graphics3d_l

COLLABORATORS			
	TITLE : graphics3d_I		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		October 9, 2022	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

graphics3d_I

1.1 graphics3d_I.doc

graphics3d.library

GD_addobjpoly()

GD_addobjvertex()

GD_ambientlight()

GD_aspectratio()

GD_cascene()

GD_cattpoly()

GD_changeviewmode()

GD_changeviewmodeobj()

GD_clipbox()

GD_clipmode()

GD_close_display3d()

GD_colldetect()

GD_createlightsource()

GD_deleteobject()

GD_dfl2fix()

GD_display3d()

GD_fix2dfl()

GD_fix2int()

GD_fix2sfl()

GD_frustum()

GD_genpalette()

GD_getobj()

GD_int2fix()

GD_loadobject()

GD_modobj()

GD_modpoly()

GD_moveforward()

GD_newobj()

GD_newtmap()

GD_newtmapf()

GD_newview()

GD_over()

GD_paintframe()

GD_pickobj()

GD_positioncamera()

GD_positionobject()

GD_recalcobj()

GD_rmtmap()

GD_rotateobject()

GD_scaleobject()

GD_setobj()

GD_sfl2fix()

GD_switch_rp()

GD_touchpalette()

GD_translateobject()

GD_viewangle()

1.2 graphics3d.library/GD_display3d()

```
                NAME
     GD_display3d  --  Inizializza tutto l'ambiente necessario per la libreria.

SYNOPSIS
     ambient3d=GD_display3d(win, x0, y0, scrw, scrh, vdist)
                            A0   D0  D1  D2    D3    D4
     struct ambient3d *GD_display3d(struct Window*,LONG,LONG,LONG,LONG,LONG);

FUNCTION
     Crea ed inizializza la struttura ambient3d che serve da descrittore
     della scena 3d ed e' usata come input da TUTTE le altre.

INPUTS
     win   = puntatore a struttura Window della finestra su cui si vuol
             visualizzare la scena 3d.
     x0,y0 = coordinate vertice in alto a sinistra del box che definisce
             i limiti di visualizzazione della scena.
     scrw  = larghezza di tale box (deve essere multiplo di 16) max.3000.
             E' anche usato come massimo valore X per il box di visualizzazione ↩
                 .
     scrh  = altezza di tale box max.3000.
             E' anche usato come massimo valore Y per il box di visualizzazione ↩
                 .
     vdist = distanza tra osservatore e piano di proiezione scena 3d va
             espresso come intero.

RESULT
     ambient3d = puntatore a struttura ambient3d se uguale a 0 allora
                 si e' verificato un errore e l'inizializzazione e'
                 fallita.

BUGS
     nessuno noto, se ne trovate segnalatemeli.

NOTES
     questa funzione va sempre usata PRIMA di poter usare tutte le altre.
     Pero' e' possibile usarla piu' volte all'interno dello stesso
     programma e memorizzandone separatamente i risultati, si puo' agire
     contemporaneamente e indipendentemente su tutte le scene cosi
     definite.
     In futuro, forse rendero' possibile creare piu' scene indipendenti
     dello stesso spazio 3D (ad esempio per creare piu' viste) mantenendo
     comuni le aree di memoria per gestire gli oggetti. Adesso si rialloca
     sempre tutte le aree e se gli oggetti sono molti o complessi la
     memoria usata e' parecchia.

SEE ALSO

              GD_close_display3d
```

1.3 graphics3d.library/GD_close_display3d()

```
                NAME
     GD_close_display3d  --  elimina tutto cio' che riguarda la scena 3d  ↩
        visualizzata.

SYNOPSIS
     GD_close_display3d(in)
                        A0
     void GD_close_display3d(struct ambient3d *);

FUNCTION
     elimina tutto cio' che era stato aperto e definito con
             GD_display3d
                     inclusi tutti gli eventuali oggetti inseriti in quella  ↩
                        scena.

INPUTS
     in = puntatore a struttura ambient3d che si vuol eliminare.
          Se questo puntatore e' 0 allora non fa nulla.

RESULT

BUGS
     nessuno noto, se ne trovate segnalatemeli.

NOTES
     da usarsi tipicamente al termine del programma per eliminare tutto
     quello che riguarda questa libreria.

SEE ALSO

             GD_display3d
```

1.4 graphics3d.library/GD_changeviewmode()

```
                NAME
     GD_changeviewmode  --  cambia il modo di visualizzare tutti gli oggetti

SYNOPSIS
     esi=GD_changeviewmode(in, modo, b_col)
                           A0  D0    D1
     LONG GD_changeviewmode(struct ambient3d *,LONG,LONG);

FUNCTION
     cambia in una volta sola il modo di visualizzazione di tutti gli
     oggetti definiti nella scena3d, per adesso si puo' visualizzare
     in wire frame , solid e flat.

INPUTS
     in    = puntatore a struttura ambient3d della scena 3d su cui
             operare.
             Deve essere maggiore di 0 altrimenti esito indefinito.
```

```
     modo  = nuovo modo di visualizzazione : vedi
             GD_modobj()
             .
     b_col = n# registro colore da usare per il bordo dei poligoni
             degli oggetti.

RESULT
     esi = esito operazione , se diverso da 0 tutto ok altrimenti errore
           operazione non eseguita.

BUGS
     nessuno noto, se ne trovate segnalatemeli.

NOTES

SEE ALSO

             GD_changeviewmodeobj
```

1.5 graphics3d.library/GD_changeviewmodeobj()

```
                     OBSOLETA - usare
             GD_modobj()
              .

NAME
     GD_changeviewmodeobj  --  cambia il modo di visualizzare dell'oggetto  ↩
        selezionato

SYNOPSIS
     esi=GD_changeviewmodeobj(in, modo)
                              A0  D0
     LONG GD_changeviewmodeobj(struct ambient3d *,LONG);

FUNCTION
     cambia il modo di visualizzazione dell'oggetto attualmente
     selezionato nella scena 3d considerata.

INPUTS
     in    = puntatore a struttura ambient3d della scena 3d su cui
             operare.
             Deve essere maggiore di 0 altrimenti esito indefinito.
     modo  = nuovo modo di visualizzazione :
             0 -> wireframe          (usare la macro WIREF)
             1 -> ombreggiatura flat (usare la macro FLAT)
             2 -> solido             (usare la macro SOLID)
             3 -> sfumato (goraud)   (usare la macro GORAUD)

RESULT
     esi = esito operazione , se diverso da 0 tutto ok altrimenti errore
           operazione non eseguita.

BUGS
     in realta' non e' stata ancora testata ma dovrebbe funzionare.
```

```
NOTES

SEE ALSO

             GD_changeviewmode
```

1.6 graphics3d.library/GD_touchpalette()

```
NAME
     GD_touchpalette  --  crea una palette sfumata di colori

SYNOPSIS
     GD_touchpalette(in, fr, lr, init_color, lastcolor)
                     A0  D0  D1  A1          A2
     void GD_touchpalette(struct ambient3d *,LONG,LONG,struct rgbtype *,struct  ↩
        rgbtype *);

FUNCTION
     crea una palette sfumata tra due registri colore im modo da poter
     usare correttamente il modo di visualizzazione flat shading.

INPUTS
     in         = puntatore a struttura ambient3d della scena 3d su cui
                  operare.
                  Deve essere maggiore di 0 altrimenti esito indefinito.
     fr         = primo registro colore da settare.
     lr         = ultimo registro colore da settare.
     init_color = puntatore a struttura rgbtype con valore RGB iniziale
                  del colore.
     lastcolor  = puntatore a struttura rgbtype con valore RGB iniziale
                  del colore.

RESULT

BUGS
     nessuno noto, se ne trovate segnalatemeli.

NOTES
     tenete presente che il colore che si assegna all'oggetto sara' usato
     come riferimento a fr dal flat shading e sara' la sfumatura piu'
     scura ,lr la piu' chiara.
     Va usata sempre poiche' il solid shading usa la sfumatura centrale
     come colore di visualizzazione dei poligoni, usando il colore
     dell'oggetto come riferimento a fr.
     Ora ci sono 2 campi di valori per RGB init_color e RGB last color :
     1- la componente RGB e' nel campo tra 0 e 15 quindi si puo' usare
        con macchine con il chipset ECS e S.O. <3.0 .
     2- la componente RGB e' nel campo tra 0 e 255 allora si DEVE usare
        con macchine con almeno il chipset AGA e S.O. >=3.0 .

SEE ALSO
```

