Altair

HyperWorks

Altair HyperMesh 2019 Tutorials

HM-8080: Calculate the Radius of an Arc

HM-8080: Calculate the Radius of an Arc

In this tutorial you will create a Tcl script that determines the radius of an arc.

Tools

The Tcl commands `if` and `expr` will be used to add logic and mathematical functions to the script. The command `hm_getentityvalue` is used to extract information from HyperMesh entities, based on data names.

Data names are generic references to the information that physically define an entity in the HyperMesh environment. An example of this is the x, y, and z coordinates that define a node location in three-dimensional space. The available data names for each entity can be found in the HyperMesh Reference Guide *Data Names* topic.

Data names are accessed using the `hm_getentityvalue` command. This command uses the data names available for an entity to return the particular value of interest. The command will return a value that is either a string or a numeric value, depending on the command syntax and the value stored in that particular data name field. The basic syntax of the command is:

`hm_getentityvalue` *`entity_type id data_name flag`*

where *`entity_type`* is the requested entity type (elements, loads, nodes, etc…), *`id`* is the entity ID, the *`data_name`* is the data field name of interest, and *`flag`* is either 0 or 1 depending on whether the command should return a numeric value (0) or a string (1).

To retrieve the x-component of a force with ID 12, the following command can be used:

```
set force_x [hm_getentityvalue loads 12 "comp1" 0]
```

Note that to assign the value from the command to a variable, the command is placed within square brackets.

Exercise

Create a Tcl script that determines the radius of a user selected arc. One point on the line and the center of the arc will need to be calculated.

1. Define the process.
2. Determine the data names to use to extract the node coordinates.
3. Create the Tcl script and add logic as necessary.
4. Test the script.

Step 1: Define the process.

The script should automate the following process:

- Prompt the user to select a line.
- Make sure the user has selected only one line.

- Determine the center of the arc by creating temporary nodes.
- Calculate the distance between one end of the arc and the center node using node coordinate data names.

Step 2: Determine the data names to use to extract the node coordinates.

The following table lists several relevant data names for nodes:

globalx	x coordinate in the global system
globaly	y coordinate in the global system
globalz	z coordinate in the global system

Steps 3-12: Create the Tcl script and add logic as necessary.

A Tcl script to perform this function might be similar to the following:

Step 3: Open a text file and save the file as HM8080.tcl.

Step 4: Allow the user to select the desired line which defines a circle or an arc and then add that line to a variable

The `*createmarkpanel` command is used to allow the user to graphically select the line which defines a circle or an arc from the HyperMesh interface and add it to the mark. The command below adds the line to mark 1. Once the line has been added to mark 1, the line id is assigned to a variable called `line_list`, using the TCL command set. Add the following 2 lines to the file `HM8080.tcl`:

```
*createmarkpanel lines 1 "Select line to find radius";
set line_list [hm_getmark lines 1];
```

Step 5: Begin an if loop which checks to see if the variable line_list has values. If it does, proceed with the macro.

Before continuing with the macro, we should check to make sure that the variable `line_list` has values in it. This is done by using an `if` loop. In the `if` loop below, we are checking to see if the variable `line_list` is empty. If the variable is empty, an error message is given to the user. Also, using the `elseif` statement in the if loop, we can check to see if more than one line is selected. If more than one line is selected, an error message is reported. If neither of those conditions are met, the macro proceeds under the else statement. Add the following line to the TCL file to define the if loop:

```
if {$line_list == ""} {
    hm_errormessage "No lines selected";
} elseif {[llength $line_list] != 1} {
    hm_errormessage "Only one line may be selected";
} else {
```

Step 6: Create 3 nodes on the line selected and then create a node at the circle center of the 3 nodes. Add those nodes to a variable.

Use the `*nodecreateonlines` command to create 3 nodes on the line which is in mark 1. This is done with the first command below. Then, use the `*createcenternode` to create a node at the center of a circle formed by the three nodes that were just created in the `*nodecreateonlines` command. These three nodes are referenced by using `-1`. `-2`, and `-3` which reference the last node created, the second to last node created, and the third to last node created. Then, the nodes are added to the nodes mark 1 using the `*createmark` command. Again, the nodes are referenced using `-1`, `-2`, `-3`, and `-4` to add the last 4 nodes created to the mark. Finally, the nodes in mark 1 are added to the variable `node_list`. Add the following 4 lines to the TCL script:

```
*nodecreateonlines lines 1 3 0 0;
*createcenternode -1 -2 -3;
*createmark nodes 1 -1 -2 -3 -4;
set node_list [hm_getmark nodes 1];
```

Step 7: Use the `lindex` command to get the node id of the first node in the list `node_list`. Then get the x, y, and z coordinates for the node.

