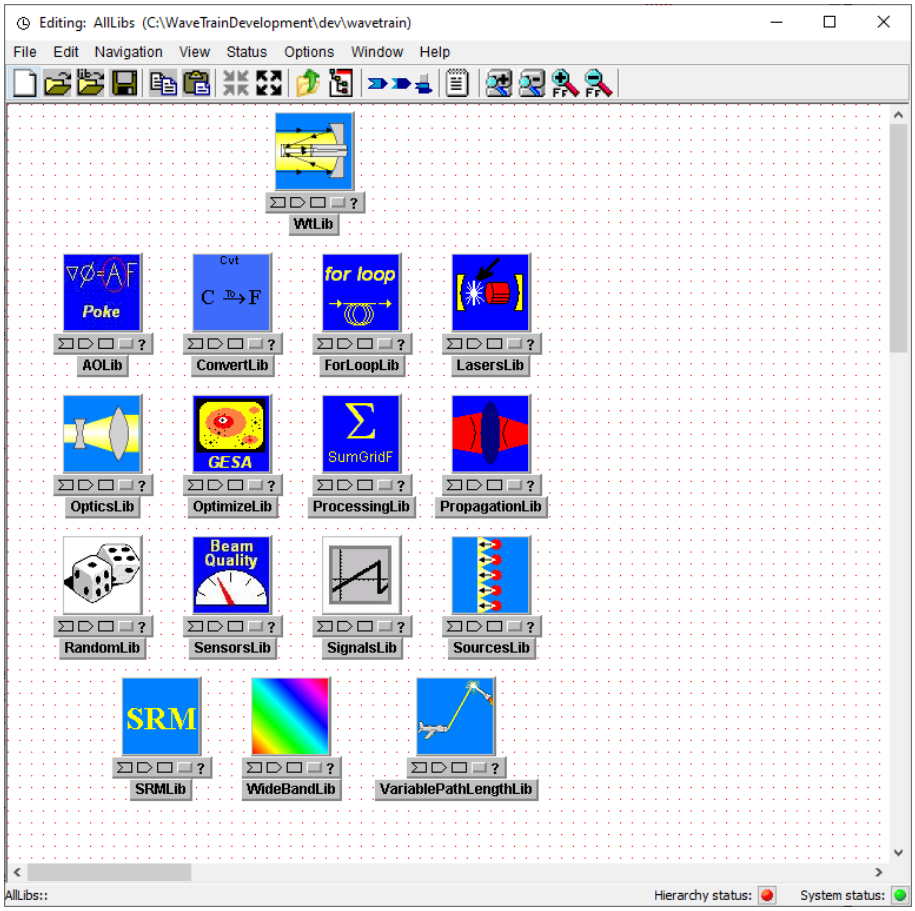

Figure 3.3: WaveTrain™ Master Component Library

The six sub libraries of the core *WtLib* library are:

- *AtmosLib*, containing components for modeling optical propagation through atmospheric turbulence or vacuum
- *ControlsLib*, containing components for modeling control systems for tracking and adaptive optics
- *OpticsLib*, containing components for modeling a wide variety of optical components and effects
- *SensorLib*, containing components for modeling optical sensors such as targetboards, cameras, and wavefront sensors
- *SignalLib*, containing components for modeling signal generators such as square wave generators (used for triggering sensors)
- *SourceLib*, containing components for modeling light sources such as pulsed and continuous wave lasers, point sources, and optically rough reflectors (which are classed as "sources" in WaveTrain™)

The rest of this Tutorial will use the components of the *WtLib*.

