


```

;resize events go here and keep the window square
'WIDGET_BASE' : object->resize, newSize = (event.x + event.y)/2
'WIDGET_TIMER':begin ;this timer rotates the ornament
    object->turnTheOrnament
    widget_control,event.top,timer=1 ;set the timer again
end

```

```

else:
end

```

```

return
end

```

```

;{{{}}}{{{}}}{{{}}}{{{}}}{{{}}}{{{}}}{{{}}}{{{}}}{{{}}}{{{}}}

```

```

pro ornamentCleanup, base
;called no matter how the GUI is destroyed
widget_control,base,get_uvalue=object
obj_destroy,object
return
end

```

```

;{{{}}}{{{}}}{{{}}}{{{}}}{{{}}}{{{}}}{{{}}}{{{}}}{{{}}}{{{}}}

```

```

pro ornament::turnTheOrnament
;rotates the ornament by 10 degrees each time around the y axis
self.oRotateModel->rotate,[0,1,0],10
self.oWindow->Draw, self.oView

```

```

return
end

```

```

;{{{}}}{{{}}}{{{}}}{{{}}}{{{}}}{{{}}}{{{}}}{{{}}}{{{}}}{{{}}}

```

```

pro ornament::resize, newSize=newSize
;resizes the draw widget and redraws the view

```

```

    widget_control,self.drawId,draw_xsize=newSize,draw_ysize=new Size
    self.oWindow->Draw, self.oView

```

```

return
end

```

```

;{{{}}}{{{}}}{{{}}}{{{}}}{{{}}}{{{}}}{{{}}}{{{}}}{{{}}}{{{}}}

```

```

function ornament::init
;initialization routine for the ornament

```

```

;initialize the super class

```

```

if (self->IDLgrModel::init(_extra=extra) ne 1) then return, 0

;Get the screen size.
device, get_screen_size = screenSize
xdim = screenSize[0]*.50 ;about 50 percent of the screen size
ydim=xdim ;keep isotropic

base=widget_base(title='Christmas Ornament',column=1,/tlb_size_events)
self.drawId=widget_draw(base, xsize=xdim, ysize=ydim,renderer=1, $
    graphics_level=2, retain=0, /expose_events)

widget_control, base, /realize
widget_control, hourglass=1
widget_control, self.drawId, get_value=oWindow
self.oWindow=oWindow

;create view object.
self.oView = obj_new('IDLgrView', projection=1, eye=2.1, $
    color=[0,0,0], view=[-1,-1,2,2], zclip=[2,-2])
self.oRotateModel = obj_new('IDLgrModel') ;need a model to rotate
;read in the earth image although any true color image will work.
read_jpeg, demo_filepath('earth.jpg', $
    subdir=['examples','demo','demodata']), earthImage, true=1

oImage1 = obj_new('IDLgrImage', earthImage)
oPoly1 = obj_new('orb', COLOR=[255, 255, 255], RADIUS=0.5, $
    DENSITY=0.8, /TEX_COORDS, TEXTURE_MAP=oImage1,style=2,hide=0,/zero)
;Invert texture coordinates, otherwise image is mirrored.
oPoly1->GetProperty, POBJ=p
p->GetProperty, TEXTURE_COORD=t
t[0,*] = 1.0 - t[0,*]
p->SetProperty, TEXTURE_COORD=t
;create the string as an ellipse
a = .1 & b=1
ap = findgen(360)*!dtr
xt = a*cos(ap)
yt= fttarr(360)
zt = -(b*sin(ap)+1.5)
oPolyLine = obj_new('IDLgrPolyline',xt,yt,zt,color=[255,255,255],thick=3 )

;add the objects to the model
self.oRotateModel->add, oPoly1
self.oRotateModel->add, oPolyLine
self.oRotateModel->rotate,[1,0,0],90

;add the rotating model to the view
self.oView -> add, self.oRotateModel

