Altair

HyperWorks

Altair MotionView 2019 Tutorials

MV-1040: Model Building using Tcl

altairhyperworks.com

MV-1040: Model Building using Tcl

In this tutorial you will:

- Learn the advantages of using Tcl programming to save time and effort in MBD model building with MotionView
- Work with the ***HyperWorks Desktop*** – Tcl interface
- Build a simple model using Tcl commands

About Tcl

Tool Command Language or Tcl (typically pronounced as "tickle" or "tee-see-ell") is a scripting language that is commonly used for quick prototyping, scripted applications, GUIs, and testing.

More About Tcl/Tk

- Tcl has a simple and programmable syntax.
- Tcl is open source.
- ***HyperWorks*** has an inbuilt Tcl interpreter which has libraries to help end users.
- Tcl can be used as a standalone or embedded in applications like HyperWorks Desktop (including ***MotionView***).
- Unlike C which is a complied language, TCL is interpreted. Tcl programs are simple scripts consisting of Tcl commands that are processed by a Tcl interpreter.
- Tcl is extensible. New Tcl commands can be implemented using C language and integrated easily. Many people have written extension packages for common tasks and are freely available on the internet.
- Engineering teams use different resources and application. Tcl can be used to glue those resources together. This greatly helps in automating the work flow.
- Tk is a Graphical User Interface toolkit that makes it possible to quickly create powerful GUIs.
- Tcl/Tk is highly portable, and runs on different flavors of UNIX, windows, Macintosh and more. This proves useful to those who work on various platforms.

Tcl with MotionView

- When building huge multibody models in MotionView, you will come across cases where the same steps are repeated multiple times. Such steps, or the set of steps, can be automated suing Tcl in order to save time and effort wasted in performing repetitive and time consuming tasks.
- Like all of the HyperWorks Desktop applications, MotionView has Tcl command layers which help in accessing the various functionalities of the product and utilizing them to write scripts to automate processes.
- The Tcl scripts can be called by Tk applications or tied in to a process manager.
- Tcl scripts can be registered in a preference file and be made a part of product with Menu shortcuts.

The ***HyperWorks Desktop*** handles and the ***HyperWorks*** database consists of a hierarchy of objects, the root of which is the ***hwi*** object which is automatically created. The ***hwi*** provides access to the ***hwiSession*** object and a few high level utilities. Currently, ***HyperWorks*** supports only one session per run. The session object can be retrieved by issuing the following command at the Tcl command prompt:

- `(System32) 1 % hwi GetSessionHandle sess1`

Once the session handle is retrieved, it can be used to access all objects in the ***HyperWorks*** database as shown below:

- `(System32) 2 % sess1 GetProjectHandle proj1`
- `(System32) 3 % proj1 GetPageHandle page1 1`

Windows are retrieved as shown below. Windows are assigned a client type, which can be modified.

- `(System32) 4 % page1 GetWindowHandle win1 1`
- `(System32) 5 % win1 SetClientType "Animation"`
- `(System32) 6 % win1 GetClientHandle post1`

A window's client type cannot be changed after the client handle has been retrieved. The client handle must be released and retrieved again if the window's client type is changed.

Every ***HyperWorks*** command object supports the following utility commands:

- **ListMethods**: Displays the method commands which can be performed on an object.
- **ListHandles**: Lists the names of all command objects of the same type.
- **ReleaseHandle**: Releases the command object.

The top level ***hwi*** command object supports the following utility commands:

- **ListAllHandles**: Displays all command objects currently in use.

Exercise: Model Building using Tcl

Step 1: Building a Simple Pendulum through Tcl commands.

In this exercise you will write a simple Tcl script to build a simple pendulum model.

Note Putting a '#' character in the beginning of any line makes it a comment and that line is not evaluated. In addition, all HyperWorks Tcl commands are case sensitive.

