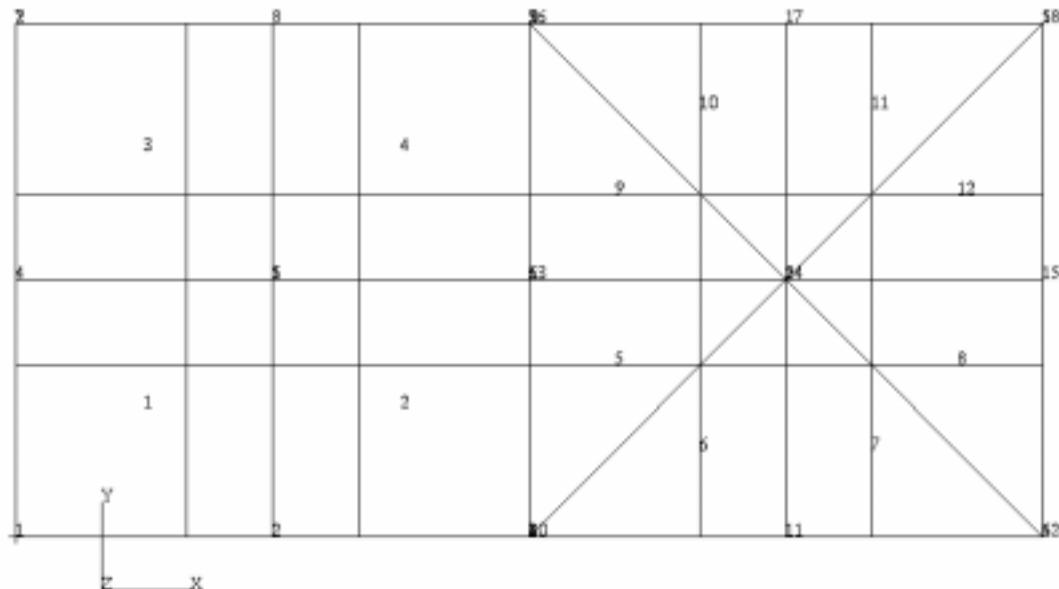

EXERCISE 12

Group Elements



Objectives:

- Write a class that creates a form with a switch, databox, apply and cancel buttons. The callback for the apply button will get the user defined element shape and group name and call put_element_in_group, created in Exercise 7.

Exercise Description:

Create a form that handles the user interface for the addition of elements of a given shape to groups using the **put_element_in_group** function created in **Exercise 7**. The function requires a switch that contains the options for the shape of element to be added to the new group. Finally, the apply button will fetch the information supplied by the user and call the **put_element_in_group** function to create the group.

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_12.template	Supplied	A template file that you may use to fill in the missing PCL calls that should be added to the function.
group_elementspcl	Created	This file should be created after all the blanks are filled in the exercise_12.template file.
group_elements.cpp	Created	This file should be created when the PCL file is run through the C Pre-Processor.
p3epilogpcl	Created	A file that needs to be created in order to input the files needed for the exercise upon startup of PATRAN.

Exercise Procedure:

1. Edit the PCL function in the file **exercise_12.template**. Replace the blanks with the appropriate PCL expressions. Rename the file **group_elementspcl** when you are done editing the file.

The form you create should look like the one shown below. You should use the pre-defined variables for the widget size and spacing to make the form presentable. These variables are accessed by placing

```
#include appforms.p
```

at the top of your PCL function and running the file through the C pre-processor prior to compilation.



You may also use the template file your instructor will provide for you to add the missing PCL code.

If no errors are found in the PCL code then run the C Pre-Processor.

2. To run the C Pre-processor on the PCL function that you just created type in one of the following commands:

For Sun Solaris:

```
% /usr/ccs/lib/cpp -I/patran/patran3/customization  
group_elementspcl group_elements.cpp
```

For SUNOS:

```
% /usr/lib/cpp -I/patran/patran3/customization  
group_elementspcl group_elements.cpp
```

For HP-UX(Earlier than version 10.2)

```
%/lib/cpp -I/patran/patran3/customization
group_elementspcl group_elements.cpp
```

For HP-UX(Version 10.2 or later):

```
%/usr/ccs/lbin/cpp -I/patran/patran3/customization
group_elementspcl group_elements.cpp
```

Note: The command for SUNOS should also work for all Silicon Graphics machines.

