Subject: Re: IDL Objects Graphics cache and crash
Posted by Karl Schultz on Tue, 12 Mar 2002 16:30:53 GMT
View Forum Message <> Reply to Message

"Karl Schultz" <karl_schultz@yahoo.com> wrote in message
news:e415b359.0203111711.159d69ee@posting.google.com...
> I have a little test program that draws three parallel polygons with
> transparency. The program uses a trackball and changes the order of
> the polygon drawing as the model "flips" from one side to another so
> that the transparency always looks right. The problem is pretty easy
> to solve for simple data like this, but gets really messy for more
> complex scenes. I'll clean up the program and send it to anyone who
> is interested.

Here's the program. It is pretty simple, but should show the general idea.

```
;;
;; Simple opacity and rendering order demonstration
;;
;; The small blue square is opaque. The red and green squares are
translucent.
;; Rotate the scene to observe that you can always see the other two squares
;; through the front square. This is accomplished by changing the drawing
order
;; of the squares as the trackball transform changes their "stacking" order
;; relative to the viewer.
;;
;; Run with /NOFLIP to observe what happens when the drawing order is not
changed.
;;
;; S. Houston / K. Schultz RSI
;;

PRO OPACITY_TEST_EVENT, sEvent

   widget_control, sEvent.id, GET_UVALUE=uval

   ; Handle KILL requests.
   IF TAG_NAMES(sEvent, /STRUCTURE_NAME) EQ 'WIDGET_KILL_REQUEST' THEN
BEGIN
      WIDGET_CONTROL, sEvent.top, GET_UVALUE=sState

      ; Destroy the objects.
      OBJ_DESTROY, sState.oView
      OBJ_DESTROY, sState.oTrack
      WIDGET_CONTROL, sEvent.top, /DESTROY
      RETURN
   ENDIF
```

```
   ; Handle other events
   CASE uval OF
      'DRAW': BEGIN
         widget_control, sEvent.top, GET_UVALUE=sState, /NO_COPY

         ; expose
         IF (sEvent.type EQ 4) THEN BEGIN
            sState.oWindow->Draw, sState.oView
            widget_control, sEvent.top, SET_UVALUE=sState, /NO_COPY
            RETURN
         ENDIF

         ; Trackball updates
         bHaveTransform = sState.oTrack->Update(sEvent, TRANSFORM=qmat)
         if (bHaveTransform NE 0) THEN BEGIN
            sState.oTopZPlus->GetProperty, TRANSFORM=t
            t = t # qmat
            sState.oTopZPlus->SetProperty, TRANSFORM=t
            sState.oTopZMinus->SetProperty, TRANSFORM=t
            print, "Z direction of trackball:", t[2,2]
            ;;
            ;; If the Z direction of the trackball is positive, hide
            ;; the model that draws the polygons sorted from +Z to -Z
            ;; and draw the model that draws the polygons sorted from -Z
to +Z.
            ;; Vice-versa for a trackball Z direction that is negative.
            if not sState.noflip then begin
               if t[2,2] ge 0 then begin
                  print, "Draw green square before red square"
                  sState.oTopZPlus->SetProperty, HIDE=0
                  sState.oTopZMinus->SetProperty, HIDE=1
               endif else begin
                  print, "Draw red square before green square"
                  sState.oTopZPlus->SetProperty, HIDE=1
                  sState.oTopZMinus->SetProperty, HIDE=0
               endelse
            endif else begin
               print, "Draw green square before red square"
            endelse
            sState.oWindow->Draw, sState.oView
         ENDIF

         ; button press
         IF (sEvent.type EQ 0) THEN BEGIN
            sState.btndown = 1b
            widget_control, sState.wDraw, /DRAW_MOTION
