# Eliminarea partilor nevizibile ale scenelor 3D din imagini -2

*Prof. univ. dr. ing. Florica Moldoveanu*

# Algoritmul BSP pentru eliminarea partilor nevizibile ale primitivelor

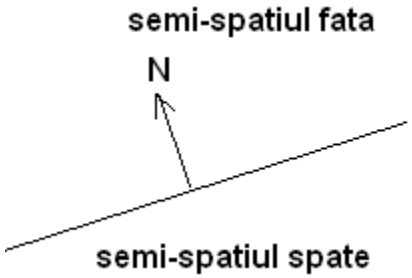
- Scena 3D este reprezentata printr-un arbore binar, denumit sugestiv arbore BSP (**Binary Space Partitioning)**
- Initial, arborele BSP a fost folosit pentru operatia de *object culling*.
- Ulterior a fost folosit şi ca algoritm pentru eliminarea părților nevizibile ale primitivelor grafice (hidden surface removal), fiind folosit în acest scop în jocuri ca *Doom* şi *Quake*, pentru a creste viteza de redare a scenelor 3D.
- Algoritmul este eficient pentru eliminarea partilor nevizibile ale poligoanelor. Eliminarea partilor nevizibile ale liniilor se rezolva eficient la nivel de fragment.
- **Arborele BSP al unei scene este independent de pozitia observatorului - reprezinta scena în sistemul coordonatelor globale.**

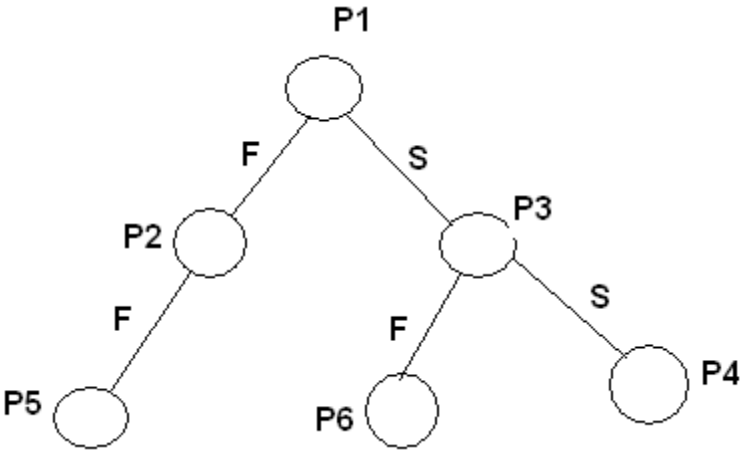
**Intrarea algoritmului de construire a arborelui**: lista poligoanelor care compun scena 3D, în orice ordine; nu are importanta din care obiect face parte fiecare poligon.

# Construirea arborelui BSP

## Construirea arborelui BSP al unei scene – PA BSP tree

- Fiecare nod al arborelui corespunde unui plan de partitionare a spatiului 3D
- PA (Polygon aligned) BSP tree: poligoanele de partitionare sunt planele poligoanelor scenei
- Fiecare plan de partitionare împarte spatiul în 2 semi-spatii:
  - cel din faţa planului (de aceeasi parte cu normala la plan)
  - cel din spatele planului

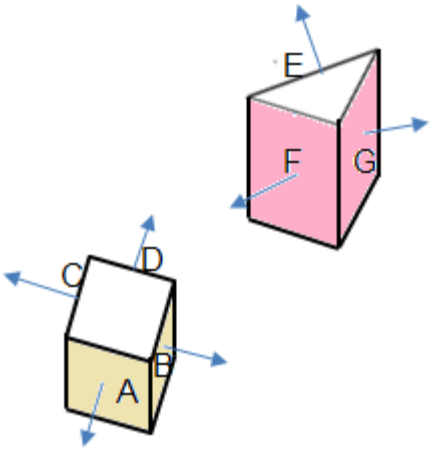


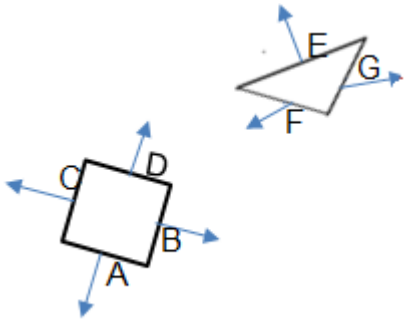

1. Se începe cu un poligon oarecare din lista (de regula prim pentru care se creaza nodul radacina al arborelui
   - Poligoanele aflate in semi-spatiul “faţă” formeaza “lista-faţă”, care va genera subarborele “faţă” al nodului
   - Primitivele din semi-spatiul “spate” formeaza “lista-spate”, care va genera subarborele “spate” al nodului.
2. Se alege un poligon din lista-faţă si se creaza nodul radacina al subarborelui “faţă, etc.

