L'algorisme de visualització per partició binaria de l'espai

P. Brunet
J. Pascual
R. Vila

Report LSI–89–3

Abstract: This work presents the Binary Space Partition algorithm for the visualization of polyedric scenes (BSP algorithm, Fuchs 1980). It also includes the criteria for the election of the root face. The complexity of the involved algorithms is studied. A new algorithm for choosing the root face of the subtrees is presented and discussed. The performance of the algorithm is presented through seven examples.

Resum: El treball presenta l'algorisme de visualitzacio d'escenes poliedriques basat en la particio binaria de l'espai (algorisme BSP, Fuchs 1980), junt amb els criteris per a l'eleccio de la cara arrel. S'analitza la complexitat dels algorismes associats, i es presenta i discuteix un nou algorisme per a l'eleccio de la cara arrel dels subarbres. Es presenta el comportament de l'algorisme en set objectes example.

1. INTRODUCCIO

La generació d´una imatge realista a partir d´una escena composada per un nombre elevat de cares planes, és un procés complexe i que dificilment pot fer-se en temps real. Després de transformar els punts a l´espai imatge i realitzar un retallat en 3D per tal d´eliminar polígons que cauen fora del camp de vista, la generació efectiva de la imatge implica el càlcul del color en cada un dels pixels de la pantalla, ordenant els poligons segons la profunditat, o bé cercant el més proper a l´observador (NeS79).

Els algorismes d´eliminació de parts amagades es poden classificar en els que treballen en espai objecte, i els que fan els seus càlculs en espai imatge (SSS74, NeS79). Els primers modifiquen el model geomètric dels objectes, deixant-ne només les parts (habitualment d´arestes) que es veuen; són aptes per la generació d´imatges en pantalla i/o plotter, i la sortida es pot canviar d´escala. En canvi, els segons treballen directament en coordenades de pantalla (raster), calculant la visibilitat en cada un dels seus pixels; en aquest cas, la sortida es calcula específicament per a un determinat tipus de pantalla.

No obstant, existeix un tercer tipus d´algorismes mixtes, anomenats de llista de prioritats (SSS74), que treballen en espai objecte en una primera fase, i en espai imatge en una segona fase. La llista de prioritats, que es calcula en espai objecte, no és més que una ordenació topológica de totes les cares dels objectes, segons la seva profunditat respecte l´observador. Més exactament, a la llista ja ordenada de cares c1,c2,...,ci,...cj,...cn, per tot i<j es cumpleix que cj pot tapar total o parcialment a la cara ci, pero mai ci pot tapar a cj (es pot definir també com una ordenació a la qual, per tota parella de cares consecutives es vol que la segona pugui tapar a la primera pero no al revés). La construcció de la llista de prioritats pot requerir en algún cas la subdivisió d´alguna de les cares de l´escena, com

es fa per exemple a l´algorisme de Newell, Newell i Sancha (NeS79). Un cop obtinguda la llista de prioritats, ja és inmediat d´obtenir la imatge: o bé es fa un retallat de cada cara ci per totes les ck, k>i i després es dibuixa el que queda de la cara i, o bé es pot utilitzar l´algorisme del pintor, que treballa forçosament en espai imatge (es pinten amb color sòlid, directament sobre pantalla, les cares en l´ordre en que són a la llista, c1..cn; el pintat de cada cara tapa automàticament les parts de les cares anteriors que eren ocultades per ella). En aquest darrer cas, la generació de la imatge a partir de la llista de prioritats té una complexitat linial respecte el nombre de cares a l´escena, i es pot resoldre si es disposa d´alguna primitiva d´omplert de poligons amb colors sólids (com el Fill_Area del GKS). D´altra banda, la llista de prioritats és evidentment una entitat pertanyent a l´espai objecte, ja que pot ser usada per generació d´imatges a pantalles de característiques diferents.

Un conjunt important dins dels algorismes de llista de prioritats, està constituit pels que generen una llista de prioritats entre cares, que és independent de la posició del punt de vista. Aquesta idea va ser analitzada per primer cop per Schumacker (SBG69, SSS74), i es basa en que les prioritats de les cares tenen una component lligada a característiques purament geomètriques de l´objecte e independents del punt de vista (per exemple, les cares de dins d´un forat són més difícils de veure que les de la part exterior d´un objecte, i caldrà posar-les abans a la llista de prioritats. El principal interés d´aquest tipus d´algorismes apareix en els cassos en que una mateixa imatge ha de ser vista per un observador que va canviant de posició (per exemple, en els simuladors de vol, FAG83); en aquest cas val la pena destinar un temps important al preprocés de l´escena i càlcul de la llista de prioritats, si després cada generació de la imatge amb l´algorisme del pintor serà de baixa complexitat i podrà fer-se en temps real (o quasi). Les idees de Schumacker van ser desenvolupades posteriorment per Fuchs, i han donat lloc a l´algorisme de partició binaria de l´espai.

En el que segueix, limitarem el nostre estudi a l´algorisme de llista de prioritats de Fuchs (FKN80, FAG83) per una escena formada per sólids emmagatzemats en model de fronteres, i dels que es guarda per tant una descripció completa de la geometría de les seves cares planes. En el cas que l´escena contingui elements superficials sense gruix, únicament caldria modificar lleugerament l´algorisme de generació de la imatge per

recorregut de l´arbre per tal de no eliminar les cares en direcció contraria a l´observador (FAG83).

Els dos apartats seguents presenten l´algorisme BSP de Fuchs, i descriuen els algorismes concrets de generació i recorregut de l´arbre de prioritats. Després, l´apartat 4 discuteix els criteris per l´elecció de la cara arrel que es presenten a (FKN80) i (FAG83), i l´apartat 5 inclou un anàlisi de la complexitat dels algorismes. Finalment, a l´apartat 6 presenta i discuteix les característiques específiques de l´algorisme implementat en el Sistema de Modelatge Geomètric DMI (Mètodes Informàtics, ETSEIB,UPC).

2. L´ALGORISME BSP. Construcció de l´arbre de prioritats

La principal característica de l´algorisme BSP (FKN80), dins dels algorismes de llista de prioritats, és que la construcció de la llista de prioritats és independent de la posició de l´observador respecte de l´escena. Per tal d´aconseguir això, el BSP no construeix una llista sino un arbre binari, en que cada nus és una cara que intervé en l´escena a visualitzar. En realitat, les sigles BSP (Binary Space Partition), provenen d´aquesta manera de subdividir l´espai.

