Altair

HyperWorks

Altair HyperMesh 2019 Tutorials

HM-8090: Create an OptiStruct PSHELL property

altairhyperworks.com

HM-8090: Create an OptiStruct PSHELL property

In this tutorial you will create a Tcl script that:

- Prompts the user for a property name and thickness
- Creates the property collector
- Assigns the OptiStruct PSHELL card image to the property collector
- Assigns the thickness value to the property

Tools

The Tcl command `if` will be used to add logic to the script. The commands `*dictionaryload` and `*attributeupdatedouble` are used to assign information to the property collector.

Solver-specific data created from the HyperMesh template system is stored in card images. Each piece of data that defines a card image has a text string (data name) and a numeric attribute ID. An example is the Young's Modulus for a material.

Templates exist for each solver supported by HyperMesh and are located in sub-folders under the `<altair_home>\templates\feoutput` directory. These templates define every solver-specific attribute including data names, attribute IDs, card image formats, and the format of the data upon export. The `*defineattribute` command is used to define attribute data names and IDs in a template file.

In order to determine the commands required to create template-specific data, it is best to run through the process in HyperMesh and to review the commands that are written to the `command.cmf` file.

Property collectors can be created and edited using the **Model Browser**. Simply right click in the **Model Browser** and select ***Create > Property*** to create one. To change the name, color, or card image of a property collector, right click on the property name in the **Model Browser** and select ***Edit***

Exercise

Create a Tcl script to create a property collector and assign a thickness. This requires that the script prompt the user for a name and a thickness value.

1. Define the process.
2. Delete the existing `command.cmf` file. This file is located in either the start-in directory or the current working directory.
3. Perform the operations in HyperMesh that the script should run.
4. Extract the commands from the `command.cmf` file.
5. Create a Tcl script by converting the commands to Tcl format and modifying as necessary.
6. Test the script.

Step 1: Define the process.

The script should automate the following process:

- Prompt the user to enter a name and a thickness value.
- Make sure a property collector with the supplied name does not already exist.
- Create the new property collector.
- Assign the PSHELL card image to the property.
- Assign the thickness to the PSHELL card.

Step 2: Delete the existing `command.cmf` file.

The current `command.cmf` file is located in the current working directory. When first opening HyperMesh, the file is created in the directory HyperMesh is launched from. As soon as you begin working in HyperMesh, all executed commands are written to the `command.cmf` file. If the file already exists, the commands are appended to the file. Deleting the file allows HyperMesh to create a new file and allows the user to easily find the relevant commands.

Step 3: Perform the operations in HyperMesh.

Execute the full process within HyperMesh. Every command issued in HyperMesh appears in the order executed and is reflected in the `command.cmf` file.

1. If the **OptiStruct** user profile is not currently loaded, please load it at this time.
2. Right click in the **Model Browser** and select ***Create*** > ***Property***.
3. Leave **Type** set to **all** and in the **Name** field, enter a name for the property.
4. For **card image=**, select `PSHELL`.
5. Check the option for **Card edit property upon creation**.
6. Click ***Create***.
7. Activate the ***T*** field and enter a thickness value.
8. Click ***return***.

Step 4: Extract the commands from the `command.cmf` file.

1. Open the `command.cmf` file using any text editor.
2. Select and copy all lines in the file.

Step 5: Create a Tcl script by converting the commands to Tcl format and modify it as necessary.

1. Create a new Tcl file using any text editor.
2. Paste the copied commands from the `command.cmf` file inside the Tcl file.

3. Remove all () and , and replace them with spaces. Also place semi-colons (;) at the end of each line. The commands should look something like:

```
*collectorcreateonly properties "my_prop" "" 11;

*createmark properties 2 "my_prop";

*dictionaryload properties 2
"C:/Altair/hw12.0/templates/feoutput/optistruct/optistruct" "PSHELL";

*initializeattributes properties “my_prop”;

*attributeupdatedouble properties 1 95 1 1 0 0.25;
```

These commands can now be run to duplicate the creation of the PSHELL property. However, simply running these commands as-is is not very flexible. The property ID, name and values are all hard coded. The template file location in the `*dictionaryload` command is also hard coded. Finally, there are a lot of extra commands that set unnecessary attributes.

4. In the `*attributeupdatedouble` command, the ID of the property is hard coded. In order to make this flexible, you need to replace the hard coded ID with the ID of the new property collector: (Changes to the above commands are shown below in **bold** print).

```
*collectorcreateonly properties "my_prop" "" 11;

*createmark properties 2 -1

set prop_id [hm_getmark props 2];

*dictionaryload properties 2
"C:/Altair/hw12.0/templates/feoutput/optistruct/optistruct" "PSHELL";

*attributeupdatedouble properties $prop_id 95 1 1 0 0.25;
```

Supplying an ID of -1 to the `*createmark` command can be used to select the most recently created entity.

5. The template file path is also hard coded. You can make this flexible using the `hm_info` command:

```
*collectorcreateonly properties "my_prop" "" 11;

*createmark properties 2 "my_prop";

set prop_id [hm_getmark props 2];

*dictionaryload properties 2 "[hm_info -appinfo SPECIFIEDPATH
TEMPLATES_DIR]/feoutput/optistruct/optistruct" "PSHELL";

*attributeupdatedouble properties $prop_id 95 1 1 0 0.25;
```

The user also needs to be prompted to enter a property name and thickness value. You can then substitute those variables in the relevant commands:

```
set prop_name [hm_getstring "Name="];

set prop_thick [hm_getfloat "Thickness="];

*collectorcreateonly properties "$prop_name" "" 11;

*createmark properties 2 "$prop_name";
```

```
set prop_id [hm_getmark props 2];
*dictionaryload properties 2 "[hm_info -appinfo SPECIFIEDPATH
TEMPLATES_DIR]/feoutput/optistruct/optistruct" "PSHELL";
*attributeupdatedouble properties $prop_id 95 1 1 0 $prop_thick;
```

6. Finally, You need to add logic to test in order to make sure that the property name and thickness values are valid:

```
set prop_name [hm_getstring "Name="];
if {$prop_name == ""} {
    hm_errormessage "No name specified.";
    return;
} elseif {[hm_entityinfo exist properties $prop_name -byname] == 1} {
    hm_errormessage "Property already exists.";
    return;
}
set prop_thick [hm_getfloat "Thickness="];
if {$prop_thick == "" || $prop_thick <= 0.0} {
    hm_errormessage "Invalid thickness value specified.";
    return;
}
*collectorcreateonly properties "$prop_name" "" 11;
*createmark properties 2 "$prop_name";
set prop_id [hm_getmark props 2];
*dictionaryload properties 2 "[hm_info -appinfo SPECIFIEDPATH
TEMPLATES_DIR]/feoutput/optistruct/optistruct" "PSHELL";
*attributeupdatedouble properties $prop_id 95 1 1 0 $prop_thick;
```

Step 6: Test the script.

1. From the menu bar, select ***View > Command Window*** to display the **Command Window** at the bottom of the screen.
2. Click and drag to open the **Command Window** from the top or bottom edge of the screen.
3. Use the `source` command to execute the script. For example:

```
source filename.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.