The MaxGUI Beginner Tutorial Series - Tutorial 3: More Buttons and Events

In the previous tutorial I used buttons as examples because they were fairly easy to understand. Lets cover the remaining button topics so that we can move on to other more interesting gadgets.

As a reminder, the syntax for the **CreateButton** function is:-

```
Function
CreateButton:TGadget(label$,x,y,w,h,group:TGadget,style=BUTTON_PUSH)
```

Buttons can be created with the following styles:-

Constant	Meaning
BUTTON_PUSH	Standard push button
BUTTON_CHECKBOX	A check box button that displays a tick when selected
BUTTON_RADIO	A radio button is accompanied by a small circular indicator, filled when selected
BUTTON_OK	Standard push button that is also activated when the user presses the RETURN key
BUTTON_CANCEL	Standard push button that is also activated when the user presses the ESCAPE key

We've already seen the default button style BUTTON_PUSH. Lets now look at how the BUTTON_OK style behaves

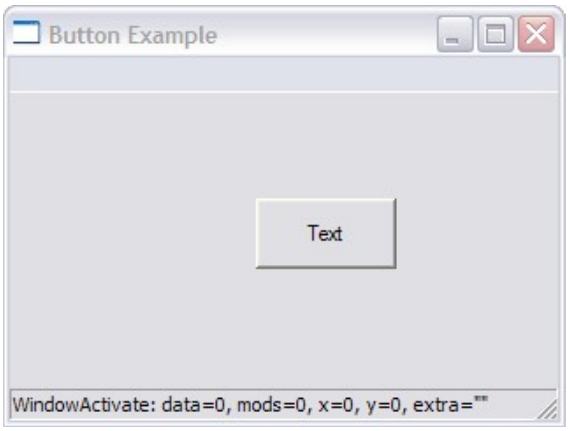
```
SuperStrict

Local MyWindow:TGadget=CreateWindow("Button Example", 200,200,320,240)
Local MyButton:TGadget=CreateButton("Text",140,60,80,40,
MyWindow,BUTTON_OK)

Repeat
  WaitEvent()
  Select EventID()
  Case EVENT_WINDOWCLOSE
     End
  Case EVENT_GADGETACTION
    SetGadgetText(MyButton,"Button clicked")
   End Select
   SetStatusText MyWindow, CurrentEvent.ToString()
Forever
```

Notice that I have added a **SetStatusText** function in the above program so that we can see on the statusbar what is happening. **SetStatusText**, as we saw earlier, allows us to print text messages on a status line. **CurrentEvent.ToString()** is a MaxGUI method which gives information on the current event. Very useful for debugging.

If you cut and paste the above program and run it, you should see the following:-



Notice the text in the statusbar, now if you were to play with the window or click the button the text will display the action that you have just carried out.

Now close the window and re-run the program. According to the manual, pressing the <ENTER> key will be the same as clicking the button but nothing happens (see the statusbar). But if you click the button first and then press the <ENTER> key it behaves as per what the manual said.

This is because the button is not in **focus**. Focus is another important concept in the GUI world. Only the gadget in focus will be able to receive keyboard input. In order to set the focus to the button we need to add another line of code:-

```
SuperStrict

Local MyWindow:TGadget=CreateWindow("Button Example", 200,200,320,240)
Local MyButton:TGadget=CreateButton("Text",140,60,80,40,
MyWindow,BUTTON_OK)

Repeat
  ActivateGadget MyButton
  WaitEvent()
  Select EventID()
  Case EVENT_WINDOWCLOSE
    End
  Case EVENT_GADGETACTION
   SetGadgetText(MyButton,"Button clicked")
  End Select
   SetStatusText MyWindow, CurrentEvent.ToString()
Forever
```

The **ActivateGadget** function will bring the focus to the gadget specified. Pressing the <ENTER> key now works. Trying pressing the <ESC> key, nothing should happen (try moving the window first so you can see the status changing).