El procediment és el seguent: Es selecciona una cara de la llista de cares inicials, i es posa com arrel de l´arbre BSP. Per cada cara de les que resten a la llista, es compara amb el pla de la cara arrel per determinar a quin semiespai es troba, i s´inclou a la llista corresponent a aquest semiespai (anomenarem semiespai del davant de la cara arrel, al definit per la normal a la cara en el sentit cap a l´exterior de l´objecte, i semiespai del darrera, a l´altre). Si una cara travessa el pla de la cara arrel, es talla per aquest pla i les parts resultants s´inclouen a les llistes corresponents. Amb això, obtenim una cara arrel i dues llistes amb la resta de les cares classificades. Aquest procediment es repeteix recursivament: per cada una de les llistes que penjen de la cara arrel, es selecciona una cara com a arrel del subarbre corresponent, i es classifica la resta de cares de la llista respecte aquesta. El procediment acaba quan no queden cares a la llista per processar.

L´algorisme concret és el seguent:

```
Procediment construir_arbre (llista_de_cares, arbre_BSP)
  Si llista_de_cares és buida llavors arbre_BSP:=arbre buit
  altrament
      arrel de l'arbre := cara seleccionada dins de llista_de_cares
    Per cada cara que resta a la llista fer
        classificar la cara respecte la cara arrel
      Si cara davant de la cara arrel llavors insertar-la a llista_davant
      altraSi cara darrera de la cara arrel llavors insertar-la a
                                                    llista_darrera
      altrament  {la cara és tallada pel pla de la cara arrel}
          tallar la cara pel pla
           afegir cada una de les parts de la secció a les llistes
      fisi
      fiper
       construir_arbre (llista_davant, arbre_davant)
       construir_arbre (llista_darrera, arbre_darrera)
      arbre_BSP := (arbre_darrera, cara arrel, arbre_davant)
  fisi
 fi
```

A part de la selecció de la cara arrel, que será estudiada als apartats 4 i 6, l'algorisme únicament conté una classificació de cares respecte la cara arrel i el procés de tall d'una cara pel pla de l'arrel. Pel que fa a la classificació, consisteix simplement en fer la classificació de tots els punts de la cara respecte el pla; i el procés de tall és un cas particular dels coneguts algorismes de clipping de poligons.

A la figura 1 es pot veure un exemple d'objecte senzill (a dalt), i tres fases diferents de la construcció del seu arbre BSP. Es considera que la cara 4 té un forat que conecta amb les cares 1 i 3, i per simplicitat no es consideren les cares paraleles al pla del paper. A la primera fase es pren com a cara arrel la 4, amb la qual cosa el subarbre de l'esquerra - que conté la llista de cares del darrera de la 4 - pasa a estar format per les cares 5, 6, 7, 8 i 9 i el subarbre de la dreta per les cares 1, 2 i 3. A la segona fase, s'escull com arrel del subarbre de l'esquerra la cara 8, i com arrel del subarbre de la dreta la cara 2; tant en un cas com en l'altre, totes les cares són darrera de les dues cares arrel, i per tant aquestes no tenen apuntador per la dreta. Finalment, a la figura es presenta l'arbre complet;

cal observar, per exemple, que la cara 7 té la resta de cares (5 i 6) davant; per tant, en aquest cas, el subarbre no nul és el de la dreta de la cara 7.

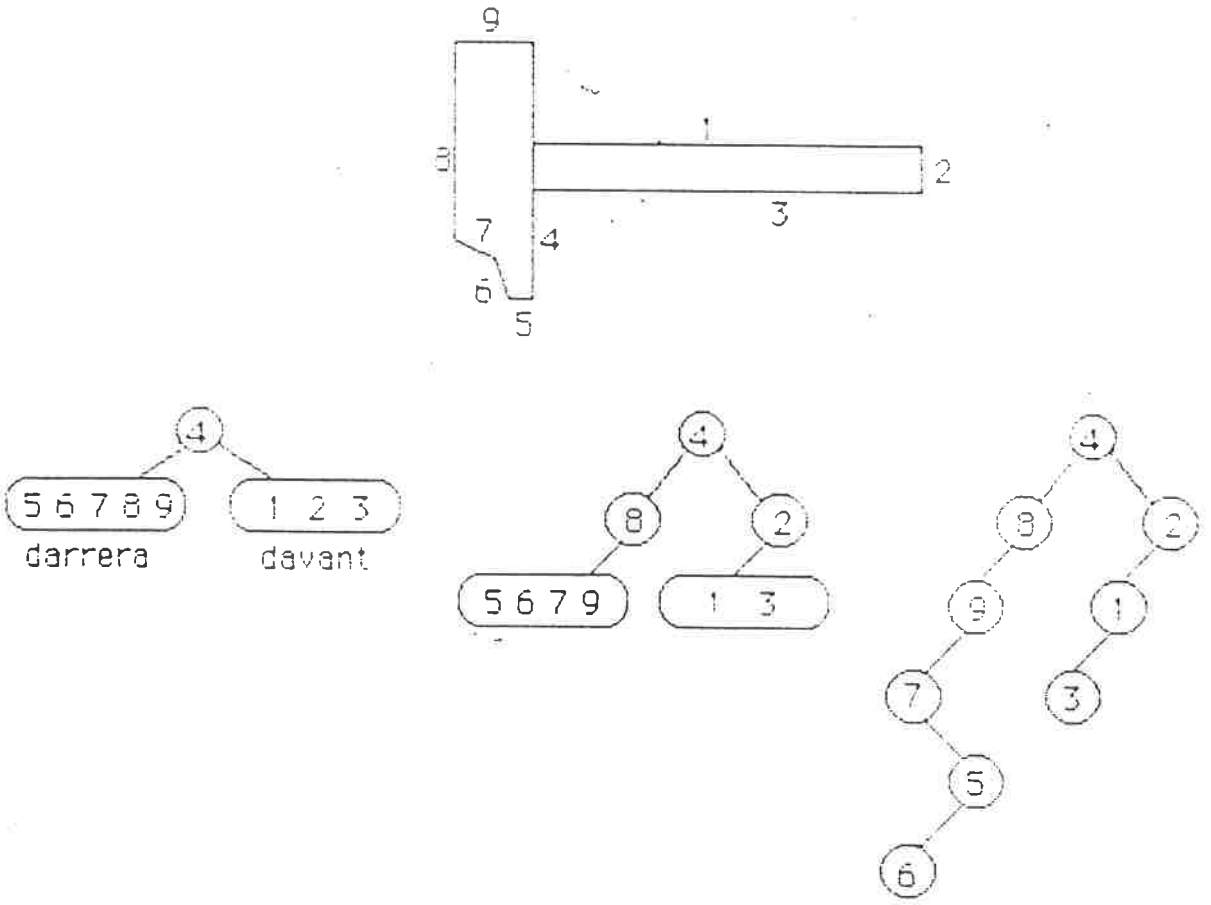


Figura 1. Tres fases de la construcció de l'arbre BSP d'un objecte

3. PROCES DE GENERACIO DE LA IMATGE

Un cop construit l'arbre, la generació de la imatge significa únicament fer un recorregut de l'arbre, equivalent a donar els poligons en el ordre del més allunyat al més proper a l'observador (back-to-front order). Aixo és possible sempre que l'observador estigui a l'infinit, sigui quina sigui la seva orientació a l'espai respecte el centre de l'escena. El cas de distancies finites és igualment tractable, tot i que és recomanable que totes les cares de l'escena estiguin a la mateixa banda del punt de vista.