# Construirea arborelui BSP – exemplu (1)

Fie o scena 3D compusa din 2 camere, reprezentate numai prin peretii laterali, şi vederea sa de sus (proiectie ortografica). Sunt reprezentate şi normalele la fețele laterale.




Scena 3D

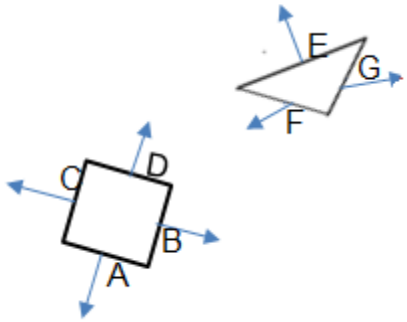


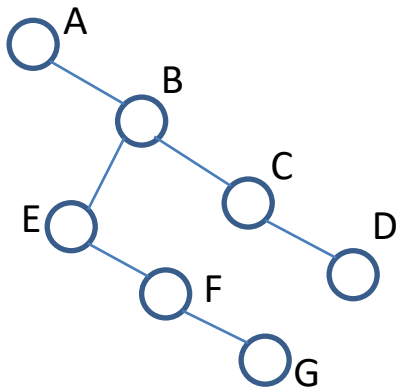

Vederea de sus a scenei

Consideram ca la construirea arborelui poligoanele sunt tratate în ordinea din lista:

A,B,C,D,E,F,G.

# Construirea arborelui BSP – exemplu (2)




1. Se incepe cu poligonul A, care devine radacina arborelui.
   Se elimina A din lista de poligoane
   Lista față: vida. Lista spate: B,C,D,E,F,G
2. Se alege poligonul B din lista spate si se creaza nodul B in arbore. Se elimina B din lista.
   Lista față: E,F,G. Lista spate: C,D
3. Se alege poligonul E din lista față si se creaza nodul E. Se elimina E din lista.
   Lista față: vida. Lista spate: F,G
4. Se alege poligonul F din lista spate si se creaza nodul F. Se elimina F din lista.
   Lista față: vida. Lista spate: G
5. Se alege G din lista spate si se creaza nodul G. Se elimina G din lista.
   Lista față: vida. Lista spate: vida
6. Se alege poligonul C din lista spate a nodului B si se creaza nodul C. Se elimina C din lista.
   Lista față: vida. Lista spate: D
7. Se alege D din lista spate si se creaza nodul D
   Lista față: vida. Lista spate: vida

# Algoritmul de creare a arborelui BSP (1)

**arbore * creareBSP (Poligon * LP)**

{ Poligon P, * ListaFata, * ListaSpate, * ListaNod;

daca (LP este vida) return NULL;

ListaFata = ListaSpate = ListaNod = NULL;

* alege un poligon P din LP;

* elimina P din LP; adauga P la ListaNod;

**pentru (fiecare poligon Q din LP) executa**

{ **daca** (Q este in semispatiul față al planului lui P) **atunci**

* adauga Q in ListaFata

**altfel**

# Algoritmul de creare a arborelui BSP (2)

**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 Q in ListaNod

**altfel** // Q este intersectat de planul lui P

* divizeaza Q cu planul lui P
* adauga fiecare poligon rezultat din intersectie in ListaFata sau ListaSpate, in functie de pozitia sa

} // pentru fiecare poligon

arbore * radacina = combina(creareBSP(ListaFata), creareBSP(ListaSpate));

*ataseaza ListaNod la radacina;

return radacina;

}

# Afisarea scenei reprezentata prin arbore BSP

❖ **Tine cont de pozitia observatorului**

- **Afişarea "back-to-front"** - poligoanele sunt trimise in banda grafica in ordinea: de la cel mai indepartat de observator pana la cel mai apropiat de observator.

❖ Poligoanele mai apropriate de observator suprascriu parti din poligoanele mai indepartate.

