# The MaxGUI Beginner Tutorial Series - Tutorial 12:  Canvas, Events and 2D Graphics

(c) Assari Dec 31 2005

One of the strength of BlitzMax is its OOPs features. What most people does not realise is that the power of BlitzMax and MaxGUI are also available without the OOPs. In these tutorials I have shyed away from using OOP so as  not to complicate the tutorials and my examples.

The Canvas gadget, the subject of this tutorial, is also one gadget that can be used easily without OOP as well. This gadget actually deserves a whole series of tutorial by itself as it actually brings the whole power of Max2D with it. However we are not going to cover the complete functionality of Max2D but to give you a glimpse of what is possible.

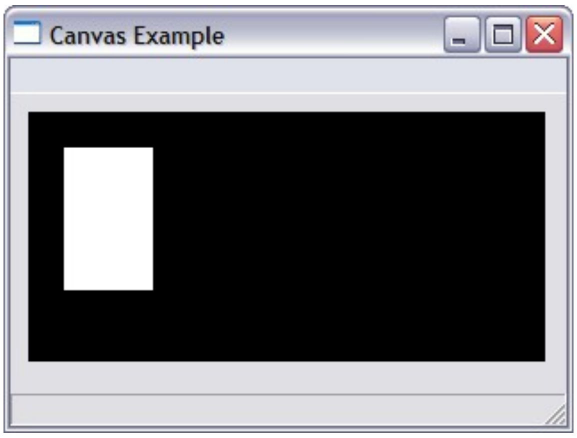
Lets take a look at a very simple Canvas Gadget in action-

```
SuperStrict

Local MyWindow:TGadget=CreateWindow("Canvas Example", 200,200,320,240)
Local MyCanvas:TGadget=CreateCanvas(10,10,290,140,MyWindow)

Repeat
  WaitEvent()
  Select EventID()
  Case EVENT_WINDOWCLOSE
     End
  Case EVENT_GADGETPAINT
    SetGraphics CanvasGraphics (MyCanvas)
    DrawRect  20,20,50,80
    Flip
   End Select
Forever
```

As we can see below, a rectangle has been drawn inside our window, on top of the canvas that we have created with the **CreateCanvas** function.



The program lines of interest to us are:-

```
Case EVENT_GADGETPAINT
    SetGraphics CanvasGraphics (MyCanvas)
    DrawRect  20,20,50,80
    Flip
```

The EVENT_GADGETPAINT event is the trigger we use to draw stuff on our canvas. The

**SetGraphics** function directs the drawing to the MyCanvas gadget.
**DrawRect** draws a rectangle at position 20,20 and of size 50x80.
**Flip** is a max2D function which effectively displays the drawn items to the screen.

I have picked the **DrawRect** function for this example, but any max2D graphics command can be used on Canvas gadgets making this an uber powerful gadget.

Let's take a look at more examples

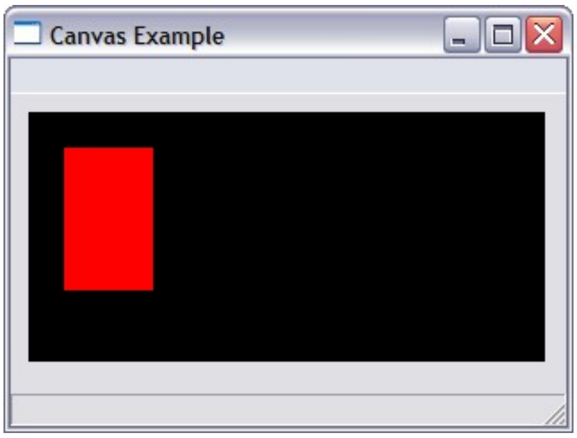
We can draw a colored rectange as follows:-

```
SuperStrict

Local MyWindow:TGadget=CreateWindow("Canvas Example", 200,200,320,240)
Local MyCanvas:TGadget=CreateCanvas(10,10,290,140,MyWindow)

Repeat
  WaitEvent()
  Select EventID()
  Case EVENT_WINDOWCLOSE
     End
  Case EVENT_GADGETPAINT
    SetGraphics CanvasGraphics (MyCanvas)
    SetColor  255,0,0
    DrawRect  20,20,50,80
    Flip
   End Select
Forever
```

a voila, we have a red rectangle (from our **SetColor** function)