El recorregut de l´arbre que cal realitzar en el procés de visualització, depen de la posició relativa observador-cares de l´objecte. A cada node de l´arbre, es determina si l´observador està situat davant o darrera de la cara corresponent al node. Aquest test, determina quin dels dos subarbres cal recorrer en primer lloc. El criteri és de tipus inordre, e implica recorrer primer el subarbre del semiespai del ´darrera´ (que no conté el punt de vista), dibuixar després la cara del node si no ha estat eliminada en el procés de culling (supresió de les cares orientades en sentit contrari a l´observador, FoV82), i finalment recorrer el subarbre del semiespai del davant, que conté el punt de vista:

```
Procediment Generació_imatge (arbre_BSP)
  Si arbre_BSP no buit llavors
    Si observador és davant de la cara arrel de l´arbre BSP llavors
        Generació_imatge (subarbre_BSP esquerra)
        Fill_area (cara arrel de l´arbre BSP)
        Generació_imatge (subarbre_BSP dreta)
    altrament
        Generació_imatge (subarbre_BSP dreta)
        Generació_imatge (subarbre_BSP esquerra)
     fisi
    fisi
 fi
```

El test "observador és davant de la cara arrel de l´arbre BSP", és molt simple, ja que es reduieix a fer el producte escalar entre el vector normal a la cara arrel de l´arbre, i el vector que indica la direcció del punt de vista. Aquest test, junt amb l´ordre en que s´han classificat les cares en el procés de generació de l´arbre, és el que assegura que les cares s´envien al frame buffer en ordre de profunditat major a menor respecte el punt de vista, tal com es pot deduir de l´algorisme recursiu.

Cal observar, pel que fa al procediment Fill_area(Cara arrel de l´arbre BSP) que:

- El conjunt de crides a Fill_area signifiquen l´aplicació de l´algorisme

del pintor (NeS79) a les cares dels objectes de l´escena a representar.

- El color del pintat de cada una de les cares s´ha de calcular d´acord amb el model d´il.luminació (FoV82), i és forçosament uniforme a tota la cara. Així, la major velocitat de visualització té com a contrapartida que les cares de les aproximacions facetades dels cilindres i superficies corbes, no es suavitzen.

- El pintat de cada cara ha de evitar els seus forats. Això es contempla, per exemple, en el GKS 3D, pero no en el GKS 2D.
- Si el perifèric de sortida no disposa d´una doble memoria de pantalla (frame buffer), l´algorisme del pintor harà de treballar directament sobre la imatge que es visualitza, i la generació de la imatge produirà un transitori amb cares a pantalla que seran més tard eliminades.

4. ELECCIO DE L´ARREL DEL SUBARBRE

En l´algorisme de generació de l´arbre BSP l´unic procés que mereix una atenció especial és el de la selecció de la cara candidata a ser arrel de l´arbre, entre la llista de cares rebudes pel procediment recursiu. Aquest procés influeix notablement la complexitat i necessitat de memoria de l´algorisme, i és l´unic que admet diversitat de solucions, tal com veurem als apartats seguents. A tall d´exemple, observi´s que, a la figura 1, si s´hagués agafat com a primera arrel la cara 3 enlloc de la 4, s´haurien d´haver tallat les 4 i 8; en canvi, amb l´elecció que s´ha fet no ha calgut tallar cap cara en tot el procés de construcció de l´arbre.

Una primera idea podria ser la seguent,

4.0 La cara candidata a arrel és sempre la primera de la llista de cares

A FKN80, la selecció de la cara arrel es basa en el seguent principi,

4.1 La cara candidata és aquella el pla de la qual talla el mínim nombre de cares a la llista.

tot i que es proposa com a millora un algorisme més complexe basat en la minimització de conflictes binaris a les llistes resultants. En concret,

4.2 Si S1(c), S2(c) i S3(c) són els subconjunts de cares de la llista respectivament davant del pla de la cara c, tallades pel pla de la cara c i darrera del pla de c, i es defineix,

$f(s_i, s_j) = 1$ si la cara c_j és tallada pel pla de la cara c_i , i 0 en cas contrari.

a la vegada que es defineix $I_{m,n}$= sumatori de $f(s_i,s_j)$ per totes les s_i de Sm i totes les s_j de Sn, la cara c seleccionada és aquella per la qual es maximitza el valor $I_{1,3} + I_{3,1}$ - $factor_{pes}$*(nº cares a S2)

D´altra banda, a l´article posterior FAG83, es proposa un algorisme heurístic molt més simple, i que no carrega tant el temps de càlcul a la construcció de l´arbre:

4.3 S´analitzen N cares a l´atzar d´entre les de la llista d´entrada al procediment, i es pren com a cara arrel del subarbre aquella d´entre les les N, el pla de la qual talla el mínim nombre de cares de la llista.

A FAG83 es discuteix el nombre de cares generades per tall durant l´algorisme en funció de N, per a 9 exemples concrets. Es comparen els resultats per N=1, 3, 5 i 15, i es conclou que N=5 dona ja arbres que són quasi óptims.

Un cop analitzada la complexitat dels algorismes, a l´apartat 6 es proposaran i discutiran algorismes alternatius de selecció de la cara candidata a arrel de l´arbre.

5. ANALISI DE LA COMPLEXITAT

La complexitat de la fase de generació de l´arbre depen tant del temps de cerca de la cara candidata a arrel, com del temps de classificació de cada una de les cares respecte l´arrel. El primer pot ser constant (quan es pren com arrel la primera cara de la llista, 4.0, o bé en l´algorisme 4.3), pero habitualment és proporcional al nombre k de cares de la llista, en implicar un recorregut d´aquesta. En aquesta darrer cas podem definir,

Que c*k és el temps de cerca de la cara arrel dins d'una llista de k cares
(c és el que anomenarem temps de cerca unitari)
Que d és el temps de classificar una cara respecte el pla de la arrel.

