The MaxGUI Beginner Tutorial Series - Tutorial 4: Panels
(c) Assari Dec 20 2005

In the previous tutorial we covered buttons. In this tutorial we are going to look at a more complicated MaxGUI gadget called **Panels**.

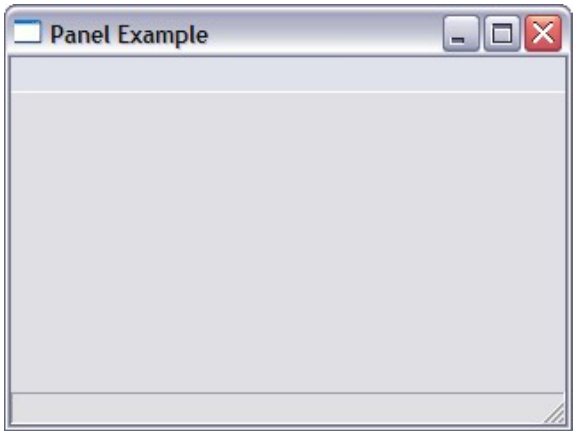
To create panels, we use the CreatePanel function with the following syntax:-

```
Function CreatePanel:TGadget(x,y,w,h,group:TGadget,style=0,title$="")
```

Let us see how the simplest panel would look like

```
SuperStrict

Local MyWindow:TGadget=CreateWindow("Panel Example", 200,200,320,240)
Local MyPanel:TGadget=CreatePanel(5,5,200,200, MyWindow)

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

Nothing very exciting as can be seen below. In fact it look exactly like our first **CreateWindow** tutorial.



A panel can actually have 3 styles:-

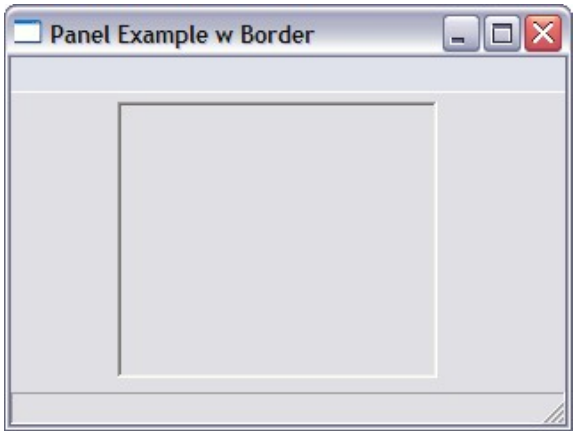
Constant	Meaning
PANEL_BORDER	Panel is drawn with a border
PANEL_ACTIVE	Panel generates mouse move events
PANEL_GROUP	Panel is drawn with a titled etched border

So lets put a border around our panel by using the PANEL_BORDER style

```
SuperStrict

Local MyWindow:TGadget=CreateWindow("Panel Example w Border",
```

```
200,200,320,240)
Local MyPanel:TGadget=CreatePanel(60,5,180,155, MyWindow, PANEL_BORDER)

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

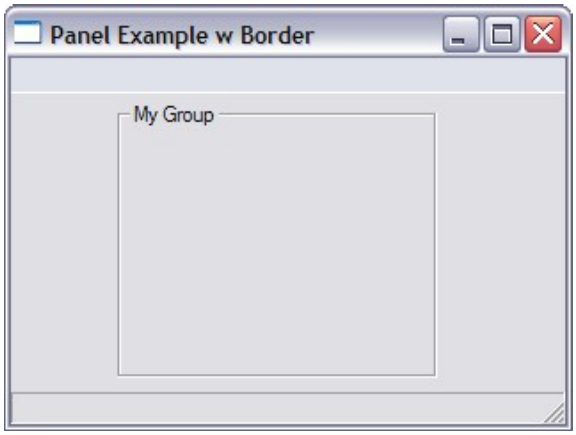
Now there is a sunken border on our panel



The next style, PANEL_GROUP, allows us the display a name on our panel

```
SuperStrict

Local MyWindow:TGadget=CreateWindow("Panel Example w Border", 200,200,320,240)
Local MyPanel:TGadget=CreatePanel(60,5,180,155, MyWindow, PANEL_GROUP, "My Group")

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

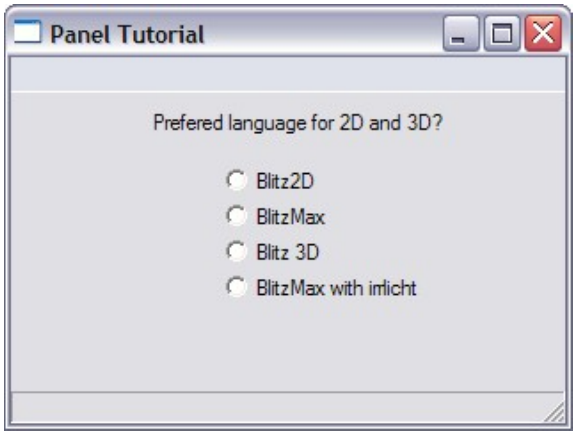
The panel now has a nice border and a name to go with it.



Apart from making nicer windows, panels actually have some use. Grouping of radio gadgets is one of them.
Lets revisit our radio button example from the previous tutorial, now with a different survey