The -I switch tells cpp to search in /patran/patran3/customization directory for the include files.

3. Type in the command p3pclcomp into your x-term window.

At the compiler's prompt type in the following command:

```
!!input group_elements.cpp
```

If there are no errors in the code then you will need to create a p3epilogpcl file.

4. Create or edit your p3epilogpcl file. The file should now contain the following commands:

```
!!input put_element_in_grouppcl
!!input group_elements.cpp
!!input trainingpcl
training.init()
```

5. Use the vi or jot editors to edit the **trainingpcl** file in order to “unghost” the exercise 12 menu option.

To unghost the rest of the exercise options you will need to comment out several of the lines in the code. In order to comment out these lines you will need to enclose each line with a:

“/*” at the beginning of each line and,

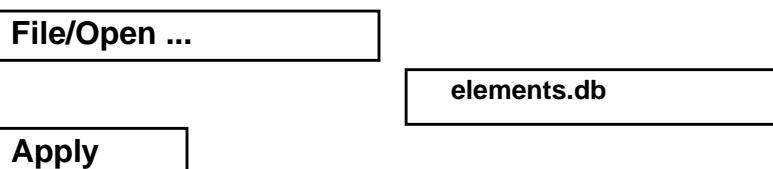
“*/” at the end of each line.

The lines that you will need to comment out are similar to the one shown below and the rest should all be grouped together.

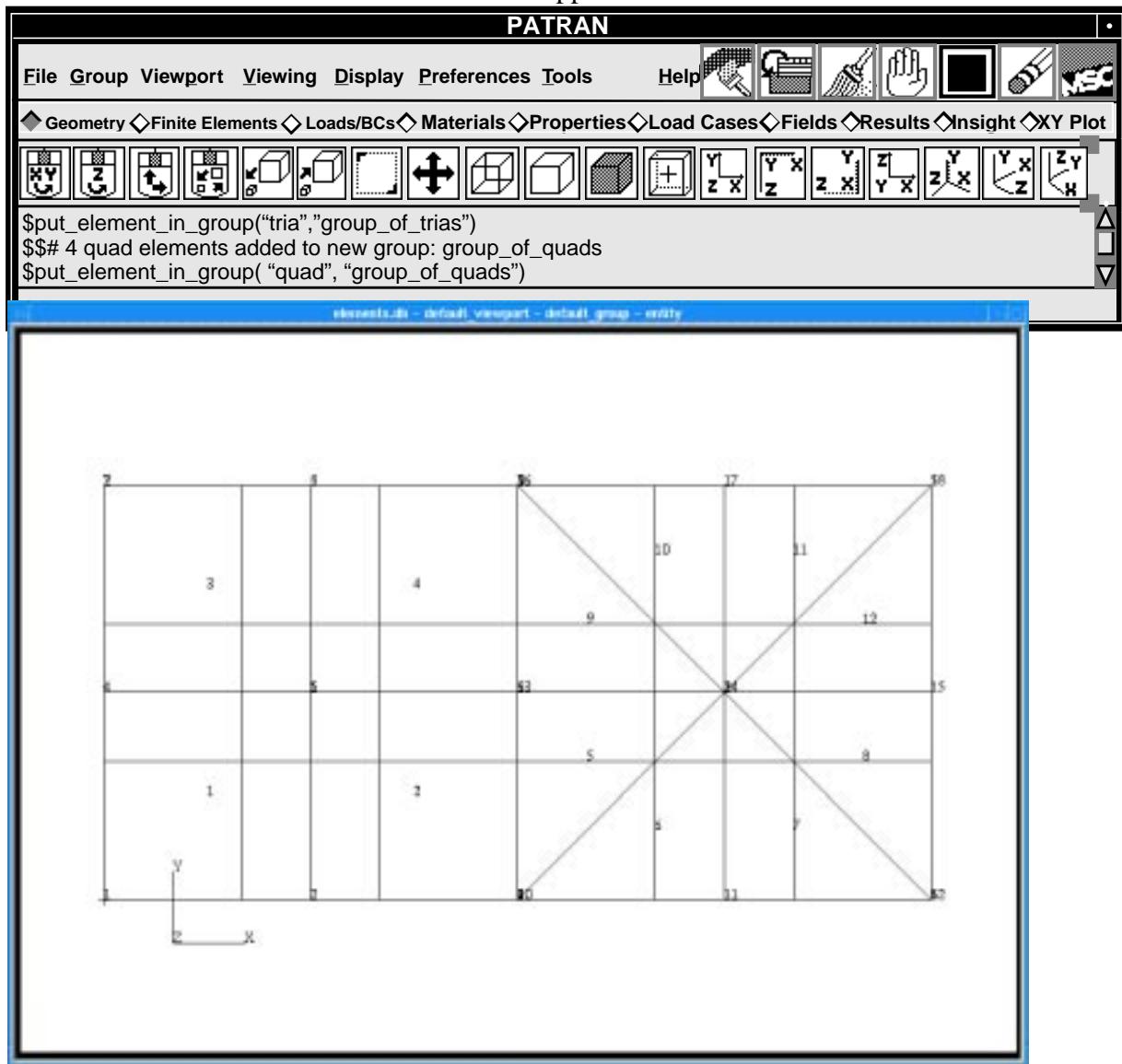
```
/*ui_wid_set(item12, "ENABLE", FALSE)*/
```