A més d'això, i com veurem, la complexitat de la generació de l'arbre depén de la seva mateixa estructura. No és el mateix un arbre BSP equilibrat, que un arbre en que cada node té un únic apuntador; en aquest darrer cas, l árbre té realment l'estructura d'una llista linial. La interpretació geomètrica és que s'ha pogut trobar una successió de cares arrel tals que en cada cas tenien totes les cares davant o totes les cares darrera. Un exemple trivial és el dels cossos convexes, que donen lloc a un arbre linial independentment de l'algorisme emprat per a la selecció de la cara arrel.

Suposem que es construeix l'arbre a partir d'una llista inicial amb n cares:

En el cas extrem de que l'algorisme de selecció de la cara arrel porti a un arbre BSP linial, aquest contindrà n-1 cares arrel. El temps de càlcul necessari a cada pas de l'algorisme recursiu és,

Cara	Temps de cerca	Temps de classificació
1	c*n	d*(n-1)
2	c*(n-1)	d*(n-2)
.....	.....	.....
n-1	c*2	d

En consequència, el temps total per a construir l'arbre BSP linial és,

$$T_L = c*(n+2)*(n-1)/2 + d*n*(n-1)/2 \leq (c+d) * (n-1)*(n+2)/2$$

$$T_L = O\left(\frac{c+d}{2} n^2\right)$$

En l'altre cas extrem, en que es construeix un arbre BSP equilibrat, i suposant que $n=2^r-1$, es té,

Nivell k a l´arbre	Num. de cares arrel	Temps de cerca	Temps classif.
1	1	c*n	d*(n-1)
2	2	c*(n-1)	d*(n-3)
3	4	c*(n-3)	d*(n-7)
4	8	c*(n-7)	$d*(n-2^k+1)$
.....	.....	.....	.....
r-1			

Operant, es pot trobar el temps de càlcul com,

$$T_B = c*n + d*(n+1)/2 + (d+c) \sum (n+1-2^k) = c*n + d*(n+1)/2 + \\ + (c+d)*(n+1)*(\log_2(n+1)-2) - (c+d)*(n-3)/2$$

i en resum,

$$T_B = O\ (\ (c+d)*(n+1)*\log_2(n+1)\ )$$

En definitiva, i suposant cerca de la cara arrel de complexitat linial (algorisme 4.1, per exemple), el temps de creació de l´arbre és algun valor intermig entre el quadràtic T_L i el de tipus n log n T_B . No obstant, la majoría de criteris de selecció de la cara arrel tendeixen a donar arbres fortament linials, i per tant en molts casos, el temps de càlcul és proper a la cota superior quadràtica. La raó és força lógica: per tal de generar un arbre equilibrat, cal que les cares candidates a arrel generin dues llistes de davant i darrera amb el mateix nombre de cares; això només es pot aconseguir amb plans que passen pel centre de l´escena, i que molt probablement tallaran un nombre elevat de cares. Per la mateixa raó, un dels criteris lògics per a evitar talls de cares és prendre com a cares arrel aquelles que no tallen cares perque les tenen totes a una banda - per exemple, quan s´agafa com a candidata una cara del "convex hull" de l´objecte a visualitzar -. En realitat, sempre és preferible un arbre quasi-linial amb poques cares tallades, que un arbre equilibrat amb un nombre superior de cares tallades, ja que la fase de recorregut de l´arbre i

visualització, com veurem té una complexitat proporcional al nombre total de cares de l'arbre, després de la construcció i tall. I se suposa que la generació de l'arbre no té perque fer-se en temps real (per tant pot ser de complexitat quadràtica), mentre que les diferents visualitzacions sí que ho han de ser.(Observi's que aquesta és una raó que desaconsella l'algorisme 4.0).

Una manera molt senzilla de reduir la complexitat de l'algorisme de creació de l'arbre, quan l'escena es composa de diversos objectes, es basa en la creació de conjunts linealment separables. Es diu que una escena es composa de dos conjunts C1 i C2 d'objectes que són linealment separables entre sí, si existeix un pla tal que C1 pertany totalment a un dels semiespais que defineix, mentre que C2 pertany totalment a l'altre. En aquest cas, només cal calcular l'arbre BSP de cada un dels conjunts C1 i C2, i després composar-los com a subarbres d'un node arrel amb el pla fictici - normalment no pertanyent a l'escena -, que s'ha usat per la separació linial de l'escena. Aquest procés es pot repetir si a la seva vegada, els conjunts C1 i C2 són linealment separables, i la complexitat va tendint altra cop de n*n a n*log n. A més, com és inmediat de veure i s'indica a FAG83, no cal recalcular l'arbre BSP si l'escena canvia d'estructura per moviment dels seus components i aquests no surten dels semiespais de separació linial.

Finalment (apartat 3), és inmediat veure que l'algorisme de recorregut de l'arbre i visualització té una complexitat proporcional al nombre de cares resultants de la construcció de l'arbre BSP, n+t (on t és el nombre de cares que han estat tallades). Per a cada cara, l'unic que es fa es

- el producte escalar de la seva normal amb la direcció d'observació
- l'omplert amb color del seu interior (fill area)

Per tant, podem dir que el temps de cada visualització T_V és,

$$T_V = \beta * (n+t)$$

Existeixen altres algorismes d'eliminació de parts amagades (per exemple, els Z-buffer de linia), que comporten una complexitat proporcional al

nombre de cares de l´escena, encara que amb un coeficient de proporcionalitat força elevat (proporcional al nombre de pixels de la pantalla, en el cas del Z-buffer). Comparant l´algorisme BSP amb aquests, podem dir:

- Que sempre, la fase de visualització a l´algorisme BSP serà molt més ràpida que en aquests algorismes, degut a que el coeficient ß és molt petit.
- Que, per escenes no massa complexes, el temps total de visualització en BSP, T_L+T_V , és també inferior al de visualització d´aquests algorismes.
- En canvi, per escenes molt complexes, el temps de construcció de l´arbre, que és quadràtic, esdevé molt elevat, i l´algorisme BSP no és aconsellable quan s´ha de realitzar una (o poques) visualitzacions. No obstant, quan s´han de realitzar moltes visualitzacions de la mateixa escena, pot tornar a ser interessant usar el mètode BSP, generant l´arbre un sol cop en procés batch, i guardant-lo en disc per a les properes visualitzacions.

