Eliminarea partilor nevizibile ale scenelor 3D din imagini

Prof. univ. dr. ing. Florica Moldoveanu

Eliminarea partilor nevizibile ale scenelor 3D, din imagini

Vizibilitatea obiectelor dintr-o scena 3D depinde de:

–Pozitia observatorului

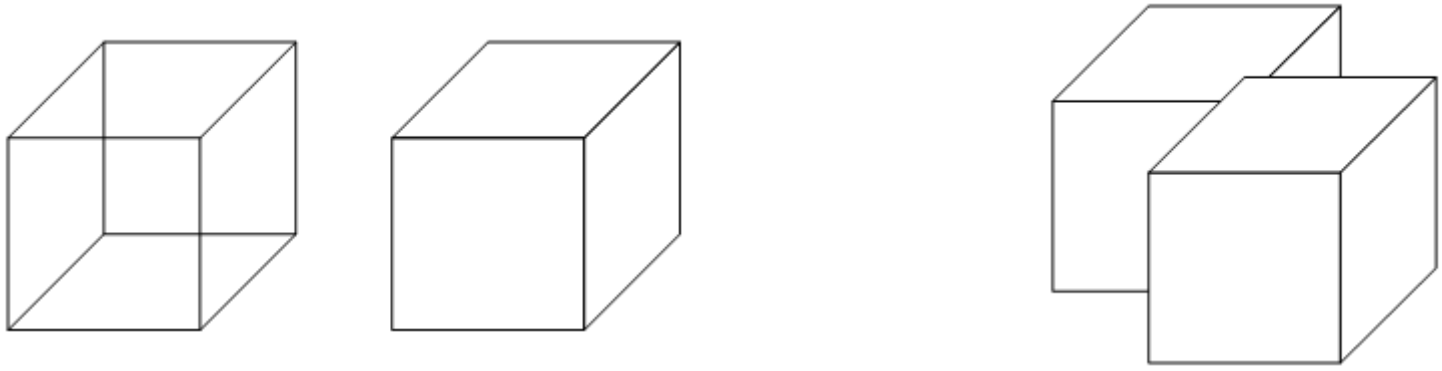
–Pozitionarea obiectelor din scena unul fata de celalalt

–Volumul vizual

Partile nevizibile ale unui obiect 3D sunt:

–Fețe auto-obturate

–Parti obturate de alte obiecte, aflate in fata obiectului in raport cu observatorul



Eliminarea partilor nevizibile din scenele 3D

Terminologie:

- ***Object culling*** – eliminarea din banda grafica a obiectelor sau grupurilor de obiecte care sunt in afara volumului vizual
  - nu este efectuata de OpenGL
  - poate fi efectuata prin algoritmi implementati intr-o biblioteca folosita de aplicatia grafica 3D (motorul grafic)
- ***Back face culling*** – eliminarea din banda grafica a fețelor auto-obturate ale obiectelor (poate fi efectuata de OpenGL)
- ***Hidden surface removal*** – eliminarea partilor nevizibile ale fețelor obiectelor:
  - este efectuata de GPU, ca operatie raster la nivel de fragment (z-buffer)
  - poate fi implementata si la nivel de poligoane, prin algoritmi implementati intr-o biblioteca folosita de aplicatia grafica 3D (motorul grafic)

Determinarea fetelor auto-obturate pentru poliedre convexe (Back face culling)(1)

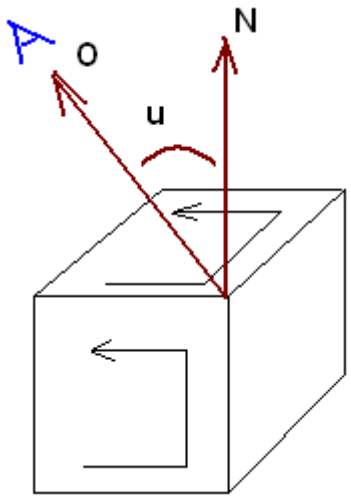
- Poate reduce substantial numarul de poligoane procesate in banda grafica.

