Lambda Research Corporation

Rev: 12.28.2018

Working with MATLAB® and TracePro® through Component Object Model (COM)

TracePro includes a COM (Component Object Model) programming interface to allow TracePro to communicate with other Windows programs. COM support in TracePro allows you to write programs that control the execution of TracePro. This is done by passing strings containing Scheme commands to TracePro via the COM interface.

TracePro COM

TracePro is an out-of-process COM server that will serve objects for clients on the same computer. Clients can create COM objects whether the server is running or not. The COM client will be able to start the registered release of TracePro. When TracePro is started via COM, the startup options, auto load scheme and check for updates will not be run. The COM programmer can use COM function calls to implement any desired startup behavior.

TracePro<edition> Object

The TracePro<edition> COM objects are registered in the Windows registry and can be used to programmatically control TracePro. The edition can be either ST (TracePro Standard) or XP (TracePro Expert). Each TracePro edition executable will have a unique TracePro object that will provide methods to interact with TracePro. Programmers will use the version independent programmatic identifier for the edition of TracePro that they have. TracePro.TraceProXP for TracePro Expert or TracePro.TraceProST for TracePro Standard to access these objects. The edition COM objects implement the ITracePro interface to provide methods to interact with TracePro.

ITracePro Interface

The following methods are used in this example, see TracePro help for a complete listing of the ITracePro methods.

HRESULT ExecuteSchemeString([in] BSTR Expressions, [out] BSTR* pResult, [out, retval] LONG* pNestLevel);

TracePro will execute the scheme code in Expressions and wait for it to complete. This behaves like the command line in the TracePro Message/Macro window.

Expressions: A string that contains the scheme code to execute. This string can represent several expressions and end with an incomplete scheme expression.

pResult: This will contain the string returned by the last complete command in Expressions or an error string if there was an argument error in a TracePro scheme command in Expressions. The processing of the Expressions will stop on an error or exception.

pNestLevel: This will contain the number of right parentheses that are required to complete an incomplete expression. It will be 0 if the last expression was complete or -1 if there was an error.

NOTE: We detect COM argument errors from TracePro scheme commands but ACIS prints the error messages and doesn't provide a way for us to know that there was a problem in ACIS scheme command or general scheme errors like the unbound variable.

HRESULT AllowUserControl();

By default, an MFC/ COM server application will not display the user interface when a COM object is created and the application is not running. Use this command to make the TracePro user interface visible if it wasn't already. NOTE: All COM objects will be destroyed if the user chooses to exit TracePro while the user interface is displayed and the user is in control.

EXAMPLES

You can download the example MATLAB functions and model from the Lambda Research website www.lambdares.com. The functions (*.m files) can be placed in you MATLAB documents folder. The default location is C:/Users/<UserName>/Documents/MATLAB.

You can place RGB_3LEDs.oml and RGB_3LEDs_Properties.txt in any folder on your computer. You will need to modify the definition of ExampleFolder in the MATLAB function com_openexample.

MATLAB Example 1

The following example illustrates how to start TracePro Expert from MATLAB, open a model, perform a ray trace and find the number of rays incident on a surface.

Function com_start_tracepro:

```
function [tracepro] = com_start_tracepro()
%COM_START_TRACEPRO: Start TracePro and display UI
%
tracepro = actxserver('TracePro.TraceProXP');

tracepro.AllowUserControl();
```

Function com_get_rays:

```
function [rays]=com_get_rays(tracepro,surface)
% COM_GET_RAYS: Demonstration function to run a raytrace in TracePro Run
% and return the number of incident rays for surface using COM.

% Execute a raytrace
tracepro.ExecuteSchemeString('(raytrace:source)');

% Get the number of incident rays
cmd = ['(raytrace:get-total-rays (entity:get-by-name "', surface, '") )'];
[ ~, return_str ] = tracepro.ExecuteSchemeString(cmd);

% The return value from the command is returned as a string and needs
% to be converted to a number.
rays = str2num(return_str);
```