We can also do animations. In order to this we need to use the timer gadget as follows:-

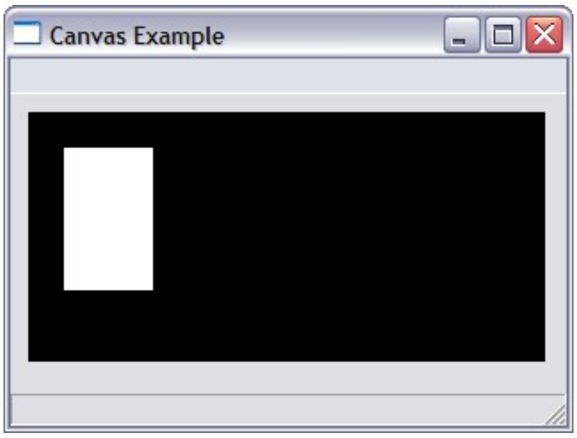
```
SuperStrict

Local MyWindow:TGadget=CreateWindow("Canvas Example", 200,200,320,240)
Local MyCanvas:TGadget=CreateCanvas(10,10,290,140,MyWindow)
Local timer:TTimer=CreateTimer(60)
Local x:Int=0

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

```
        Case EVENT_TIMERTICK
          x=x+1
          If x>240 x=0
          RedrawGadget(MyCanvas)
        Case EVENT_GADGETPAINT
          SetGraphics CanvasGraphics (MyCanvas)
          Cls
          DrawRect  x,20,50,80
          Flip
         End Select
      Forever
```

You should now see a white rectangle moving across the screen.



The **CreateTimer** function creates a special timer which will trigger an event 60 times per second. This event will be captured as an EVENT_TIMERTICK event. The integer variable x is used as the horizontal position of our rectangle.

```
      Local timer:TTimer=CreateTimer(60)
      Local x:Int=0
```

When the EVENT_TIMERTICK event is detected (which will be once every 1/60 th of a second), we will move the rectangle coordinate by one pixel. We then need to check that if it has gone over the screen, we reset the location back to zero.

The **RedrawGadget** function will cause the EVENT_GADGETPAINT event to be generated thus causing our rectangle to be drawn on the canvas.

```
          Case EVENT_TIMERTICK
            x=x+1
            If x>290 x=0
            RedrawGadget(MyCanvas)
```

Note that in the **DrawRect** function, we now use the integer variable x instead of a constant so that we can programatically increase the rectangle's location.

```
        Case EVENT_GADGETPAINT
          SetGraphics CanvasGraphics (MyCanvas)
          DrawRect  x,20,50,80
          Flip
```

The other powerful thing with Canvases is that we can capture mouse events as well. Lets take a look at an example where we use the mouse to move our rectangle around the canvas

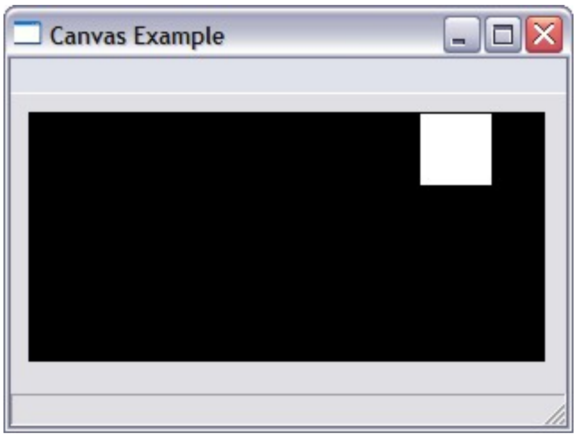
```
SuperStrict