```

```

;don't want light sources to rotate so give them a separate model
oLightModel = obj_new('IDLgrModel')
oLight1 = obj_new('IDLgrLight',direction=[0,0,0],location=[0,0,1],type =2, $
    color=[255,255,255])
oLight2 = obj_new('IDLgrLight', direction=[1,0,0],location=[-1,1,0],type=1,
$
    color=[255,255,255],intensity=0.75)
oLightModel->add,oLight1
oLightModel->add,oLight2

;add the background image, define image dimensions.
;larger images make smaller lights
xsize = 256 & ysize = 256
image = bytarr(3,xsize,ysize)
;define input function parameters:
A = [ 1., 1., 5., 5., xsize/2., ysize/2.]
;create X and Y arrays:
X = findgen(xsize) # replicate(1.0, ysize)
Y = replicate(1.0, xsize) # findgen(ysize)
;create a circle:
U = ((X-A[4])/A[2])^2 + ((Y-A[5])/A[3])^2
;create gaussian Z for a blurred light
Z = bytscl(A[0] + A[1] * exp(-U/2))
ind = where(z gt 15) ;points where the blurring has a value gt 15
xPoints = ind mod xsize ;get the x and y points
yPoints = fix(ind/xsize)
;create 16 random lights
for i =0,15 do begin
    im = bytarr(xsize,ysize) ;used as a template for the lights
    xoffset = randomn(seed,1)*xsize/4. ;random positions
    yoffset = randomn(seed,1)*ysize/4.
    ;fill in the z points on a random place in im.
    im[xPoints-xoffset[0],yPoints-yoffset[0]] = z[xPoints,yPoints]
    ;spread them between the bands
    band = fix(randomu(seed,2)*3)
    ;if two bands are the same only do one. Otherwise the values exceed 255
    if band[0] ne band[1] then for j=0,1 do image[band[j],*,*] =
image[band[j],*,*] + im $
        else image[band[0],*,*] = image[band[0],*,*] + im
endfor

;background tree color (dark green)
treeColor = 90B
;find out all the green plane indices that have a color less than the
;desired tree color. We will make these all dark green
ind = where(image[1,*,*] lt treeColor)
;we have a pixel interleaved image and we only want the green indices
;to change so we have ind*3+1. If we had wanted the blue plane to change

```

```

;this would have been ind*3+2 (red is just ind*3). This is much faster than
a loop or trying to
;do it as a single plane. surprisingly, image[1,ind] = treeColor does not
work.
image[ind*3+1] = treeColor
;create the image object
oImage2 = obj_new('IDLgrImage',image)
;the polygon will fill the entire view and be placed at z=-1
face = [[0,2,0],[2,2,0],[2,0,0],[0,0,0]] - 1.0
oPoly2 = obj_new('IDLgrPolygon', face ,color=[255,255,255], $
    texture_map=oImage2, texture_coord = [[0,0], [1,0], [1,1],
[0,1]],$,
    style=2) ;texture_coord and style=2 are necessary for the image
to be seen.
oLightModel->add,oPoly2

;add some text that will not rotate
oFont = obj_new('IDLgrFont','courier')
displayText = ['M','E','R','R','Y',
',','C','H','R','I','S','T','M','A','S','!']
numDisplay = n_elements(displayText)
xPos = ((findgen(numDisplay)/(numDisplay-1.))*1.5) - 0.75 ;from -.25 to .75
yPos = fltarr(numDisplay)-0.75
zPos = fltarr(numDisplay) + 0.5
oText =
    obj_new('IDLgrText',string=displayText,location=transpose([[ xPos],[yPos],[zP
os]]), $
        color=[255,0,0],char_dim=[.15,.15],font=oFont)
oLightModel->add,oText

self.oView-> add, oLightModel

self.oWindow->Draw, self.oView
;store objects for cleanup
self.oContainer = obj_new('IDL_Container')
self.oContainer -> add, oPoly1
self.oContainer -> add, oPoly2
self.oContainer-> add,oLight1
self.oContainer-> add,oLight2
self.oContainer-> add,oImage1
self.oContainer-> add,oImage2
self.oContainer-> add,oFont
self.oContainer-> add,oText
self.oContainer-> add, oLightModel
;store the object reference in the base
widget_control, base,set_uvalue=self
widget_control,base,timer=1.0 ;timer for rotating the ornament

```