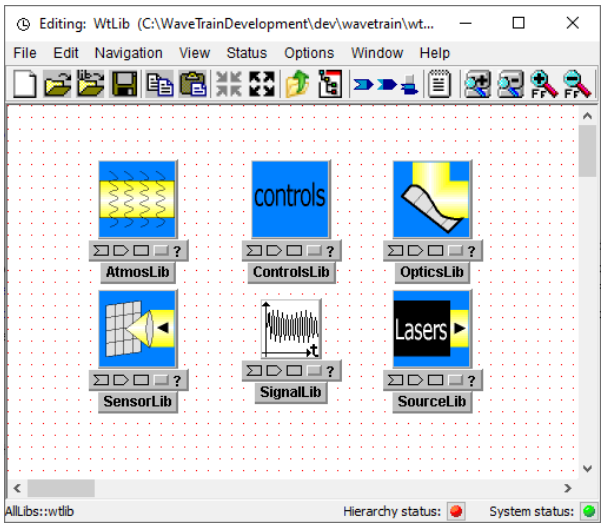

Figure 3.4: The *WtLib* Library

3.3 Copy Components From One TSE Window to Another

Now we will begin constructing a new WaveTrain™ system by copying components from the *WtLib* library into the "NewSystem". The order in which the component are copied makes no difference, however it seems logical to start with a source. In the *WtLib* window double-left-click on the *SourceLib* icon. This causes you to "descend" into the sub-library so that you now see (in the same window) a variety of individual WaveTrain™ source modules (components) which have been collected for convenience into the sub-library. The resulting window should look similar to Fig. 3.5 although the exact icons and their arrangement may differ. Note that the image in Fig. 3.5 the window size hides some of the individual components.

Note the toolbar button that is marked with the red arrow in Fig. 3.5. This button has Window's standard "up-directory" icon. In the TSE contest, this signifies going up one level in a system hierarchy. If you left-click that button you will return to *WtLib* level, and then left-clicking again will return you to the *AllLib* level. Alternatively, you can go up one level by double-left-clicking on the blank space in the TSE window, which is arguably more convenient. As you will see as we go along, the libraries are just nested systems built in the same way as systems used for simulations but without any connections or eiher missing or default parameter settings.

New you can begin your system construction by locating and copying the *PoointSource* component into your "NewSystem" window. The steps are:

1. Find the *PointSource* component in the *SourceLib* window
2. Select *PointSource* by moving the cursor over the icon and left-clicking (notice this outlines the icon in red)
3. Either use the TSE's menu bar to execute **Edit → Copy PointSource**, left-click on the standard **Copy** icon on the toolbar, or right-click on the *PointSouce* component and select **Copy**. You can also simply use the keyboard shortcut **Ctrl-C**. This copies the *PointSource* component to the TSE clipboard.
4. In the "NewSystem" window either use the TSE's menu bar to execute **Edit → Paste PointSource**, left-click on the standard **Paste** icon on the toolbar, or right-click on the *PointSouce* component and select **Paste**. You can also simply use the keyboard shortcut **Ctrl-V**. This pastes the current clipboard contents to the current system.
5. When first pasted the component will likely be put in the upper left hand corner of the TSE. By lef-clicking the icon and dragging it you can move it to any desired location with the current TSE window.

To select and past additional components, simply repeat the copy-paste procedure for the additional components. In order to follow along with the illustrations and exercises in the remainder of this chaper you should now copy and paste the remaining components in Fig. 3.6. You will ultimately be building the WtDemo system that was shown in Fig. 2.1.