```
SuperStrict

Local MyWindow:TGadget=CreateWindow("Panel Tutorial", 200,200,320,240)

Local Label0:TGadget=CreateLabel("Prefered Language for 2d and 3D?",80,10,300,20,
MyWindow)
Local Radio1:TGadget=CreateButton("Blitz2D",120,40,100,20,
MyWindow,BUTTON_RADIO)
Local Radio2:TGadget=CreateButton("BlitzMax",120,60,100,20,
MyWindow,BUTTON_RADIO)
Local Radio3:TGadget=CreateButton("Blitz 3D",120,80,100,20,
MyWindow,BUTTON_RADIO)
Local Radio4:TGadget=CreateButton("BlitzMax with irrlicht",120,100,120,20,
MyWindow,BUTTON_RADIO)

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

The problem with our set of radio buttons is that we can only select one out of the four. Whereas we would like the survey to allow the users to select their favourite 2D and 3D languages

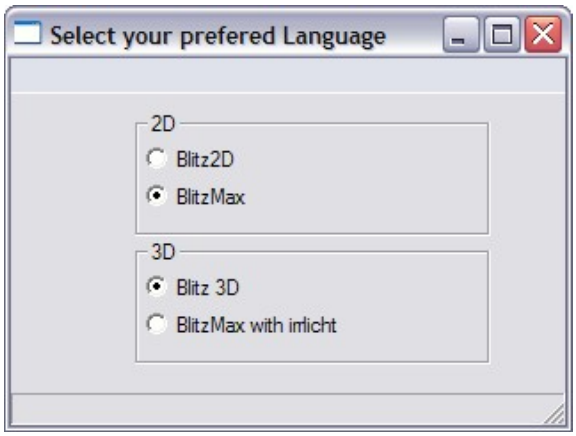


This is where our panel grouping capability comes in handy. So our program can be re-coded as follows:-

```
SuperStrict

Local MyWindow:TGadget=CreateWindow("Select your prefered Language",
200,200,320,240)

Local Panel2D:TGadget=CreatePanel(70,10,200,70, MyWindow, PANEL_GROUP, "2D")
Local Radio1:TGadget=CreateButton("Blitz2D",1,1,100,20, Panel2D,BUTTON_RADIO)
Local Radio2:TGadget=CreateButton("BlitzMax",1,22,100,20, Panel2D,BUTTON_RADIO)
```

```
Local Panel3D:TGadget=CreatePanel(70,82,200,70, MyWindow, PANEL_GROUP, "3D")
Local Radio3:TGadget=CreateButton("Blitz 3D",1,1,100,20, Panel3D,BUTTON_RADIO)
Local Radio4:TGadget=CreateButton("BlitzMax with irrlicht",1,22,120,20,
Panel3D,BUTTON_RADIO)

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

Now we can see that the panel grouping allows us to select an answer for each 2D and 3D group.

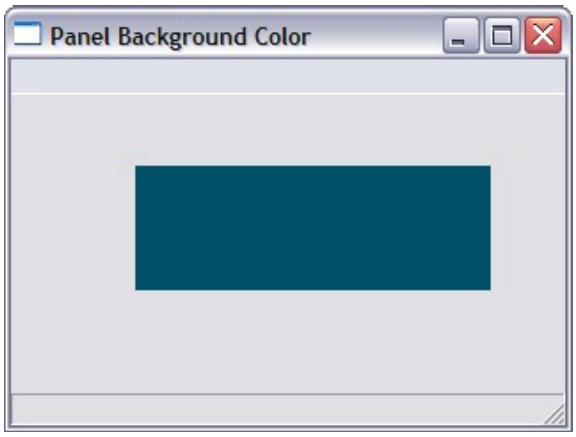


Before we move on the the last panel style PANEL_ACTIVE, let us explore another interesting property of panels, namely the ability to change background color and have an image background.

Lets start with changing its background color.