6. SOLUCIONS APORTADES. DISCUSSIO

Seguint les consideracions anteriors, a la implementació de l´algorisme BSP que s´ha realitzat al Sistema de Modelatge DMI, s´ha intentat minimitzar el nombre de cares tallades, pero sense anar a algorismes de complexitat excessiva, com podrien ser els del tipus 4.2. D´altra banda, l´algorisme 4.3 sembla excessivament heurístic, i el seu comportament pot ser especialment dolent en determinats cassos. L´algorisme de selecció implementat es basa en que les cares de l´escena que pertanyen a la frontera convexa, que no tallen a les altres i produeixen arbres BSP linials, contenen vertex de coordenades extremes dins del conjunt de vertex de l´escena - en alguna sistema de coordenades ortonormal -. En concret, l´algorisme conté els seguents passos,

- Abans d´iniciar la creació de l´arbre, s´ordenen les cares per les coordenades dels seus vértex mínims: Si V_i és el vértex de coordenada x mínima a la cara C_i , i V_j és el vértex de coordenada x mínima a la cara C_j , direm que $C_i < C_j$ si $x(V_i) < x(V_j)$. En el cas d´igualtat de les

coordenades x, la ordenació ve definida per les coordenades y, i sino, per les z. Aquest procés té complexitat n * log n, i per tant és menor que la corresponent a la creació de l´arbre.

- A cada selecció de cara arrel, es cerca la cara amb mínim nombre de talls seguint, dins de la llista de cares, l´ordre definit a l´ordenació previa. Observi´s que l´ordenació es fa una sola vegada i serveix per a totes les llistes de cares dels subarbres, donat que són subconjunts de la llista inicial ordenada, i no es modifica la posició espaial de les cares. Es pot pensar en dos possibles algorismes, que anomenarem d i e:

    d) Es recorre la llista de cares en l´ordre indicat, fins trobar la primera cara que no produeixi talls, o acabar la llista. En aquest darrer cas, es pren com a cara arrel la cara que donava el mínim nombre de talls a tota la llista (com el criteri 4.1).

    e) Es recorre la llista de cares en l´ordre indicat, fins trobar la primera cara que no produeixi talls, o haver tractat els M primers elements de la llista. En aquest darrer cas, es pren com a cara arrel la cara que dona el mínim nombre de talls d´entre les M tractades.

S´ha fet un estudi comparatiu de prestacions d´aquests dos algorismes, junt amb els seguents més clàssics, que no inclouen el pretractament d´ordenació,

a) Recorregut sense memoria del mínim. La cara arrel és, o bé la primera que no produeix talls, o be la darrera de la llista.

b) Algorisme 4.1 (amb finalització abans d´acabar la llista si es troba una cara que no produeix talls)

c) Algorisme b, amb alternancia en el sentit de recorregut de les llistes e implementació enllaçada de les mateixes.

Els resultats, pel que fa a nombre de cares n+t a l´arbre final i temps de procés, es poden veure a les Taules 1 i 2. Els objectes 1...7 són els corresponents a les figures 2.1 ...2.7, i comporten diferents complexitats geomètriques.

TAULA 1
Cares inicials a cada un dels models exemple,
i total n+t de cares a l'arbre BSP construit

Objecte	1	2	3	4	5	6	7
Cares Inicials	73	26	168	320	216	254	448
a	96	26	168	320	560	450	*
b	75	26	168	320	337	316	*
c	75	26	168	320	338	314	*
d	75	26	168	320	338	*	*
e	75	27	*	320	410	338	840

* : Valors encara no avaluats

TAULA 2
Temps de procés a cada un dels models exemple,
a la fase de generació de l'arbre (segons)

Objecte	1	2	3	4	5	6	7
Cares Inicials	73	26	168	320	216	254	448
a	21	1.5	13.2	47.1	57.4	81.4	*
b	19.4	0.6	22.1	59	375	638	*
c	11	0.5	9.7	25.2	100	167	*
d	4	0.2	5.6	28	188	*	*
e	1.2	0.2	*	*	22	31	668

* : Valors encara no avaluats

Pel que fa al nombre de cares tallades, es pot observar que l´increment és moderat, excepte a l´algorisme a i el cas de l´objecte 5 a l´algorisme e. En canvi, analitzant el temps de càlcul, s´observa que els temps són significativament més baixos en l´algorisme e, com ja era d´esperar. Per tant, es pot concloure que aquest darrer algorisme permet de reduir significativament el temps de construcció de l´arbre, sense incrementar el nombre de cares del model.

7. REFERENCIES

FKN80 H.Fuchs, Z.M.Kedem, B.F.Naylor, "On Visible Surface Generation by a Priori Tree Structures". ACM Computer Graphics, Vol 14 num 3, pp 124-133; Proc Siggraph 1980.

FAG83 H.Fuchs, G.D.Abram, E.D.Grant, "Near Real-Time Shaded Display of Rigid Objects", ACM Computer Graphics, VOL 17 num 3, pp 65-72; Proc Siggraph 1983.

FoV82 J.D.Foley, A.Van Dam, "Fundamentals of Interactive Computer Graphics", Addison Wesley Publ. Co., 1982.

NeS79 W.Newman, R.Sproull, "Principles of Interactive Computer Graphics", Mac Graw Hill, 1979.

SBG69 R.A.Schumacker, B.Brand, M.Gilliland, W.Sharp, "Study for Applying Computer Generated Images to Visual Simulation". Technical Report AEHRL-TR-69-14 (AD 700375), U.S. Air Force Human Resources Laboratory, 1969.

SSS74 I.E.Sutherland, R.F.Sproul, R.A.Schumacker, "A Characterization of Ten Hidden-Surface Algorithms", Computing Surveys of the ACM, Vol 6 num 1, 1974.