Note: Be sure to comment out items 13 thru 17 as well

6. Start PATRAN once again with all the changes made to those files. Pull down the training menu in PATRAN, you should notice that all of the menu options are able to be selected. If you cannot select the exercise 12 option then you need to make sure that those lines were commented out in the **training.pcl** code and then re-execute the function **training.init()**
7. Open the database **elements.db**.



Your model should appear as shown below.



8. Test the form and function that you built..

**Training/Exercise12**

◆ Quad

Group Name

my_Quads

Apply

MSC/PATRAN responds:

4 Quad elements added to new group: my_QUADs

Try the function with the group of trias as well to see if both functions work.

Sample Solution:

```
/*$$ Use of PCL in creating customized forms/widgets
*
* Purpose:
*   Create a form with a switch, a databox, apply and cancel buttons.
*   The callback for the apply button will get the element shape and
*   the group name and call put_element_in_group which was written
*   for an earlier exercise.
*
* Input:
*   <None>
*
* Output:
*   <None>
*
* Log:
*
* Notes:
*   Things to learn from this exercise
*   Using the cpp pre-processor (appforms.p)
*   Positioning widgets with parameters
*   Databox, switch widgets
*   Callback and get widget values
*   Default button
*
*/
#include "appforms.p"

CLASS group_elements

/* Variable initialization */

CLASSWIDE      widget form_id,          @
                el_shape_switch,    @
                group_name_box,    @
                apply_button,       @
                item_1,             @
                cancel_button

FUNCTION INIT()

REAL y_loc

/*
* Create the form
*/
form_id=ui_form_create(
    /* callback */     "",           @
    /* x */           FORM_X_LOC,   @
    /* y */           FORM_Y_LOC,   @
    /* position */    "UL",         @
    /* width */        FORM_WID_SML, @
    /* height */       FORM_HGT_FULL, @
    /* label */        "Group Elements", @
    /* iconname */    "")           )

y_loc = FORM_T_MARGIN

/*
* Create the "Element Shape Switch"
*/
```

```

el_shape_switch = ui_switch_create(
    /* parent */      form_id, @
    /* callback */   "", @
    /* x */          UNFRAMED_L_MARGIN, @
    /* y */          y_loc, @
    /* num_columns */ 1, @
    /* label */       "Element Shape", @
    /* always_on? */ TRUE ) @

/*
 * Add the items (element shapes) to the switch
 */

ui_item_create(
    /* parent */      el_shape_switch, @
    /* name */        "Point", @
    /* label */       "Point", @
    /* toggleable */ FALSE ) @

ui_item_create(
    /* parent */      el_shape_switch, @
    /* name */        "Bar", @
    /* label */       "Bar", @
    /* toggleable */ FALSE ) @

ui_item_create(
    /* parent */      el_shape_switch, @
    /* name */        "Tria", @
    /* label */       "Tria", @
    /* toggleable */ FALSE ) @

item_1 = ui_item_create(
    /* parent */      el_shape_switch, @
    /* name */        "Quad", @
    /* label */       "Quad", @
    /* toggleable */ FALSE ) @

ui_item_create(
    /* parent */      el_shape_switch, @
    /* name */        "Tet", @
    /* label */       "Tet", @
    /* toggleable */ FALSE ) @

ui_item_create(
    /* parent */      el_shape_switch, @
    /* name */        "Wedge", @
    /* label */       "Wedge", @
    /* toggleable */ FALSE ) @

ui_item_create(
    /* parent */      el_shape_switch, @
    /* name */        "Hex", @
    /* label */       "Hex", @
    /* toggleable */ FALSE ) @

y_loc += SWITCH_4R_HGT_LABEL + 3 * SWITCH_HGT_LABEL_INCR +@ INTER_WIDGET_SPACE
ui_wid_set( item_1, "VALUE", TRUE )