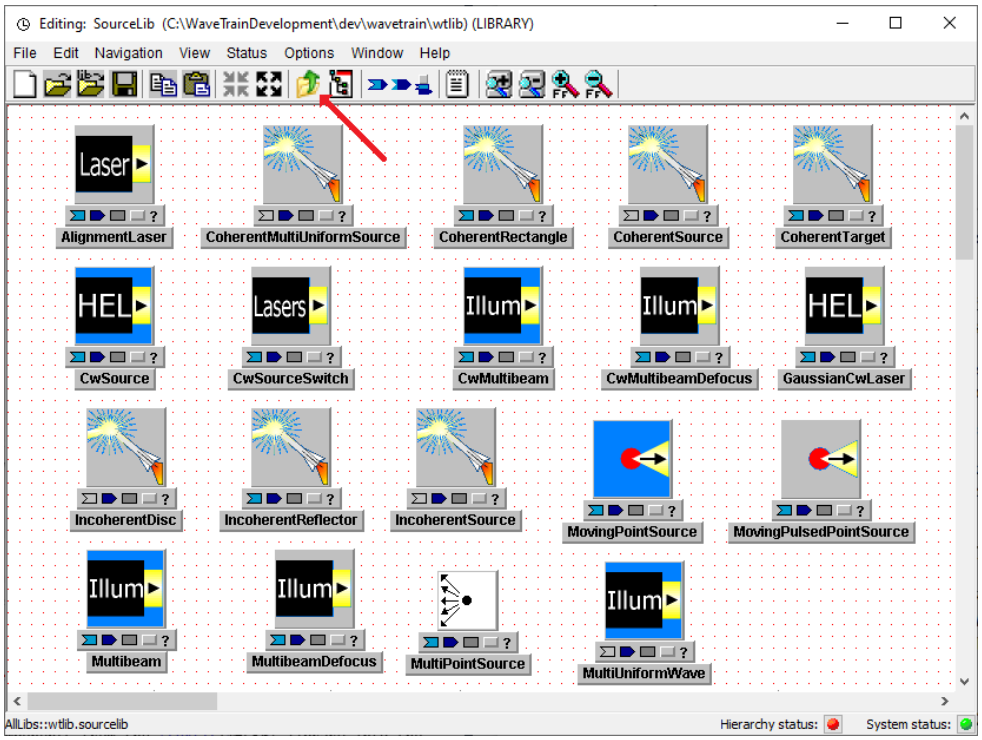

Figure 3.5: The Contents of the *SourceLib* sub-library of the *WtLib* library

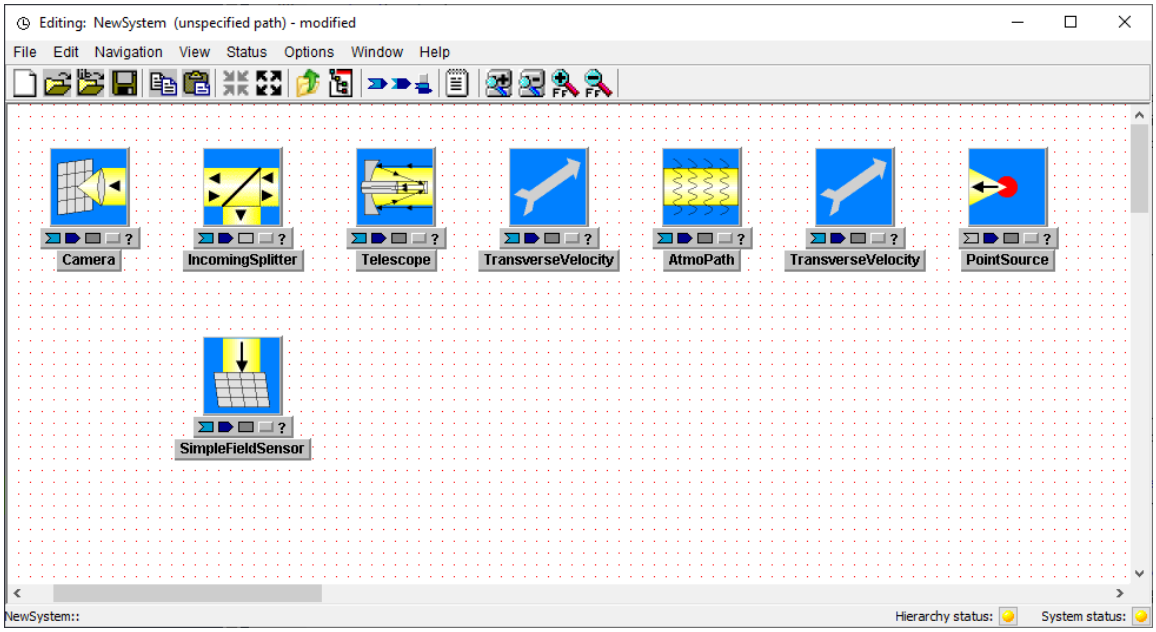

Figure 3.6: Assembled components for the new system

Do not be concerned if your component icons look somewhat different than the ones in the Fig. 2.1. The icon pictures have occasionally changed during WaveTrain™ development cycles. However the names of the systems in the title bars at the bottom of the icons should be identical. The icon pictures themselves should be interpreted loosely, and have impact on the actual contents of the component or its function. For example, your *IncomingSplitter* may have the incorrect orientation of the split relative to the location of the *SimpleFieldSensor*. The icons themselves can be changed to reflect whatever orientation you choose based on your system layout, and can even be modified as described in **??**, which can be useful when building up a layout of an existing, real world system.

If you make a mistake, or are just experimenting, you can remove a component from your system by selecting it with a left-click and use the TSE menu sequence **Edit** → **Delete**. You can also right-click on the icon and select delete. Finally, you can just use the keyboard shorcut **Delete** with the desired icon selected.

It is also allowed to select more than one component at a time. Using a left-click, and holding the mouse button down, you can drag a box around the desired components. You can also use a Ctrl-left-click sequence to select a second, third, etc. components. The selected components will all be copied at once and pasted at once. When first pasted, they may all appear in the upper-left corner, stacked on top of each other. You can then drag them individually to their desired locations. However, if components that are group copied are connected to each other, their connections will be preserved in the copy.