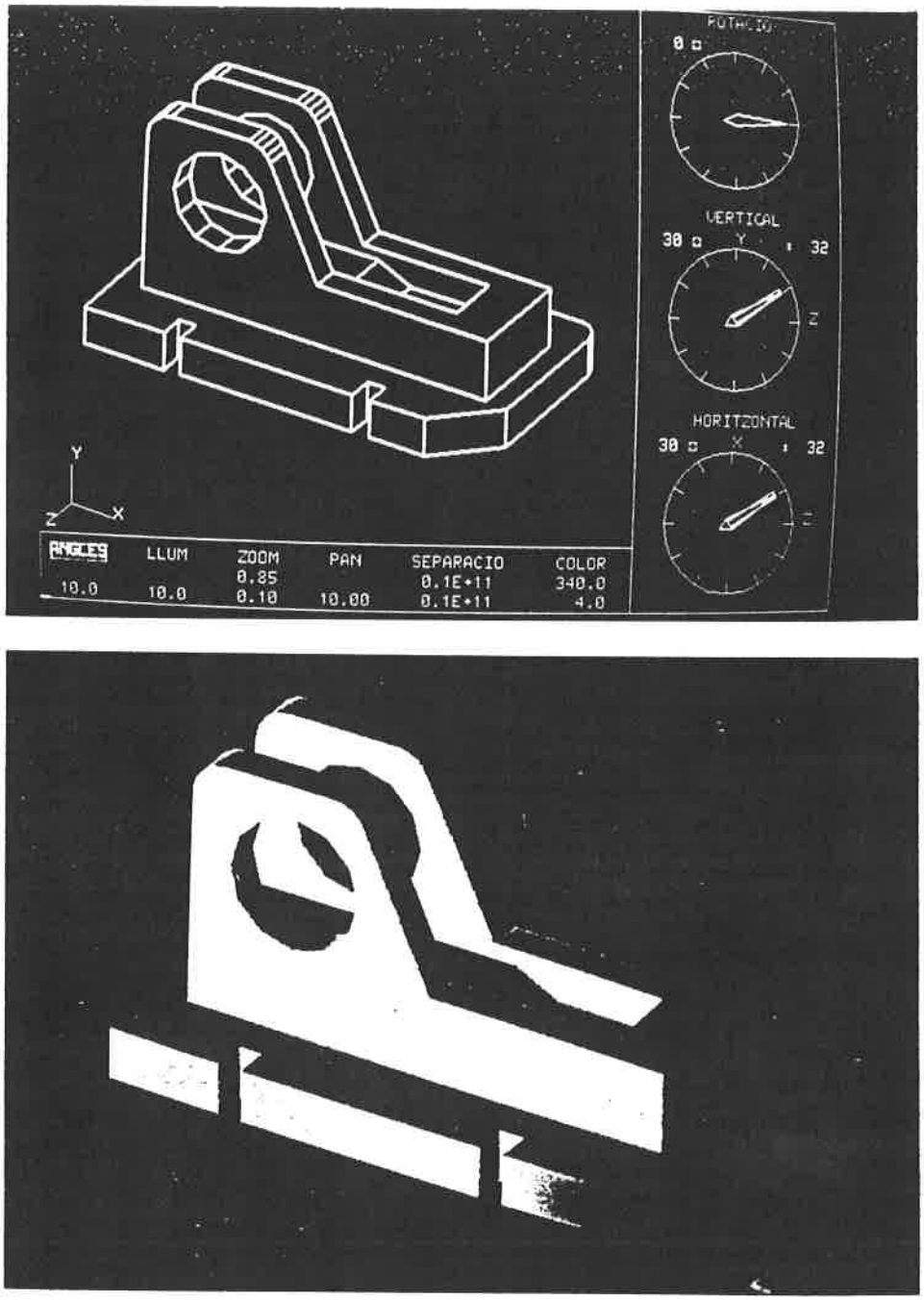


Figura 2.1 Objecte de test amb 73 cares planes.

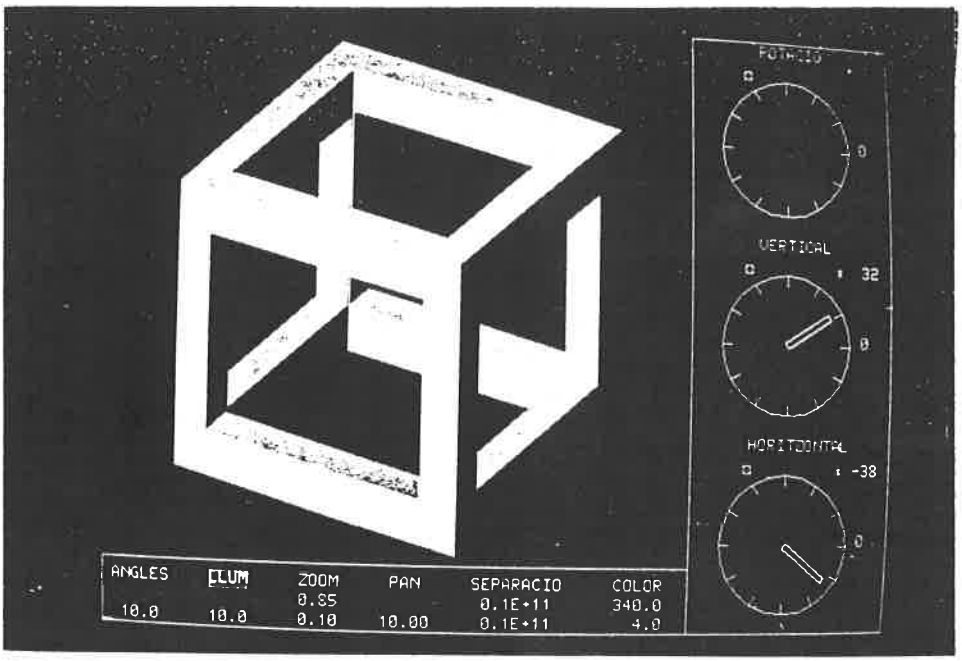


Figura 2.2 Objecte test amb 26 cares. Resultat de la visualització BSP.