Algoritmul:

-Observatorul si obiectul sunt raportati la acelasi sistem de coordonate.

-Conturul fiecarei fețe a obiectului este orientat in sens trigonometric atunci cand poliedrul este vazut din exterior.

-Observatorul este situat in afara obiectului.



Fața este vizibila daca : $0 <= u < 90^o$

sau

$0 < \cos(u) <= 1$

N: normala la fața

O: vectorul orientat catre observator

Determinarea fetelor auto-obturate pentru poliedre convexe(2)

Produsul scalar:

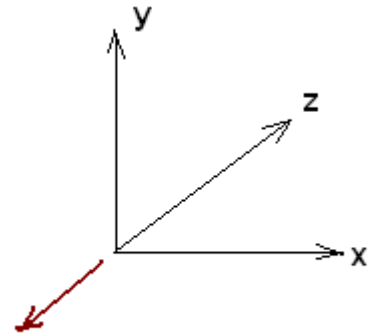
$N \bullet O = ||N|| * ||O|| * \cos(u) \rightarrow \cos(u) = N \bullet O / (||N|| * ||O||) = Nu \bullet Ou$

Conditia de vizibilitate: Nu • Ou > 0 (produsul scalar la versorilor)

Determinarea fetelor auto-obturate poate fi efectuata:

1. In sistemul coordonatelor globale (in care este definit si observatorul)
2. In sistemul de coordonate observator
3. In sistemul coordonatelor de decupare (dupa aplicarea transformarii "model-view-projection")

In cazul 3:



Nu • Ou = [nx, ny, nz] • [0, 0, -1] = - nz

Conditia de vizibilitate devine: nz <0

O[0, 0, -1] - este acelasi pentru orice punct al unui obiect

Observatorul este la infinit, pe axa z negativa, si priveste in directia axei Z pozitive

Determinarea fetelor auto-obturate pentru poliedre convexe(3)

Calculul normalei la o față a poliedrului

Fie V1(x1, y1, z1), V2(x2, y2, z2), V3(x3, y3, z3), 3 varfuri succesive ale conturului fetei.
Normala la față se poate obtine calculand produsul vectorial: (V1-V2)x(V2-V3)

$$\mathbf{N} = (V1-V2)x(V2-V3) = [(z3-z2)*(y2-y1) - (z2-z1)*(y3-y2)]*\mathbf{i}$$
$$- [(x2-x1)*(z3-z2) - (x3-x2)*(z2-z1)*\mathbf{j}$$
$$+[(x2-x1)*(y3-y2) - (x3-x2)*(y2-y1)]*\mathbf{k}$$

unde **i**, **j**, **k** sunt versorii directiilor axelor sistemului de coordonate carteziene 3D

Normalele:

- Pot fi atasate varfurilor in aplicatia OpenGL → sunt transformate de OpenGL in sistemul coordonatelor de decupare.
- Nu sunt atasate varfurilor: sunt calculate de OpenGL (daca s-a cerut eliminarea fetelor auto-obturate) folosind varfurile transformate in sistemul coordonatelor de decupare.

Determinarea fetelor auto-obturate pentru poliedre convexe(4)

Pentru eliminarea fetelor auto-obturate de catre OpenGL:

glCullFace(GL_FRONT/ GL_BACK/GL_FRONT_AND_BACK);

- eliminare fete: din fata / din spate / toate (nu vor fi afisate fețe, ci numai alte primitive: puncte, linii)

glFrontFace(GL_CCW/GL_CW);

- specifica orientarea fetelor din fata(in spatiul coordonatelor globale): trigonometrica/sensul acelor de ceas

Daca ciclul de varfuri al fiecarei fete este orientat trigonometric atunci cand obiectul este privit din exteriorul sau:

- atunci cand observatorul este in spatiul exterior obiectului se va apela glFrontFace(GL_CCW)
- atunci cand observatorul este in spatiul interior obiectului se va apela glFrontFace(GL_CW)