- Se porneste din radacina arborelui si se avanseaza in arbore pana la frunze, in fiecare nod coborandu-se in subarborele nodului care se afla de partea opusa observatorului fata de planul nodului.

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

# Afisarea scenei reprezentata prin arbore BSP

- **Afişarea "front-to-back"** - poligoanele sunt trimise in banda grafica in ordinea: de la cel mai apropiat de observator pana la cel mai îndepartat de observator.

❖ Poligoanele mai indepartate de observator sunt "decupate" de poligoanele deja afisate, prin testul de adancime (pixelii acoperiti de poligoanele din fata nu sunt suprascrisi).

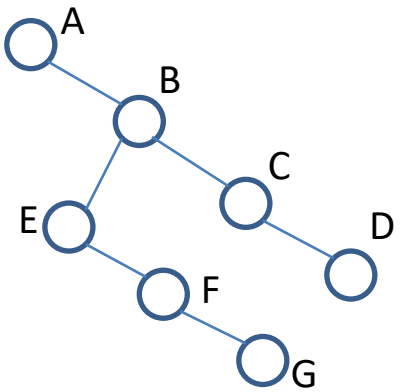
- Se porneste din radacina arborelui si se avanseaza in arbore pana la frunze, in fiecare nod coborandu-se in subarborele nodului care se afla de aceeasi parte cu observatorul fata de planul nodului.