1.7 graphics3d.library/GD_moveforward()

```
                NAME
     GD_moveforward  --  sposta l'osservatore

SYNOPSIS
     GD_moveforward(in, dist)
                    A0  D0
     void GD_moveforward(struct ambient3d *,LONG);

FUNCTION
     sposta l'osservatore di dist unita' nella direzione del punto di
     vista.

INPUTS
     in   = puntatore a struttura ambient3d della scena 3d su cui operare.
            Deve essere maggiore di 0 altrimenti esito indefinito.
     dist = n# di unita' di spostamento osservatore, puo' essere negativo
            (spostamento all'indietro) ma deve essere in FIX POINT ovvero:
            valore intero * 256 (o meglio * FIXV) +
            valore frazionario che sara' considerato in 256 esimi
            (o meglio in FIXV esimi).

RESULT

BUGS
     nessuno noto, se ne trovate segnalatemeli.

NOTES
     I valori in fix point sono cosi composti :
     valore intero*moltiplicatore + valore frazionario.
     dove moltiplicatore e' la posizione della virgola ed e' sempre
     uguale ,in questa libreria e' = 256 (usare la macro FIXV).
     es.: il numero in floating point 10.2 espresso in fix point sara' :
             parte_intera * moltiplicatore +
             moltiplicatore / inverso_parte_frazionaria
             e cioe con moltiplicatore uguale a 256 (macro FIXV):
             (10 * 256) + (256 / (1/0.2)) = 2611
     ovvero anche :
             numero float * moltiplicatore -> 10.2 * 256 = 2611

SEE ALSO

             GD_viewangle
             ,
             GD_positioncamera
```

1.8 graphics3d.library/GD_viewangle()

```
                NAME
     GD_viewangle  --  gira l'osservatore

SYNOPSIS
```

```
     GD_viewangle(in ,ax ,ay ,az )
                  A0  D0  D1  D2
     void GD_viewangle(struct ambient3d *,LONG,LONG,LONG);

FUNCTION
     permette di variare l'angolo di visuale dell'osservatore.

INPUTS
     in = puntatore a struttura ambient3d della scena 3d su cui operare.
          Deve essere maggiore di 0 altrimenti esito indefinito.
     ax = valore rotazione su asse X osservatore.
          Va espresso in valori interi (non fixpoint) ed e' considerato
          in gradi sessagesimali (0-90-180-360).
     ay = valore rotazione su asse Y osservatore.
          Va espresso in valori interi (non fixpoint) ed e' considerato
          in gradi sessagesimali (0-90-180-360).
     az = valore rotazione su asse Z osservatore.
          Va espresso in valori interi (non fixpoint) ed e' considerato
          in gradi sessagesimali (0-90-180-360).

RESULT

BUGS
     nessuno noto, se ne trovate segnalatemeli.

NOTES

SEE ALSO

             GD_moveforward
             ,
             GD_positioncamera
```

1.9 graphics3d.library/GD_frustum()

```
                NAME
     GD_frustum  --  setta i piani che delimitano lo spazio visibile

SYNOPSIS
     GD_frustum(in ,near ,far )
                A0  D0    D1
     void GD_frustum(struct ambient3d *,LONG,LONG);

FUNCTION
     setta la distanza sull'asse Z dell'osservatore dei piani ad esso
     perpendicolari e che ne delimitano il campo visivo.

INPUTS
     in   = puntatore a struttura ambient3d della scena 3d su cui operare.
            Deve essere maggiore di 0 altrimenti esito indefinito.
     near = valore intero (non fixpoint) distanza piano che segnala
            inizio campo visivo dello spazio definito.
     far  = valore intero (non fixpoint) distanza piano che segnala
            fine campo visivo dello spazio definito.
```

```
RESULT

BUGS
     nessuno noto, se ne trovate segnalatemeli.

NOTES

SEE ALSO

             GD_clipmode
```

1.10 graphics3d.library/GD_createlightsource()

```
                NAME
     GD_createlightsource  --  posiziona la fonte di luce nello spazio

SYNOPSIS
     GD_createlightsource(in ,x ,y ,z )
                          A0  D0 D1 D2
     void GD_createlightsource(struct ambient3d *,LONG,LONG,LONG);

FUNCTION
     crea e posiziona una fonte di luce nello spazio.

INPUTS
     in = puntatore a struttura ambient3d della scena 3d su cui operare.
          Deve essere maggiore di 0 altrimenti esito indefinito.
     x  = coordinata X della luce rispetto all' origine dello spazio.
          Valore espresso in fix point (vedi notes di
             GD_moveforward
             ).
     y  = coordinata Y della luce rispetto all' origine dello spazio.
          Valore espresso in fix point (vedi notes di
             GD_moveforward
             ).
     z  = coordinata Z della luce rispetto all' origine dello spazio.
          Valore espresso in fix point (vedi notes di
             GD_moveforward
             ).

RESULT

BUGS
     temo che non posizioni correttamente la luce, devo verificarla
     meglio.

NOTES
     in realta' questa funzione posiziona soltanto una luce non la crea,
     visto che se ne puo' avere solo una ed e' presente dal momento in
     cui si usa
             GD_display3d
              questo perche' altrimenti il flat shading
     non potrebbe funzionare subito.
```

```
     In futuro puo' darsi che permetta di gestirne piu' di una e in tal
     caso la creera anche .

SEE ALSO

             GD_ambientlight
```

1.11 graphics3d.library/GD_ambientlight()

```
                NAME
     GD_ambientlight  --  setta l'intensita' della luce ambientale

SYNOPSIS
     GD_ambientlight(in ,inte )
                     A0  D0
     void GD_ambientlight(struct ambient3d *,LONG);

FUNCTION
     setta l'intensita' della luce ambientale, non il colore .

INPUTS
     in   = puntatore a struttura ambient3d della scena 3d su cui operare.
            Deve essere maggiore di 0 altrimenti esito indefinito.
     inte = valore intensita' luce espresso in fix point(vedi notes di

             GD_moveforward
             ).

RESULT

BUGS

NOTES
     questa funzione in pratica ha effetto solo se si usa il flat shading
     almeno per adesso.

SEE ALSO

             GD_createlightsource
```

1.12 graphics3d.library/GD_positioncamera()

```
                NAME
     GD_positioncamera  --  posiziona l'osservatore

SYNOPSIS
     GD_positioncamera(in ,x ,y ,z )
                       A0  D0 D1 D2
     void GD_positioncamera(struct ambient3d *,LONG,LONG,LONG);
```

```
FUNCTION
     posiziona l'osservatore rispetto all'origine dello spazio.

INPUTS
     in = puntatore a struttura ambient3d della scena 3d su cui operare.
          Deve essere maggiore di 0 altrimenti esito indefinito.
     x  = coordinata X dell'osservatore rispetto all' origine dello
          spazio.
          Valore espresso in fix point (vedi notes di
             GD_moveforward
             ).
     y  = coordinata Y dell'osservatore rispetto all' origine dello
          spazio.
          Valore espresso in fix point (vedi notes di
             GD_moveforward
             ).
     z  = coordinata Z dell'osservatore rispetto all' origine dello
          spazio.
          Valore espresso in fix point (vedi notes di
             GD_moveforward
             ).

RESULT

BUGS
     nessuno noto, se ne trovate segnalatemeli.

NOTES

SEE ALSO

             GD_moveforward
             ,
             GD_viewangle
```

1.13 graphics3d.library/GD_aspectratio()

```
                NAME
     GD_aspectratio  --  varia aspect ratio

SYNOPSIS
     GD_aspectratio(in ,ratio )
                    A0  D0
     void GD_aspectratio(struct ambient3d *,LONG);

FUNCTION
     varia l'aspect ratio della scena visualizzata ,di default e' uguale
     a 1:1.

INPUTS
     in    = puntatore a struttura ambient3d della scena 3d su cui
             operare. Deve essere maggiore di 0 altrimenti esito
             indefinito.
     ratio = valore nuovo aspect ratio cosi espresso, es.: se 1:2 allora
```

```
              e' uguale a 0.5.
              Valore espresso in fix point (vedi notes di
              GD_moveforward
              ).

RESULT

BUGS
     nessuno noto, se ne trovate segnalatemeli.

NOTES

SEE ALSO
```

1.14 graphics3d.library/GD_clipmode()

```
                NAME
     GD_clipmode  --  setta un particolare clip mode

SYNOPSIS
     GD_clipmode(in ,mode )
                 A0  D0
     void GD_clipmode(struct ambient3d *,LONG);

FUNCTION
     setta il modo di clippaggio degli oggetti nello spazio, tra i due
     disponibili :
     ZETA PLANE : per clippare l'oggetto ne considera il bounding box
                  solo sull'asse Z rispetto ai piani near e far.
     FRUSTUM    : per clippare l'oggetto ne considera il boundig box
                  su tutti e 3 gli assi , la Z rispetto ai piani near e
                  far, la X e la Y rispetto ai limiti massimi del box di
                  visualizzazione.

INPUTS
     in   = puntatore a struttura ambient3d della scena 3d su cui operare.
            Deve essere maggiore di 0 altrimenti esito indefinito.
     mode = valore intero nuovo clip mode :
             0 - ZETA PLANE (usare macro ZPLANE)
             1 - FRUSTUM    (usare macro FRUSTUM)

RESULT

BUGS
     nessuno noto, se ne trovate segnalatemeli.

NOTES

SEE ALSO

              GD_frustum
```