The structure of every Tcl script created for HyperWorks Desktop products should follow the following structure:

```
hwi OpenStack

Obtain All necessary handles

Perform some function

Release All obtained handles individually

hwi CloseStack
```

1. Open a new ***MotionView*** session.
2. Go to ***View*** menu, and click on ***Command Window***.
3. A ***TkCon*** window opens up and displays the version of Tcl and Tk installed with Hyperworks.
4. In the Command Prompt type:

   ```
   hwi GetSessionHandle sess
   ```

   The prompt prints sess as the command output if the command is successful. This command assigns the Session Handle to the variable "`sess`".
5. To view all the option/commands available with the ***hwi*** class type in `hwi ListMethods` at the command prompt. This will list all the options available under ***hwi***.
6. Now, type:

   ```
   sess GetProjectHandle proj
   ```

   This command will assign the Project handle to the variable "`proj`".
7. The next step is to obtain the Page handle, the command for it is:

   ```
   proj GetPageHandle page 1
   ```

   The variable "`page`" now points to the page handle of the first page of the session.

Note Please refer the "*Programming with Tcl/Tk Commands*" online help under the *"HyperView, MotionView and HyperGraph" Reference Guide* for the explanation on the syntax of these commands.

8. To get the control of the window we need to get the window handle the command for that is:

   ```
   page GetWindowHandle win 1
   ```

   This assigns the window handle of the first window to the variable "`win`".

9. Now to get the client handle type in:

   ```
   win GetClientHandle mc
   ```

Note A HyperWorks session has multiple clients (HyperView, MotionView, HyperGraph 2D, etc). When MotionView is invoked, the default client is MotionView. The `GetClientHandle` command gets you the access to the MotionView model object through the Client Handle.

10. To be able to set different views and fit the model in the graphics window the view control handle is required, the command to get view control handle is:

    ```
    win GetViewControlHandle vch
    ```

11. To start with a new blank model we will run the command:

    ```
    mc CreateBlankModel
    ```

12. To obtain the handle of the model just created in the previous step type in the command:

    ```
    mc GetModelHandle m
    ```

Note Once the model handle is obtained we can now start creating entities using the `InterpretEntity` and `InterpretSet` statements.

To build a simple pendulum we will be using 2 points, 1 body, 3 graphic entities, and 1 revolute joint.

The syntax for the `InterpretEntity` command is given below:

```
modelHandle InterpretEntity EntityHandle Entitytype
EntityVariableName EntityLabel <Parameters>
```

Where:

`EntityHandle` - The handle for the entity.

`Entitytype` - The type of entity to create (Point, Body, Graphic, etc.).

`EntityVariableName` – The variable name for the entity to view in MotionView.

`EntityLabel` – The label for entity to view in MotionView.

`Parameters` – The parameters which are required to create the respective entity (for example, CM point for Body).

13. To start with Add a point for the pendulum pivot with variable p_0 and label Pivot with a command:

    ```
    m InterpretEntity p Point p_0 "\"Pivot\""
    ```

14. Now to set the properties of the point just created, the command is:

    ```
    m InterpretSet SetPoint p_0 0.0 0.0 0.0
    ```

15. p is the Point handle for Tcl and is released with `p ReleaseHandle` command:

```
p ReleaseHandle
```

16. To create a point for the location of the pendulum mass and set the property for it, the set of commands are:

```
m InterpretEntity p Point p_1 "\"Mass\""

m InterpretSet SetPoint p_1 p_0.x+100 p_0.y p_0.z

p ReleaseHandle
```

17. Add the pendulum body and set its mass and inertia properties type in the following commands:

```
m InterpretEntity b Body b_0 "\"Pendulum\"" p_1

m InterpretSet SetBodyInertia b_0 0.5 100 100 100

m InterpretSet SetOrientation b_0.cm TWOAXES ZX

b ReleaseHandle
```

18. To add the graphics on the body for visualization the three graphic entities are added using the commands:

```
m InterpretEntity g Graphic gra_0 "\"Graphic_Pivot\"" CYLINDER B_Ground
p_0 V_Global_Y 1 1 10 -5 CAPBOTH

g ReleaseHandle

m InterpretEntity g Graphic gra_1 "\"Graphic_Pendulum_Cylinder\""
CYLINDER b_0 p_0  p_1 1 CAPBOTH

g ReleaseHandle

m InterpretEntity g Graphic gra_2 "\"GraphicMass_Cylinder\"" CYLINDER
b_0 p_1 V_Global_Y 5 5 3 -2 CAPBOTH

g ReleaseHandle
```

19. The pendulum will need to be connected to the ground with a revolute:

```
m InterpretEntity j RevJoint j_0 "\"Joint_Pivot_Rev\"" B_Ground b_0 p_0
V_Global_Y

j ReleaseHandle
```

20. After adding any entity to the model the database has to be updated by using the evaluate command:

```
m Evaluate
```

21. To the fit model in the graphics window:

```
vch Fit
```

22. The model is ready to be run. Go to the **Run** panel, specify a name for the result file and click on the ***Run*** button to run the model using MotionSolve. Use the ***Animate*** button to view the animation.

