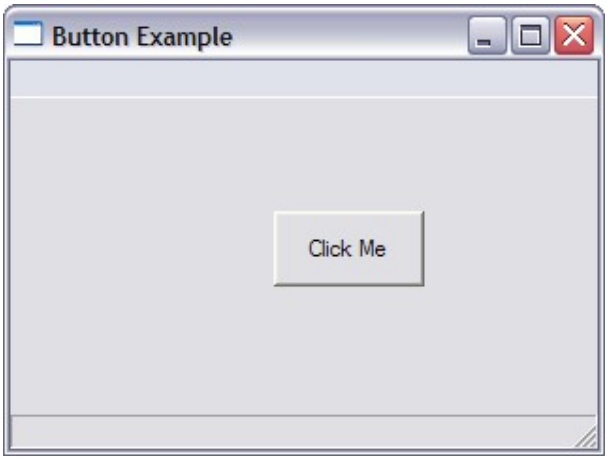
The MaxGUI Beginner Tutorial Series - Tutorial 2: Buttons and Events

Gadgets and Events are central to MaxGUI. Unfortunately we cannot understand one without the other. What I'm trying to do here is introduce both concepts slowly. I am assuming that you have read my previous tutorial on **CreateWindows** and also knows some basic programming concepts.

Lets start by introducing the Button gadget, something very familiar to us (whether on XP, Mac or Linux).

```
SuperStrict

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

Repeat
 WaitEvent()
Until EventID()=EVENT_WINDOWCLOSE

End
```

Cut and paste the above code into the MaxIDE and run the program. You should see the following (for Windows XP users):-



You can click the button but nothing very exciting happens. We will do more later.

Now lets try and see what all those lines of Blitzmax code means

```
SuperStrict
```

In the previous tutorial I have kept thing simple and has not introduce the BlitzMax command **Superstrict** before. From now on this will appear in all my example. The only thing we need to know at this stage is that this command will make BlitzMax very strict on how we define variables in our program. Trust me, using this at the start of our BlitzMax code saves a lot of grief.

I will not explain the **CreateWindow** command anymore as its been covered in the

previous tutorial. The function of interest to us is the **CreateButton** function.

```
Local MyButton:TGadget=CreateButton("Click Me",140,60,80,40, MyWindow)
```

The above code creates a button and displays it in the window we have created just before this line. The central piece here is the MaxGUI function **CreateButton.**

If we were to look at the Blitzmax Helpfile we will see the following syntax requirements

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

As you can see, the **CreateButton** function has almost identical parameters to the **CreateWindow** function.

- The items between brackets are parameters, lets go through them one by one:-
  - **label$** is the text that will appear on the button
  - **x,y** is the x and y coordinates of the location where the button will appear within the parent window
  - **w,h** is the **w**idth and **h**eight of the window (ie its size)
  - **group:TGadget** is the group to whom this created button belongs to. The gadget which owns this button is also called its parent
  - **style** is the style of the button with BUTTON_PUSH as the default style

Like **CreateWindow, CreateButton** also returns a Gadget object (see CreateButton:TGadget)

Introducing Events
Before we continue further with the CreateButton function, lets talk a little bit about Events

Blitzmax (and Blitzplus before this) introduces the concept of Events to the Blitz family. The Events system allows a Blitzmax program to react to an event that has happened. Examples of events:-

- A user clicking the left mouse button
- A user moving a window
- A user typing on his keyboard

One of the function that Blitzmax has to handle events is called **WaitEvent()**. Lets look at the following piece of code from our earlier examples:-

```
Repeat
 WaitEvent()
Until EventID()=EVENT_WINDOWCLOSE

End
```

I hope you are familiar with the **Repeat Until** loop. Basically, BlitzMax will repeat the **WaitEvent()** function call until the function **EventID()** returns a windowclose event (**EVENT_WINDOWCLOSE**)

What **WaitEvent()** does is it checks whether an event has occured. It is important to note that **WaitEvent()** waits for an event to happen before it proceeds to the next line of code

The **EventID()** function returns the identity number of the event which we then check against EVENT_WINDOWCLOSE. If they are the same, the program exits the repeat loop.