Now that we know about focus, we should be able to make a button of style BUTTON_CANCEL work first time:-

```
SuperStrict

Local MyWindow:TGadget=CreateWindow("Button Example", 200,200,320,240)
Local MyButton:TGadget=CreateButton("Text",140,60,80,40,
MyWindow,BUTTON_CANCEL)

Repeat
  ActivateGadget MyButton
```

```
  WaitEvent()
  Select EventID()
  Case EVENT_WINDOWCLOSE
    End
  Case EVENT_GADGETACTION
   SetGadgetText(MyButton,"Button clicked")
  End Select
  SetStatusText MyWindow, CurrentEvent.ToString()
Forever
```

By specifying BUTTON_CANCEL as the style of the created button, we can now receive an EVENT_GADGETACTION event by pressing the <ESC> key.

Radio Buttons

Before we move on to radio buttons, let me introduce you to another MaxGUI gadget called **Labels.**
They can be easily created using the, you guess it, the **CreateLabel** function which has the following syntax:-

```
Function CreateLabel:TGadget(name$,x,y,w,h,group:TGadget,style=0)
```

Again the syntax is very similar to **CreateWindow** and **CreateButton**. One of the beauty of MaxGUI, is syntax consistency.

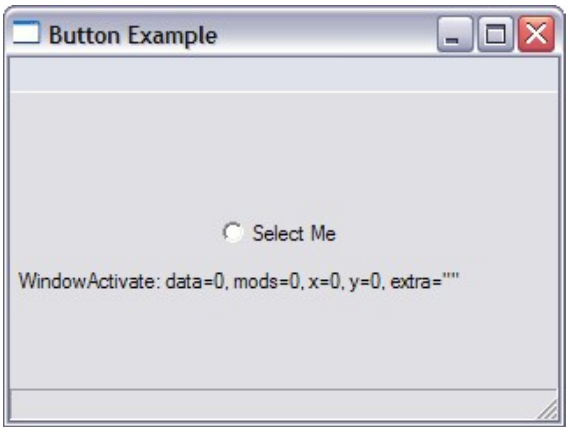
So instead of using **SetStatusText** as our debugging tool, lets use a label instead. Note that in the program below I have made a few changes (see texts in bold)

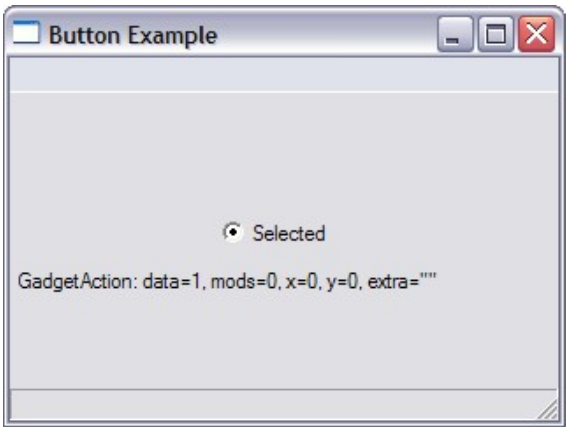
```
SuperStrict

Local MyWindow:TGadget=CreateWindow("Button Example", 200,200,320,240)
Local MyButton:TGadget=CreateButton("Select Me",120,60,100,40,
MyWindow,BUTTON_RADIO)
Local Label:TGadget=CreateLabel("Blank",5,100,300,40, MyWindow)

Repeat
  WaitEvent()
  Select EventID()
  Case EVENT_WINDOWCLOSE
    End
  Case EVENT_GADGETACTION
   SetGadgetText(MyButton,"Selected")
  End Select
  SetGadgetText Label, CurrentEvent.ToString()
Forever
```

Notice the radio button before and after selection.