Function com_openexample:

```
function [ExampleFolder]=com_openexample(tracepro)
%COM_OPENEXAMPLE Open RGB_3LEDs.oml and read properties
%
% TODO: Change ExampleFolder to the folder that you have downloaded
% RGB_3LEDs.oml and RGB_3LEDs_Properties.txt
ExampleFolder='C:/TracePro/Website/TracePro_Tools/Operational_Tools/MATLAB/';

% Use ExecuteSchemeString to open model
ExampleOml= 'RGB_3LEDs.oml';
cmd = ['(file:open "', ExampleFolder, ExampleOml, '")'];
tracepro.ExecuteSchemeString(cmd);

% Use ExecuteSchemeString to read properties
PropertiesForOml='RGB_3LEDs_Properties.txt';
cmd = ['(tools:database-import "', ExampleFolder, PropertiesForOml, '" #f)'];
tracepro.ExecuteSchemeString(cmd);
```

Enter the following commands in the MATLAB command window.

```
tracepro=com_start_tracepro();

com_openexample(tracepro);

nrays=com_get_rays(tracepro,'Observation');
```

nrays should be 30000.

MATLAB Example 2

The following example demonstrates how to save data from a TracePro Irradiance map and convert the data into a matrix for plotting in MATLAB.

Function read_irradiance:

This function will open a data file saved from TracePro and return the Irradiance data in a matrix. The function will read the data size from the file.

```
function [data]=read_irradiance(fname)
% READ_IRRADIANCE: Read an irrradiance map file and return the data in a
matrix.
%
% Read the irradinace map file
fid = fopen(fname, 'r');
% Find the grid size
grid_size = textscan(fid,'Grid Size is %d x %d', 1, 'headerlines', 25);
% Build a format string to get all the data
size = cell2mat(grid_size);
% Build a format string for the textscan command.
format = '';
for i = 1:size(1)
    format = [format, ' %f'];
end
% Convert the data by reading the file and building a cell array.
cells = textscan(fid, format, 'headerlines', 3);
% Convert to a matrix.
data = cell2mat(cells);
% Close the file.
fclose(fid);
```

Request and Plot Irradiance Data

This function establishes a conversation from MATLAB to TracePro. TracePro is assumed to have traced some rays and has a surface named "Observation". The function will select the surface in TracePro, display an Irradiance map and save the data to a file. The Irradiance data is returned using the *read_irradiance* function and plotted in a Contour plot.

```
function com_plot_irradiance(tracepro,tmp_folder,surface)
% COM_PLOT_IRRADIANCE: Demonstration function to retrieve irradiance
% map data for surface and create a contour plot using COM.

% Select surface
cmd = ['(edit:select (entity:get-by-name "', surface, '") )'];
tracepro.ExecuteSchemeString(cmd);

% Set the map type to irradiance
tracepro.ExecuteSchemeString('(analysis:irradiance-set-plot-quantities 0)');

% Update and display the map
tracepro.ExecuteSchemeString('(analysis:irradiance)');

% Save the irradiance map data to a file
tmp_nam = [tmp_folder, 'temp1.dat'];
cmd = ['(analysis:irradiance-save  "', tmp_nam, '")'];
tracepro.ExecuteSchemeString(cmd);

% Read the irradinace map file
mat_data = read_irradiance(tmp_nam);

% Flip the data to match TracePro
data = flipud(mat_data);

% Plot the data as a contour
contour (data, 'DisplayName', 'data', 'ZDataSource', 'data');
axis square; % set the axes aspect ratio
figure(gcf);
```

Example Output

The following function will execute this example.

```
function com_tracepro_demo()
%COM_TRACEPRO_DEMO: Demonstrate TracePro COM support
%    1) Start Tracepro if it is not running and make the UI visible
%    2) Open demo model in TracePro
%    3) Perform raytrace
%    4) Save irradiance map to file
%    5) Read the irradiance map file and display contour map in MATLAB
%

% Start TracePro if not running and make it visible
tracepro=com_start_tracepro();

% Open Example
ExampleFolder = com_openexample(tracepro);

% Run ray trace and get the number of rays incident on surface Observation
srf='Observation';
nrays=com_get_rays(tracepro,srf);

% If we have any incident rays create an irradiance contour plot
if nrays > 0
      com_plot_irradiance(tracepro,ExampleFolder,srf);
end
```

Type the following at the MATLAB prompt:

```
>> com_tracepro_demo();
```

The Irradiance plot and MATLAB contour plot are shown below and can be compared.