```
SuperStrict

Local MyWindow:TGadget=CreateWindow("Panel Background Color", 200,200,320,240)
Local Panel2D:TGadget=CreatePanel(70,40,200,70, MyWindow)
SetPanelColor Panel2D, 1,81,107

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

If you were to execute the above program, you should get the following:-



As you can see we now has a colored panel, courtesy of the **SetPanelColor** function. The parameters for SetColorPanel are

- The name of the panel to be colored
- The r,g,b value of the color. If you don't know what this means, experiment with putting 3 numbers at random between 0-255 e.g. try 255,255,255 or 0,0,0 or 200,154,0 etc.

The formal syntax is:-

```
Function SetPanelColor( panel:TGadget,r,g,b )
```

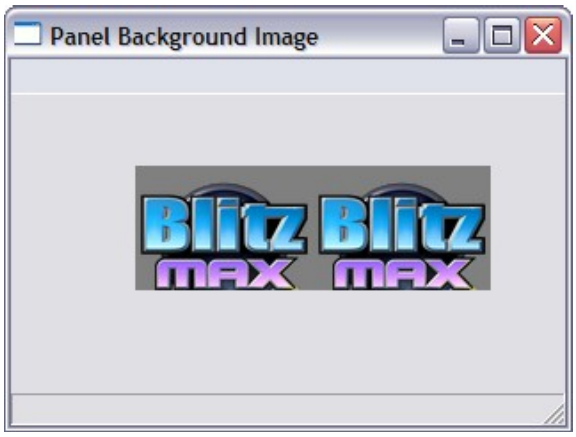
A more interesting feature is the **SetPanelPixmap** function. Instead of a background color you can specify a background image. The formal syntax looks like this:-

```
Function SetPanelPixmap(
panel:TGadget,pixmap:TPixmap,flags=PANELPIXMAP_TILE)
```

Let's us see how a simple example would look like:-

```
SuperStrict

Local MyWindow:TGadget=CreateWindow("Panel Background Image", 200,200,320,240)
Local Panel2D:TGadget=CreatePanel(70,40,200,70, MyWindow)

Local image:TPixmap=LoadPixmap("D:\My Documents on E\_Tutorials\B-max.png")
SetPanelPixmap Panel2D, image

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

Our panel now has a tiled image of the Blitzmax logo.




```
Local image:TPixmap=LoadPixmap("D:\My Documents on E\_Tutorials\B-max.png")
```

Note that the above line loads a image from a file. You will need to specify your own directory for this example to work. Otherwise you will just get a blank panel with no image on it.

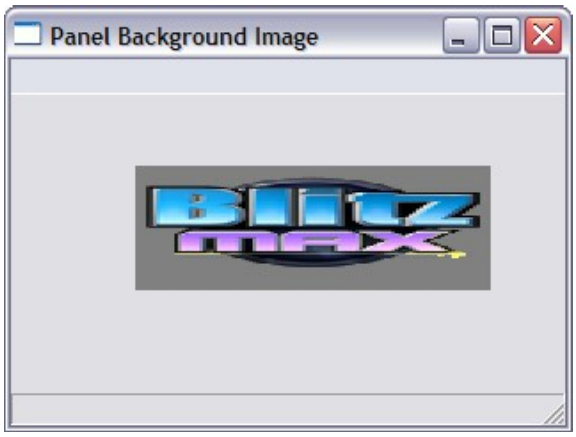
The SetPanelPixmap function has several flags (similar to style that we have seen before) as follows:-

Constant	Meaning
PANELPIXMAP_TILE	The panel is filled with repeating tiles (default)
PANELPIXMAP_CENTER	The pixmap is positioned at the center of the Panel.
PANELPIXMAP_FIT	The pixmap is scaled to best fit the Panel size.
PANELPIXMAP_STRETCH	The pixmap is stretched to fit the entire Panel.

The TILE style is the default style, similar to tiling of images on webpages. Let us see how a STRETCH style would look like:-

```
SuperStrict

Local MyWindow:TGadget=CreateWindow("Panel Background Image", 200,200,320,240)
Local Panel2D:TGadget=CreatePanel(70,40,200,70, MyWindow)

Local image:TPixmap=LoadPixmap("D:\My Documents on E\_Tutorials\B-max.png")
SetPanelPixmap Panel2D, image, PANELPIXMAP_STRETCH

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

Our Blitzmax logo is now stretched to fill up the size of the panel.



I will leave it as an exercise for you to see how the other style works.

Summary

That ends the tutorial for now. In the next installment, we will learn about mouse events, the last style of **CreatePanel**; PANEL_ACTIVE

So before we end this tutorial let us recap what we have learnt.

- Panels are gadgets which allows us to group our gadgets together
- Panels can have borders, background color and background images

The real power of panels is from their ability to track mouse events, which we shall cover next.

Return to Tutorial Index