So basically we can say that the above code does the following

- Waits for an event to happen. If something happens goto the next line otherwise keep checking
- Check that the kind of event that happens was a Window being closed event.
- If yes, we are done, otherwise go wait for the next event

Now lets re-write the above program in a slightly different way

```
SuperStrict

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

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

Instead of using the Repeat/Until loop, we are now using the **Repeat/Forever** loop which as the name suggests will loop forever.

We are also now using another programming construct call **Select/Case**. In the above example we **select** for the value of **EventID()** and when the **Case** is EVENT_WINDOWCLOSE the program **end**s.

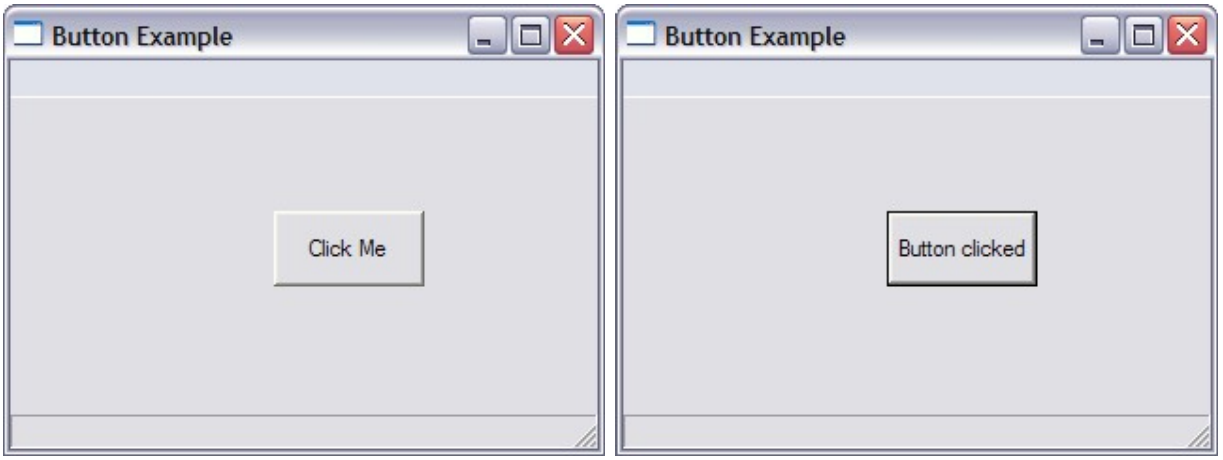
Now that we understand the above programming construct we are ready to check for other cases of **EventID**s.

```
SuperStrict

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

Repeat
  WaitEvent()
  Select EventID()
  Case EVENT_WINDOWCLOSE
     End
  Case EVENT_GADGETACTION
    SetGadgetText(MyButton,"Button clicked")
   End Select
Forever
```

Notice now we have two cases to select from, EVENT_WINDOWCLOSE and EVENT_GADGETACTION.
EVENT_WINDOWCLOSE as we have seen earlier is triggered when a user clicks the close window icon. EVENT_GADGETACTION on the other hand is triggered when a user

clicks our Button.

Cut and Paste the above code and run it. You can see the text on the button changing when you click it. This is done via the MaxGUI function **SetGadgetText**.



So lets try and summarize what we have learnt so far.

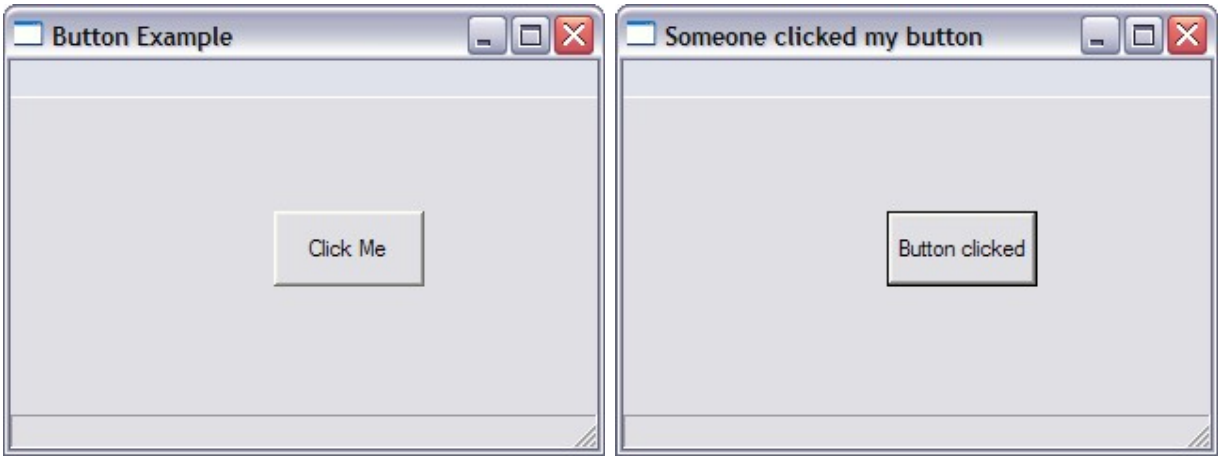
For the time being, think of events as triggers that our Blitzmax program can react to. Most MaxGUI gadgets can be used to trigger events. Gadgets are objects within MaxGUI. Examples of gadgets are

- Windows created via CreateWindow
- Buttons created via CreateButton

As an illustration, try the program below