1.15 graphics3d.library/GD_pickobj()

```
                NAME
     GD_pickobj  --  dato un punto ne identifica poligono e oggetto

SYNOPSIS
     idobj=GD_pickobj(in ,np ,x ,y )
                      A0  A1  D0 D1
     LONG GD_pickobj(struct ambient3d *,LONG *,LONG,LONG);

FUNCTION
     dato un punto rispetto alla finestra di visualizzazione (2D quindi)
     identifica all'interno di quale poligono e di quale oggetto si trova
     tra quelli visibili in quel momento ,cioe se si tiene fisso il punto
     ma si varia la vista puo' cambiare il risultato.

INPUTS
     in = puntatore a struttura ambient3d della scena 3d su cui operare.
          Deve essere maggiore di 0 altrimenti esito indefinito.
     np = puntatore ad intero dove mettere n# poligono trovato.
          Sara' un valore intero che parte da 0 ma avra senso solo se il
          risultato e maggiore o uguale a 0.
     x  = valore intero(non fix point) coordinata X rispetto alla finestra
          di visualizzazione della scena 3d.
     y  = valore intero(non fix point) coordinata Y rispetto alla finestra
          di visualizzazione della scena 3d.

RESULT
     valore intero(non fix point) con identificativo univoco dell'oggetto
     nel cui interno si trova il punto dato.
     Se uguale a 0 il punto e' all'esterno di tutti gli oggetti
     attualmente visibili.

BUGS
     non e' infallibile poiche' non trova tutti i punti all'interno di un
     poligono.

NOTES
     l'algoritmo usato e' del tutto empirico per motivi di velocita' e
     non riesce ad individuare tutti i punti all'interno di un poligono
     ma sicuramente i punti che trova sono dentro il poligono segnalato.
     Se qualcuno conosce algoritmi piu' affidabili e me li puo' spiegare
     sara' il benvenuto.
     Per informazione tale algoritmo deve poter individuare se un punto
     e' o no all'interno di un triangolo o quadrangolo(questo non e'
     essenziale) ma deve farlo nel modo piu' veloce possibile, poiche'
     puo' doverne esaminare a centinaia.

SEE ALSO

               GD_setobj
               ,
               GD_getobj
```

1.16 graphics3d.library/GD_newobj()

```
                NAME
     GD_newobj  --  crea un nuovo oggetto

SYNOPSIS
     esi=GD_newobj(in ,name ,pol ,vert)
                   A0  A2    D0   D1
     LONG GD_newobj(struct ambient3d *,char *,LONG,LONG);

FUNCTION
     crea ed inizializza le aree di memoria per generare un nuovo oggetto
     lo posiziona sull'origine degli assi dello spazio e lo fa diventare
     l' attualmente selezionato.

INPUTS
     in   = puntatore a struttura ambient3d della scena 3d su cui operare.
            Deve essere maggiore di 0 altrimenti esito indefinito.
     name = puntatore a stringa (terminata da 0x00) con nome oggetto.
     pol  = valore intero indicante il n# totale di poligoni da assegnare
            all'oggetto.
     vert = valore intero indicante il n# totale di vertici da assegnare
            all'oggetto.

RESULT
     se esi uguale a 0 operazione fallita, altrimenti tutto ok e il valore
     ritornato e' l'identificativo (univoco) dell'oggetto creato.

BUGS
     nessuno noto, se ne trovate segnalatemeli.

NOTES
     ricordarsi che l'oggetto cosi' creato ha i vertici e i poligoni
     indefiniti quindi vanno usate le funzioni
             GD_addobjvertex
              per
     definirne i vertici e
             GD_addobjpoly
              e
             GD_cattpoly
              per definirne i i
     poligoni e quindi
             GD_recalcobj
              per inizializzarne correttamente
     tutti i valori interni.

SEE ALSO

             GD_deleteobject
              ,
             GD_addobjvertex
              ,
             GD_addobjpoly
              ,
             GD_recalcobj
```

1.17 graphics3d.library/GD_deleteobject()

```
                NAME
     GD_deleteobject  --  elimina un oggetto

SYNOPSIS
     GD_deleteobject(in)
                     A0
     void GD_deleteobject(struct ambient3d *);

FUNCTION
     elimina un oggetto e tutte le aree di memoria che lo riguardano
     facendo diventare l'oggetto attuale il precedente, o se il primo
     il successivo

INPUTS
     in = puntatore a struttura ambient3d della scena 3d su cui operare.
          Deve essere maggiore di 0 altrimenti esito indefinito.

RESULT

BUGS
     nessuno noto, se ne trovate segnalatemeli.

NOTES
     se nessun oggetto definito allora termina senza fare nulla.

SEE ALSO

              GD_newobj
```

1.18 graphics3d.library/GD_addobjvertex()

```
                NAME
     GD_addobjvertex  --  aggiunge un vertice all'oggetto corrente

SYNOPSIS
     esi=GD_addobjvertex(in ,num ,x ,y ,z )
                         A0  D0   D1 D2 D3
     LONG GD_addobjvertex(struct ambient3d *,LONG,LONG,LONG,LONG);

FUNCTION
     inserisce un vertice nell'oggetto attualmente selezionato alla
     posizione indicata da num.

INPUTS
     in  = puntatore a struttura ambient3d della scena 3d su cui operare.
           Deve essere maggiore di 0 altrimenti esito indefinito.
     num = numero intero indicante quale vertice si sta inserendo.
           (#1->num=0 , #2->num=1 , ....).
     x   = valore intero coordinata X del vertice da inserire.
           Valore espresso in fix point (vedi notes di
             GD_moveforward
```

```
            ).
      y   = valore intero coordinata Y del vertice da inserire.
            Valore espresso in fix point (vedi notes di
              GD_moveforward
              ).
      z   = valore intero coordinata Z del vertice da inserire.
            Valore espresso in fix point (vedi notes di
              GD_moveforward
              ).

RESULT
      se esi maggiore di 0 allora tutto ok altrimenti inserimento fallito.

BUGS
      nessuno noto, se ne trovate segnalatemeli.

NOTES
      per il momento questa funzione piu' che inserire modifica il vertice,
      dato che
              GD_newobj
               gia' crea tutti i vertici anche se con valori
      fasulli.
      In futuro pero' puo' darsi che li aggiunga effettivamente.

SEE ALSO

              GD_newobj
              ,
              GD_deleteobject
               ,
              GD_addobjpoly
```

1.19 graphics3d.library/GD_addobjpoly()

```
                NAME
      GD_addobjpoly  --  aggiunge un poligono all'oggetto corrente

SYNOPSIS
      esi=GD_addobjpoly(in ,num ,p1 ,p2 ,p3 ,p4 )
                        A0  D0   D1  D2  D3  D4
      LONG GD_addobjpoly(struct ambient3d *,LONG,LONG,LONG,LONG,LONG);

FUNCTION
      inserisce un poligono nell'oggetto attualmente selezionato alla
      posizione indicata da num.
      Per poligono si intende l'elenco dei tre o quattro vertici in senso
      orario che ne compongono gli spigoli oppure uno o due vertici che
      compongono il punto o la linea.

INPUTS
      in  = puntatore a struttura ambient3d della scena 3d su cui operare.
            Deve essere maggiore di 0 altrimenti esito indefinito.
      num = numero intero indicante quale poligono si sta inserendo.
            (#1->num=0 , #2->num=1 , ....).
```

```
     p1  = numero indice #1 vertice poligono su elenco vertici oggetto.
     p2  = numero indice #2 vertice poligono su elenco vertici oggetto.
           In particolare se questo e' uguale a -1 allora poligono con
           solo un punto ovvero disegna un solo punto.
     p3  = numero indice #3 vertice poligono su elenco vertici oggetto.
           In particolare se questo e' uguale a -1 allora poligono con
           solo due lati ovvero disegna un segmento.
     p4  = numero indice #4 vertice poligono su elenco vertici oggetto.
           In particolare se questo e' uguale a -1 allora poligono con
           solo tre lati.

RESULT
     se esi maggiore di 0 allora tutto ok altrimenti inserimento fallito.

BUGS
     nessuno noto, se ne trovate segnalatemeli.

NOTES
     vale la stessa nota fatta per
             GD_addobjvertex
             , cioe' in realta non
     si aggiunge un poligono ma si modifica uno gia' presente.
     Almeno per ora in futuro chissa'.

SEE ALSO

             GD_newobj
             ,
             GD_deleteobject
              ,
             GD_addobjvertex
              ,
             GD_cattpoly
```