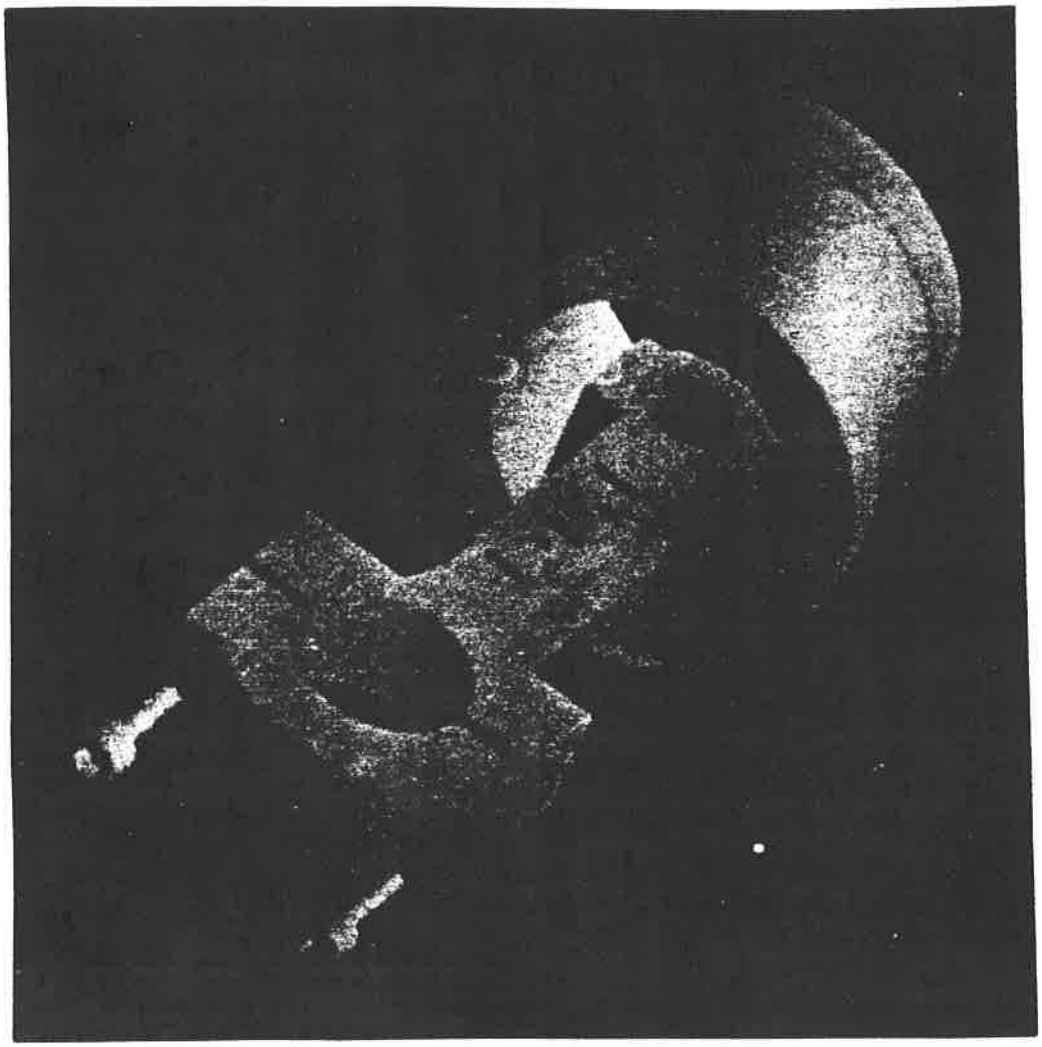


Figura 2.3 Conjunt d´objectes amb un total de 168 cares planes.

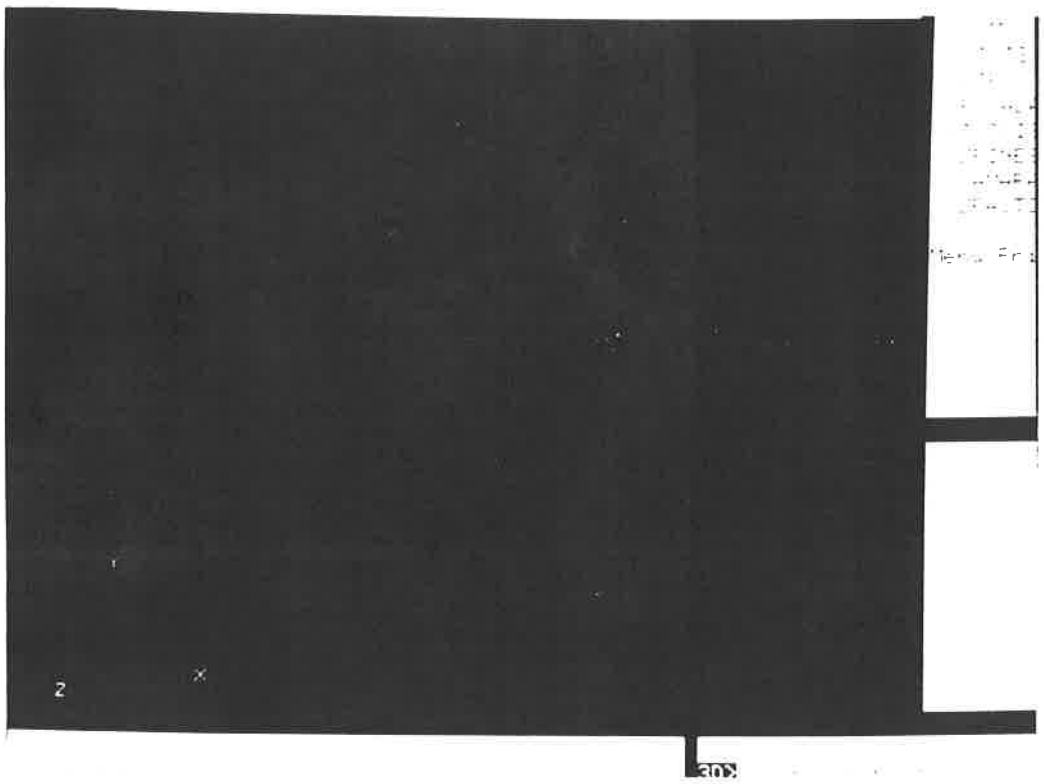


Figura 2.4 Objecte convexe de 320 cares planes

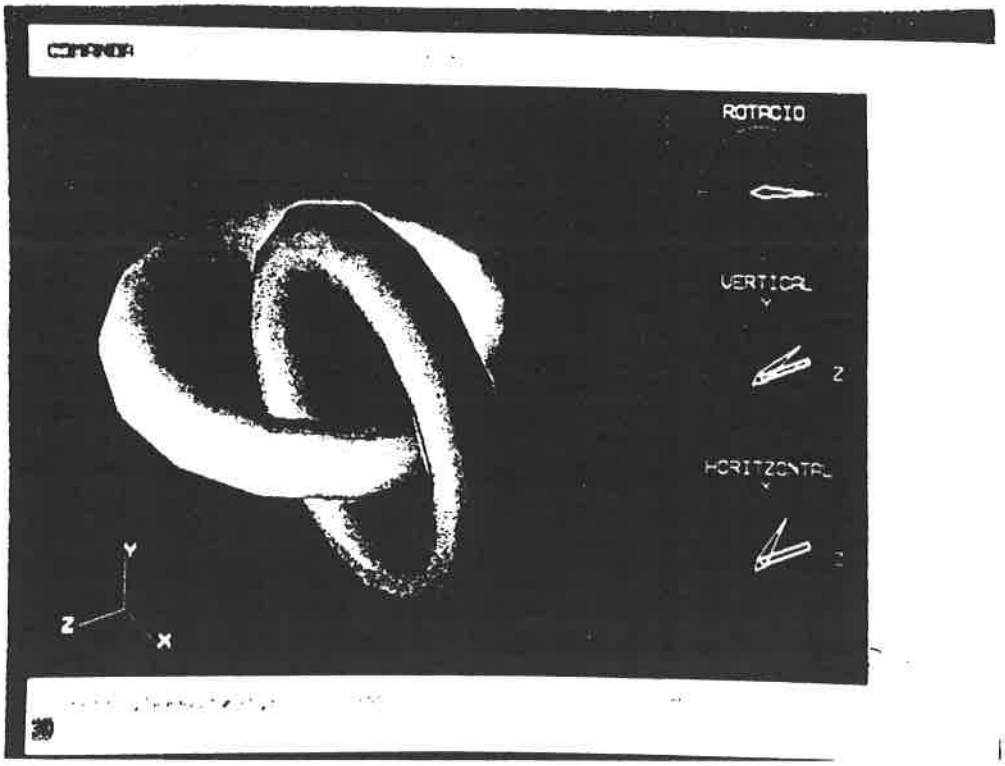


Figura 2.5 Dos torus, amb un total de 216 cares.

