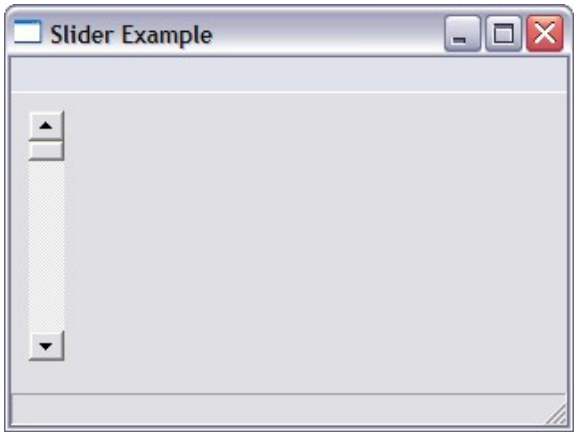
The MaxGUI Beginner Tutorial Series - Tutorial 13: Sliders and Scrollbars

In this tutorial we are going to cover sliders, another common gadget seen in all modern GUI systems.

Again let us start with the simplest MaxGUI program that demonstrates the slider functionality:-

```
SuperStrict

Local MyWindow:TGadget=CreateWindow("Slider Example", 200,200,320,240)
Local MySlider:TGadget=CreateSlider(10,10,20,140,MyWindow)

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

A vertical scrollbar can be seen below, created by our simple program above via the **CreateSlider** function. It doesnot do anything except for moving the scroll bar up and down.



The CreateSlider function has the following syntax:-

```
Function CreateSlider:TGadget(x,y,w,h,group:TGadget,style=0)
```

According to the MaxGUI helpfiles, the slider gadget can have the following styles:-

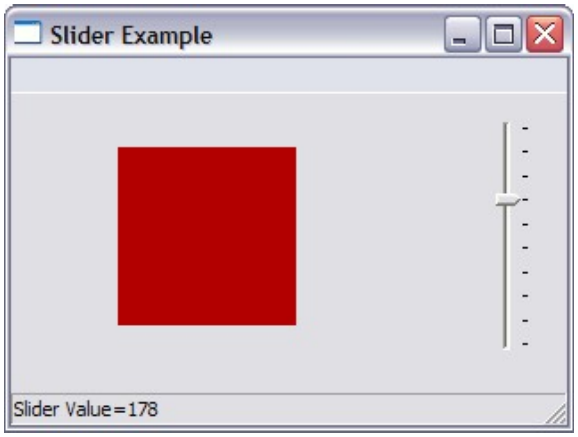
Constant	Meaning
SLIDER_HORIZONTAL	The slider is moved left and right
SLIDER_VERTICAL	User slider is moved up and down
SLIDER_SCROLLBAR	The slider uses a proportional size knob
SLIDER_TRACKBAR	The slider uses a fixed size knob
SLIDER_STEPPER	The slider has no knob, just arrow buttons

SLIDER_TRACKBAR

Let us go through the styles one by one, starting with the SLIDER_TRACKBAR style

```
SuperStrict

Local MyWindow:TGadget=CreateWindow("Slider Example", 200,200,320,240)
Local MySlider:TGadget=CreateSlider(270,10,20,140,MyWindow,SLIDER_TRACKBAR)
Local MyCanvas:TGadget=CreateCanvas(60,30,100,100,MyWindow)
Local r:Int=0

Repeat
  WaitEvent()
  Select EventID()
  Case EVENT_WINDOWCLOSE
    End
  Case EVENT_GADGETACTION
    r=int(25.5*EventData())
    SetStatusText MyWindow, "Slider Value="+r
    RedrawGadget(MyCanvas)
Case EVENT_GADGETPAINT
   SetGraphics CanvasGraphics (MyCanvas)
   SetColor  r,0,0
   DrawRect  0,0,100,100
   Flip
  End Select
Forever
```

What we can see on the screenshot below is a Trackbar slider on the right and a square canvas colored with the setting of the value of the trackbar. If you manipulate the trackbar, you can see the color changing and the status bar showing the value of the slider.



Lets take a closer look at our program. First we need to create our window and the two required gadgets, a slider and a canvas and then we place them at the appropriate locations within our window.

```
Local MyWindow:TGadget=CreateWindow("Slider Example", 200,200,320,240)
Local MySlider:TGadget=CreateSlider(270,10,20,140,MyWindow,SLIDER_TRACKBAR)
Local MyCanvas:TGadget=CreateCanvas(60,30,100,100,MyWindow)
Local r:Int=0
```

When the trackbar is moved by the user, a GADGET_ACTION event is generated and the **EventData**() function will retrieve the value of the slider (which defaults to a range of min to max of 1 to 10).

Since we want to use the value in a **SetColor** function (which has a range of 0-255) we multiply the slider value with 25.5 and turn it into an integer via the **int** function.