(vazute din interior, fetele obiectului au conturul orientat anti-trigonometric)

Algoritmul z-buffer(1)

-Algoritm de eliminare a partilor nevizibile ale poligoanelor (fețelor obiectelor)

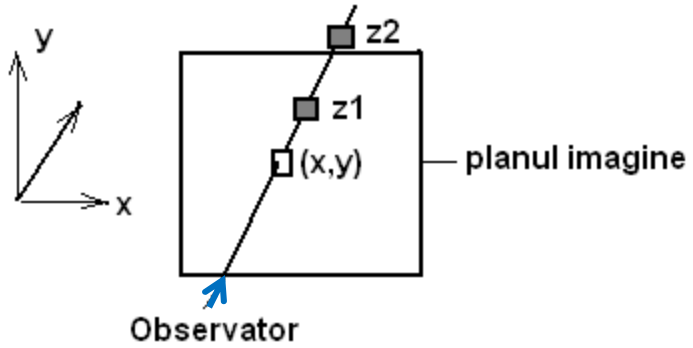
-Executat in timpul rasterizarii primitivelor (dupa transformarea de proiectie), de GPU:

-Observatorul este la infinit, pe axa z negativa

-Asupra primitivelor se efectueaza proiectie ortografica in planul XOY

Doua fragmente aflate pe acelasi proiector, rezultate din rasterizarea a doua primitive diferite, au coordonata z diferita:

- Fragmentul avand coordonata z mai mica va fi afisat in pixelul (x,y).



Algoritmul z-bufer(2)

- Primitivele grafice (poligoane, linii) in care a fost descompusa scena 3D sunt rasterizate

in ordinea in care au fost transmise de programul de aplicatie.

- Buffer-ul imagine este actualizat pe masura rasterizarii primitivelor, astfel:

daca (fragmentul curent se proiecteaza in pixelul (x,y) si
z-fragment < z-fragment afisat in pixelul (x,y))
atunci
actualizeaza culoarea pixelului (x,y) in bufer-ul imagine, la culoarea fragmentului curent

Rezulta: este necesar sa se memoreze coordonatele z ale fragmentelor afisate in pixelii imaginii

Z-buffer: - tablou bidimensional cu numar de elemente egal cu numarul de pixeli ai suprafetei de afisare

- in memoria placi grafice

Algoritmul z-bufer(3)

Algoritmul z-buffer:

*Initializeaza buffer-ul imagine la culoarea de fond

*Initializeaza Z-buffer la coordonata z a planului din spate al volumului vizual (z=1)

Pentru fiecare fragment f(x,y,z) rezultat din rasterizarea unei primitive

daca z< Z-buffer[y][x] atunci

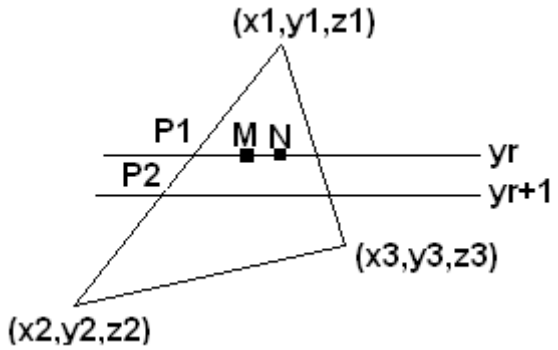
Z-buffer[y][x] = z

*actualizeaza culoarea pixelului (x,y) in buffer-ul imagine folosind culoarea fragmentului f

Algoritmul z-bufer(4)

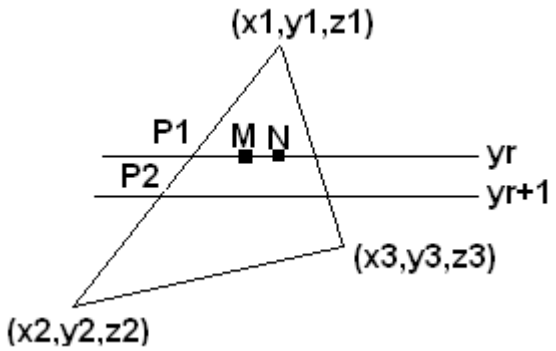
Calculul coordonatei z a unui fragment

-Ultima transformare din lantul de transformari eectuate asupra varfurilor 3D conserva coordonata z a fiecarui varf.