Set a variable called `id` which contains the node id for the first node in the list `node_list`. The id for the first node is retrieved using the `lindex` command which takes the variable `node_list` and using the index 0, retrieves the first node id in the list. Then, using the variable `id` and the `hm_getentityvalue` command with the node data names x, y, and z, the x, y, and z coordinates for the node are set to the variables `x1`, `y1`, and `z1`. Add the following 4 lines to the TCL script:

```
set id [lindex $node_list 0];
set x1 [hm_getentityvalue nodes $id "x" 0];
set y1 [hm_getentityvalue nodes $id "y" 0];
set z1 [hm_getentityvalue nodes $id "z" 0];
```

Step 8: Use the `lindex` command to get the node id of the last node in the list `node_list`. Then get the x, y, and z coordinates for the node.

Set a variable called `id` which contains the node id for the last node in the list `node_list`. The id for the first node is retrieved using the `lindex` command which takes the variable `node_list` and using the index 3, retrieves the first node id in the list. Then, using the variable `id` and the `hm_getentityvalue` command with the node data names x, y, and z, the x, y, and z coordinates for the node are set to the variables x2, y2, and z2. Add the following 4 lines to the TCL script:

```
set id [lindex $node_list 3 ];
set x2 [hm_getentityvalue nodes $id "x" 0];
set y2 [hm_getentityvalue nodes $id "y" 0];
set z2 [hm_getentityvalue nodes $id "z" 0];
```

Step 9: Define three variables which are the x, y, and z distance between the two nodes defined in the last two steps.

Three variables are defined which are simply the x, y, and z distance between the two nodes defined in Steps 7 and 8. The component difference between each node is calculated using the coordinates defined in Steps 7 and 8 and the TCL command `expr`. Add the following 3 lines to the TCL script:

```
set dx [expr $x1 - $x2];
set dy [expr $y1 - $y2];
set dz [expr $z1 - $z2];
```

Step 10: Define a variable called radius which uses the variables `dx`, `dy`, and `dz` to calculate the radius of the line which is a circle or an arc.

Using the three variables which were defined in the previous step (`dx`, `dy`, and `dz`) the magnitude of the distance is calculated. This distance corresponds to the radius of the arc/circle which is defined by the line selected. To calculate the radius, the `expr` command is used. Add the following line to the TCL script:

```
set radius [expr sqrt(($dx*$dx) + ($dy*$dy) + ($dz*$dz))];
```

Step 11: Clear the nodes in the temporary node mark.

To clear all the nodes in the temporary node mark, use the `*nodecleartempmark` command. Add the following command to the TCL script:

```
*nodecleartempmark;
```

Step 12: Report to the user the radius of the selected line.

Using the `hm_usermessage` command, the value of the variable radius is reported to the user. Also, close the if loop which was started back in Step 5. Add the following two lines to the TCL script:

```
    hm_usermessage "Radius = $radius";
}
```

Step 13: Clear the lines and nodes mark.

Using the `hm_markclear` command, the nodes mark and the lines mark are cleared. Add the following two lines to the TCL script:

```
hm_markclear lines 1;
hm_markclear nodes 1;
```

Step 14: Test the script.

1. From the ***File*** menu, load the file, `radius-tcl.hm`.
2. From the menu bar, select ***View>Command Window*** to display the **Command Window** at the bottom of the screen.
3. Click and drag to open the **Command Window** from the bottom edge of the screen.
4. Use the `source` command to execute the script. For example:

   ```
   source HM8080.tcl
   ```

   It is often necessary to debug Tcl scripts using the command window. This allows you to run the Tcl script and easily review error messages, as well as print out debug information. Additional details can be found in the *Creating Tcl Scripts* and *Running Tcl Scripts* sections.
5. Select different lines to review the calculated radius.

Important things to notice.

- The `*entityhighlighting` and `hm_commandfilestate` commands are used to speed up the execution of the script. The `*entityhighlighting` command disables highlighting entities when the `*createmark` command is used. The `hm_commandfilestate` command controls if commands are written out to the command file. It is always important to "reset" these commands after a script is complete or before exiting due to an error.