3.4 Saving Systems and Opening Existing Systems

3.4.1 Saving

At this stage you'll want to save your new system before proceeding to the next stage of system construction. A system can be saved at ny stage, no particular level of completeness needs to be achieved before saving. To save the new system the steps are

1. In the TSE menu bar left-click **File** → **Save As** or press the **Save** icon on the toolbar.
2. Navigate to any dierectory where you want to save the system (it is preferrable to *not* use any directories that are part of the core WaveTrain™ installation.
3. Specify a system name of your choice and press **Save**. The file extension **.tsd** (tsd = "tempus system definition") will be automatically appended to the name you specify.

Depending on the last location used for the TSE, the **File** → **Save As** may first place you in the WaveTrain™ library directory or sub-directory. You do *not* want to save your system there. Whereever you choose to save it, it is usually good practice to store each new WaveTrain™ system in its own directory. The process of building and executing the system will generate numerous files and folders that are easier to manage when the folder is limited to a single system. After the system has been saved for the first time, subsequent changes can be saved by using just **File** → **Save** or the **Save** icon on the toolbar.

3.4.2 Opening and existing system

Suppose that you wish to work with a system that has been previously saved but is currently not open in the TSE. To open the system the steps are:

1. Open a new TSE window from the TVE toolbar.
2. In the TSE menu bar left-click **File** → **Open** → **Browse** or press the **Open** icon on the toolbar.
3. In the resulting directory window, navigate to the directory where the system of interest is stored.
4. Select the **.tsd** file that bears the name of the system you want and press **Open** or double-left-click the **.tsd** file name.

You can also open recently opened systems using **File** → **Recent Systems** and look for the recent system of interest.

3.5 Component Parameters, Inputs, and Outputs

After you copy or add components (subsystems), you will need to assign values to their ”parameters”, and possibly to some of their ”inputs”. These two terms are used in a specialized sense in WaveTrain™: they are both inputs in a generic sense, but the distinction is that ”parameters” are fixed in time, whereas ”inputs” may or may not change with time.

A typical component ”parameter” might be an aperture diameter, a source power, or a mesh spacing.

A typical component ”input” might be a ”wavetrain” (WaveTrain™'s data structure that carries the optical field information). The data in a wavetrain incident on a component usually changes as the simulation time advances. On the other hand, timing parameters that define sensor exposures are also ”inputs”, even though most of these do not change with time.

A typical component ”output” might be a wavetrain, or a time-integrated sensor exposure map.

A component may have any number of parameters, inputs and outputs. A parameter must be assigned a value. An input usually receives its value by being connected to the output of another block, but sometimes an input is assigned a value explicitly. Outputs are usually connected to the inputs of other components; in some cases, outputs are not graphically connected to anything, but may just be recorded. These options will become clearer as you continue with the construction of the tutorial system.

In the subsequent sections of this introductory tutorial, you will see many examples of parameters, inputs, and outputs. We will refer to them without quotes from now on; it should be clear from the context whether ”input” is meant in the specialized WaveTrain™ or the generic sense.

3.6 Display or Hide Graphical Elements

Before actually setting any parameters or connecting inputs and outputs, you must become familiar with certain mechanics of the GUI related to displaying and hiding of the component graphical elements. Notice that, in the copy-paste operations you performed above, the component (subsystem) icons consisted of a picture, a miniature toolbar beneath the picture, and a component name bar below that.

3.6.1 Displaying/hiding the components parameters, inputs and outputs

Due to the finite display space available, it is frequently useful (or necessary) to hide various graphical elements. Of course to initially set the parameters and to connect the inputs and outputs, these items must be displayed. To display or hide, you will use one of the toolbars buttons in the System Editor window, and also the miniature toolbars located underneath the component icons. The master control for displaying and hiding is the toolbar button indicated by the red arrow in Fig. 3.7. In the TSE window in which you have been assembling your system, left-click the indicated button. That pulls down the palette which you see in Fig. 3.7. This palette contains six rows of buttons: for now we are just interested in the last four rows, each of which consists of three or four small buttons. The first of these rows controls display of System Names. The second row controls display of Inputs. The third row controls display of Outputs. The fourth row controls display of Parameters.