Radio buttons are good for selections when you only want a single answer. Lets look at an example

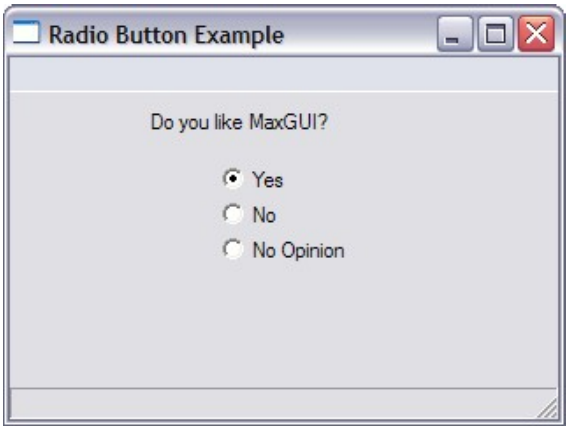
```
SuperStrict

Local MyWindow:TGadget=CreateWindow("Radio Button Example", 200,200,320,240)

Local Label0:TGadget=CreateLabel("Do you like MaxGUI?",80,10,300,20, MyWindow)
Local Radio1:TGadget=CreateButton("Yes",120,40,100,20, MyWindow,BUTTON_RADIO)
Local Radio2:TGadget=CreateButton("No",120,60,100,20, MyWindow,BUTTON_RADIO)
Local Radio3:TGadget=CreateButton("No Opinion",120,80,100,20,
MyWindow,BUTTON_RADIO)

Repeat
  WaitEvent()
  Select EventID()
  Case EVENT_WINDOWCLOSE
     End
   End Select
Forever
```

Running the above program yields a window where you can select from one of three possible answers. Notice that you can have only one selection at a time.



What we would like to able to do next is to figure out what our user has selected. Lets add some useful lines to our code

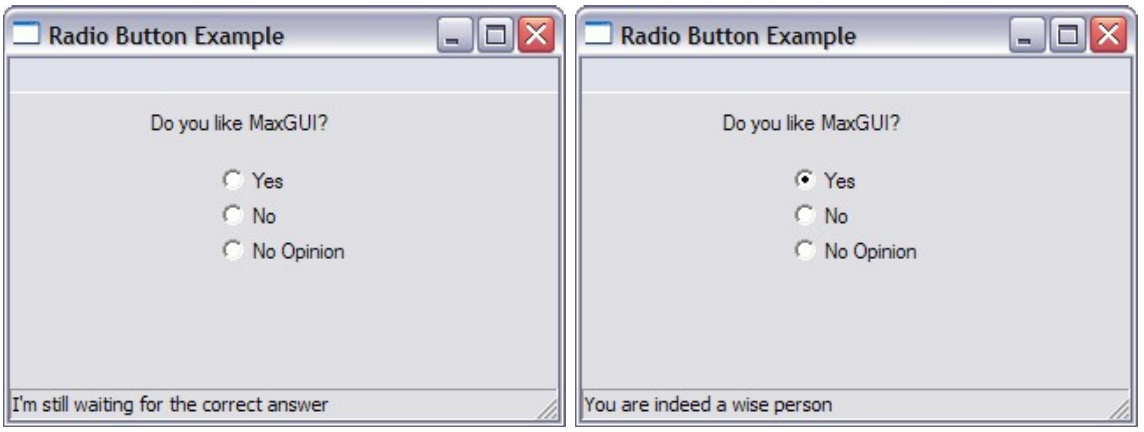
```
SuperStrict

Local MyWindow:TGadget=CreateWindow("Radio Button Example", 200,200,320,240)
```

```
Local Label0:TGadget=CreateLabel("Do you like MaxGUI?",80,10,300,20, MyWindow)
Local Radio1:TGadget=CreateButton("Yes",120,40,100,20, MyWindow,BUTTON_RADIO)
Local Radio2:TGadget=CreateButton("No",120,60,100,20, MyWindow,BUTTON_RADIO)
Local Radio3:TGadget=CreateButton("No Opinion",120,80,100,20,
MyWindow,BUTTON_RADIO)

Repeat
  WaitEvent()
  Select EventID()
  Case EVENT_WINDOWCLOSE
    End
  End Select
  If ButtonState(Radio1)=True
    SetStatusText MyWindow,"You are indeed a wise person"
  Else
    SetStatusText MyWindow,"I'm still waiting for the correct answer"
  EndIf
Forever
```

By using the MaxGUI **ButtonState** function we can check for the state of the Radio1. The state will be **true** if the user has selected it.