23. The handles obtained through the commands in the above steps now have to be released using the `ReleaseHandle` command. Type in the following:

```
m ReleaseHandle;

mc ReleaseHandle;

win ReleaseHandle;

page ReleaseHandle;

proj ReleaseHandle;

sess ReleaseHandle;
```

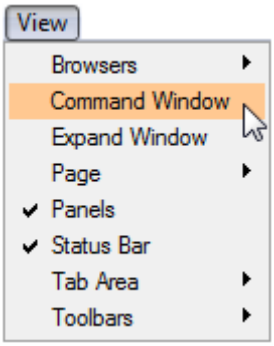
24. In a text editor paste all the above Tcl commands and save the file as `pendulum.tcl` in the working directory. This file can be "sourced" and the model can be built in one step. The complete script is given below for your reference (please see the bottom of the tutorial).

Note You can also use the file `pendulum.tcl` located in the `automation` folder.

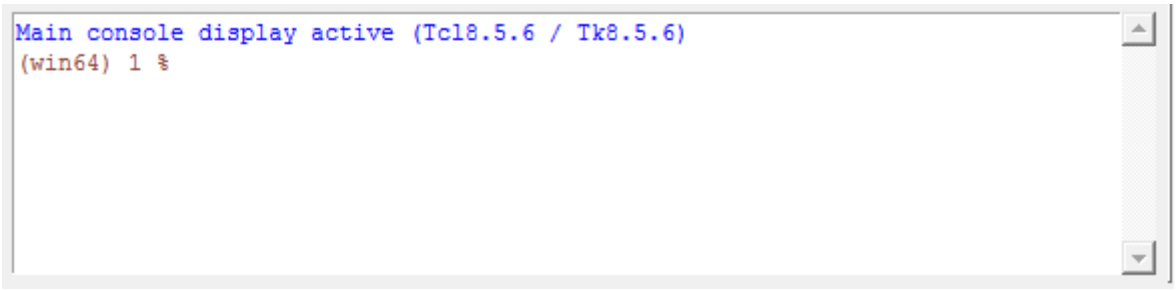
Copy this file to your `<working directory>`.

Step 2: Sourcing the Tcl file.

1. Start a new ***MotionView*** session.
2. Go to the ***View*** menu from the menu bar.



3. Click on ***Command Window***. A ***TkCon*** window opens up at the bottom of the screen.



4. Change the directory to current working directory by using the `cd` command.

5. To invoke a Tcl script, use the command `source pendulum.tcl`, where `pendulum.tcl` is the file that you saved in the previous step.
6. This will build the complete model by sequentially running the commands in the file line by line.

Step 3: Registering the Tcl in the preference file.