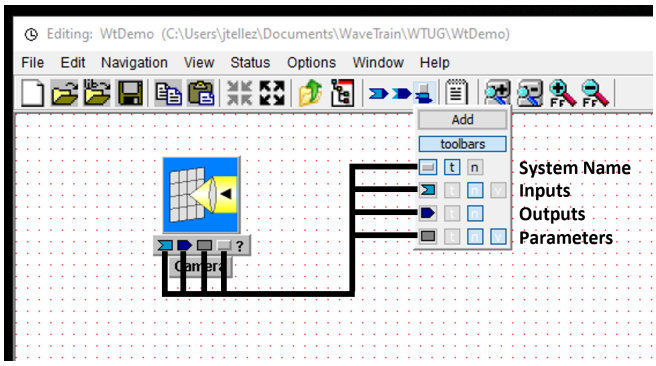

Figure 3.7: Master control for displaying and hiding component graphical elements

3.6.2 Global display/hide

As an initial exercise, after pulling down the palette, left-click on the leftmost **Inputs** button: in your new system window, you should now see that the inputs of all your components are now displayed. The display may be messy because the icons are too close to each other for present purposes. Grab one or two of the icons and move them to see everything for those icons. The palette buttons are all "toggles", so left-click the same button again to hide the inputs. The leftmost **Outputs** button serves the same function for component outputs, and the leftmost **Parameters** button serves that function for the component parameters. Any combination of inputs, outputs, and parameters may be displayed. Finally, note that the leftmost "System Name" button allows you to turn off/on the display of component (system) names; these take less room, so usually there is not much point in hiding the names.

3.6.3 Individual component display/hide

Instead of applying the display/hide toggles to all components at once, it is often very helpful to apply the toggles to one component at a time. To do this, simply select (left-click) the icon of interest, and then press the palette buttons until the you acheive the desired display state.

3.6.4 Displaying/hiding type, name, value elements

Next, consider the palette buttons labeled "t" (= type), "n" (= name), "v" (= value). The **Inputs**, **Outputs**, and **Parameters** rows in Fig. 3.7 have such buttons. As an exercise, select the *SimpleFieldSensor* icon in your new system, and use the palette buttons in the leftmost column to display the inputs, outputs, and parameters. When all elements t, n, and v are displayed, your component should look like the Fig. 3.8. Note that inputs (light blue fields) and parameters (light grey fields) have three columns: type, name, and value. Outputs have only two columns: t and n. You can use the subsidiary palette buttons labeled t, n, and v to display/hide any combination of individual elements of the **Inputs**, **Outputs**, and **Parameters** fields. As you become more familiar with the purpose of the elements you can decide to hide or display them as needed.

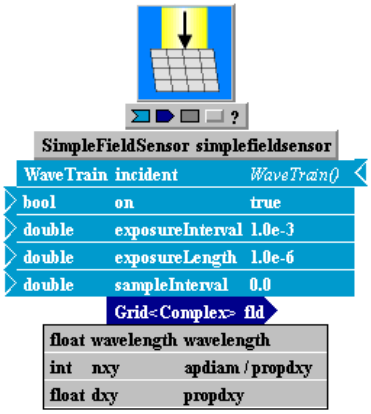

Figure 3.8: ***SimpleFieldSensor*** **with everything displayed**

3.6.5 Miniature toolbar under component icon

From inspection of Fig. 3.8, you will see a miniature toolbar directly below the component icon, above the component name. The mini-toolbar has symbols that look exactly like the buttons in the left column of the palette. Once you have used the palette to set the display/hide properties of the t, n, v elements to your liking, then you can more quickly display/hide the "System Name", **Inputs**, **Outputs**, or **Parameters** rows by just left-clicking the mini-toolbar buttons. Note that clicking the mini-toolbar buttons preserves the t, n, v settings that you have chosen with the palette.

3.6.6 The component name fields

We have not yet said much above about the "System Name" row of the palette, and the display fields that it controls. The system (component) name fields have a different function than the **Inputs**, **Outputs**, and **Parameters** fields, although the display/hide features are controlled in exactly the same way using the palette and the mini-toolbar. The example in Fig. 3.8 displays both t and n for the "System Name" row. The t (type) field in the "System Name" bar corresponds to what is technically called the "C++ class name" in the WaveTrain™ code. The text in this field, "SimpleFieldSensor", is not user-editable. The n (name) field in the "System Name" bar is also assigned by default when you insert the component into your system. However, the text in the n field is arbitrarily adjustable by the user. There is usually no need to make any adjustment to the default assignment, but this name can be changed to conform to user preference. If you look closely at your system assembled in Section 3.3 you should have two *TransverseVelocity* components as shown in Fig. 3.6 . If you make the names visible the you will see that the TSE assigned the default name for one of them as "transversevelocity2". That is because we inserted two *TransverseVelocity* components into our new system, and this was the second "instance" of the *TransverseVelocity* type that was inserted. You are allowed to change the name and suffix number to anything that seems meaningful or convenient to you, by clicking on the name and editing. The numbering need not be consecutive, or you can just use different names and no numerical suffix: the only requirements are that multiple instances of a "class" must have distinct names, names cannot be the same as the class or component name, and names cannot have any spaces or special characters that are not allowed in C++ variable names.

