Week 9: GUI – Part II

- User Interface, UI
- Layouts
- Controls

(All examples can be found in Z:\Public\gui_II.mel)

User Interface UI

We started to create dialogs using MEL script last week. It is very easy in MEL for creating user interface UI. Generally speaking, it is always nice to provide UI for driving your MEL scripts.

Last week, we had talked about few UI elements in Maya and created dialogs using those elements. This week we will take a closer look of the UI elements and learn more about the interaction of those elements.

Command:

Last week, we just created the elements on the dialog without knowing how to interact with it. One of the most typical UI elements in Maya dialog is button. For button, we can also assign commands to most control types in Maya. In the below example, we will use the -command flat to specify particular MEL script that will be executed whether the button is clicked.

This example demonstrates a common way that allows us to interact with Maya using the common control elements:

```mel
{  
    string $win = `window`;  
    columnLayout;  
    $cmd = ("deleteUI " + $win);  
    button -label "Quit" -command $cmd;  
    showWindow;  
}
```

The above code will create a simple dialog as in the Figure 1 with a “Quit” button. One thing that we need to take notice is: If we delete a window, all of its child controls will be deleted as well.
UI Layouts:
When placing UI elements in Maya dialogs, you usually want to specify a layout for those elements. Maya provides a few basic layouts for UI elements and Maya itself uses them extensively. We will try some common ones here.

The simplest of all layouts is the columnLayout. This simply places all UI elements in a vertical column. We have the following code for creating three buttons arranged in vertical fashion:

```plaintext
{ string $wnd = 'window';
columnLayout;
    button -label "button1";
    button -label "button2";
    button -label "button3";
showWindow;
}
```

Figure 2 shows the result of the code.

The rowLayout simply places all UI elements in a horizontal row. You need to specify the number of columns to be used in the row (i.e. how many elements we wish to place underneath it).

Class exercise: Use of rowLayout
Create a Maya dialog as shown in Figure 3. It will use rowLayout instead of 
columnLayout. Remember you need to specify the number of elements when you 
create rowLayout.

![Figure 3 rowLayout](image)

A frameLayout create a collapsible layout. The frame can only hold a single user 
interface item. Therefore if you want to have more than once UI elements inside the 
frame, you will need to nest another layout inside the frame layout.

As soon as we create a layout, all new UI elements will automatically be added under 
the new layout. Those elements become the children of the newly created layout. 
However we can the parent of the elements by using the setParent command.
In the above code example, you can see that “button1”, “button2”, and “button3” are created under frameLayout “frame1”. “buttonA” and “buttonB” are created frameLayout “frame2”. We achieve this using the setParent command for “jumping” out of the frame layout step by step. Actually, we did discuss this effect last week during the class.
**Symbol Button:**
Symbol buttons work in the same way as normal button except that they display an image instead of a text label. Use the –image flag to specify the image to use with particular button. The image is either in bmp or xpm image format. xpm format is X-Window format and it is common in Unix platform but it is seldom used in Windows platform.

A symbol button is created using symbolButton command and it can include a graphic icon. The following code will create a symbol button as shown in Figure 5.

```mel
{
    string $wnd = `window`;
    columnLayout;
    // create three symbol buttons with related mel command
    symbolButton -image "D:\dick\work\SM3122\mel\monster1.xpm";
    showWindow;
}
```

Figure 5 Symbol button with graphic icon

**Class exercise: Symbol button**

Create a Maya dialog as shown in Figure 6 with three symbol button using the graphic files: “monster1.bmp”, “monster2.bmp”, and “monster3.bmp”. All the bitmap files are in J: drive.

When you press button of “monster1.bmp”, you should create a circle on the Maya scene. For “monster2.bmp”, you should create a sphere. For “monster3.bmp”, you should create a polyCube.

**Hint:** Use the –command flag of the `symbolButton` command.
Figure 6 Symbol button exercise

**Checkbox:**

Checkbox(es) simply store on/off value. You can either set up commands to be trigger when the checkbox’s state changes, or you can manually query the value from the checkbox.
Radio Button:
Radio Buttons require the use of both the `radioButton` and `radioCollection` MEL commands. First we create a `radioCollection` and then insert the `radioButtons` that need to be part of the collection. The following example creates two radio collections:

```mel
// a function to be called when the checkbox gets checked.
proc on_func()
{
    print("checkbox on!
");
}

// a function to be called when the checkbox gets unchecked.
proc off_func()
{
    print("checkbox on!
");
}

// create a window
columnLayout;
$window = `window`;

// define the layout of controls added
to the window.
$columnLayout = `columnLayout`;

// create a checkbox
$c = `
checkBox
-label "thingy"
-onCommand "on_func"
-offCommand "off_func"
`;

// show the window we last created
showWindow;

// to get the current value of the checkbox, use the -query flag
$value = `checkBox
-query
-value $c`;

print("check_box value = "+ $value +"\n");
}`
Float Field:

Float fields can be used to provide numerical text input into your user interfaces. For integers you will need an `intField`, for strings you can use the `textField` control, and for float value you can use `floatField`. 

```{window; columnLayout;
   // create the first radio collection
   $radio1 = `radioCollection`;
   // add some radio buttons to the collection
   $on = `radioButton -label "On"`;
   $off = `radioButton -label "Off"`;

   separator -w 50 -style "single";

   // create the second radio collection
   $radio2 = `radioCollection`;

   // add some radio buttons to the collection
   $X = `radioButton -label "X"`;
   $Y = `radioButton -label "Y"`;
   $Z = `radioButton -label "Z"`;

   // edit the radio collections to set the required radio button
   radioCollection -edit -select $on $radio1;
   radioCollection -edit -select $X $radio2;

   // now show the window
   showWindow;

   // If you need to query the selected radio button, use...
   $selected = `radioCollection -query -select`;
}`

Figure 7 Checkbox
You will notice that we are using the global valuable to pass around the name for the floatField in order to capture the value that the user had input. It is one way of getting the floatField value. There is another way of doing that.
Scroll Field:

The scroll field, created by `scrollField` command, defines a large multiple line text editable area for inputting large amount of text. For example, Maya's Script Editor itself is composed out of two scroll fields; one for the results, one for the code being edited.

```plaintext
{
    // create the window
    window;

    // create a pane layout to hold the 4 scroll fields
    paneLayout -configuration "horizontal4";

    // create 4 scroll fields
    scrollField -wordWrap true
                  -text "Non editable with word wrap" -editable false;
    scrollField -wordWrap false
                  -text "Non editable with no word wrap" -editable false;
    scrollField -wordWrap true
                  -text "Editable with word wrap";
    scrollField -wordWrap false
                  -text "Editable with no word wrap";
    showWindow;
}
```
Class exercise: Create the UI

Create the UI as shown in Figure 10. You can do it any way you like by using everything that you had learnt so far.

Figure 10 Class Exercise

** Week 8 End **