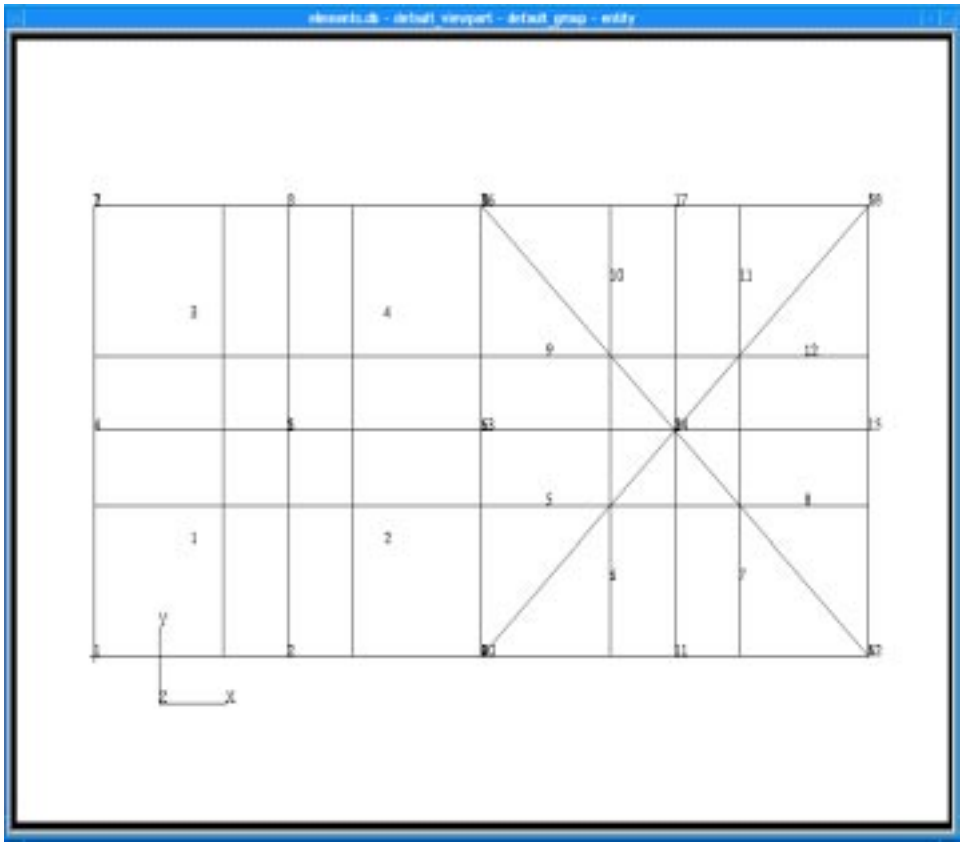
EXERCISE 7

Put Elements In Groups



Objectives:

- Write a function to add all elements of the same shape into a new group.

Put Elements in Groups

Exercise Description:

This exercise, `put_element_in_group ( element_type, group_name)`, creates a group named *`group_name`*, with all elements of shape *`element_type`*. The function needs to check that a database is open. Then as the function fills each array, check that there are no problems with the reading from the database. Appropriate error messages should be displayed, and the function terminated if it fails. When the function creates and fills the group, send a message to the command window indicating what was created and how many elements it contains.

Files:

All the files that used in this exercise are listed below. Each list includes the file, where it originated, and a summary of information of how it relates to the exercise.

File	Supplied/Created	Description
exercise_07.template	Supplied	A template to be edited with the proper PCL syntax.
put_element_in_group.pcl	Created	Must be created from the exercise_07.template. This file is the corrected template renamed.

Exercise Procedure:

1. Enter the vi editor and edit the PCL function in a file called `exercise_07.template`. Subsititute the approporiate PCL syntax for the lines which contain:

```
********** # **********
```

There are quick reference pages for vi and UNIX in the appendix of this workbook for your reference.

After filling in the blanks, save the function with the file name **`put_element_in_group.pcl`**.

2. Compile the function.

Start the compiler by typing p3pclcomp.

Enter the command

```
!!input put_element_in_group.pcl
```

into the command line.

All the error messages and diagnostics will be written below that.