1. Open a text editor with a new file.
2. Write the following statements:

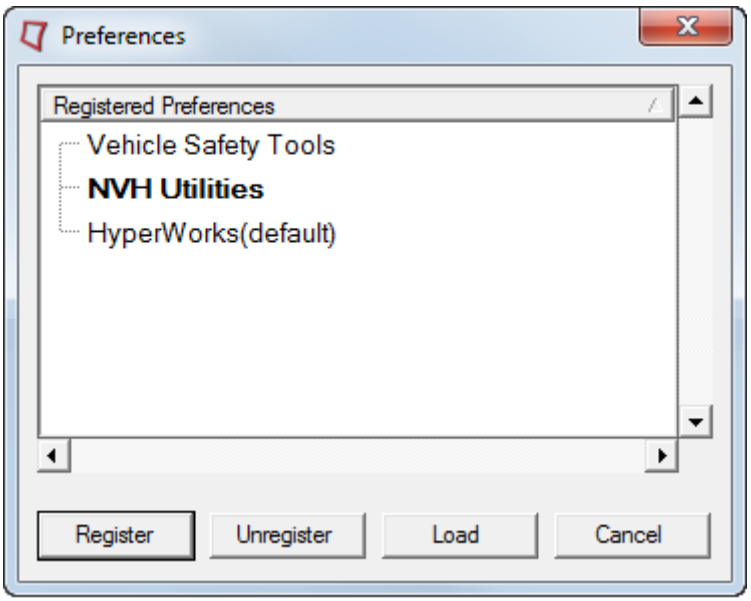
```
*Id("HyperWorks vXX.X")

*BeginModelDefaults()

*BeginMenu(scripts, "My Scripts")

*MenuItem(flexprep, "Build Simple Pendulum", Tcl, "<working
directory>/pendulum.tcl")

*EndMenu()

*EndModelDefaults()
```

Note Please refer to online help if you need to know more about the syntax.

Replace <working_directory> with the actual path on your machine to the working directory, using forward slashes for the path, if necessary.

3. Save the file as `mypreference.mvw` in the `<working directory>`.
4. Start a new ***MotionView*** session.
5. From ***File*** menu select ***Load > Preference File***.

   The ***Preferences*** dialog is displayed.



6. Click ***Register***.

7. Select the file `mypreference.mvw` you created.

   A new registered preference is added to the list.

8. Select the new preference and click ***Load***.

9. Close the session and start a new one.

10. You should see a new menu ***My Scripts*** in the modeling client. This should be available every time you open the MotionView session as long you have the preference file registered.

11. Click on ***My Scripts -> Build Simple Pendulum*** menu and run the script.

The complete script is given below for your reference:

```
## Macro to Build a Simple Model in MotionView ##

## Requesting for the Handles required to Use MotionView ##
hwi OpenStack
hwi GetSessionHandle sess
sess GetProjectHandle proj
proj GetPageHandle page [proj GetActivePage]
page GetWindowHandle win [page GetActiveWindow]
win GetClientHandle mc
win GetViewControlHandle vch
mc CreateBlankModel
mc GetModelHandle m

## Building the Model using the InterpretEntity statements ##
      m InterpretEntity p Point p_0 "\"Pivot\""
      m InterpretSet SetPoint p_0 0.0 0.0 0.0
      p ReleaseHandle
      m InterpretEntity p Point p_1 "\"Mass\""
      m InterpretSet SetPoint p_1 p_0.x+100 p_0.y p_0.z
      p ReleaseHandle

      m InterpretEntity b Body b_0 "\"Pendulum\"" p_1
      m InterpretSet SetBodyInertia b_0 0.5 100 100 100
      m InterpretSet SetOrientation b_0.cm TWOAXES ZX
      b ReleaseHandle
```

```
## Adding graphics to the pendulum and the Ground to improve result
visualization

      m InterpretEntity g Graphic gra_0 "\"Graphic_Pivot\"" CYLINDER
      B_Ground p_0 V_Global_Y 1 1 10 -5 CAPBOTH
      g ReleaseHandle
      m InterpretEntity g Graphic gra_1 "\"Graphic_Pendulum_Cylinder\""
      CYLINDER b_0 p_0  p_1 1 CAPBOTH
      g ReleaseHandle
      m InterpretEntity g Graphic gra_2 "\"GraphicMass_Cylinder\"" CYLINDER
      b_0 p_1 V_Global_Y 5 5 3 -2 CAPBOTH
      g ReleaseHandle


## Adding the Revolute joint between the Ground and the pendulum body

      m InterpretEntity j RevJoint j_0 "\"Joint_Pivot_Rev\"" B_Ground b_0
      p_0 V_Global_Y
      j ReleaseHandle


m Evaluate

vch Fit


after 1000


## Running the Model ##

mc ExportModel simple_pendu.xml

mc RunSolverScript simple_pendu.xml


## Releasing All the Handles

m ReleaseHandle;

mc ReleaseHandle;

win ReleaseHandle;

page ReleaseHandle;

proj ReleaseHandle;

sess ReleaseHandle;

hwi CloseStack;


## End of Script
```