Coordonatele z ale punctelor (fragmentelor) P1, P2, M, N se obtin prin calcul incremental, in algoritmul de rasterizare.

Algoritmul z-bufer(5)



P1(xP1, yr, zP1), P2(xP2, yr+1, zP2)

m- panta laturii (x1,y1,z1) – (x2,y2,z2)

xP2 = xP1 + 1/m (vezi curs - rasterizare)

A*x + B*y + C*z + D = 0 ec planului poligonului

zP1 = (-A*xP1 - B*yr –D)/C

zP2 = (-A*xP2 - B*(yr +1) – D)/C = (-A(xP1 + 1/m) - B*yr –B – D)/C = zP1 + (-A/m –B) = zP1 + K1

M(xM, yr, zM), N(xM + 1, yr, zN)

zN = (-A*(xM + 1) – B*yr – D)/ C = zM + (-A/C) = zM + k2

Algoritmul z-bufer(6)

Aprecieri asupra algoritmului z-Buffer

1) Numarul de comparatii de valori z pentru un pixel: numarul de fragmente care se proiecteaza in acel pixel.

   Timpul de calcul tinde sa devina independent de numarul de poligoane: in medie, numarul de pixeli acoperiti de un poligon este invers proportional cu numarul de poligoane.

2) "Depth complexity": numarul de suprascrieri ale unui pixel in buffer-ul imagine, la generarea unui cadru imagine

   - ❖ calcul culoare fragment: model iluminare, utilizare texturi

3) Valorile z (in Z-bufer) sunt reprezentate prin numere intregi: 16, 32 biti → pierdere precizie

Algoritmul BSP (Binary Space Partitioning)(1)

Poate fi folosit pentru:

–Eliminarea obiectelor aflate in afara volumului vizual (object culling)

–Eliminarea partilor nevizibile ale fețelor obiectelor(hidden surface removal)

Intrarea: lista poligoanelor care compun scena 3D

(nu are importanta din care obiect face parte fiecare poligon)

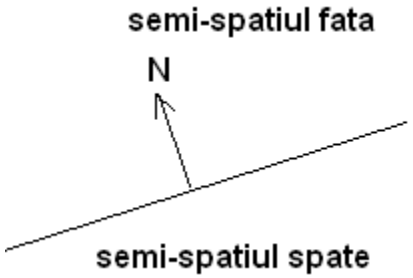
–Scena 3D este reprezentata printr-un arbore binar: arborele BSP

–Arborele BSP al scenei este independent de pozitia observatorului (reprezinta scena in sistemul coordonatelor globale).

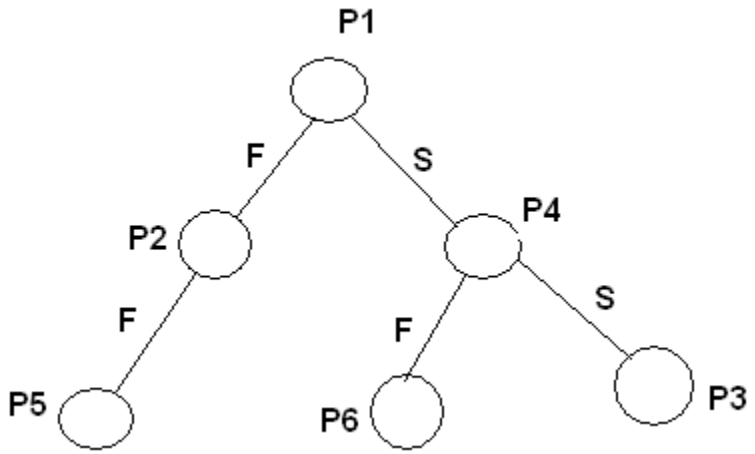
Algoritmul BSP (2)