            sState.oWindow->Draw, sState.oView
```

```
        ENDIF

        ; button release
        IF (sEvent.type EQ 1) THEN BEGIN
          IF (sState.btndown EQ 1b) THEN $
            sState.oWindow->Draw, sState.oView
          sState.btndown = 0b
          widget_control, sState.wDraw, DRAW_MOTION=0
        ENDIF

        WIDGET_CONTROL, sEvent.top, SET_UVALUE=sState, /NO_COPY
      END
  ENDCASE
END

;  ------------------------------------------------------------ -------------
-
pro opacity_test, NOFLIP=noflip

  noflip = KEYWORD_SET(noflip)

  xdim = 512
  ydim = 512

  ; Create the widgets
  wBase = WIDGET_BASE(/COLUMN, XPAD=0, YPAD=0, $
                      TITLE="Opacity Test", $
                      /TLB_KILL_REQUEST_EVENTS)

  wDraw = WIDGET_DRAW(wBase, XSIZE=xdim, YSIZE=ydim, UVALUE='DRAW', $
                      RETAIN=0, /EXPOSE_EVENTS, /BUTTON_EVENTS, $
                      GRAPHICS_LEVEL=2)

  ; Realize the widgets
  widget_control, wBase, /REALIZE

  ; Get the is of the drawable
  widget_control, wDraw, GET_VALUE=oWindow

  oView = OBJ_NEW('IDLgrView', COLOR=[255,255,255], PROJECTION=2)
  oTopZPlus = OBJ_NEW('IDLgrModel')
  oTopZMinus = OBJ_NEW('IDLgrModel')
  oTrack = OBJ_NEW('Trackball', [xdim/2, ydim/2], xdim/2)


  ;; Red transparent polygon
  texData = BYTARR(4,64,64)
  texData[0,*,*] = 255
```

```
   texData[1,*,*] = 0
   texData[2,*,*] = 0
   texData[3,*,*] = 64

   oImage1 = OBJ_NEW('IDLgrImage', texData, INTERLEAVE=0)
   oPoly1 = OBJ_NEW('IDLgrPolygon', [-0.5, 0.5, 0.5, -0.5], $
                              [-0.5, -0.5, 0.5, 0.5], $
                              [0.5, 0.5, 0.5, 0.5], $
                              COLOR=[255,255,255], $

TEXTURE_COORD=[[0,0],[1,0],[1,1],[0,1]], $
                              TEXTURE_MAP=oImage1)
   oModel1 = OBJ_NEW('IDLgrModel')
   oModel1->Add, oPoly1

   ;; Green transparent polygon
   texData[0,*,*] = 0
   texData[1,*,*] = 255
   texData[2,*,*] = 0
   texData[3,*,*] = 64

   oImage2 = OBJ_NEW('IDLgrImage', texData, INTERLEAVE=0)
   oPoly2 = OBJ_NEW('IDLgrPolygon', [-0.5, 0.5, 0.5, -0.5], $
                              [-0.5, -0.5, 0.5, 0.5], $
                              [-0.5, -0.5, -0.5, -0.5], $
                              COLOR=[255,255,255], $

TEXTURE_COORD=[[0,0],[1,0],[1,1],[0,1]], $
                              TEXTURE_MAP=oImage2)
   oModel2 = OBJ_NEW('IDLgrModel')
   oModel2->Add, oPoly2

   ;; Blue opaque polygon
   oPoly3 = OBJ_NEW('IDLgrPolygon', $
                              [-0.2, 0.2, 0.2, -0.2], $
                              [-0.2, -0.2, 0.2, 0.2], $
                              COLOR=[0,0,255])
   oModel3 = OBJ_NEW('IDLgrModel')
   oModel3->Add, oPoly3


   oTopZPlus->Add, oModel3 ; Non transparent first
   oTopZPlus->Add, oModel2 ; Green (back) next
   oTopZPlus->Add, oModel1 ; Red (front) last
   oView->Add, oTopZPlus
   oTopZMinus->Add, oModel3, /ALIAS ; Non transparent first
   oTopZMinus->Add, oModel1, /ALIAS ; Red (front, but back when rotated)
next
```

```
    oTopZMinus->Add, oModel2, /ALIAS ; Green (back, but front when rotated)
last
    oView->Add, oTopZMinus

    ; Hide the back-oriented model
    oTopZMinus->SetProperty, HIDE=1

    sState = {  btndown: 0b, $
                wDraw: wDraw, $
                oWindow: oWindow, $
                oView: oView, $
                oTopZPlus: oTopZPlus, $
                oTopZMinus: oTopZMinus, $
                oTrack: oTrack, $
                noflip: noflip }

    widget_control, wBase, SET_UVALUE=sState, /NO_COPY
    xmanager, 'opacity_test', wBase, /NO_BLOCK

end
```