```
SuperStrict

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

Repeat
  WaitEvent()
  Select EventID()
  Case EVENT_WINDOWCLOSE
     End
  Case EVENT_GADGETACTION
    SetGadgetText(MyButton,"Button clicked")
    SetGadgetText(MyWindow,"Someone clicked my button")
   End Select
Forever
```

Note the before clicking and after clicking picture. The title in the window also changed as a result of our 2nd **SetGadgetText** command which directs the text **Someone clicked my button** to the MyWindow Gadget. A button is a gadget, so is a window




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

Both of the gadgets that we created, MyWindow and MyButton,are of type Gadget. This you can see from the construct of the statement

```
Local GadgetName:TGadget= ....
```

Local indicates that we are creating a local variable, limited in scope but faster. Then we put the name of the variable and finally the construct **:TGadget** indicate that we want a variable of type Gadget.
To illustrate how these variable declaration work, the following 3 lines of code creates an integer, float and string variables respectively

```
Local MyNumber:Int
Local A:Float = 1.234
Local Text:String
```

It is not the intention of this tutorial to teach you about the concept of variables. But for MaxGUI it is important to declare the variables explicitly.

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

The above works, but it is wrong and will trip you up later. So declare them like so

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

When we use **Superstrict** at the start of our BlitzMax code, the compiler will catch these errors.

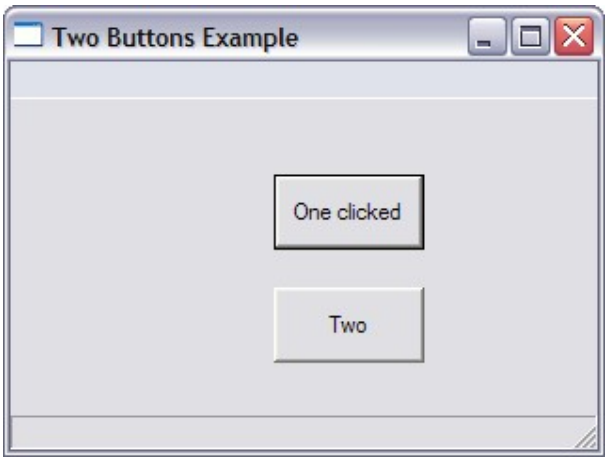
EventSource

So far our program has been checking the id of events, differentiating between EVENT_WINDOWCLOSE and EVENT_GADGETACTION. Now let see how we cope with 2

buttons.

```
SuperStrict

Local MyWindow:TGadget=CreateWindow("Two Buttons Example",
200,200,320,240)
Local Button1:TGadget=CreateButton("One",140,40,80,40, MyWindow)
Local Button2:TGadget=CreateButton("Two",140,100,80,40, MyWindow)

Repeat
  WaitEvent()
  Select EventID()
  Case EVENT_WINDOWCLOSE
    End
  Case EVENT_GADGETACTION
    Select EventSource()
      Case Button1
        SetGadgetText(Button1,"One clicked")
      Case Button2
        SetGadgetText(Button2,"Two clicked")
      End Select
   End Select
Forever
```

As you can see from the created window below, the **EventSource()** function allows us to check for the gadget that generated the mouse click allowing us to differentiate the action to be taken via another set of **Select/Case** construct.



Lets summarize what we have learnt in this tutorial

- We've learnt that we can create gadgets via for example the **CreateButton** function that can trigger events via for example user clicks
- These events can then be acted upon by our Blitzmax code via functions such as **WaitEvent()** and **EventSource()** and standard programming constructs such as loops and **Select/Case**.

There are more to learn, so lets move on to the next Tutorial.

Return to Tutorial Index