Local MyWindow:TGadget=CreateWindow("Canvas Example", 200,200,320,240)
Local MyCanvas:TGadget=CreateCanvas(10,10,290,140,MyWindow)

Local x:Int=0
Local y:Int=0
ActivateGadget MyCanvas

Repeat
  WaitEvent()
  Select EventID()
  Case EVENT_WINDOWCLOSE
     End
  Case EVENT_MOUSEMOVE
     x=EventX()
     y=EventY()
     RedrawGadget(MyCanvas)
  Case EVENT_GADGETPAINT
    SetGraphics CanvasGraphics (MyCanvas)
    Cls
    DrawRect  x,y,40,40
    Flip
   End Select
Forever
```

You can now move the white 40x40 square around the canvas.



Let us look into more detail at our program. We first need to declare the integer variables for our rectangle's coordinates. Then we must activate our canvas gadget, otherwise our mouse events will not be generated for the canvas.

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

Once we have done the activation, whenever the mouse move over the canvas, an EVENT_MOUSEMOVE event will be generated and the coordinates of the mouse will be available via the **EventX** and **EventY** functions. Once we have updated the x,y variables, we call the **RedrawGadget** function to generate the EVENT_GADGETPAINT event which will redraw our canvas.

```
            Case EVENT_MOUSEMOVE
               x=EventX()
               y=EventY()
               RedrawGadget(MyCanvas)
```

We can improve the above program by making our cursor dissappear as follows:-

```
SuperStrict

Local MyWindow:TGadget=CreateWindow("Canvas Example", 200,200,320,240)
Local MyCanvas:TGadget=CreateCanvas(10,10,290,140,MyWindow)

Local x:Int=0
Local y:Int=0
ActivateGadget MyCanvas

Repeat
  WaitEvent()
  Select EventID()
  Case EVENT_WINDOWCLOSE
     End
  Case EVENT_MOUSEENTER
     HideMouse
 Case EVENT_MOUSELEAVE
     ShowMouse
  Case EVENT_MOUSEMOVE
     x=EventX()
     y=EventY()
     RedrawGadget(MyCanvas)
  Case EVENT_GADGETPAINT
    SetGraphics CanvasGraphics (MyCanvas)
    Cls
    DrawRect  x,y,40,40
    Flip
   End Select
Forever
```

The canvas gadget supports the following events:-

| | |
|---|---|
| EVENT_MOUSEDOWN | Mouse button pressed. Event data contains mouse button code |
| EVENT_MOUSEUP | Mouse button released. Event data contains mouse button code |
| EVENT_MOUSEMOVE | Mouse moved. Event x and y contain mouse coordinates |
| EVENT_MOUSEWHEEL | Mouse wheel spun. Event data contains delta clicks |
| EVENT_MOUSEENTER | Mouse entered gadget area |
| EVENT_MOUSELEAVE | Mouse left gadget area |
| EVENT_KEYDOWN | Key pressed. Event data contains keycode |
| EVENT_KEYUP | Key released. Event data contains keycode |
| EVENT_KEYCHAR | Key character. Event data contains unicode value |

I will leave it as an exercise to use these other events that I have not covered:)

**Summary**

The Canvas Gadget is a very powerful gadget which allows us to utilise the full power of the

Max2D functionality on its canvas. On top of the Max2D commands, you have the MaxGUI events also available  for use as well allowing for some very interesting programs.

To recap what we have learnt so far

- Canveses can be created using the **CreateCanvas** function
- We need to capture the EVENT_GADGETPAINT event to draw our canvas
- The **RedrawGadget** function must be used whenever we wish to update the drawings on our canvas.
- Timers can be used to animate our canvas via the **CreateTimer** function and then capturing the EVENT_TIMERTICK event.
- Mouse and Key events can be captured on our canvas but requires the ActivateGadget function to be called before the events can be trapped.

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