3.7 Set Component Parameters

Now that you have mastered the displaying and hiding of display elements, you are ready to set the parameters. In your new system from Section 3.3 shown in Fig. 3.6, select the *SimpleFieldSensor* component, and display all the **Inputs**, **Outputs**, and **Parameters** elements to follow along. After you've copied *SimpleFieldSensor* from *WtLib*, and you've displayed all the elements, it should look like what is shown in Fig. 3.8.

We now want to enter desired values in the value fields (rightmost column) of the **Parameters** block (light grey) and the **Intputs** block (light blue). In WaveTrain™, we use the term "setting expression" to denote the expression entered in a value field. Note that in this example all the value fields are already filled in with some default setting expressions. This will not be the case with all components: sometimes the value fields are initially blank. In either case, these initial settings are usually not what you want in your new system.

Note that some setting expressions in Fig. 3.8 are simple numbers ("1.0e-3"), some are symbols ("wavelength", "propdxy"), some are boolean expressions ("true"), and some are algebraic expressions ("apdiam / propdxy"). After you finish the introductory tutorial, you can obtain full details on the expressions and functions that are allowed in setting expressions by referring to the detail chapter on Data Entry.

The simplest method of assigning a setting expression is to enter a number. If you enter a number and press <Enter>, you are completely done with that parameter. However, this is often undesirable for two possible reasons:

1. You may want to link the value to that of another parameter value.
2. You may want to vary the value later at the level of the TRE (to be discussed later in this chapter).

To handle both situations you must assign a symbolic name or algebraic expression as the setting expression.

To change or initially assign a setting expression, left-click in the value field of interest and begin typing. A few rudimentary text-editing features are available, e.g. text selection and copy-paste via <Ctrl>-C and <Ctrl>-V. Press <Enter>to complete the edit. You will now practice entry of setting expressions by changing or accepting all the value fields of *SimpleFieldSensor*:

3.7.1 Enter setting expressions for *SimpleFieldSensor*

1. Parameter name *wavelength*
   - Currently, the value of this parameter is also "wavelength". For practice, enter the new setting expression "wvln". You are allowed to use the same symbol for a parameter setting as the parameter name, or a different one: the default here was the same, but you just changed the value to "wvln".
   - When you pressed <Enter>, the TSE popped up another window, entitled "Expression Contains Undefined Identifiers", as shown in Fig. 3.9.

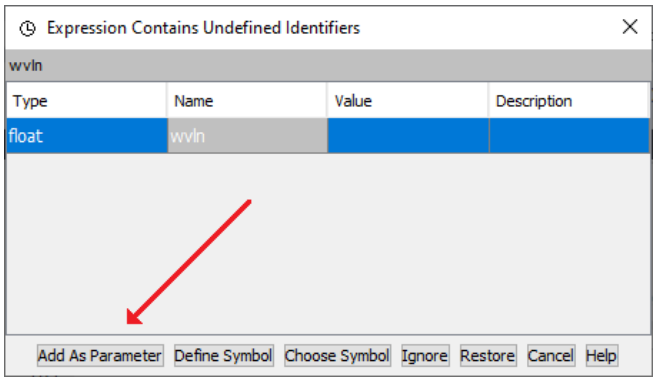

Figure 3.9: Dialog box when using undefined variables in setting expressions

- Note that symbols in WaveTrain™ are case sensitive, e.g. "Wvln" is different than "wvln".

2. Parameter name *nxy*

   nxy and *dxy* define the space mesh on which the *SimpleFieldSensor* will report its results. Let's just accept the default expression "apdiam / propdxy" that is already present.

   - Left-click in the value field, and simply press <Enter>
   - The TSE again pops up the "Undefined Identifiers" window: press "Add as Parameter" twice to register the symbols "apdiam" and "propdxy"

3. Parameter name *dxy*

   Again, let's just accept the default expression.

   - Left-click in the value field, and simply press <Enter>
   - This time, the System Editor does not produce the "Undefined Identifiers" window: that is because the symbol "propdxy" has already been registered in the previous step, when you defined the setting expression for parameter name "nxy".

   In the *SimpleFieldSensor* module, the last four inputs (in addition to the parameters) also require setting expressions. On the other hand, the input of type "WaveTrain" will recieve its input by being connected to the output of another block, so we enter nothing in the value field of that first input bar:

4. Input name *on*

   - Left-click in the value field, and simply press <Enter>
   - The boolean symbol "true", which you've just accepted is already defined in the WaveTrain™ code, so the "Undefined Identifiers" window did not pop up.

