Subject: IDL Object Graphics Draw snooper
Posted by Mark Hadfield on Fri, 22 May 1998 07:00:00 GMT
View Forum Message <> Reply to Message

Here's something fairly interesting. The IDL 5.1 Object Graphics help file documents the IDLgrModel::Draw method (whereas version 5.0 didn't) and describes its purpose thus:

"The IDLgrModel::Draw procedure method draws the specified picture to the specified graphics destination. This method is provided for purposes of sub-classing only, and is intended to be called only from the Draw method of a subclass of IDLgrModel.

This sub-classing is used, for example, by the new IDLgrLegend class to allow the legend dimensions to be fitted to the text. (This has to be done when the object is drawn, because the relationship between text dimensions and normalised corrdinates depend on the destination object.) It also gives us a chance to design objects that respond dynamically to the drawing process ("Whoa! I'm being drawn to a high-resolution printer, I'd better thicken up all my lines!")

The graphics atoms, like IDLgrAxis, seem to have Draw methods too, but they are not documented, so of course I wouldn't dream of using them!

The code below defines a rudimentary class called DrawCounter--a sub-class of IDLgrModel--which appears as a rectangle but also keeps count of the number of times its Draw method is called. On my system (win32) DrawCounter::Draw is called twice when an IDLgrWindow containing the object is drawn, once when the window's select method is called (whether the object is caught by the selection or not) and 560 times (!) when the object is drawn to A4 a 600 dpi PCL printer. The number of hits goes down to 140 when the printer resolution is lowered to 300 dpi. I surmise that the view is rendered to the printer in patches of about 240x240 pixels (1 A4 page at 600 dpi = 4672 x 6774 pixels) and that each time a patch is drawn, all the objects in the view are queried to find out if they extend into the patch.

Class exercise: Design a WhackTheMole object that always appears in an area that isn't being drawn right now.

The possibilities for unproductive fiddling are limitless!

--
Mark Hadfield, m.hadfield@niwa.cri.nz http://www.niwa.cri.nz/~hadfield/
National Institute for Water and Atmospheric Research
PO Box 14-901, Wellington, New Zealand

===== Snip ==========

```
function DrawCounter::Init, LOCATION=location, COLOR=color

    if n_elements(location) eq 0 then location = [0,0]
    if n_elements(color) eq 0 then color=[0,0,0]

    if not self->IDLgrModel::Init() then return, 0

    oShape = obj_new('IDLgrPolygon', location[0]+[-0.1,0.1,0.1,-0.1],
location[1]+[-0.1,-0.1,0.1,0.1], COLOR=color)

    self->IDLgrModel::Add, oShape

    self.count = 0

    return, 1

end

pro DrawCounter::Draw, oSrcDest, oView

    self.count = self.count + 1

    self->IDLgrModel::Draw, oSrcDest, oView

end

pro DrawCounter::GetProperty, COUNT=count

    if arg_present(count) then count = self.count

end

pro DrawCounter::SetProperty, COUNT=count

    if n_elements(count) ne 0 then self.count = count

end

pro DrawCounter__define
    struct = { DrawCounter, inherits IDLgrModel, count:0L}
end



pro draw_counter

    oView = obj_new('IDLgrView', NAME='Test graph',
VIEWPLANE_RECT=[0,0,1,1])
```

```
   themodel = obj_new('IDLgrModel')

   oView->Add, themodel

   oCounter = obj_new('DrawCounter', LOCATION=[0.5,0.3])

   themodel->Add, oCounter

   oWindow = obj_new('IDLgrWindow', UNITS=2, DIMENSIONS=[10,10],
RENDERER=1)

   message, /INFORM, 'Drawing window...'

   oWindow->Draw, oView

   oCounter->GetProperty, COUNT=count
   message, /INFORM, 'DrawCounter::Draw was called for oCounter
'+strtrim(count,2)+' times'
   oCounter->SetProperty, COUNT=0

   message, /INFORM, 'Selecting...'

   objs = oWindow->Select(oView, [0.5,0.8], UNITS=3)

   oCounter->GetProperty, COUNT=count
   message, /INFORM, 'DrawCounter::Draw was called for
oCounter'+strtrim(count,2)+' times'
   oCounter->SetProperty, COUNT=0

   message, /INFORM, 'Selection returned objects:'
   print, objs

   oPrinter = obj_new('IDLgrPrinter')

   if dialog_printersetup(oPrinter) then begin
      message, /INFORM, 'Printing...
      oPrinter->Draw, oView
      oPrinter->NewDocument
      oCounter->GetProperty, COUNT=count
      message, /INFORM, 'DrawCounter::Draw was called for oCounter
'+strtrim(count,2)+' times'
      oCounter->SetProperty, COUNT=0
   end

   obj_destroy, oView
   obj_destroy, oWindow
   obj_destroy, oPrinter
```

end