Construirea arborelui BSP al unei scene

- Fiecare nod al arborelui corespunde unui plan de partitionare a spatiului 3D (de regula, planul unui poligon al scenei)



- Fiecare plan de partitionare imparte spatiul in 2 semi-spatii:
  - –cel din fata planului (de aceeasi parte cu normala la plan)
  - –cel din spatele planului
- Se incepe cu un poligon oarecare din lista (de regula primul), pentru care se creaza nodul radacina al arborelui
- Poligoanele din semi-spatiul “față” formeaza “lista-față”, care va genera subarborele “față” al nodului
- Poligoanele din semi-spatiul “spate” formeaza “lista-spate”, care va genera subarborele “spate” al nodului



- Se alege un poligon din lista-față si se creaza nodul radacina al subarborelui “față, etc.

Algoritmul BSP (3)

Afisarea arborelui BSP al unei scene

-Tine cont de pozitia observatorului

-Afisare "back-to-front": se incepe cu poligonul cel mai indepartat de observator

-Se porneste din radacina arborelui si se avanseaza in arbore pana la frunze, in functie de pozitia observatorului fata de planul atasat fiecarui nod

-Exemplu: se adauga observatorul in scena 3D a arborelui din figura precedenta

Eliminarea obiectelor nevizibile din banda grafica (Object culling):

- Daca planul de partitionare al unui nod nu intersecteaza volumul vizual, atunci numai subarborele aflat de aceeasi parte cu volumul vizual va fi afisat, celalalt subarbore fiind eliminat din banda grafica.

Algoritmul BSP (4)

Algoritmul de creare a arborelui BSP:

Arbore * creareBSP (Poligon * LP)

```
{ Poligon P, * ListaFata, * ListaSpate, * ListaNod;
  daca (LP este vida) return NULL;
  * alege un poligon P din LP;
  * elimina P din LP;
   ListaFata = ListaSpate = NULL;
   pentru (fiecare poligon Q din LP) executa
    {  daca (Q este in semispatiul fata al planului lui P) atunci
              * adauga Q in ListaFata
       altfel
            daca (Q este in semispatiul spate al planului lui P) atunci
             * adauga Q in ListaSpate
            altfel
               daca Q este in acelasi plan cu P atunci
                  * adauga  in ListaNod
```

Algoritmul BSP (5)

```
            altfel // Q este intersectat de planul lui P
               * divizeaza Q cu planul lui P
               * adauga fiecare poligon rezultat in ListaFata sau ListaSpate, in functie de pozitia sa
   } // pentru fiecare poligon
return combina(creareBSP(ListaFata), creareBSP(ListaSpate);
}
```

Afisarea arborelui BSP

```
void afisareBSP(arbore * A, Pozitie Observator)
{ daca (! A) return;
  daca (Observator este in semispatiul fata al planului radacinii) atunci
    { afisareBSP(A->spate); *afisare poligoane din nodul radacina; afisareBSP(A->fata);}
  altfel
    { afisareBSP(A->fata); *afisare poligoane din nodul radacina; afisareBSP(A->spate);}
}
```

Algoritmul BSP (6)

Aprecieri:

- Arborele nu trebuie sa fie construit pentru fiecare cadru imagine: avantaj pentru scenele statice
- Generarea unui cadru = executia functiei de afisare
- Permite eliminarea din banda grafica a unui numar mare de poligoane care sunt in afara volumului vizual(Object culling)
- Diverse adaptari ale arborelui pentru cazul obiectelor in miscare in scena 3D
- Numarul de suprascrieri ale unui pixel imagine este mai redus decat in algoritmul Z-buffer

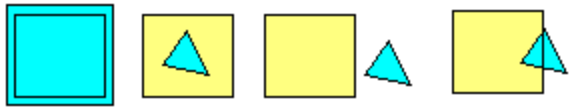
- Sistemele grafice actuale, OpenGL si Direct3D, nu construiesc arborele BSP al unei scene.
- Construirea si afisarea arborelui BSP sunt functii de management al scenei 3D, implementate de regula intr-un motor grafic 3D.

