Subject: Re: Progress and Widgets
Posted by davidf on Wed, 10 Dec 1997 08:00:00 GMT
View Forum Message <> Reply to Message

J.D. Smith (jdsmith@astrosun.tn.cornell.edu) writes:

> I was inspired by having way to much other stuff to do to rewrite
> David's routine as an object-based progress bar. In my opinion, it is
> much simpler to use.

Ooghh, this is nice! Thanks, J.D. As coincidence would have it I was just thinking about you, wondering why we haven't heard from you in awhile. I was thinking you had probably moved on to C++ land. I'm glad to see you are still with us.

I've got a couple of more old programs lying around that can use some work. Please let me know if you have any more "free" time. :-)

Cheers,

David

David Fanning, Ph.D.
Fanning Software Consulting
E-Mail: davidf@dfanning.com
Phone: 970-221-0438
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: Progress and Widgets
Posted by J.D. Smith on Wed, 10 Dec 1997 08:00:00 GMT
View Forum Message <> Reply to Message

I was inspired by having way to much other stuff to do to rewrite David's routine as an object-based progress bar. In my opinion, it is much simpler to use. One simply gets the bar with a call to obj_new, informing the bar what interval you will be updating it with (e.g. defaults to 5%), and then call Update when appropriate during your time consuming calculation/whatever. The cancel button works by evaluating whether it has been hit when each update happens... this means that if the update interval is too large, it could be slow, but it has the advantages of speed and simplicity over a real-time event driven cancel button.

Here's the code... put it in a file called showprog__define.pro. Also

attached is "testbar", a simple routine which uses the showProg object.