1.20 graphics3d.library/GD_cattpoly()

```
                     OBSOLETA -- usare la
             GD_modpoly()
              .

NAME
     GD_cattpoly  --  varia gli attributi di un poligono

SYNOPSIS
     esi=GD_cattpoly(in ,num ,color ,twoside )
                     A0  D0   D1     D2
     LONG GD_cattpoly(struct ambient3d *,LONG,LONG,LONG);

FUNCTION
     modifica le caratteristiche del poligono alla posizione indicata da
     num nell'oggetto attualmente selezionato.

INPUTS
     in      = puntatore a struttura ambient3d della scena 3d su cui
```

```
                operare. Deve essere maggiore di 0 altrimenti esito
                indefinito.
     num      = numero intero indicante quale poligono si sta variando.
                (#1->num=0 , #2->num=1 , ....).
     color    = numero intero indicante il colore base per il poligono.
                Per la visualizzazione tipo FLAT usera i colori successivi
                a questo per sfumare su toni piu' chiari.
     twoside = numero intero indicante se poligono a 2 facce (1) o ad 1
                faccia (0).
                Se a 2 facce ovvero con faccia anteriore e posteriore,
                risultera' sempre visibile.
                Se con 1 faccia allora sara' visibile solo se la faccia
                che guarda all'esterno dell'oggetto sara' di fronte
                all'osservatore.
                Questo e' un modo veloce per ridurre il n# di poligoni
                presenti in una scena 3d.
                Non e' considerato in caso di poligoni con meno di
                tre lati.

RESULT
     se esi maggiore di 0 allora tutto ok altrimenti modifica fallita.

BUGS
     nessuno noto, se ne trovate segnalatemeli.

NOTES

SEE ALSO

              GD_addobjpoly
```

1.21 graphics3d.library/GD_recalcobj()

```
                NAME
     GD_recalcobj  --  ricalcola i parametri fissi dell'oggetto

SYNOPSIS
     GD_recalcobj(in )
                  A0
     void GD_recalcobj(struct ambient3d *);

FUNCTION
     ricalcola alcuni parametri normalmente non variabili dell'oggetto
     attualmente selezionato (come il bounding box) va richiamata solo
     se si varia le coordinate di uno o piu' vertici .

INPUTS
     in    = puntatore a struttura ambient3d della scena 3d su cui operare.
             Deve essere maggiore di 0 altrimenti esito indefinito.

RESULT

BUGS
     nessuno noto, se ne trovate segnalatemeli.
```

```
NOTES
     questa funzione deve essere sempre usata dopo il termine della
     definizione completa di un nuovo oggetto.
     Ovvero dopo l'uso di
             GD_newobj
             ,
             GD_addobjvertex
             ,
             GD_addobjpoly
             .

SEE ALSO

             GD_newobj
              ,
             GD_addobjvertex
              ,
             GD_addobjpoly
```

1.22 graphics3d.library/GD_setobj()

```
                NAME
     GD_setobj  --  setta come attualmente selezionato un oggetto

SYNOPSIS
     esi=GD_setobj(in ,num )
                   A0  D0
     LONG GD_setobj(struct ambient3d *,LONG);

FUNCTION
     fa diventare oggetto attualmente selezionato quello che ha come
     identificativo num.

INPUTS
     in  = puntatore a struttura ambient3d della scena 3d su cui operare.
           Deve essere maggiore di 0 altrimenti esito indefinito.
     num = numero intero indicante identificativo oggetto da settare.

RESULT
     se esi maggiore di 0 allora tutto ok altrimenti settaggio fallito.

BUGS
     nessuno noto, se ne trovate segnalatemeli.

NOTES

SEE ALSO

             GD_getobj
```

1.23 graphics3d.library/GD_getobj()

```
                NAME
     GD_getobj  --  ritorna identificativo di un oggetto

SYNOPSIS
     id=GD_getobj(in)
                  A0
     LONG GD_getobj(struct ambient3d *);

FUNCTION
     ritorna l'identificativo dell' oggetto attualmente selezionato.

INPUTS
     in = puntatore a struttura ambient3d della scena 3d su cui operare.
          Deve essere maggiore di 0 altrimenti esito indefinito.

RESULT
     se maggiore di 0 allora identificativo oggetto, altrimenti nessun
     oggetto attualmente selezionato.

BUGS
     nessuno noto, se ne trovate segnalatemeli.

NOTES

SEE ALSO

             GD_setobj
```

1.24 graphics3d.library/GD_paintframe()

```
                NAME
     GD_paintframe  --  disegna effettivamente tutti i poligoni

SYNOPSIS
     rast=GD_paintframe(in)
                        A0
     struct RastPort *GD_paintframe(struct ambient3d *);

FUNCTION
     disegna effettivamente tutti i poligoni effettivamente visibili nella
     vista corrente ma non li visualizza.

INPUTS
     in = puntatore a struttura ambient3d della scena 3d su cui operare.
          Deve essere maggiore di 0 altrimenti esito indefinito.

RESULT
     puntatore a RastPort usata per disegnare i poligoni (attualmente non
     visibile), puo' essere utilizzato come riferimento da altre funzioni
     grafiche purche' queste considerino anche i layer (utilizzati per
     il clipping).
```

```
     Inoltre tenete presente che questa rastport ha come 0,0 e larghezza
     e altezza i valori originariamente impostati con
             GD_display3d
              e non
     quelli eventualmente modificati con
             GD_clipbox
             .

BUGS
     nessuno noto, se ne trovate segnalatemeli.

NOTES
     per eliminare le facce nascoste, prima di disegnare i poligoni si
     riordinano in base alla distanza del punto medio dall'osservatore.
     Tenete presente pero' che questo algoritmo sbaglia in caso di
     intersezioni.
     Se si usa lo Z-buffering il risultato e' perfetto ed e' anche
     leggermente piu' veloce con grossi oggetti.

SEE ALSO

             GD_newview
              ,
             GD_switch_rp
```

1.25 graphics3d.library/GD_newview()

```
                NAME
     GD_newview  --  ricalcola la vista attuale della scena 3d

SYNOPSIS
     GD_newview(in)
                A0
     void GD_newview(struct ambient3d *);

FUNCTION
     ricalcola la lista dei poligoni effettivamente visibili nella vista
     attuale e li proietta sul piano di proiezione.

INPUTS
     in = puntatore a struttura ambient3d della scena 3d su cui operare.
          Deve essere maggiore di 0 altrimenti esito indefinito.

RESULT

BUGS
     nessuno noto, se ne trovate segnalatemeli.

NOTES
     questa funzione va usata sempre se si vuol vedere effettivamente
     l'effetto delle trasformazioni sugli oggetti.
     Dopo aver invocato questa funzione occorre invocare
             GD_paintframe()
                     per disegnare i poligoni e poi
```

```
              GD_switch_rp()
               per visualizzarli
     effettivamente.

SEE ALSO

              GD_paintframe
               ,
              GD_switch_rp
```

1.26 graphics3d.library/GD_switch_rp()

```
                 NAME
     GD_switch_rp  --  visualizza la vista disegnata con
              GD_paintframe()
                 SYNOPSIS
     GD_switch_rp(in)
                   A0
     void GD_switch_rp(struct ambient3d *);

FUNCTION
     visualizza la vista disegnata con
              GD_paintframe
               e le eventuali
     aggiunte fatte successivamente.

INPUTS
     in = puntatore a struttura ambient3d della scena 3d su cui operare.
          Deve essere maggiore di 0 altrimenti esito indefinito.

RESULT

BUGS
     nessuno noto, se ne trovate segnalatemeli.

NOTES
     Nella versione _CPU della libreria uso la WritePixelArray() per
     visualizzare il chunky buffer nella finestra.
     Nella versione _BLT della libreria
     per effettuare la visualizzazione si usa la funzione ClipBlit cioe'
     si copia dalla rastport usato da
              GD_paintframe
               alla rastport della
     finestra di visualizzazione il box che si intende visualizzare,
     in questo modo si eliminano(quasi) i fastidiosissimi effetti di
     flickering che si avrebbero disgnando direttamente sulla rastport
     della finestra ed in piu' si semplifica il clipping sul box.
     Ho provato ad usare la funzione BltBitMap al posto di ClipBlit
     visto che sembra piu' veloce, ma almeno sulla mia macchina a volte
     la manda in crash spettacolari.
     Quindi per ora uso ClipBlt , si accettano suggerimenti per risolvere
     il problema.

SEE ALSO
```

```
            GD_paintframe
             ,
            GD_newview
```