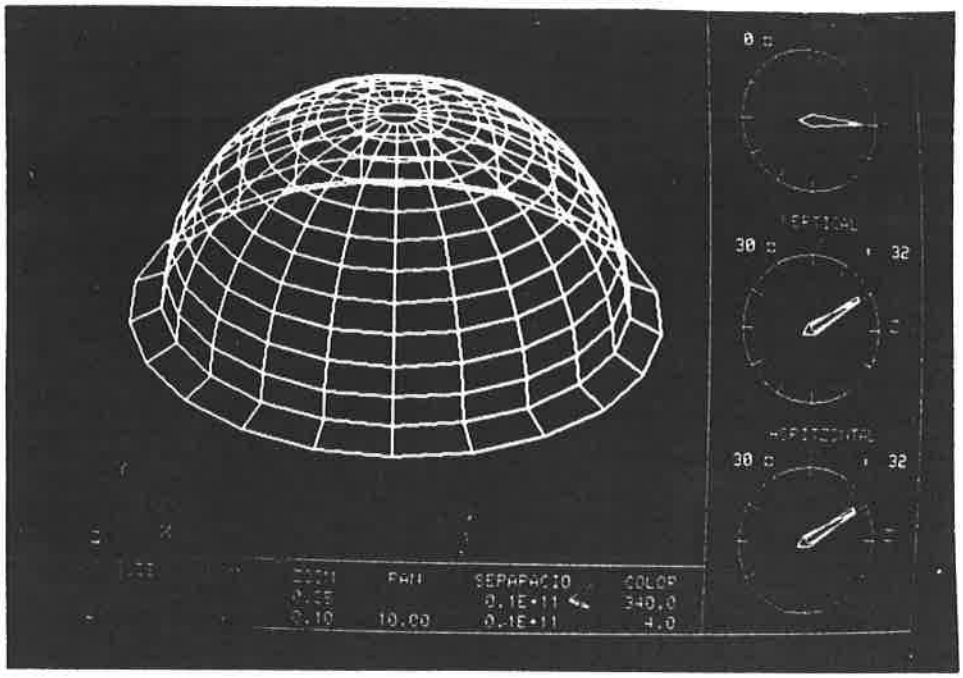
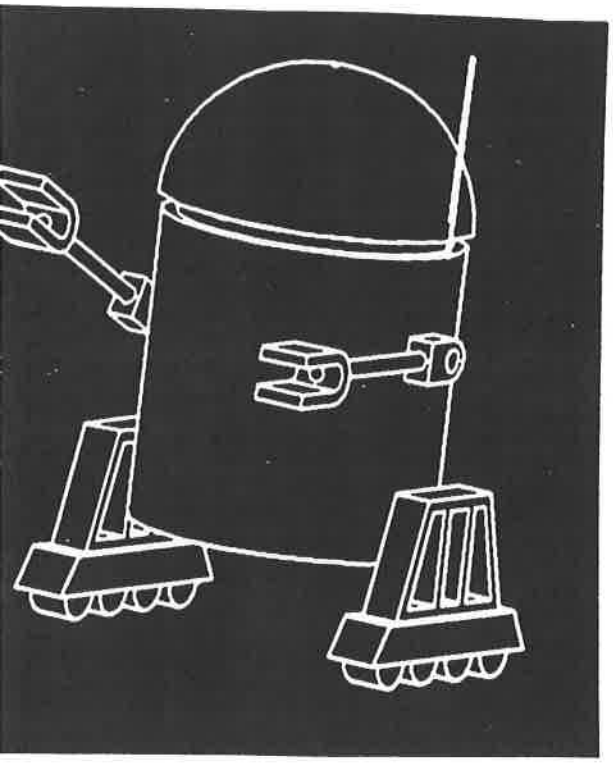


Figura 2.6 Objecte amb 254 cares, que és no convexe degut a la vorera inferior



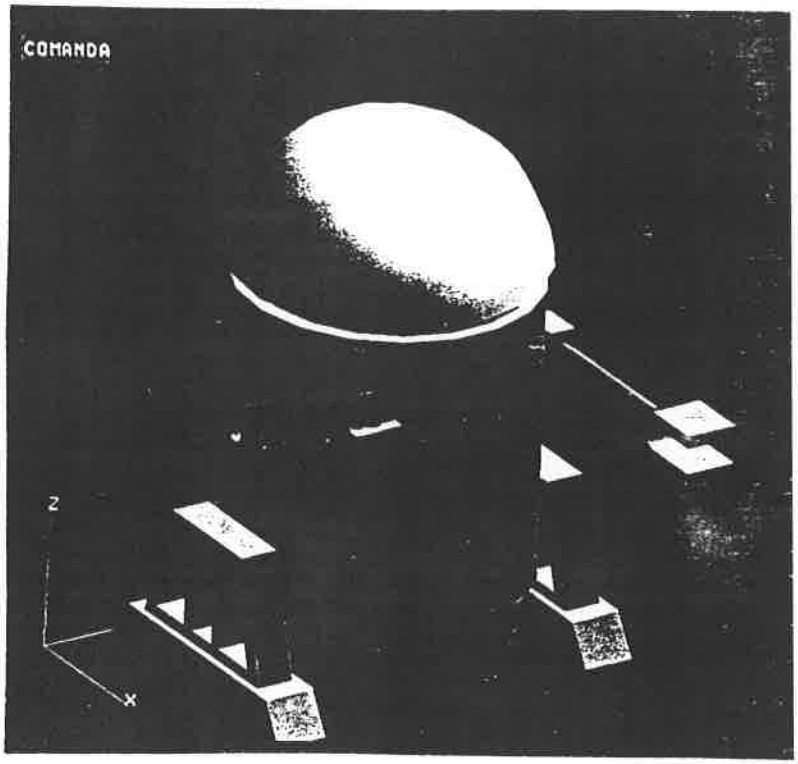


Figura 2.7 Conjunt amb un total de 448 cares. Es presenten visualitzacions per filferros, eliminació de linies ocultes y Z-buffer de linies amb suavització de Gouraud.