   The "true" setting means that the sensor's first exposure window will start at simulation time $t = 0.0$.

   The next two inputs mean exactly what they say. They define, respectively, the interval between start of successive exposure windows, and the length of each exposure window. All the basic WaveTrain™ sensors are time-integrating sensors, although later you may encounter sensors that directly report power or irradiance.

5. Input name *exposureInterval*

   - Accept the default numerical value

6. Input name *exposureLength*

   - Accept the default numerical value

Numerical values WaveTrain™ are in meter-kilogram-second system (MKS) units, unless explicitly specified otherwise. Therefore, all time quantities are in seconds.

7. Input name *sampleInterval*

   This name may be a little confusing. It allows you to perform multiple WaveTrain™ propagations during each *exposureLength* window.

   - Leave it set at 0.0 for this tutorial. This means that the optical field at the sensor will be computed once during each *exposureLength*.

After completing the introductory tutorial, you can consult the detail section **??** for further information about timing options.

Parameters sub-window:

Before proceeding with the setting expressions for other components, you should now become aware of a sub-window of the System Editor. Using the System Editor View menu, execute the menu commands View - Parameters. This makes visible a sub-window within the System Editor, which we call the Parameters Panel, as shown in the following picture:

The Parameters Panel contains a listing of all the symbols that you have defined so far: "wvln", "apdiam", and "propdxy" all appear. If undesired symbols still appear in the list (e.g., from some experimentation you might have done), you can (and should) delete those by left-clicking in their row, and pressing the button at the top right of the Parameters Panel.

For now, let's ignore the "Default value" and "Description" columns in the Parameters Panel. You can toggle the Parameters Panel display on and off, as convenient, using View - Parameters.

Help pages for individual WaveTrain components

In this tutorial, we guide you by giving you valid setting expressions to enter for all parameters and inputs. Naturally, when you eventually do this on your own, you will need reminders of the meaning of, and valid possible settings for, the many component parameter and input names. This information can be obtained for each component by left-clicking the question-mark button, , at the bottom right of any component icon.

Try this, for example, with the SimpleFieldSensor component whose settings you entered above. You will see that a HTML help page opens in your web browser: this page contains a summary description of the component's function, and short physical definitions of each parameter, input and output. Sometimes these descriptions may be terse, and you may still have to consult sections of the User Guide for interpretation. Nevertheless, the component help page is the place to start.

The picture below shows the help page that you should see for SimpleFieldSensor: .

Enter setting expressions for remaining components:

At this point, you have completed the setting expressions for the SimpleFieldSensor module, and you should now enter the setting expressions for the remaining components as follows.

Enter setting expressions for PointSource:

Initially, PointSource copied from Wtlib has the default values shown below. (*) Since we decided above to represent the propagation wavelength by the symbol "wvln", you should change the symbol "wavelength" in the value field to "wvln". (*) Accept the default numerical values of the other parameters as they are.

(Note that the PointSource parameter named "power" is poorly named: this parameter is actually the Watts/steradian produced by PointSource).

Enter setting expressions for TransverseVelocity (first instance named transversevelocity, and second instance named transversevelocity2):

You will assign the parameter values to represent a uniform atmospheric wind speed, transverse to the propagation path, with stationary source and receiver. (Picture the three velocities with respect to an earth frame, which is considered inertial). (*) Change the default settings to match those in the pictures below:

Notice that we have entered "vWindX" in one motion module and "- vWindX" in the other. To understand the physical meaning of these settings (after completing the present tutorial), you can consult the User Guide detail sections for further information about the specification of transverse displacement and transverse motion in WaveTrain.

Enter setting expressions for AtmoPath:

This is a complicated component, with many parameters. AtmoPath contains specifications for both the Fourier propagation mesh and the turbulence phase screen parameters. For present tutorial purposes, we will enter settings here without explaining all the details. The details can be learned later by consulting various sections of the Modeling Details section of the User Guide.

Parameter name "atmSpec": This parameter contains most of the settings that define the atmospheric turbulence along the propagation path. The setting expression in this value field is something new compared with the components you worked

with above: it is a special function, "AcsAtmSpec(...)", that is part of the WaveTrain code. You will now set the arguments of this function: (*) Left-click in the value field, and change "wavelength" to "wvln", to accord with your previous symbol definitions. (*) Press <Enter>, and you will once again see the "Undefined Identifiers" window. Left-click the "Add As Parameter" button five times, to register the remaining default symbols that appear in the argument list, namely "nscreen, clear1Factor, hPlatform, hTarget, range". We will briefly discuss the physical significance of these symbols later in the tutorial, when we finally assign them numerical values.