1.27 graphics3d.library/GD_translateobject()

```
                NAME
     GD_translateobject  --  spostamento relativo origine di un oggetto

SYNOPSIS
     GD_translateobject(in ,dx ,dy ,dz)
                        A0  D0  D1  D2
     void GD_translateobject(struct ambient3d *,LONG,LONG,LONG);

FUNCTION
     sposta l'origine dell'oggetto attualmente selezionato in modo
     relativo rispetto all'origine del propio asse.

INPUTS
     in = puntatore a struttura ambient3d della scena 3d su cui operare.
          Deve essere maggiore di 0 altrimenti esito indefinito.
     dx = valore spostamento origine oggetto sull'asse X .
          Valore espresso in fix point (vedi notes di
             GD_moveforward
             ).
     dy = valore spostamento origine oggetto sull'asse Y .
          Valore espresso in fix point (vedi notes di
             GD_moveforward
             ).
     dz = valore spostamento origine oggetto sull'asse Z .
          Valore espresso in fix point (vedi notes di
             GD_moveforward
             ).

RESULT

BUGS
     nessuno noto, se ne trovate segnalatemeli.

NOTES
     ogni richiamo di questa funzione adesso si comporta come la

             GD_scaleobject
              o la
             GD_rotateobject
              , cioe non e' cumulativa.
     es: due traslazioni di 2 unita' sull'asse X equivalgono ad una di 2.

SEE ALSO

             GD_positionobject
```

1.28 graphics3d.library/GD_positionobject()

```
                NAME
     GD_positionobject  --  spostamento assoluto origine di un oggetto

SYNOPSIS
     GD_positionobject(in ,x ,y ,z )
                       A0  D0 D1 D2
     void GD_positionobject(struct ambient3d *,LONG,LONG,LONG);

FUNCTION
     sposta l'origine dell'oggetto attualmente selezionato in modo
     assoluto rispetto all'origine della scena 3d.

INPUTS
     in = puntatore a struttura ambient3d della scena 3d su cui operare.
          Deve essere maggiore di 0 altrimenti esito indefinito.
     x  = nuovo valore origine oggetto sull'asse X .
          Valore espresso in fix point (vedi notes di
             GD_moveforward
             ).
     y  = nuovo valore origine oggetto sull'asse Y .
          Valore espresso in fix point (vedi notes di
             GD_moveforward
             ).
     z  = nuovo valore origine oggetto sull'asse Z .
          Valore espresso in fix point (vedi notes di
             GD_moveforward
             ).

RESULT

BUGS
     nessuno noto, se ne trovate segnalatemeli.

NOTES
     e' SEMPRE applicata dopo tutte le altre.

SEE ALSO

             GD_translateobject
```

1.29 graphics3d.library/GD_scaleobject()

```
                NAME
     GD_scaleobject  --  riscala un oggetto

SYNOPSIS
     GD_scaleobject(in ,xscale_fact,yscale_fact,zscalefact)
                    A0  D0          D1          D2
     void GD_scaleobject(struct ambient3d *,LONG,LONG,LONG);

FUNCTION
```

```
     riscala l'oggetto attualmente selezionato lungo gli assi della
     sua origine ma in modo temporaneo (vale solo per la frame attuale).

INPUTS
     in           = puntatore a struttura ambient3d della scena 3d su
             cui operare. Deve essere maggiore di 0 altrimenti esito
             indefinito.
     xscale_fact  = valore riscalatura oggetto sull'asse X .
             Valore espresso in fix point (vedi notes di
             GD_moveforward
             ).
     yscale_fact  = valore riscalatura oggetto sull'asse Y .
             Valore espresso in fix point (vedi notes di
             GD_moveforward
             ).
     zscale_fact  = valore riscalatura oggetto sull'asse Z .
             Valore espresso in fix point (vedi notes di
             GD_moveforward
             ).

RESULT

BUGS
     nessuno noto, se ne trovate segnalatemeli.

NOTES
     ogni volta che si usa questa funzione la riscalatura sara' applicata
     alle originali dimensioni dell'oggetto, quindi se si vuol riscalare
     l'oggetto in piu' volte occorre che i valori scale_fact siano variati
     di conseguenza.
     Es: due riscalature sull'asse X di 2 volte equivalgono ad una di 2
         volte (e non di 4 come puo' sembrare).
     Da notare inoltre che lo stesso vale anche per combinazioni di
     scalature e rotazioni, ovvero per variazioni graduali contemporanee
     occorre ogni volta riapplicare entrambe le trasformazioni con i
     rispettivi valori all'oggetto prima di richiamare la
             GD_newview
             .

SEE ALSO

             GD_rotateobject
```

1.30 graphics3d.library/GD_rotateobject()

```
               NAME
     GD_rotateobject  --  ruota un oggetto

SYNOPSIS
     GD_rotateobject(in ,angle_x ,angle_y ,angle_z)
                     A0  D0       D1       D2
     void GD_rotateobject(struct ambient3d *,LONG,LONG,LONG);

FUNCTION
```

```
     ruota l'oggetto attualmente selezionato rispetto alla sua origine
     ma in modo temporaneo (vale solo per la frame attuale).

INPUTS
     in      = puntatore a struttura ambient3d della scena 3d su
               cui operare. Deve essere maggiore di 0 altrimenti esito
               indefinito.
     angle_x = valore intero (non fix point) in gradi sessagesimali
               indicante angolo di rotazione su asse X oggetto.
     angle_y = valore intero (non fix point) in gradi sessagesimali
               indicante angolo di rotazione su asse Y oggetto.
     angle_z = valore intero (non fix point) in gradi sessagesimali
               indicante angolo di rotazione su asse Z oggetto.

RESULT

BUGS
     nessuno noto, se ne trovate segnalatemeli.

NOTES
     anche questa trasformazione si riferisce sempre alla posizione
     originale dell'oggetto a cui si applica.
     Per maggiori chiarimenti vedi NOTES di
             GD_scaleobject
             .

SEE ALSO

             GD_scaleobject
```

1.31 graphics3d.library/GD_clipbox()

```
               NAME
     GD_clipbox  --  varia le dimensioni del box di visualizzazione.

SYNOPSIS
     esi=GD_clipbox(in ,minx ,miny ,dx ,dy )
                    A0  D0    D1    D2  D3
     LONG GD_clipbox(struct ambient3d *,LONG,LONG,LONG,LONG)

FUNCTION
     varia le dimensioni del box che delimita l'area di visualizzazione
     della scena 3d.

INPUTS
     in   = puntatore a struttura ambient3d della scena 3d su cui operare.
            Deve essere maggiore di 0 altrimenti esito indefinito.
     minx = coordinata X angolo superiore sinistro box, rispetto alla
            finestra di visualizzazione.
     miny = coordinata Y angolo superiore sinistro box, rispetto alla
            finestra di visualizzazione.
     dx   = valore larghezza box deve essere multiplo di 16.
     dy   = valore altezza box.
```

```
RESULT
     se maggiore di 0 allora tutto ok ed indica l'effettivo valore di dx usato,
     altrimenti variazione fallita per errore.

BUGS
     nessuno noto, se ne trovate segnalatemeli.

NOTES
     attenzione a non eccedere mai le dimensioni originali del box di
     visualizzazione definite con la
             GD_display3d
              pena probabili blocchi.
     Questa funzione non fa alcun controllo in proposito.

SEE ALSO

             GD_display3d
```

1.32 graphics3d.library/GD_over()

```
                NAME
     GD_over  --  cambia il modo di tracciamento nella rastport nascosta

SYNOPSIS
     GD_over(in, mod )
             A0  D0
     void GD_over(struct ambient3d *,LONG);

FUNCTION
     cambia il modo video di tracciamento nella rastport usata da

             GD_paintframe
              non influenzando pero' questa funzione.

INPUTS
     in  = puntatore a struttura ambient3d della scena 3d su cui operare.
           Deve essere maggiore di 0 altrimenti esito indefinito.
     mod = nuovo modo video tra i seguenti :
           0 = JAM1 (usare la macro JAM1)
           1 = JAM2 (usare la macro JAM2)
           2 = COMPLEMENT (usare la macro COMPLEMENT)
           4 = INVERSVID (usare la macro INVERSVID)

RESULT

BUGS
     nessuno noto, se ne trovate segnalatemeli.

NOTES

SEE ALSO
```