Algoritmul Warnock – bazat pe divizarea spatiului de afisare(1)

Intrare: lista poligoanlor care compun scena 3D, transformate in spatiul imagine

- **Față de o zona dreptunghiulara a spatiului de afisare, poligoanele din lista curenta pot fi in una dintre categoriile:**

1. Poligon acoperitor, care acopera intreaga zona
2. Poligon inclus in zona
3. Poligon disjunct cu zona
4. Poligon intersectat de zona: se divide in poligon inclus + poligon/poligoane disjunct/disjuncte



- **In zona curenta sunt vizibile:**

- Poligoanele incluse si unul dintre cele acoperitoare

- **Lista poligoanelor vizibile se sorteaza crescator dupa coordonata zmin a fiecarui poligon**: primul din lista este cel mai apropiat de observator → toate poligoanele care urmeaza in lista dupa unul acoperitor pot fi eliminate din lista.
- **Algoritmul este recursiv:** la un apel se transmit functiei de afisare: zona de afisare si lista de poligoane de afisat. Iesirea din recursivitate:
  - lista de poligoane este vida
  - lista are un singur element, care este un poligon inclus in zona
  - primul in lista este un poligon acoperitor al zonei
  - zona are un singur pixel

Algoritmul Warnock(2)

Daca iesirea din recursivitate nu este posibila, se imparte zona curenta in 4 subzone egale si se apeleaza functia recursiv pentru fiecare zona, cu lista poligoanelor vizibile.

void Warnock(lista_poligoane L, RECT zona)
{ se formeaza lista V, a poligoanelor vizibile in zona, plecand de la L
daca V este vida return;
daca V are un singur element, P, atunci:
daca P este inclus in zona, atunci:
- se afiseaza zona in culoarea fondului
- se afiseaza poligonul in culoarea proprie

altfel (P este poligon acoperitor)
- se afiseaza zona in culoarea poligonului

return;
daca zona are un singur pixel (x,y), atunci:
- se calculeaza coordonata z in pixelul (x,y) pentru toate poligoanele din lista V;
- se afiseaza pixelul in culoarea poligonului cu zmin in (x,y);

return;

Algoritmul Warnock(3)

se sorteaza lista V crescator dupa zmin poligoane (primul in lista este cel mai apropiat de observator);
daca primul in lista V este un poligon acoperitor, atunci:
- se afiseaza zona in culoarea poligonului;
return;
se elimina din V toate poligoanele care urmeaza dupa primul poligon acoperitor;

```
Warnock( V, z1);
Warnock( V, z2);
Warnock( V, z3);
Warnock( V, z4);
return;
}
```

Aprecieri:

-Principalul efort de calcul este clasificarea poligoanelor fata de zona curenta: intersectii poligoane – zona pentru calcul varfuri si informatii asociate varfurilor (normale, culori, coord textura)

- La fiecare apel recursiv, lista poligoanelor scade foarte mult (deci si efortul de clasificare)

- Algoritmul poate fi paralelizat cu usurinta.

- Nu sunt suprascrisi pixelii imaginii (ca in Z-buffer) – exceptand cazul poligon inclus inscris peste fond.