Parameter name "propnxy": Parameter names "propnxy" and "propdxy" define the size and spacing of the mesh that WaveTrain will use for its Fourier propagations. Obviously, these are critical specifications. (*) Left-click in the value field, press <Enter>, and "Add As Parameter" to accept the default symbol "propnxy". (You already defined the symbol "propdxy" in a previous component").

Parameter names "xp1" through "yt2", and "screenDxy": These parameters define the dimensions and mesh spacing of the turbulence phase screens. The symbols in the respective setting expressions have all been registered previously, so you need do nothing here.

Remaining parameter names: The setting expressions are either numerical values, or symbols previously defined. You will accept all these for now, so no further action is required to complete the AtmoPath settings.

Enter setting expressions for Telescope:

This component performs two functions: (a) it applies a binary aperture (in general, annular), and (b) it undoes the wavefront curvature acquired due to propagation from a point source to the telescope. Function (b) works together with the Camera module that follows.

Parameter name "range": Here you simply enter the distance to an OBJECT plane whose image you want to generate in the focal plane of a subsequent Camera module: (*) The desired setting expression in the value field, "range", is already present by default, and you have previously registered this symbol. No further action is required.

Parameter names "apertureRadius" and "annulusRadius": (*) These already have the desired setting expressions entered by default, and the symbol "apdiam" has been registered previously. No further action is required. Note the mixed nomenclature: "apertureRadius" signifies the outer radius of the annular aperture, while "annulusRadius" signifies the inner radius, all in MKS units (i.e., meters).

No setting expressions required for IncomingSplitter

This component makes a copy of the incident wavefront and retransmits it into two separate optical paths. Note that the wavefront is duplicated: it is NOT split in energy.

Enter setting expressions for Camera:

Patience, you're almost done! Camera is another relatively complicated component, with many parameters. Again, for present tutorial purposes, we will enter settings here without explaining the details. The details can be learned later by consulting the basic sensor and Camera-specific detail sections of the User Guide.

(*) Edit the setting expressions in Camera's parameter value fields to match the picture below.

(*) One CAUTION: the parameters named "nxyPupil" and "dxyPupil" unfortunately have quite misleading names. Unless you explicitly use an esoteric feature of WaveTrain ("wavesharing"), these parameter values are NOT used in the calculation. The aperture which controls the diffractive spreading in Camera's image is the Telescope aperture that you specified above. (*) The setting expression you entered for the parameter named "dxyDetector" exploits the full resolution available from Camera's Fourier transform calculation. (*) Note that the numerical setting expressions for Camera's timing inputs are chosen to match those of the SimpleFieldSensor (matching is not required).

Execute the menu commands File - Save

Of course, you can do this at any time during the above work, but make sure you do so before proceeding to the next stage of construction.

References

[1] J. W. Goodman, *Statistical Optics*. New York: Wiley, 1985.

[2] V. I. Tatarskii, *Wave Propagation in a Turbulent Medium*. New York: Dover, 1961.

[3] A. Ishimaru, *Wave Propagation and Scattering in Random Media*, vol. 2. New York: Academic Press, 1978.

[4] R. K. Tyson, *Principles of Adaptive Optics*. Boston: Academic Press, 3rd ed., 2010.

[5] M. C. Roggemann and B. Welsh, *Imaging Through Turbulence*. Boca Raton: CRC Press, 1996.

[6] J. Hardy, *Adaptive Optics for Astronomical Telescopes*. Oxford: Oxford University Press, 1998.

[7] L. C. Andrews and R. L. Phillips, *Laser Beam Propagation through Random Media*. Bellingham: SPIE Press, 2nd ed., 2005.

[8] R. R. Beland, "Propagation through atmospheric optical turbulence," in *The Infrared and Electo-Optical Systems Handbook. Vol. 2, Atmospheric Propagation of Radiation* (F. G. Smith, ed.), pp. 157–232, Bellingham: ERIM and SPIE Optical Engineering Press, 1993.

[9] J. D. Schmidt, *Numerical Simulation of Optical Wave Propagation with Examples in MATLAB*. Bellingham: SPIE Press, 2010.

AO	adaptive optics
DM	deformable mirror
GUI	graphical user interface
MKS	meter-kilogram-second system
RHEL	Red Hat Enterprise Linux
TRE	Tempus Runset Editor
TSE	Tempus System Editor
TVE	Tempus Visual Editor
WFS	wavefront sensor