```
Case EVENT_GADGETACTION
  r=Int(25.5*EventData())
```

The **SetStatusText** function then writes our **r** value to the status bar. We then issue a **RedrawGadget** function call to generate the Gadgetpaint event so that our canvas will be re-painted with the new color.

```
SetStatusText MyWindow, "Slider Value="+r
RedrawGadget(MyCanvas)
```

When we receive the Gadgetpaint event, we then set the ink color to r,0,0 and redraw our rectangle.

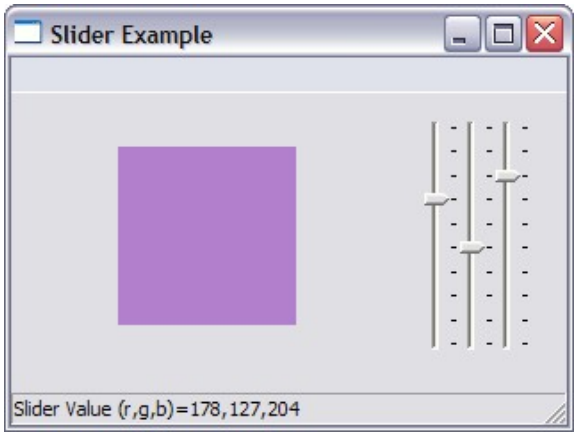
```
Case EVENT_GADGETPAINT
  SetGraphics CanvasGraphics (MyCanvas)
  SetColor  r,0,0
  DrawRect  0,0,100,100
  Flip
```

Lets get a bit more complicated and enhance our program to actually be able to manipulate the rgb values of the SetColor function. To do this we need 3 sliders, one for each rgb color as follows:-

```
SuperStrict

Local MyWindow:TGadget=CreateWindow("Slider Example", 200,200,320,240)
Local
RedSlider:TGadget=CreateSlider(230,10,20,140,MyWindow,SLIDER_TRACKBAR)
Local
GreenSlider:TGadget=CreateSlider(250,10,20,140,MyWindow,SLIDER_TRACKBAR)
Local
BlueSlider:TGadget=CreateSlider(270,10,20,140,MyWindow,SLIDER_TRACKBAR)

Local MyCanvas:TGadget=CreateCanvas(60,30,100,100,MyWindow)
Local r:Int=0
Local g:Int=0
Local b:Int=0

Repeat
 WaitEvent()
 Select EventID()
 Case EVENT_WINDOWCLOSE
   End
 Case EVENT_GADGETACTION
   Select EventSource()
   Case RedSlider
     r=Int(25.5*EventData())
   Case GreenSlider
     g=Int(25.5*EventData())
   Case BlueSlider
     b=Int(25.5*EventData())
   End Select
   SetStatusText MyWindow, "Slider Value (r,g,b)="+r+","+g+","+b
```

```
    RedrawGadget(MyCanvas)
Case EVENT_GADGETPAINT
   SetGraphics CanvasGraphics (MyCanvas)
   SetColor  r,g,b
   DrawRect  0,0,100,100
   Flip
  End Select
Forever
```

Running the program will yield us the following window, with better range of colors available for our canvas.



As usual, lets go through our program in greater detail. First we create our three sliders, calling then RedSlider, GreenSlider and BlueSlider.

```
Local
RedSlider:TGadget=CreateSlider(230,10,20,140,MyWindow,SLIDER_TRACKBAR)
Local
GreenSlider:TGadget=CreateSlider(250,10,20,140,MyWindow,SLIDER_TRACKBAR)
Local
BlueSlider:TGadget=CreateSlider(270,10,20,140,MyWindow,SLIDER_TRACKBAR)
```

Then we need to create the rgb variables

```
Local r:Int=0
Local g:Int=0
Local b:Int=0
```

Now that we have 3 different sliders, we need to know which slider causes the gadgetaction event to occur. For this, we use the **EventSource** function and the select/case statements

```
Case EVENT_GADGETACTION
  Select EventSource()
  Case RedSlider
    r=Int(25.5*EventData())
  Case GreenSlider
    g=Int(25.5*EventData())
  Case BlueSlider
    b=Int(25.5*EventData())
  End Select
```

The rest of the programs are as before except for the SetColor function which now has 3

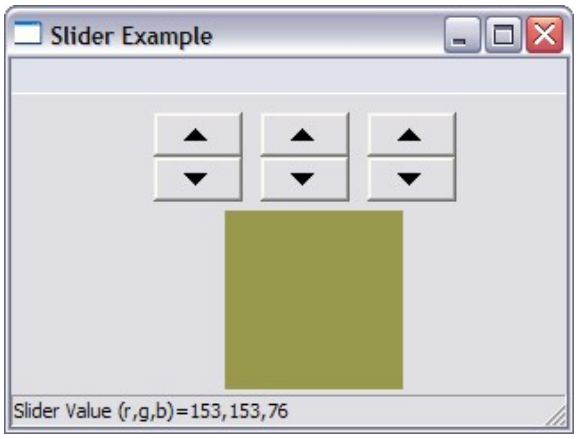
variable parameters instead of just one.

SLIDER_STEPPER

The next style we want to look at is the **SLIDER_STEPPER** style. We will just modify the above program to see how it works with the stepper style as follows:-