The last style of button that we want to look at is the BUTTON_CHECKBOX style. Let's see how this works

```
SuperStrict

Local MyWindow:TGadget=CreateWindow("Checkbox Button Example", 200,200,320,240)

Local Label0:TGadget=CreateLabel("Select the Product you wish to buy",80,10,300,20,
MyWindow)
Local BlitzMax:TGadget=CreateButton("Blitzmax $80",120,40,100,20,
MyWindow,BUTTON_CHECKBOX)
Local MaxGUI:TGadget=CreateButton("MaxGUI  $25",120,60,100,20,
MyWindow,BUTTON_CHECKBOX)
Local BlitzPlus:TGadget=CreateButton("I am a registered Blitzplus owner",80,80,200,20,
MyWindow, BUTTON_CHECKBOX)

Local Total:TGadget=CreateLabel("Total in Basket $",60,110,300,20, MyWindow)
Local Amount:Int

Repeat
  WaitEvent()
  Amount=0
```

```
  Select EventID()
  Case EVENT_WINDOWCLOSE
    End
  End Select
  If ButtonState(BlitzMax)=True Amount=80
  If ButtonState(MaxGUI)=True
    If ButtonState(BlitzPlus)=False Amount=Amount+25
  EndIf
  SetGadgetText Total,"Total in Basket $"+Amount+".00"

Forever
```

If you cut&paste and run the above program and click on the two checkboxes, you should get the following:-

Note that unlike radio button check boxes can have multiple selections. Now lets try and understand the program a bit better

```
Local BlitzMax:TGadget=CreateButton("Blitzmax $80",120,40,100,20,
MyWindow,BUTTON_CHECKBOX)
Local MaxGUI:TGadget=CreateButton("MaxGUI  $25",120,60,100,20,
MyWindow,BUTTON_CHECKBOX)
Local BlitzPlus:TGadget=CreateButton("I am a registered Blitzplus owner",80,80,200,20,
MyWindow, BUTTON_CHECKBOX)
```

The three lines of code creates the three checkboxes that we saw. Note the spacing of the three lines defined by their y positions (40,60 and 80)

```
Local Total:TGadget=CreateLabel("Total in Basket $",60,110,300,20, MyWindow)
Local Amount:Int
```

We next create the label to write our total amount due and creatively called it total. We then create an integer variable called Amount to keep our total number in.

```
  If ButtonState(BlitzMax)=True Amount=80
```

The above **If ButtonState** statement will increase the amount due by 80 if the user checked the BlitMax checkbox.

```
  If ButtonState(MaxGUI)=True
    If ButtonState(BlitzPlus)=False Amount=Amount+25
  EndIf
```

The above convoluted **If** statements are required to cater for the fact that you get MaxGUI for free if you are already a registered user of BlitzPlus i.e. if you checked the MaxGUI button (ButtonState is true) the amount will only be increased if you do not check the BlitzPlus checkbox.

```
SetGadgetText Total,"Total in Basket $"+Amount+".00"
```

Finally the above statement uses the **SetGadgetText** function to write the total amount in the basket onto the Total label.

Summary

In this tutorial we covered the use of the **CreateButton** function for the various button styles, which can really be grouped into 3 types; push buttons, radio buttons and checkboxes.

We also learned about **focus**, our gadget must be in focus before any keyboard entry can be made to it.

We also learned about using labels via the **CreateLabel** function

And lastly we learned to check whether our user has checked/selected our buttons by using the **ButtonState** function.

We are now ready to move on to the next tutorial and more interesting gadgets

Return to Tutorial Index.