3. Start MSC/PATRAN.

4. Open up a new database called **elements.db**

File/New...

elements.db

OK

5. Click OK in the New Model Preference form.

6. Create a surface.

◆ **Geometry**

Action:	**Create**
Object:	**Surface**
Method:	**XYZ**

Apply

7. Create another surface next to that one.

Action:	**Transform**
Object:	**Surface**
Method:	**Translate**

◆ **Cartesian in Refer. CF**

Refer. Coordinate Frame	**Coord 0**
Translation Vector	**<1 0 0>**
Surface List	**Surface 1**

Apply

8. Mesh surface 1 with quad4 elements..

◆ **Finite Elements**

Action:	**Create**

Object:	**Mesh**
Type:	**Surface**
Global Edge Length	**0.5**
Element Topology	**Quad4**

◆ **IsoMesh**

Surface List	**Surface 1**

Apply

9. Now add the mesh for the second surface with Tria Elements

Action:	**Create**
Object:	**Mesh**
Type:	**Surface**
Global Edge Length	**0.5**
Element Topology	**Tria3**

◆ **IsoMesh**

Surface List	**Surface 2**

Apply

10. At the command line of MSC/PATRAN type the following command:

```
!!input put_element_in_group.pcl
```

The file should compile with no problems once again.

11. Execute the command that will group the Tria Elements together by typing:

```
put_element_in_group ( “tria” , “group_of_trias”)
```

You should get the following output:

```
8 tria elements added to new group: group_of_trias
```

12. Do the same thing for the group of Quad elements that were created:

put_element_in_group (“quad” , “group_of_quads”)

```
4 quad elements added to new group: group_of_quads
```

13. Quit PATRAN.

Sample Solution:

```
FUNCTION put_element_in_group( element_type, group_name )

/* Purpose: Place all elements of a specified shape into
 *          a user-specified group
 *
 * Input: element_type S element type
 *        group_name  S  group name to put elements in
 *
 * Output: none
 *
 * Side Effects;
 *           A new group is created in the database with all elements
 *           of the specified shape.
 *
 */

STRING      element_type[], group_name[]
INTEGER     numels, i, status, num_found = 0
INTEGER     elem_ids(VIRTUAL), topo_codes(VIRTUAL)
INTEGER     shape(VIRTUAL), nnod(VIRTUAL)
STRING      type[5](8) = “Point”, “Bar”, “Tria”, “Quad”, @
            “Tet”, “ “, “Wedge”, “Hex”

/*
* Check to see that a database is open
*/
num_found = 0
IF(db_is_open() ) THEN
/*
* Get number of elements and allocate arrays
*/

status = ************ 1 *****************
IF( status != 0 ) THEN
           msg_to_form( status, 4, 13000000, 1, 1., “” )
           RETURN status
END IF
IF( numels == 0 ) THEN
           ui_write( “No elements exist in the database.”)
           RETURN 1
END IF
sys_allocate_array( elem_ids, 1, numels )
sys_allocate_array( topo_codes, 1, numels )
sys_allocate_array( shape, 1, numels )
sys_allocate_array( nnod, 1, numels )

/*
* Extract from database eids, topology, shape codes
* and nnodes
*/

status = ************** 2 ************** /* elem ids */
IF( status != 0 ) THEN
           msg_to_form( status, 4, 13000000, 1, 1., “” )
           RETURN status
END IF
```

```
status = ************** 3 ****************** /* topology codes */
IF( status != 0 ) THEN
           msg_to_form( status, 4, 13000000, 1, 1., "" )
           RETURN status
END IF
status = *************** 4 ******************* /* topology data */
IF( status != 0 ) THEN
           msg_to_form( status, 4, 13000000, 1, 1., "" )
           RETURN status
END IF

/*
* Now loop thru elements. If shape code is of desired type,
* add the element to the group.
*/

status = ************** 5 ****************** /* Create the group */
IF( status != 0 ) THEN
           msg_to_form( status, 4, 11000000, 1, 1., "" )
           RETURN status
END IF
FOR (i = 1 TO numels )
           IF (!em_proceed_normal())THEN RETURN
           IF( type(shape(i)) == element_type ) THEN
               num_found += 1
               ***************** 6 **********************
           END IF
END FOR

/* Release virtual arrays */

        sys_free_array( elem_ids )
        sys_free_array( topo_codes )
        sys_free_array( shape )
        sys_free_array( nnod )
       ui_writef( "I8,' ', A5,' elements added to new group: ',A32", @
              num_found, element_type, group_name )
ELSE
  ui_write("A database must be open to create a group.")
END IF

END FUNCTION
```

```
*1* db_count_elems( numels )
*2* db_get_elem_ids( numels, elem_ids )
*3* db_get_elem_etop( numels, elem_ids, topo_codes )
*4* db_get_elem_topology_data( numels, topo_codes, @
                                shape, nnod)
*5* ga_group_create( group_name )
*6* ga_group_entity_add( group_name, @
                          "E "//str_from_integer( elem_ids(i) ))
```