JD

```
***** start showProg
;+
; NAME: showProg
;
; PURPOSE: Display a progress bar
;
; CATEGORY: Objects.
;
; CALLING SEQUENCE:
;  Creating a new progress bar:
;   bar=obj_new('showProg',OVER=ov,SIZE=sz,MESSAGE=msg,
;  TITLE=ttl,CANCEL=cncl)
; Updatig the progress bar:
;  ret=bar->Update()
; Closing the progress bar (only for special case -- usually closes at
; 100% completion or cancel button hit):
;  bar->Done
;
; INPUTS:
; PERCENT: The percentage increments that will be supplied
;  (defaults to 5 -- for 20 intervals.)  The user is responsible
;  for updating (with showProg.Update) after each PERCENT% of
;  the calculation is complete.
; OVER: The widget_id of a widget for positioning the showProg bar over
; SIZE:  A vector -- the size of the bar in screen coordinates...
;  defaults to 150x15
; TITLE: The title to put on the widget. Defaults to no title bar
; MESSAGE:  A message to add to the widget.  Defaults to none.
; CANCEL: The text of the cancel button, defaults to 'Cancel'
; COLOR: The color to draw
;
; OUTPUTS: ; ret=bar->Update() :
;       the return value is 1 unless either of two things occur: The
;       bar moves to completion (100%) (ret=0), or the user clicks the
;       "cancel" button (ret=-1).  The bar is destroyed for these two
cases.
;       Acting on these cases is left up to the user.
;
; RESTRICTIONS: The user is responsible for updating the bar at the ;
;       interval he specified upon initialization (the PERCENT value
;       -- defaults to 5%), using ret=bar->Update().
;
; MODIFICATION HISTORY:
```

```
;-

 ;=============================================================
==================
; DrawBox.  Internal routine for Drawing the progress bar box
 ;=============================================================
==================
pro showProg::DrawBox
   ybox=[0,self.size[1],self.size[1],0]

right=round((1.<(float(self.complete)*self.interval/100.)>0.)*self.size[0])
   xbox=[0,0,right,right]
   oldwin=!D.WINDOW
   wset,self.barwin
   polyfill, xbox,ybox,color=self.color,/DEVICE
   wset,oldwin
   return
end

 ;=============================================================
==================
; Done - Finish the progress bar
 ;=============================================================
==================
pro showProg::Done,COMPLETE=complete
   if n_elements(complete) ne 0 then begin ;finish the bar
      self.complete=ceil(100./self.interval)
      self->DrawBox
   endif
   widget_control, widget_info(self.wDraw,/PARENT),/DESTROY
   return
end

 ;=============================================================
==================
; Update - Update the progress bar
; ret=updatebar->Update()
; the returned value is 0 if we're complete, -1 if 'Cancel' hit.
 ;=============================================================
==================
function showProg::Update
   self.complete=self.complete+1
   self->DrawBox              ;draw the updated

   ;; check if we're done
   if self.complete ge (100/self.interval) then begin
      self->Done,/COMPLETE
      return,0
```

```
   endif

   ;; grab any events from the cancel button
   event= widget_event(self.wCancel,/nowait)
   type = tag_names(event, /structure_name)
   ;; check if cancel is hit
   if type eq 'WIDGET_BUTTON' then begin
      self->Done
      return,-1
   endif

   return,1
end

 ;=============================================================
==================
; Init - Initialize showProg bar.
;  updatebar=obj_new('showProg',PERCENT=per,OVER=ov,SIZE=sz,MES SAGE=msg,
;  TITLE=ttl,CANCEL=cncl)
; PERCENT: The percentage increments that will be supplied
;  (defaults to 5 -- for 20 intervals.)  The user is responsible
;  for updating (with showProg.Update) after each PERCENT% of
;  the calculation is complete.
; OVER: The widget_id of a widget for positioning the showProg bar over
; SIZE:  A vector -- the size of the bar in screen coordinates...
;  defaults to 100x10
; TITLE: The title to put on the widget. Defaults to no title bar
; MESSAGE:  A message to add to the widget.  Defaults to none.
; CANCEL: The text of the cancel button, defaults to 'Cancel'
; COLOR: The color to draw
 ;=============================================================
==================
function showProg::Init,PERCENT=percent,OVER=over,MESSAGE=msg, SIZE=sz,
$
                TITLE=ttl, CANCEL=cncl, COLOR=clr
   if n_elements(sz) eq 2 then self.size=sz else self.size=[150,15]
   if n_elements(clr) ne 0 then self.color=clr else
self.color=!D.N_COLORS/2
   if n_elements(cncl) eq 0 then cncl='Cancel'
   if n_elements(percent) eq 0 then self.interval=5 else    $
    self.interval=percent       ;percent interval of each

   device, get_screen_size=ss
   if n_elements(over) ne 0 then begin
      g=widget_info(over, /GEOMETRY)
      xoff=ss[0]<(g.xoffset+g.xsize-ss[0]/20)>0
      yoff=ss[0]<(g.yoffset+g.ysize-ss[1]/20)>0
   endif else begin
```

```
    xoff=ss[0]/2. & yoff=ss[1]/2.
  endelse

  if n_elements(ttl) eq 0 then begin
    ttl='' & atr=5
  endif else atr=1            ;no title


 base=Widget_Base(/column,/Base_Align_Center,xoff=xoff,yoff=y off,TITLE=ttl,
$
                TLB_FRAME_ATTR=atr)
  if n_elements(msg) ne 0 then lbl=widget_label(base,value=msg)

   self.wDraw=widget_draw(base,xsize=self.size[0],ysize=self.si ze[1])
  self.wCancel=widget_button(base,value=cncl)

  widget_control, base,/realize
  Widget_Control, self.wDraw, get_value=win
  self.barwin=win
  return,1
end

pro showProg__define
  struct={showProg, $
        interval:0., $       ;intervals (%) showProg will be updated
        complete:0, $        ;the number of intervals completed
        size:[0,0], $       ;size in screen units of bar
        wDraw:0L, $          ;id of draw widget
        wCancel:0L, $        ;id of cancel button
        barwin:0,$           ;window id
        color:0} ;color of the progress bar
  return
end

******* end showProg

******* start testbar

pro testbar,_EXTRA=e,PERCENT=percent
  if n_elements(percent) eq 0 then percent=5

  ;; set up some fake widget

 base=widget_base(xsize=300,ysize=300,xoffset=200,yoff=100,TI TLE='DUMB_BASE')
  label=widget_label(base,
value='',yoffset=120,xoffset=120,/dynamic_resize, $
                FONT='-adobe-times-medium-r-normal--34-240-' + $
                '100-100-p-170-iso8859-1')
```

```
  widget_control, base,/realize


  pbar=obj_new('showProg',_EXTRA=e,OVER=base,PERCENT=percent)

  i=0 & update=1
  repeat begin
    wait,.1                ;simulate a calculation
    i=i+1
    widget_control, label,set_value=strtrim(i,2)
    if i mod percent eq 0 then update=pbar->Update() ;update every
percent%

    ;; make sure we have only 100 iterations....
    if i eq 100 then begin
      update=0
      ;; finish up -- only needed here if percent is not a factor of
100
      if 100 mod percent ne 0 then pbar->Done
    endif
  endrep until update ne 1
  widget_control, base,/destroy
  return
end

********  end testbar
```

Subject: Re: Progress and Widgets
Posted by davidf on Wed, 10 Dec 1997 08:00:00 GMT

View Forum Message <> Reply to Message

Matthew Hanson (matt@ktaadn.com) writes:

> I am currently working on a program which has substantial computation
> and wait times (incidentally, the object graphics seem to take an
> unusually long time to render. . .).

No c-o-m-m-e-n-t.

> I want to be able to show progress and was wondering if anyone knows how
> to do the following things:
> 1) Change the pointer/mouse to an hourglass or similar 'hold on' icon.
> I am using IDL for UNIX.

Widget_Control, /Hourglass

> 2) Show a progress bar.

I put an example of this on my anonymous ftp site for you. The
file is named showprogress.pro. Compile it and then type "test"
to see it in action. You can find it at:

ftp://ftp.dfanning.com/pub/dfanning/outgoing/miscellaneous/

> 3) Have a modal message widget come up, but with no user buttons, and
> some text. Basically it needs an ID so i can destroy it later in the
> program. Is the best way to do this just make a window?

See the code above. I like to make it a function that returns the
ID value when its called.

Cheers,

David

David Fanning, Ph.D.
Fanning Software Consulting
E-Mail: davidf@dfanning.com
Phone: 970-221-0438
Coyote's Guide to IDL Programming: http://www.dfanning.com/