Altair HyperMesh 2019 Tutorials

HM-8020: Create a Utility Menu Macro From a Command File

HM-8020: Create a Utility Menu Macro From a Command File

In this tutorial you will:

- Determine the commands to save the current HyperMesh model
- Create a **Utility Menu** macro to execute the command
- Create a new button on the **User** page of the **Utility Menu** to run the macro

Tools

In order to execute command file commands or Tcl scripts from a button on any of the HyperMesh **Utility Menu** pages, a **Utility Menu** macro must first be defined. A **Utility Menu** macro contains valid command file or templex commands that execute the appropriate operations, and is defined using the `*beginmacro` and `*endmacro` commands. Macros may accept data passed to them using the arguments `$1`, `$2`, etc. Each argument specifies where the values should be substituted. These macros are defined within the `.mac` files, including the `userpage.mac` file.

The following skeleton code shows the format of a **Utility Menu** macro:

```
*beginmacro(macroname)
    command statements go here
*endmacro()
```

Utility Menu macros consist of HyperMesh Tcl modify commands.

Exercise

Create a **Utility Menu** macro from a command file that saves the model and add a button on the **User** page that will launch the macro:

1. Define the task.
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. Add the commands to the `userpage.mac` file.
6. Modify as necessary and add macro wrapper commands `*beginmacro` and `*endmacro`.
7. Add macro button using `*createbutton` that calls the new macro defined in Step 6.
8. Reload the current `.mac` file into HyperMesh to load the modified `userpage.mac`.
9. Test the macro.

Step 1: Define the task.

The first step in creating a macro is to define the process you want to automate and recognize the individual tasks to reach the desired conclusion. Here, you want to create a one-button macro to automate saving the current HyperMesh model to a file named `temp.hm`. The actions necessary to complete this task are:

- From the menu bar, select *File > Save as > Model…*.

- Use the file browser to locate a directory and enter the name for the filename.

- Click *Save*.

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. From the menu bar, select *File > Save as > Model…*.

2. Using the file browser, locate a directory to save the temporary file with the name `temp.hm`.

    Remember this is just a temporary file and will be overwritten each time the macro is executed.

3. Click *Save*.

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

1. Open the `command.cmf` file using any text editor.

2. Locate the `*writefile` command at or near the end of the `command.cmf` file.

    This is the command that writes the model file.

3. Select and copy this line.

Step 5: Add the commands to the `userpage.mac` file.

1. Open the `userpage.mac` file using any text editor.

2. Paste the `*writefile` command copied from the `command.cmf` file inside the `userpage.mac` file.

Step 6: Modify as necessary and add Utility Menu macro wrapper commands.

1. Remove the path in the `*writefile` command so that it looks like:

    ```
    *writefile("temp.hm",0)
    ```

2. Enclose the commands from Step 5 between the wrapper commands `*beginmacro` and `*endmacro`. In the `*beginmacro` command, name the macro `macroSave`.

    ```
    *beginmacro(macroSave)
        *writefile("temp.hm",0)
    *endmacro()
    ```

    The macro name `macroSave` will be used to connect the button with the macro via the `macroName` field in the `*createbutton` command.

3. Add the command `*answer(yes)` after the `*writefile` command.

    The command `*answer(yes)` automatically answers "yes" if prompted to overwrite the file in the event `temp.hm` already exists.

4. Save the `userpage.mac` file.

Step 7: Add the macro button.

Create a button on the **User** page to execute the macro.

1. Create a new button in the `userpage.mac` file.

    ```
    *createbutton(5,"Save File",20,0,10,GREEN,"Save file","macroSave")
    ```

    This creates a button on page 5 (**User** page), names it, places it in the 20th row, starts it at column 0, sets the width at 10 columns, applies to it the color green, provides a help string and references the macro macroSave defined in Step 6.

2. Save the `userpage.mac` file.

Step 8: Reload the current `.mac` file into HyperMesh to load the modified `userpage.mac`.

To reload the current macro menu `.mac` file while HyperMesh is open, select ***Preferences > Menu Config*** from the menu bar and click on ***retrieve*** next to **macro file**. Make sure to load the proper `.mac` file from the `hm\scripts\<profile name>` directory based on the current user profile, or load the default `hm.mac` in the `hm\bin\<platform>` directory if no user profile is loaded.

Step 9: Test the macro.

1. Click the ***User*** button on the **Utility Menu**.

    The new button labeled **Save File** should be on the **User** page.

2. Click this button to automatically save your file.

   The file is saved to the directory specified in the `*writefile` command. In this case no directory is specified so HyperMesh saves the file to the start-up or current working directory. It will always save with the name specified in the macro (in this case, `temp.hm`).