The MaxGUI Beginner Tutorial Series - Tutorial 17: Toolbars and Progress Bars

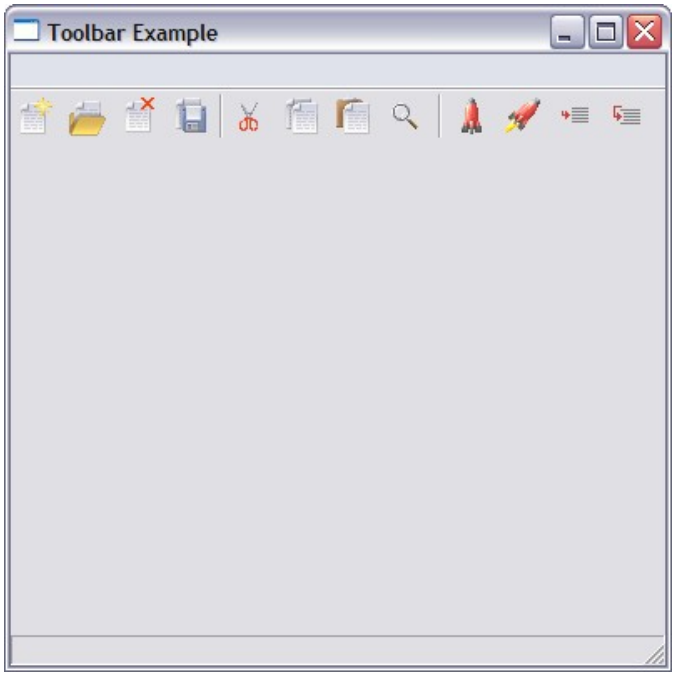
This will be a relatively short tutorial as the gadgets we are going to learn about in this tutorial have very specific purposes.

Displaying a ToolBar in a window using CreateToolBar

We can display the toolbar seen on the MaxIDE by running the following program

```
SuperStrict

Local MyWindow:TGadget=CreateWindow("Toolbar Example", 40,40,400,400)
Local MyToolBar:TGadget=CreateToolBar(BlitzMaxpath()+"/src/maxide/icons.PNG",
..
      0,0,400,20,MyWindow)

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

Notice that we can now see the familiar toobars as seen in the MaxIDE. Note though that the toolbar is shorter than the one on the MaxIDE as our window is not as wide. But the toolbar works as it should, you can hover over a button and see a reaction and click a button and it behaves as expected, except that nothing happens when we click the buttons.



The syntax for CreateToolBar function is

Function CreateToolBar:TGadget(source:Object,x,y,w,h,group:TGadget,style=0)

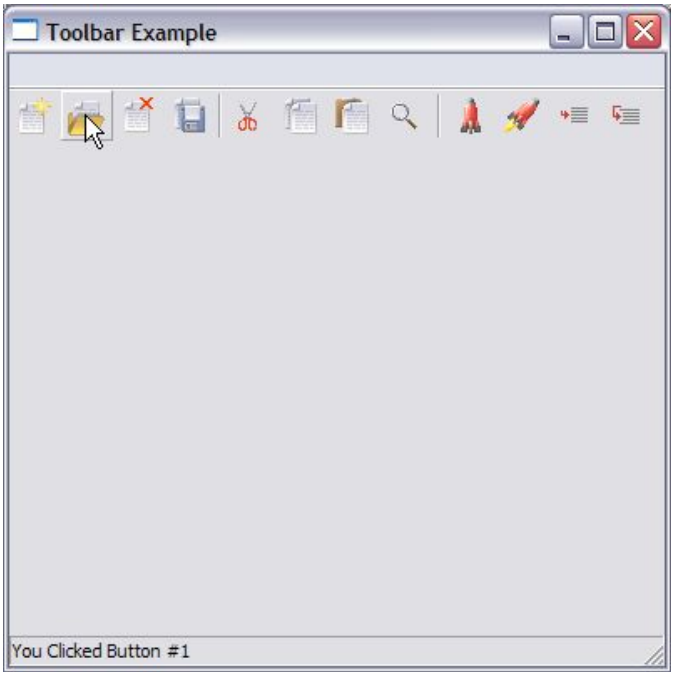
The image to be used as the source for the toolbar buttons can be any image format recognised by Blitzmax. Using png (with transparency) will allow for buttons which blends into the GUI colors as seen in these examples.

Taking action on Toolbar clicks

When we click a button on the toolbar, a **GadgetAction** event is generated. Note the code that we have added below to capture this. The **EventData**() function will retrieve the index of the clicked toolbar button.

