Subject: Re: Q: A good way to interrupt and kill procedures?
Posted by J.D. Smith on Fri, 22 May 1998 07:00:00 GMT
View Forum Message <> Reply to Message

Nancy Levenson wrote:
>
> I am looking for a sensible way to set up a time-consuming program so
> it can be interrupted.
>
> The time-consuming tasks are image manipulations for astronomical data
> reduction. They are run with a widget. Ideally, I would like to have
> a 'kill process' button whose events could be recognized while the
> data reduction is in progress.
>
> Any suggestions?
>
> Nancy Levenson
> Department of Astronomy
> UC Berkeley
> levenson@astro.berkeley.edu

If your calculation is encoded in some form of loop to which you have access, such that, at any given iteration through the loop, you can figure out what percent complete the calculation is, then you can use the showProg object class (based on a D. Fanning idea) I posted previously here for the purpose of cancelling *and* giving your user an idea of how much time remains. Note that the process can only be cancelled every PERCENT%, which you can choose, and that it is your responsibility to determine when the calculation is PERCENT% complete, for updating the progress bar.

An example using showProg, testbar.pro, is also attached.

Good Luck,

JD

**********BEGIN showprog__define.pro**********
```
;+
; NAME: showProg
;
; PURPOSE: Display a progress bar, and allow user cancellation.
;
; CATEGORY: Objects.  Progress indication.
;
; CALLING SEQUENCE:
;  Creating a new progress bar:
;   bar=obj_new('showProg',OVER=ov,SIZE=sz,MESSAGE=msg,
```

```
;  TITLE=ttl,CANCEL=cncl)
; Updating the progress bar:
;  ret=bar->Update()
; Closing the progress bar.  Used only for special cases, such as after
;       incorrectly estimating the percent completion. ShowProg 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 2 element 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().
;-


 ;=============================================================
=================
; 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've done enough intervals
   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 bar if no title present

    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__define.pro**********************************

**********************BEGIN testbar.pro*****************************************
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)
  widget_control, base,/realize

   pbar=obj_new('showProg',_EXTRA=e,OVER=base,MESSAGE='CALCULAT ING...', $
            PERCENT=percent,CANCEL='Abort Calculation')

  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 some factor of 100
      if 100 mod percent ne 0 then pbar->Done
    endif
    if update eq -1 then  $
     tmp=dialog_message('Calculation Cancelled',/ERROR, $
                  DIALOG_PARENT=base, TITLE='This Calculation')
  endrep until update ne 1
  widget_control, base,/destroy
  return
end
**************************END testbar.pro*************************************
```

--

J.D. Smith |*| WORK: (607) 255-5842
Cornell University Dept. of Astronomy |*| (607) 255-4083
206 Space Sciences Bldg. |*| FAX: (607) 255-5875
Ithaca, NY 14853 |*|