1.33 graphics3d.library/GD_cascene()

```
                NAME
     GD_cascene -- varia alcuni parametri di visualizzazione della scena
                   3d

SYNOPSIS
     GD_cascene(in, new )
                A0  A1
     LONG GD_cascene(struct ambient3d *,struct tag3d *);

FUNCTION
     Permette di variare alcuni parametri di visualizzazione della scena
     3d definita con dislplay3d.

INPUTS
     in  = puntatore a struttura ambient3d della scena 3d su cui operare.
           Deve essere maggiore di 0 altrimenti esito indefinito.
     new = puntatore ad array di strutture tag3d con nuovi parametri,
           funziona allo stesso modo delle TagList implementate nelle
           librerie di sistema.
           Per il momento i valori possibili per .tipo sono:
           CS_PROJECT - tipo di proiezione da usare valori usabili:
                        PROSP_P=proiezione prospettica (l'attuale).
                        PARAL_P=proiezione parallela (sperimentale).
           CS_SBUFF   - non implementato ancora.
           CS_GCOLOR  - n# registro colore sfondo scena 3d (default n#0).
           CS_VDIST   - nuovo valore (intero non fix) per distanza
                        tra osservatore e piano di proiezione.
           CS_VIEWP   - inserisce le coordinate attuali della camera nella
                        struttura vertex puntata dal campo .val dell'elemento
                        tag3b.
           CS_NPX0    - val. intero (non fix) con nuova origine X
                        box nella finestra.
           CS_NPY0    - val. intero (non fix) con nuova origine Y
                        box nella finestra.
           CS_ZBUF    - on/off(1/0) z-buffer.
           CS_ZOOM    - val. in fixpoint fissa il livello di zoom della scena
                        (max: 256 min: 1/256).

RESULT
     se uguale a 0 nessuna variazione effettuata, se >0 allora indica
     numero di variazioni effettuate.

BUGS
     nessuno noto, se ne trovate segnalatemeli.

NOTES
     Trattare l'array di strutture tag3d esattamente come gli
     array di strutture TagItem dette anche tag list implementate
     nel sistema operativo amiga dal 2.0 in poi.
     L'ultima struttura dell'array deve essere vuota e deve avere
     come primo elemento la costante END_T.
     NON USARE MAI NEL PRIMO VALORE DIRETTAMENTE I NUMERI MA USATE
     SEMPRE LE LABEL CHE LI INDICANO.
     Per abilitare lo Z-Buffer si deve avere almeno maxXbox*maxYbox*4
```

```
     bytes di memoria liberi altrimenti non si abilita.
     Attenzione Se si abilita e poi disabilita la memoria usata non
     e' subito rilasciata ma verra liberata' solo alla chiusura della
     scena (
             GD_close_display3d
             ).

SEE ALSO
```

1.34 graphics3d.library/GD_fix2int()

```
                NAME
     GD_fix2int -- converte un numero fix point in un intero.

SYNOPSIS
     GD_fix2int(in,out)
                A0  A1
     LONG GD_fix2int(LONG *,LONG *)

FUNCTION
     converte un numero fix point nel formato della libreria in un intero
     a 32bit approssimando all'intero piu' vicino.

INPUTS
     in  = puntatore ad intero a 32 bit con valore fix point da
           convertire.
     out = puntatore ad intero a 32 bit dove mettere risultato.

RESULT
     se uguale a 0 tutto ok se diverso da 0 errore e out non modificato.
BUGS
     nessuno noto, se ne trovate segnalatemeli.

NOTES

SEE ALSO

             GD_fix2sfl
             ,
             GD_fix2dfl
```

1.35 graphics3d.library/GD_fix2sfl()

```
                NAME
     GD_fix2sfl -- converte un numero fix point in un single float.

SYNOPSIS
     GD_fix2sfl(in,out)
                A0  A1
     LONG GD_fix2sfl(LONG *,float *)
```

```
FUNCTION
     converte un numero fix point nel formato della libreria in un float
     in singola precisione.

INPUTS
     in  = puntatore ad intero a 32 bit con valore fix point da
           convertire.
     out = puntatore a numero single float dove mettere risultato.

RESULT
     se uguale a 0 tutto ok se diverso da 0 errore e out non modificato.
BUGS
     nessuno noto, se ne trovate segnalatemeli.

NOTES

SEE ALSO

             GD_fix2int
             ,
             GD_fix2dfl
```

1.36 graphics3d.library/GD_fix2dfl()

```
               NAME
     GD_fix2dfl -- converte un numero fix point in un double float.

SYNOPSIS
     GD_fix2dfl(in,out)
                A0  A1
     LONG GD_fix2dfl(LONG *,double *)

FUNCTION
     converte un numero fix point nel formato della libreria in un float
     in doppia precisione.

INPUTS
     in  = puntatore ad intero a 32 bit con valore fix point da
           convertire.
     out = puntatore a numero double float dove mettere risultato.

RESULT
     se uguale a 0 tutto ok se diverso da 0 errore e out non modificato.
BUGS
     nessuno noto, se ne trovate segnalatemeli.

NOTES

SEE ALSO

             GD_fix2sfl
             ,
             GD_fix2int
```

1.37 graphics3d.library/GD_int2fix()

```
                NAME
     GD_int2fix -- converte un intero in un numero in fix point.

SYNOPSIS
     GD_int2fix(in ,out)
                A0  A1
     LONG GD_int2fix(LONG *,LONG *);

FUNCTION
     converte un intero a 32bit in un numero fix point nel formato
     richiesto dalla libreria.

INPUTS
     in  = puntatore a intero a 32 bit da convertire.
     out = puntatore a intero a 32 bit dove mettere numero in fix point
           calcolato.

RESULT
     se uguale a 0 tutto ok se diverso da 0 errore e out non modificato.
BUGS
     nessuno noto, se ne trovate segnalatemeli.

NOTES

SEE ALSO

             GD_sfl2fix
             ,
             GD_dfl2fix
```

1.38 graphics3d.library/GD_sfl2fix()

```
                NAME
     GD_sfl2fix -- converte un single float in un numero in fix point.
SYNOPSIS
     GD_sfl2fix(in ,out)
                A0  A1
     LONG GD_sfl2fix(float *,LONG *);

FUNCTION
     converte un numero float in singola precisione in un numero fix
     point nel formato richiesto dalla libreria.

INPUTS
     in  = puntatore a numero float da convertire.
     out = puntatore a intero a 32 bit dove mettere numero in fix point
           calcolato.
```

```
RESULT
     se uguale a 0 tutto ok se diverso da 0 errore e out non modificato.
BUGS
     nessuno noto, se ne trovate segnalatemeli.

NOTES

SEE ALSO

             GD_int2fix
             ,
             GD_dfl2fix
```

1.39 graphics3d.library/GD_dfl2fix()

```
               NAME
     GD_dfl2fix -- converte un double float in un numero in fix point.

SYNOPSIS
     GD_dfl2fix(in ,out)
                A0  A1
     LONG GD_dfl2fix(double *,LONG *);

FUNCTION
     converte un numero float in doppia precisione in un numero fix
     point nel formato richiesto dalla libreria.

INPUTS
     in  = puntatore a numero double float da convertire.
     out = puntatore a intero a 32 bit dove mettere numero in fix point
           calcolato.

RESULT
     se uguale a 0 tutto ok se diverso da 0 errore e out non modificato.
BUGS
     nessuno noto, se ne trovate segnalatemeli.

NOTES

SEE ALSO

             GD_sfl2fix
             ,
             GD_int2fix
```