```
SuperStrict

Local MyWindow:TGadget=CreateWindow("Toolbar Example", 40,40,400,400)
Local MyToolBar:TGadget=CreateToolBar(BlitzMaxpath()+"/src/maxide/icons.PNG",
..
      0,0,400,20,MyWindow)

Repeat
  WaitEvent()
  Select EventID()
  Case EVENT_WINDOWCLOSE
    End

  Case EVENT_GADGETACTION
    Select EventSource()
    Case MyToolBar
       SetStatusText MyWindow, "You Clicked Button #"+EventData()
    End Select

  End Select
Forever
End
```

When we click the 2nd toolbar button, the number 1 was displayed on the status bar. Remember that the toolbar index starts with a count of zero.



Let us now make the toolbar do something useful, the open file button will actually request for a file and the close button will END the program. We can do that as follows:-

```
SuperStrict

Local MyWindow:TGadget=CreateWindow("Toolbar Example", 40,40,400,400)
Local MyToolBar:TGadget=CreateToolBar(BlitzMaxpath()+"/src/maxide/icons.PNG",
..
      0,0,400,20,MyWindow)

Repeat
  WaitEvent()
  Select EventID()
  Case EVENT_WINDOWCLOSE
    End

  Case EVENT_GADGETACTION
    Select EventSource()
    Case MyToolBar
        Select EventData()
          Case 1
            Local file:String=RequestFile("Open a File")
          Case 2
            Local result:Int=Confirm("Are You Sure you want to Quit?")
            If result=1 Then End
          Default
            SetStatusText MyWindow, "You Clicked Button #"+EventData()
    End Select