```
SuperStrict

Local MyWindow:TGadget=CreateWindow("Slider Example", 200,200,320,240)
Local RedSlider:TGadget=CreateSlider(80,10,50,50,MyWindow,SLIDER_STEPPER)
Local
GreenSlider:TGadget=CreateSlider(140,10,50,50,MyWindow,SLIDER_STEPPER)
Local BlueSlider:TGadget=CreateSlider(200,10,50,50,MyWindow,SLIDER_STEPPER)

Local MyCanvas:TGadget=CreateCanvas(120,65,100,100,MyWindow)
Local r:Int=0
Local g:Int=0
Local b:Int=0

Repeat
  WaitEvent()
  Select EventID()
  Case EVENT_WINDOWCLOSE
    End
  Case EVENT_GADGETACTION
    Select EventSource()
    Case RedSlider
      r=Int(25.5*EventData())
    Case GreenSlider
      g=Int(25.5*EventData())
    Case BlueSlider
      b=Int(25.5*EventData())
    End Select
    SetStatusText MyWindow, "Slider Value (r,g,b)="+r+","+g+","+b
    RedrawGadget(MyCanvas)
Case EVENT_GADGETPAINT
   SetGraphics CanvasGraphics (MyCanvas)
   SetColor  r,g,b
   DrawRect  0,0,100,100
   Flip
  End Select
Forever
```

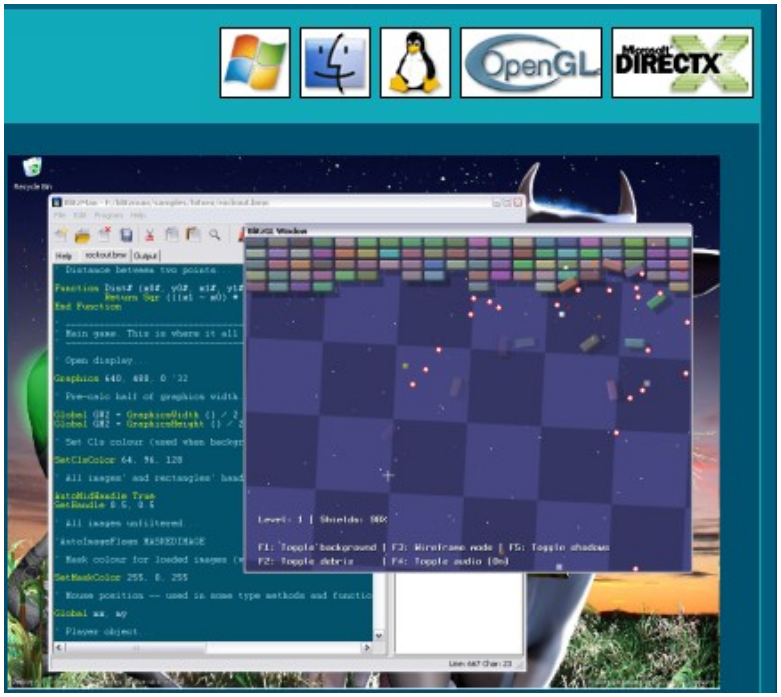
What we get are stepper style sliders which are just buttons with arrows which changes the value of the slider. Note that all we have done is just change the size and positions of the sliders and canvas to accomodate the stepper style, the rest of the program remained unchanged.



SLIDER_SCROLLBAR

Let us now look at the last style, the SLIDER_SCROLLBAR, this is what we see almost daily on our window gadgets. The scrollbars are useful to allow users to scroll through contents larger than the available display area.

In the following example we are loading a 434x387 image onto a 200x200 canvas. Download the image below so that we can load the image into our program below:-



It's a longish program but fret not as I will explain it in detail later. Let us see what it does first. Cut and Paste and run the program as usual. Pls remember to put in the correct location for the Pixmap file.