Algoritmul pictorului(1)

–Intrare: lista poligoanelor care alcatuiesc scena 3D, transformate in spatiul de afisare

Forma generala a algoritmului

1. Se calculeaza “extensia” fiecarui poligon din listă pe axele OX, OY, OZ: paralelipipedul incadrator al poligonului, cu fețele paralele cu planele principale ale sistemului de coordonate

2. Se ordoneaza poligoanele crescator dupa coordonata zmin a fiecarui poligon: primul in lista va fi cel mai apropiat de observator

3. Se descompun poligoanele ale caror extensii pe axa OZ se suprapun, astfel incat extensiile lor pe axa OZ sa fie disjuncte

4. Se afiseaza (transmit in banda grafica) poligoanele incepand cu ultimul din lista

Algoritmul pictorului(2)

Pasul 3. Se descompun poligoanele ale caror extensii pe axa OZ se suprapun

- Este necesar numai daca extensiile pe axa OZ se suprapun.
- Sunt multe aplicatii in care acest pas nu este necesar, poligoanele fiind amplasate in plane de Z-constant: cartografie, generarea straturilor circuitelor imprimate, etc.
- Poate fi optimizat, stiind ca nu intotdeauna atunci cand extensiile pe axa OZ se suprapun este necesara descompunerea poligoanelor; testele se efectueaza progresiv, in functie de complexitatea calculelor presupuse

- **Principalul efort de calcul:** sortarea listei de poligoane si descompunerea, daca este necesara.

Tehnici de eliminare a obiectelor nevizibile (1) (Object culling/Visibility culling)

❖ Scopul: de a impiedica rasterizarea obiectelor care sunt complet nevizibile

1) Eliminarea fețele auto-obturate

2) Se doreste ca eliminarea sa se efectueze la nivel de obiecte sau grupuri de obiecte→ aceasta presupune reprezentarea scenei printr-o structura de date spatiala.

Structuri de date spatiale pentru managementul scenei 3D

- Ierarhie de volume incadratoare
- Arbore BSP
- Arbore octal

Tehnici de eliminare a obiectelor nevizibile (2)
1. Scena- Ierarhie de volume incadratoare

- **Scena este reprezentata printr-un arbore avand in noduri volume incadratoare** : sfera, paralelipipedul aliniat cu axele, poliedrul convex minimal incadrator, cilindrul, elipsoidul.

- Volumele incadratoare ale obiectelor sunt frunzele arborelui
- Se grupeaza volumele incadratoare ale obiectelor, rezultand noduri ale arborelui, fiecare avand asociat volumul incadrator al grupului
- Se continua gruparea pana ce se ajunge la nodul radacina, al carui volum incadrator incadreaza intreaga scena 3D

- **Eliminarea obiectelor nevizibile:**

  - testele de vizibilitate incep cu radacina arborelui

  - daca volumul incadrator al unui nod este complet in afara volumului vizual, atunci intregul grup de obiecte din subarborele nodului este exclus din banda de redare.

Tehnici de eliminare a obiectelor nevizibile (3)
2. Scena - Arbore BSP(1)

❖ **Planele de partitionare sunt planele poligoanelor**

- Arborele scenei se construieste ca in algoritmul BSP pentru eliminare suprafetelor nevizibile

❖ **Eliminarea obiectelor nevizibile:**

daca planul de divizare atasat unui nod nu intersecteaza volumul vizual, atunci volumul vizual este situat de o parte a planului iar subarborele atasat nodului care este situat de cealalta parte a planului este complet nevizibil si poate fi eliminat din procesul de redare.

– Eliminarea se poate face in spatiul observator, folosind volumul vizual definit de aplicatie (trunchi de piramida sau paralelipiped) sau in spatiul coordonatelor de decupare, inainte de decupare.

Tehnici de eliminare a obiectelor nevizibile (3)
3. Scena - Axis Aligned BSP Tree

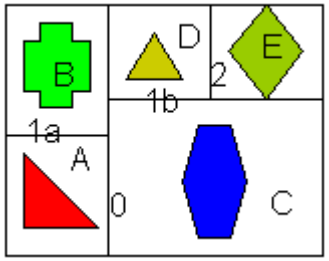
- **Planele de partitionare sunt aliniate cu axele sist de coord 3D -**perpendiculare pe cele 3 axe: Axis-Aligned BSP Tree (AA-BSP)
- AA-BSP tree este un k-d tree (*k-dimensional tree):* arbore binar in care fiecare nod corespunde unei coordonate intr-un spatiu k-dimensional. Pentru k=3, un nod va corespunde uneia dintre coordonatele x, y sau z. De exemplu, daca planul de partitionare al unui nod este perpendicular pe OX, atunci nodul este asociat coordonatei x a planului.
- **Construirea arborelui:**