  End Select
Forever
End
```

Displaying Separators in Toolbars

Notice the separators in the toolbar (see image below).

If you look at the original image, you will see blank images (imaged filled with the transparent color), these blank images are turned into separators by MaxGUI. If you noticed earlier, the image # from the eventdata() function actually includes these blank images as well.

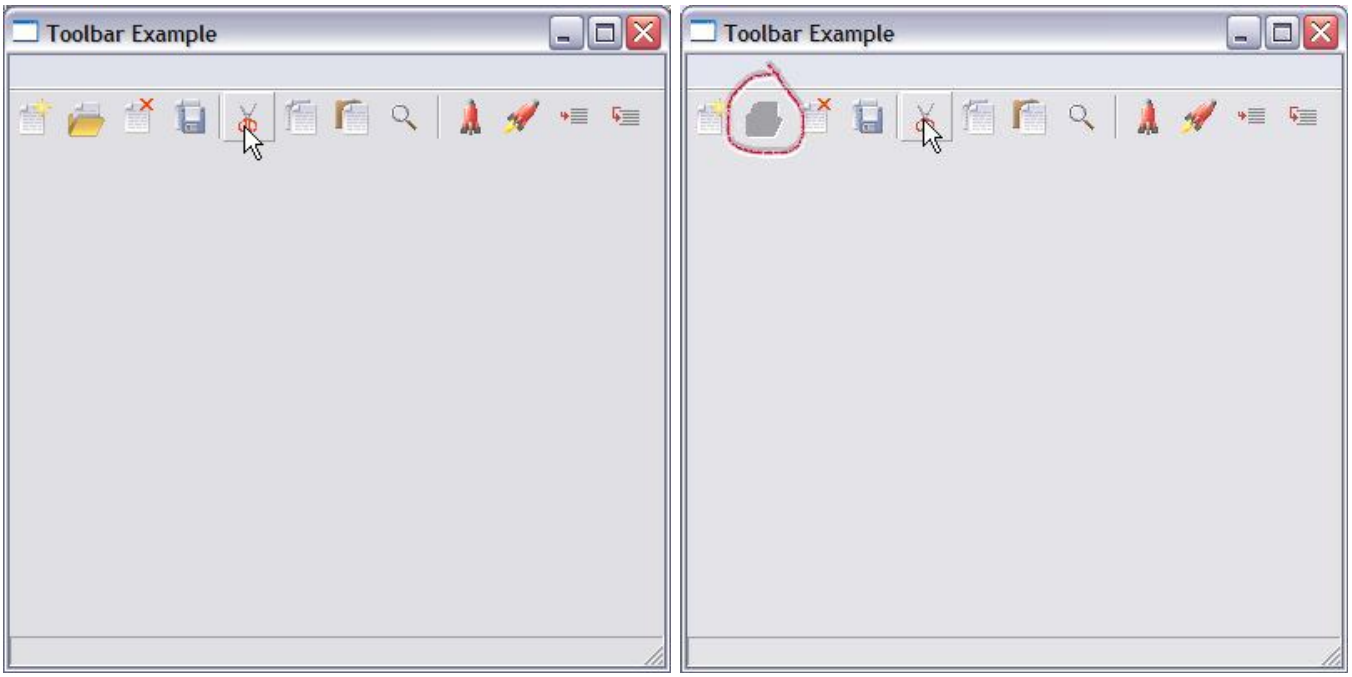
Enabling and Disabling ToolBar items

The buttons on the toolbar can be disabled and enabled via the DisableGadgetItem and EnableGadgetItem functions respectively.

```
SuperStrict

Local MyWindow:TGadget=CreateWindow("Toolbar Example", 40,40,400,400)
Local MyToolBar:TGadget=CreateToolBar(BlitzMaxpath()+"/src/maxide/icons.PNG",
..
      0,0,400,20,MyWindow)
Local state:int=1
```

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

  Case EVENT_GADGETACTION
   Select EventSource()
   Case MyToolBar
        Select EventData()
          Case 1
             Local file:String=RequestFile("Open a File")
          Case 2
             Local result:Int=Confirm("Are You Sure you want to Quit?")
             If result=1 Then End
           Case 5
              If state=1 Then
                 DisableGadgetItem MyToolbar,1
                 state=0
              Else
                 EnableGadgetItem MyToolbar,1
                 state=1
              EndIf

          Default
            SetStatusText MyWindow, "You Clicked Button #"+EventData()
        End Select
    End Select

  End Select
Forever
End
```

If we now click on the CUT button (button #5) then the Open File button will be disabled. Clicking it again will enable the button as per our code.