```
SuperStrict

Local MyWindow:TGadget=CreateWindow("Slider Example", 200,200,320,320)
Local HorizontalSlider:TGadget=CreateSlider(10,210,200,20,MyWindow,
SLIDER_SCROLLBAR | SLIDER_HORIZONTAL)
Local VerticalSlider:TGadget=CreateSlider(210,10,20,200,MyWindow,
SLIDER_SCROLLBAR | SLIDER_VERTICAL)
Local MyCanvas:TGadget=CreateCanvas(10,10,200,200,MyWindow)

Local map:TPixmap=LoadPixmap("D:\My Documents on E\_Tutorials\T13-05.jpg")
```

```
Local x:Int=0
Local y:Int=0

SetSliderRange VerticalSlider,200,PixmapHeight(Map)
SetSliderRange HorizontalSlider,200,PixmapWidth(Map)

Repeat
  WaitEvent()
  Select EventID()
  Case EVENT_WINDOWCLOSE
     End
  Case EVENT_GADGETACTION
     Select EventSource()
     Case HorizontalSlider
        x=EventData()
     Case VerticalSlider
        y=EventData()
     End Select
     RedrawGadget(MyCanvas)
Case EVENT_GADGETPAINT
    SetGraphics CanvasGraphics (MyCanvas)
    Cls
    DrawPixmap map,0,0
    Local Pix:TPixmap=GrabPixmap(x,y,200,200)
    Cls
    DrawPixmap  pix,0,0
    Flip
   End Select
   SetStatusText  MyWindow,"Slider values (x,y)="+x+","+y
Forever
```

The horizontal and vertical scrollbars now will actually scroll the image as expected.

OK, let us get into the program details. First we need to declare the horizontal and vertical sliders. It is important that we align the sliders against the canvas. Although this is purely for aesthetic reasons as it does not impact the functionality at all.

```
Local HorizontalSlider:TGadget=CreateSlider(10,210,200,20,MyWindow,
SLIDER_SCROLLBAR | SLIDER_HORIZONTAL)
Local VerticalSlider:TGadget=CreateSlider(210,10,20,200,MyWindow,
SLIDER_SCROLLBAR | SLIDER_VERTICAL)
```

```
Local MyCanvas:TGadget=CreateCanvas(10,10,200,200,MyWindow)
```

We then load our image into a Pixmap object and also declare our variables x and y which will store the start location for the image to be displayed on our canvas.

```
Local map:TPixmap=LoadPixmap("D:\My Documents on E\_Tutorials\T13-05.jpg")
Local x:Int=0
Local y:Int=0
```

The next two lines set the range for our sliders. Normally the range will be from 0-10, we now want to set it to fit the range of our image. We use the **PixmapHeight** and **PixmapWidth** functions to set the range of the Vertical and Horizontal sliders via the **SetRangeSlider** function respectively.

The syntax for SetSliderRange from the manual is as below:-

```
Function SetSliderRange(slider:TGadget,range0,range1)
```

I actually prefer the Blitzplus description, where range0 actually is the visible panel, hence we use the canvas width and height and range1 is the total size of the pixmap

```
Function SetSliderRange(slider:TGadget,Visible Range,Total Range)
```

So we set our slider range as follows:-

```
SetSliderRange VerticalSlider,200,PixmapHeight(Map)
SetSliderRange HorizontalSlider,200,PixmapWidth(Map)
```

Once the scroll bar is moved, a gadgetaction event will be generated. We then need to check the source of the Gadgetevent by using the **EventSource**() function. We then use the select/case statements to update the relevant variables.

```
Case EVENT_GADGETACTION
  Select EventSource()
  Case HorizontalSlider
    x=EventData()
  Case VerticalSlider
    y=EventData()
  End Select
  RedrawGadget(MyCanvas)
```

The **RedrawGadget** function will cause a gadgetpaint event to be generated. We then draw the original image to our canvas, then grab just the "visible" portion and redraw the "visible" part back onto our canvas to simulate the scrolling action. All these happen behind the scenes, the user will only see the final image when we issue the **Flip** function. [Update: there is another, better?, way to scroll an image with scroll bars. see this link)

```
Case EVENT_GADGETPAINT
  SetGraphics CanvasGraphics (MyCanvas)
  Cls
  DrawPixmap map,0,0
  Local Pix:TPixmap=GrabPixmap(x,y,200,200)
  Cls
  DrawPixmap  pix,0,0
```

```
Flip
```

And finally, our usual **SetStatusText** function to show us the value returned by the **eventdata**() function.

```
SetStatusText  MyWindow,"Slider values (x,y)="+x+","+y
```

Summary

The Slider Gadget, like the tabber gadget, is simply a conduit for user input. The job of managing the scroll or slide has to be done by the program. Sliders are easier as the events generated are numerical in nature but scroll-bars are harder as they need to "scroll" something, typically images or text.

To recap what we have learnt so far

- Sliders can be created using the **CreateSlider** function and have 3 styles, trackbars, steppers and scrollers. Each of this can either be vertical or horizontal.
- The value from the slider gadget can be retrieved via the **Eventdata**() function. The range returned by **Eventdata**() defaults to 0-10 but can be changed via the **SetSliderRange** function.
- **SetSliderRange** function can be used to programatically set the slider value (I did not cover this but the usage is fairly straight forward)

So thats ends our tutorial for now. Back to Tutorial Index.