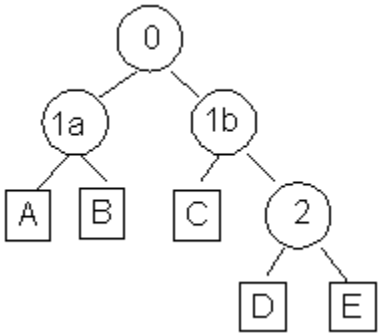
Scena este divizata recursiv prin plane paralele cu planele sistemului de coordonate, intr-o ordine fixa, tinand cont si de geometria scenei; de exemplu, in pasul 1 se alege un plan paralel cu XOY, in pasul 2 un plan paralel cu YOZ, in pasul 3 un plan paralel cu XOZ, in pasul 4 un plan paralel cu XOY, s.a.m.d.

- Se pleaca de la volumul incadrator al scenei (aliniat cu axele)
- In fiecare pas rezulta 2 volume incadratoare mai mici.
- Alegerea planelor se face astfel incat numarul de obiecte din fiecare semispatiu sa fie acelasi sau diferit cu 1, pentru ca arborele sa fie echilibrat
- Alegerea planelor se poate face astfel incat numarul de intersectii dintre planele de partitionare si obiectele scenei sa fie minimizat.

Tehnici de eliminare a obiectelor nevizibile (3)
3. Scena- Axis Aligned BSP Tree (cont)

Exemplu de divizare
cu plane aliniate cu axele, in 2D:





Pasul 1: divizare cu un plan vertical: rezulta sub-arborii 1a si 1b
Pasul 2: - subarborele 1a:
divizare cu un plan orizontal: rezulta frunzele A, B
se iese din recursivitate
- subarborele 1b:
divizare cu un plan orizontal: rezulta nodul frunza C si nodul 2
se iese din recursivitate
Pasul 3: - subarborele 2:
divizare cu un plan vertical: rezulta frunzele D, E
se iese din recursivitate

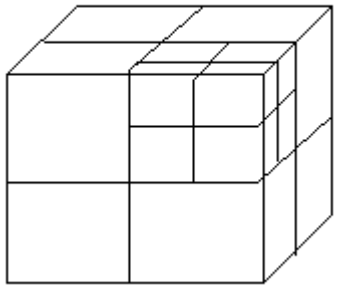
❖ **Eliminarea obiectelor nevizibile :** la fel ca in cazul ierarhiei de volume incadratoare

➢ daca volumul incadrator al unui nod este complet in afara volumului vizual, intregul grup de obiecte reprezentate prin subarborele nodului este exclus din banda grafica

Tehnici de eliminare a obiectelor nevizibile (3)
4. Scena - Arbore octal

❖ Intrarea: lista poligoanelor scenei

- In fiecare pas se divide volumul/subvolumul incadrator in 8 subvolume egale.



- Fiecare nod al arborelui are 8 copii.
- Subdivizarea spatiului are loc pana cand numarul de poligoane din fiecare nod frunza este mai mic decat un prag dat.
- Poligoanele sunt memorate in nodurile frunza; un poligon poate fi asociat mai multor noduri, care corespund unor subcuburi adiacente.

❖ Arborele octal se poate folosi pentru reprezentarea ierarhica a scenei 3D: fiecare obiect este asociat cu cel mai mic subcub al arborelui octal, care-l incadreaza complet.

❖ Eliminarea obiectelor nevizibile din banda grafica se efectueaza intersectand subcuburile arborelui cu volumul vizual:

- se elimina toate obiectele din subcuburile care sunt in afara volumului.
- testele incep de la radacina arborelui: daca subcubul unui nod este in afara volumului vizual se elimina toate obiectele din subarborele nodului.