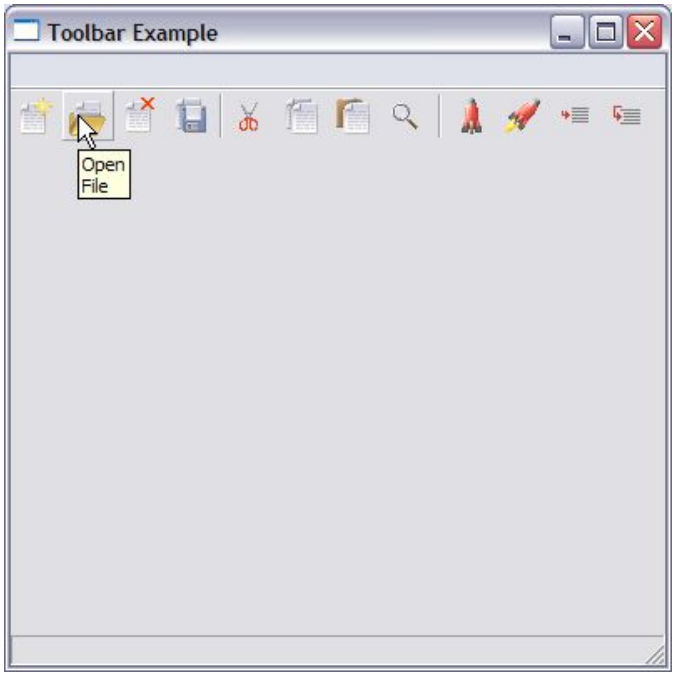


Adding tooltips to the Toolbar

The last function we want to look at is setting tooltips using the **SetToolBarTips** function.

```
SuperStrict

Local MyWindow:TGadget=CreateWindow("Toolbar Example", 40,40,400,400)
Local MyToolBar:TGadget=CreateToolBar(BlitzMaxpath()+"/src/maxide/icons.PNG",
..
      0,0,400,20,MyWindow)
SetToolBarTips MyToolBar,
["New","Open~nFile","Close","Save","","Cut","Copy","Paste","Find"]

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

Now when we hover on the buttons the tooltips appears. Note that we need to specify null strings ("") as the tips for the separators; look closely at the the above line (between "Save" and "Cut" for example). Notice also that if we use the newline code **~n** in our tooltip text we can have tooltips with multiple lines.



Using the Progress Bar Gadget

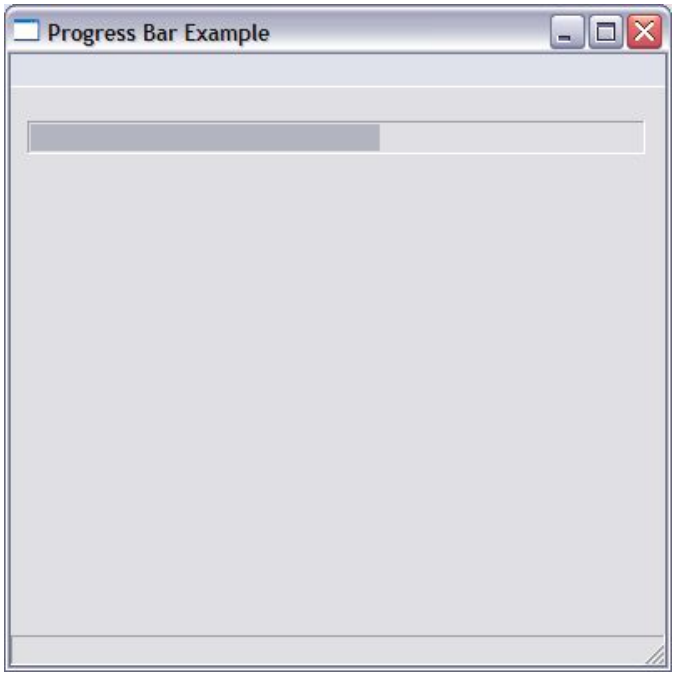
This is a very simple gadget and easy to setup as well. You basically need to do 3 things.

- Firstly create the progress bar using the **CreateProgBar** function
- Create a timer to use to update the progress via the **CreateTimer** function
- Update the progress bar using the **UpdateProgBar** function

```
SuperStrict

Local MyWindow:TGadget=CreateWindow("Progress Bar Example", 40,40,400,400)
Local MyProgBar:TGadget=CreateProgBar(10,20,370,20,MyWindow)

CreateTimer 10

Repeat
  WaitEvent()
```

```
    Select EventID()
    Case EVENT_WINDOWCLOSE
      End
    Case EVENT_TIMERTICK
     Local t:Int=EventData()
     If t=50 End
     UpdateProgBar Myprogbar,t/50.0

    End Select
  Forever
  End
```

Our progress bar should show progress like so:-



Summary

Toolbars are created using strip icons and generates gadgetaction events when clicked. Reacting to these events will allow your user to interact with your program via the toolbar buttons. You can assign tooltips to them as well as enabling and disabling them.

Progress bars allow you to indicate progress of certain activities within the program.

That's all for now. Return to Main Index.