```

```

/*
 * Create the "group name data box"
 */

group_name_box = ui_databox_create(
    /* parent */      form_id, @
    /* callback */   "", @
    /* x */          UNFRAMED_L_MARGIN, @
    /* y */          y_loc, @
    /* label_length */ 0.0, @
    /* box_length */  DBOX_WID_SINGLE, @
    /* label */       "Group Name", @
    /* value */      "", @
    /* label_above */ TRUE, @
    /* datatype */   "STRING", @
    /* num_vals */   1 ) @

y_loc += DBOX_HGT_LABOVE + INTER_WIDGET_SPACE

/*
 * Create the "Apply" button
 */

apply_button = ui_button_create(
    /* parent */      form_id, @
    /* callback */   "apply_cb", @
    /* x */          BUTTON_HALF_X_LOC1, @
    /* y */          y_loc, @
    /* width */     BUTTON_WID_HALF, @
    /* height */    0.0, @
    /* label */      "Apply", @
    /* labelinside */ TRUE, @
    /* highlight */  TRUE ) @

/*
 * Create the "Cancel" button
 */

cancel_button = ui_button_create(
    /* parent */      form_id, @
    /* callback */   "cancel_cb", @
    /* x */          BUTTON_HALF_X_LOC2, @
    /* y */          y_loc, @
    /* width */     BUTTON_WID_HALF @
    /* height */    0.0, @
    /* label */      "Cancel", @
    /* labelinside */ TRUE, @
    /* highlight */  FALSE ) @

y_loc += BUTTON_DEFAULT_HGT + FORM_B_MARGIN

ui_wid_set( form_id, "HEIGHT", y_loc )

END FUNCTION

FUNCTION DISPLAY()

    ui_form_display( "group_elements" )

END FUNCTION

FUNCTION apply_cb()

```

```
/*
 * Create the apply call back function. It needs to get the information from the form,
 * then use it to call the put elements in group function you created earlier.
 */

*****1*****
*****1*****


*****2*****
*****2*****


END FUNCTION /* apply_cb end function */FUNCTION cancel_cb()

ui_form_hide( "group_elements" )

END FUNCTION /* cancel_cb end function */

END CLASS /* group_elements end class */
```

```
***** 1 *****  
STRING group_name[32]  
STRING element_shape[5]  
  
***** 2 *****  
ui_wid_get( group_name_box, "VALUE", group_name )  
ui_wid_get( el_shape_switch, "VALUE", element_shape )  
put_element_in_group( element_shape, group_name )
```