1.40 graphics3d.library/GD_loadobject()

```
             NAME
     GD_loadobject -- Carica un oggetto con il formato propietario .3dgfo .
```

```
SYNOPSIS
        id=GD_loadobject(in,name,scale)
                         A0 A1   D0
        LONG
                GD_dfl2fix
                (struct ambient3d *,char *,LONG)

FUNCTION
        Carica un file con la descrizione di un oggetto 3d, il formato
        e' propietario vedi le note per una rapida spiegazione.
INPUT
        in   = puntatore a struttura ambient3d della scena 3d su cui operare.
               Deve essere maggiore di 0 altrimenti esito indefinito.
        name = puntatore a stringa con nome del file da caricare (path completo)
        scale= valore in FIXPOINT con il fattore di scala da applicare all'oggetto ↩
           .

RESULT
        Se uguale a 0 operazione fallita oggetto non caricato.
        Se diverso da 0 allora e' l'identificativo del nuovo oggetto creato come
        in
                GD_newobj
                .

BUGS
        Nessuno noto, se ne trovate segnalatemeli.

NOTES
        Questa e' una spiegazione del mio formato propietario per oggetti 3d , .3 ↩
           dgfo:
        Testata :
         [3][D][G][F][X][V][x][.][x][x][00][nome_oggetto][00]
         numero totale vertici (4 bytes :long int)
         numero totale poligoni (4 bytes :long int)
        Coordinate vertici :
         X coordinata (8 bytes :formato Ieee double float )
         Y coordinata (8 bytes :formato Ieee double float )
         Z coordinata (8 bytes :formato Ieee double float )
         Nota : Una tripletta per ogni vertice.
        Descrittori poligoni :
         flag+colour del poligono (4 bytes)
         numero vertici del poligono (1 byte :valori permessi 1,2,3,4)
         #1 indice vertice (4 bytes :long int ,primo=0 ultimo=numero_vertice-1)
         #2 indice vertice (4 bytes :long int ,primo=0 ultimo=numero_vertice-1)
         #3 indice vertice (4 bytes :long int ,primo=0 ultimo=numero_vertice-1)
         #4 indice vertice (4 bytes :long int ,primo=0 ultimo=numero_vertice-1)
         Nota : l'indice vertice #2,#3,#4 e' presente solo se il numero dei  ↩
            vertici
                del poligono e' di rispettivamente 2,3 o 4.
         Descrizione di flag+colour :
                flag (1 byte)
                bit 0 = 0
                        colour[0] e' la componente Rossa (1 byte: 0-255)
                        colour[1] e' la componente Verde (1 byte: 0-255)
                        colour[3] e' la componente Blu   (1 byte: 0-255)
                bit 0 = 1
```

```
                        colour[0] ignorato
                        colour[1] ignorato
                        colour[3] registro colore nella palette virtuale
                                        (1 byte: 0-255)
                Nota : per ora e' supportato solo bit 0=1
                bit 1 = 1 poligono a 2 facce.
                bit 1 = 0 poligono ad 1 faccia
                bit 2-7 = riservati per usi futuri

SEE ALSO
```

1.41 graphics3d.library/GD_genpalette()

```
                NAME
        GD_genpalette -- genera la palette dei colori virtuali per la scena 3D

SYNOPSIS
        esi=GD_genpalette(in,new)
                          A0 A1
        LONG GD_genpalette(struct ambient3d *,struct tag3d *)

FUNCTION
        Genera la palette dei colori virtuale ovvero il numero assegnato ad ogni
        colore non e' detto corrisponda al registro colore della palette inoltre
        implementa anche il colore trasparente.
        La
                GD_touchpalette()
                 ora richiama questa funzione.

INPUT
        in   = puntatore a struttura ambient3d della scena 3d su cui operare.
               Deve essere maggiore di 0 altrimenti esito indefinito.
        new  = puntatore ad array di strutture tag3d con nuovi parametri,
               funziona allo stesso modo delle TagList implementate nelle
               librerie di sistema.
               Per il momento i valori possibili per .tipo sono:
                GP_RCOL - riserva il n# di colori indicato nella palette da ↩
                   generare
                          cioe' non modifichera quei colori a partire dal reg #0.
                          Utile per non modificare i colori del sistema (Def. #4).
                GP_NCOL - definisce il n# base di colori virtuali da usare (Def. ↩
                   #1)
                          Non puo' superare il n# di colori ammesso per lo schermo ↩
                              usato
                          salvo per l'uso del colore trasparente.
                GP_NLIV - definisce il n# di livelli di luminosita' ammessi per ↩
                   ogni
                          colore virtuale, eccetto che per il trasparente (def. si
                          adatta automaticamente al massimo permesso su quello ↩
                             schermo).
                          Il prodotto GP_NLIV*GP_NCOL+GP_RCOL <= N#_MAX. ↩
                             _COLORI_SCHERMO.
                          Altrimenti questo valore sara' variato dalla funzione in ↩
                              modo
```

```
            da soddisfare sempre questa funzione.
            Si puo' sfruttare questa caratteristica per rendere la ↩
               def.
            dei colori e della scena indipendente dal n# di colori ↩
               dello
            schermo scelto.
GP_TRASP- se .val=1 attiva il colore trasparente se .val=0 lo ↩
   disattiva
            (Def. disattivato) questo sara' sempre l'ultimo dei ↩
               colori
            virtuali e non verra' usato per il controllo del massimo ↩
                n# di
            livelli di luminosita' ammissibili.
GP_COL  - fissa il n# di colore virtuale su cui agiranno le ↩
   prossime
            operazioni.(def. #0) Non puo' superare ovviamente il ↩
               valore
            indicato in GP_NCOL e inizia da 0 .
            Questa operazione puo' essere ripetuta piu' volte, in ↩
               questo
            modo si potranno modificare piu' colori virtuali in una ↩
               volta
            sola.
GP_HRGB - fissa il massimo livello di luminosita' per il colore ↩
   virtuale
            scelto, .val deve puntare ad una struttura rgbtype con ↩
               tale
            valore.  (Def ->red=15,->green=15,->blue=15).
            Attenzione se i valori saranno compresi nel range tra ↩
               0-15 si
            potra' usare anche il S.O. >=2.0 altrimenti se compresi ↩
               tra
            0-255 occorrera' per forza il S.O >= 3.0
GP_LRGB - fissa il minimo livello di luminosita' per il colore ↩
   virtuale
            scelto.  (Def ->red=0,->green=0,->blue=0).
            Per il resto e' identica in tutto a GP_HRGB.
GP_INFO - Restituisce nell'intero puntato da .val il n# di ↩
   registro
            reale della palette a cui corrisponde il registro ↩
               virtuale
            scelto.Questo tra l'altro coincide con il valore di ↩
               minore
            intensita' luminosa.
GP_PALET- in .val e' indicato il n# di registro reale della ↩
   palette a cui
            far corrispondere il colore virtuale scelto.
            ATTENZIONE questo verra' considerato' come valore con ↩
               minor
            intensita' luminosa per quel colore e i successivi ↩
               GP_NLIV
            registri saranno' i valori d'intensita' luminosa ↩
               crescente.
            Funzione utile se si vuol predefinire la palette prima ↩
               di
            usare questa funzione.
```

```
RESULT
        Numero di operazioni eseguite con successo.

BUGS
        Nessuno noto, se ne trovate segnalatemeli.

NOTES
        L'uso accorto di questa funzione permette di evitare di dover ricalcolare ↩
           la
        palette e cambiare il colore degli oggetti se si cambia il n# massimo di ↩
           colori
        dello schermo.
        Inoltre dovrebbe rendere quasi indolore il passaggio eventuale (nelle ↩
           prossime
        versioni) a schermi true color.

SEE ALSO

                GD_touchpalette
```

1.42 graphics3d.library/GD_modpoly()

```
                NAME
        GD_modpoly -- modifica alcune caratteristiche dei poligoni dell'oggetto
                      attuale

SYNOPSIS
        esi=GD_modpoly(in,new)
                        A0 A1
        LONG GD_modpoly(struct ambient3d *,struct tag3d *)

FUNCTION
        Permette la modifica di alcuni parametri dei poligoni dell'oggetto attuale
        tra questi permette il posizionamento di una texture map sul poligono.

INPUT
        in   = puntatore a struttura ambient3d della scena 3d su cui operare.
               Deve essere maggiore di 0 altrimenti esito indefinito.
        new  = puntatore ad array di strutture tag3d con nuovi parametri,
               funziona allo stesso modo delle TagList implementate nelle
               librerie di sistema.
               Per il momento i valori possibili per .tipo sono:
                MP_POLY - seleziona il n# del poligono su cui agiranno le
                          successive operazioni (Def. #0)
                          Questa operazione puo' essere ripetuta piu' volte cosi'
                          da poter modificare piu' poligoni in una sola volta.
                MP_ACTIV- attiva (1 su .val) o disattiva (0 su .val) il poligono
                          scelto (def. attivo).
                MP_COLOR- nuovo colore (tra quelli virtuali) per il poligono
                          scelto.
                MP_2SIDE- setta a due(1 su .val) o ad una(0 su .val) faccia il
                          poligono scelto.
                MP_TMAP - assegna la texture map indicata al poligono scelto.
                          In val occorre riportare il valore restituito dalle
```

```
                funzioni
            GD_newtmap()
             o
            GD_newtmapf()
             .(Def #0 nessuna
                texture map assegnata).
            MP_VTMAP- posiziona l'eventuale texture map assegnata sul poligono
                scelto.
                .val deve puntare ad una struttura vtmap dovre dovranno
                essere definiti i vertici in pixel nella texture map
                per ogni rispettivo vertice del poligono .
                Quindi si puo' fare in modo che sul poligono sia ↩
                   visualizzata
                anche solo una porzione della texture e orientata come ↩
                   si vuole.
            MP_VTAUTO-Posiziona automaticamente la texture sul poligono in ↩
               modo che
                l'angolo in alto a sinistra della texure coincida col ↩
                   primo
                vertice del poligono e il resto della texture sia ↩
                   interamente
                visualizzato nel poligono.
                ATTENZIONE impostare .val sempre a 0 poiche' in future ↩
                   versioni
                potra' assumere qualche significato.

RESULT
        Numero di operazioni eseguite con successo.

BUGS
        Nessuno noto, se ne trovate segnalatemeli.

NOTES
        Ricordarsi che se si vuol attivare il texture mapping oltre a definire ↩
           quale
        texture map applicare e come occorre anche attivare il texture map sull' ↩
           oggetto
        di cui fa parte quel poligono, usare la
                GD_modobj()
                 per fare questo.

SEE ALSO
```

1.43 graphics3d.library/GD_newtmap()

```
                NAME
        GD_newtmap -- crea una texture map usabile poi da qualunque oggetto

SYNOPSIS
        map=GD_newtmap(in,dx,dy,buf)
                       A0 D0 D1 A1
        LONG GD_newtmap(struct ambient3d *,short int,short int,unsigned char *)

FUNCTION
```

```
        Carica e prepara una texture map per poter essere successivamente piazzata
        su uno o piu' poligoni a partire da un buffer chunky che ne descrive l' ↩
           immagine.

INPUT
        in   = puntatore a struttura ambient3d della scena 3d su cui operare.
               Deve essere maggiore di 0 altrimenti esito indefinito.

        dx   = valore larghezza texture map espresso in pixel.
        dy   = valore altezza texture map espresso in pixel.
        buf  = puntatore a array di unsigned char con descrizione immagine dx*dy
               della texture map in formato chunky buffer.
               I colori li indicati coincideranno con i colori virtuali definiti
               con la
                GD_genpalette()
                 o
                GD_touchpalette().Ovviamente
                 se alcuni
               pixel saranno indicati del colore trasparente(sempre se attivo)
               allora l'immagine risultante sara' bucata in quei punti.

RESULT
        Se 0 operazione fallita , se diversa da 0 identificativo di quella texture
        da indicare tutte le volte che la si vuol applicare su qualche poligono.
        Non esistono limiti sul numero di volte che una texture puo' essere  ↩
           applicata
        e questa sara' memorizzata sempre e soltanto 1 volta.
        Non c'e alcuna differenza di velocita' se si applica una texture di 2x2  ↩
           pixel
        di lato o di 1000x1000 pixel (varia ovviamente solo l'occupazione di  ↩
           memoria)
        quindi basatevi solo sulla qualita' di visualizzazzione finale.

BUGS
        Nessuno noto, se ne trovate segnalatemeli.

NOTES

SEE ALSO

                GD_newtmapf
                ,
                GD_rmtmap
                ,
                GD_modpoly
```

1.44 graphics3d.library/GD_rmtmap()

```
                NAME
        GD_rmtmap -- elimina una texture map precedentemente creata

SYNOPSIS
        GD_rmtmap(in,map)
                  A0 A1
```

```
        GD_rmtmap(struct ambient3d *,long int *)

FUNCTION
        Elimina una texture map precedentemente preparata con
                GD_newtmap()
                 o

                GD_newtmapf()
                .

INPUT
        in   = puntatore a struttura ambient3d della scena 3d su cui operare.
               Deve essere maggiore di 0 altrimenti esito indefinito.
        map  = valore restituito da
                GD_newtmap()
                 o
                GD_newtmapf()
                 ,se =0 non
               fara' nulla.

RESULT
        Nessuno.

BUGS
        Nessuno noto, se ne trovate segnalatemeli.

NOTES

SEE ALSO

                GD_newtmap
                ,
                GD_newtmapf
```

1.45 graphics3d.library/GD_newtmapf()

```
                NAME
        GD_newtmapf -- crea una texture map leggendola da un file iff-ilbm

SYNOPSIS
        map=GD_newtmapf(in,name)
                        A0 A1
        LONG GD_newtmapf(struct ambient3d *,unsigned char *)

FUNCTION
        Funzionalmente identica alla
                GD_newtmap()
                 pero' legge i dati dal file in
        formato IFF-ILBM indicato.

INPUT
        in   = puntatore a struttura ambient3d della scena 3d su cui operare.
               Deve essere maggiore di 0 altrimenti esito indefinito.
        name = puntatore a stringa di caratteri con nome e path completo del
```

```
                file in formato IFF-ILBM in cui e' descritta la texture map.
                Le dimensioni dx e dy saranno estratte dal medesimo file.
                Non saranno lette correttamente immagini HAM ,HALF-BRITE o
                TRUE COLOR.
                Per i colori valgono le stesse indicazioni della
                 GD_newtmap()
                 .

RESULT
        Vedi
                 GD_newtmap()
                 .

BUGS
        Nessuno noto, se ne trovate segnalatemeli.

NOTES
        Vedi
                 GD_newtmap()
                 .

SEE ALSO

                 GD_newtmap
                 ,
                 GD_rmtmap
                 ,
                 GD_modpoly
```

1.46 graphics3d.library/GD_colldetect()

```
NAME
        GD_colldetect -- rileva le collisioni fra oggetti

SYNOPSIS
        ris=GD_colldetect(in,n, buf)
                          A0 D0 A1
        LONG GD_colldetect(struct ambient3d *,long int,long int *)

FUNCTION
        Permette la rilevazione di eventuali collisione dell'oggetto attuale con
        gli altri ed ovviamente indica con chi in caso positivo.

INPUT
        in   = puntatore a struttura ambient3d della scena 3d su cui operare.
               Deve essere maggiore di 0 altrimenti esito indefinito.
        n    = numero massimo di collisioni registrabili.
        buf  = array di long int con almeno 'n' elementi (non si verifica quindi
               attenzione) dove verranno messi gli identificativi degli
               oggetti con cui l'attuale collide.
               Se si rileva un numero di collisioni maggiore di n l'array
               sara' completamente riempito gli altri andranno persi.
               Se si rileva un numero minore solo i primi elementi dell'array
```

```
               saranno riempiti.

RESULT
        Se 0 nessuna collisione rilevata altrimenti e' il numero di TUTTE le
        collisioni effettivamente rilevate.

BUGS
        Nessuno noto, se ne trovate segnalatemeli.

NOTES
        Per il rilevamento delle collisioni viene considerato solo il bounding
        box dell'oggetto (aggiornato ad ogni trasformazione) quindi puo' essere
        un po' impreciso, ad esempio se si ha un toroide e si inserisce qualunque
        altro oggetto nel foro centrale si rilevera' una collisione anche se
        non e' realmente cosi'.

SEE ALSO
```

1.47 graphics3d.library/GD_modobj()

```
NAME
        GD_modobj -- modifica alcune caratteristiche dell'oggetto attuale

SYNOPSIS
        esi=GD_modobj(in,new)
                      A0 A1
        LONG GD_modobj(struct ambient3d *,struct tag3d *)

FUNCTION
        Permette la modifica di alcuni parametri dell'oggetto attuale, per
        ora lo stato e il modo di visualizzazione.

INPUT
        in   = puntatore a struttura ambient3d della scena 3d su cui operare.
               Deve essere maggiore di 0 altrimenti esito indefinito.
        new  = puntatore ad array di strutture tag3d con nuovi parametri,
               funziona allo stesso modo delle TagList implementate nelle
               librerie di sistema.
               Per il momento i valori possibili per .tipo sono:
                MO_STATE - attiva (.val=1) o disattiva (.val=0) l'oggetto attuale
                           (def. attivo).
                           Se si disattiva questo sara' ancora esistente ma non
                           sara' piu' considerato nell'aggiornamento e  ↩
                              visualizzazione
                           della scena, se l'oggetto e' grosso si otterra' una  ↩
                              discreta
                           accellerazione nella visualizzazione e nei calcoli.
                MO_VMODE - cambia il modo di visualizzazione di un oggetto valori
                           accettati:
                                WIREF           -> wireframe
                                SOLID           -> solido
                                SOLID+TMAP      -> solido con texture map
                                FLAT            -> flat shading
                                FLAT+TMAP       -> texture map con flat shading
```

```
                                GORAUD          -> goraud shading
                                GORAUD+TMAP     -> texture map con goraud shading

RESULT
        Numero di operazioni eseguite con successo.

BUGS
        Nessuno noto, se ne trovate segnalatemeli.

NOTES
        Per il modo di visualizzazione occorre tener presente che solo la versione ↩
            per
        CPU ( _CPU ) implementa il goraud shading e tutte le combinazioni con la
        texture mapping, le altre versioni accettano quei parametri ma li ignorano ↩
            .

SEE ALSO
```