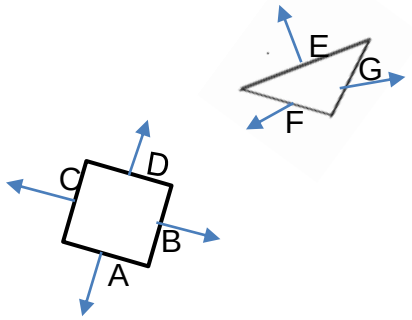
```
void afisareBSP_front_to_back(arbore * A, Pozitie Observator)
{ daca (! A) return;
  daca (Observator este in semispatiul fata al planului radacinii arborelui A) atunci
    { afisareBSP(A->fata); *afisare poligoane din nodul radacina; afisareBSP(A->spate);}
  altfel
    { afisareBSP(A->spate); *afisare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

# Afisarea arborelui BSP – exemplu (1)

## Afişare “din față în spate”

Se tine cont de pozitia observatorului în scena vizualizata.

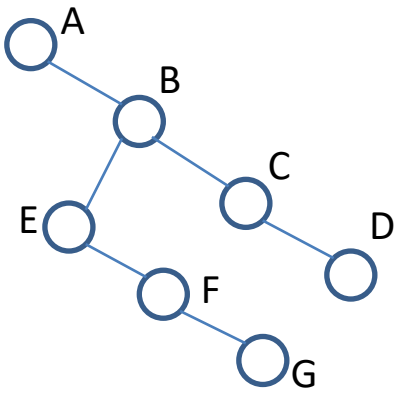


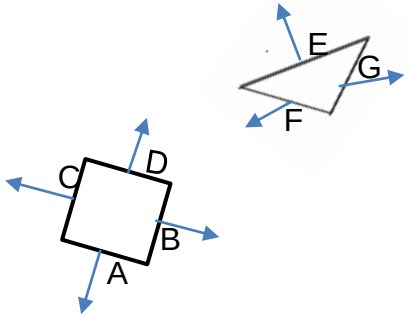




1. Fata de planul nodului A observatorul se afla in semispatiul spate.
   Se coboara in subarborele spate al nodului A.
2. Fata de planul nodului B observatorul se afla in semispatiul față.
   Se coboara in subarborele față al nodului B.
3. Fata de planul nodului E observatorul se afla in semispatiul față.
   Nodul E nu are subarbore față. **Se afiseaza poligonul din nodul E**.
   Se coboara in subarborele spate al nodului E.
4. Fata de planul nodului F observatorul se afla in semispatiul spate.
   Se coboara in subarborele spate al nodului F.

# Afisarea arborelui BSP – exemplu (2)

## Afişare “din față în spate”




5. Nodul G este frunza → **se afiseaza poligonul din nodul G**.
6. **Se afiseaza poligonul din F, apoi cel din B**.
   Se coboara in subarborele spate al nodului B
7. Fata de planul nodului C observatorul se afla in semispatiul spate.
   Se coboara in subarborele spate al nodului C
8. Nodul D este frunza. **Se afiseaza poligonul din nodul D apoi cel din nodul C.**
   **Se afiseaza poligonul din nodul A.**

Rezulta urmatoarea ordine in care sunt trimise poligoanele scenei 3D in banda grafica, la afisarea scenei “din față in spate”:

**E, G, F, B, D, C, A.**

# Algoritmul BSP - aprecieri (1)

- Arborele nu trebuie sa fie reconstruit sau modificat pentru fiecare cadru imagine, daca scena nu s-a modificat: avantaj pentru scenele statice.
- Modificarea pozitiei observatorului → numai executia functiei de afisare, daca scena nu s-a modificat.
- Pentru obiectele dinamice se pot face diverse adaptari ale algoritmului; de ex:

1. Pentru fiecare cadru imagine se insereaza in arborele BSP cu obiectele statice, noduri corespunzand obiectelor dinamice, in functie de pozitia fiecaruia.
2. Se construieste arborele BSP numai cu obiectele statice iar obiectele dinamice sunt trimise direct in banda grafica, eliminarea partilor obturate de acestea fiind realizata la rasterizare.

- Variantă avantajoasa pentru scenele cu obiecte statice foarte mari; exemplu, pereti, elemente statice dintr-un mediu extern, etc.

# Algoritmul BSP - aprecieri (2)

- **In cazul afisarii back-to front**, poligoanele mari (ex. pereti) din spate sunt afisate primele si apoi părți mari din ele sunt obturate de poligoanele mai apropriate de observator, prin suprascrierea pixelilor → timp consumat inutil cu calculul culorilor pixelilor suprascrisi.
- Nu este necesar testul de adâncime la afisarea fragmentelor (fragmentele care se proiecteaza in acelasi pixel sunt suprascrise), de aceea se poate dezactiva testul de adâncime:

  glEnable(GL_DEPTH_TEST) / glDisable(GL_DEPTH_TEST)
- Afisarea scenei in 2 pasi, prin tehnica “Deffered rendering (Deffered shading)”, reduce consumul de timp cu calculul culorilor: calculul culorilor fragmentelor pe baza modelului de iluminare se efectueaza numai pentru fragmentele vizibile.

# Algoritmul BSP – aprecieri (3)

- **In cazul afisarii front-to back**, poligoanele apropriate de observator sunt afisate primele, apoi cele din spate sunt “decupate” de poligoanele deja afisate, prin testul de adancime (pixelii acoperiti de poligoanele din fata nu sunt suprascrisi).
- Deoarece nu au loc suprascrieri de pixeli, majoritatea motoarelor grafice care folosesc BSP utilizeaza afisarea front-to-back.
- Afisarea scenei prin tehnica “Deffered rendering (Deffered shading)”, reduce consumul de timp cu calculul culorilor si in acest caz:

  Forward rendering (implicit in banda grafica). La rasterizare, pentru fiecare fragment:

  - calcul culoare fragment
  - test de adancime

  deci, culorile fragmentelor poligoanelor mai indepartate se calculeaza chiar daca nu se suprascriu pixelii in care se proiecteaza.

Construirea si afisarea arborelui BSP sunt functii de management al scenei 3D, implementate de regula intr-un motor grafic 3D.

# Algoritmul pictorului(1)

– Algoritm din categoria “Hidden surface removal”.

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

– Observatorul este la infinit pe axa -OZ

– Ideea: eliminarea partilor nevizibile se rezolva prin afisare “din spate in fata”, la fel ca in pictura.

– Eficient pentru aplicatiile in care primitivele grafice sunt situate in zone disjuncte pe axa de adancime (OZ)

**Forma generala a algoritmului**

1. Se calculeaza “extensia” fiecarui poligon din listă pe axele OX, OY, OZ (xmin,ymin,zmin – xmax, ymax, zmax): paralelipipedul incadrator al poligonului, cu feţele paralele cu planele principale.

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 transmit in banda grafica poligoanele incepand cu ultimul din lista (cel mai indepartat de observator).

# Algoritmul pictorului(2)

**Pasul 3. Se descompun poligoanele ale caror extensii 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; descompunerea nu este necesara daca:

  - extensiile pe axa OX nu se suprapun

  - extensiile pe axa OY nu se suprapun

  - proiectiile poligoanelor nu se suprapun in imagine

- 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 - Frustum culling -

❖ Scopul: eliminarea din banda grafica a obiectelor sau grupurilor de obiecte care sunt complet in afara volumului vizual

❖ Se bazeaza pe reprezentarea scenei printr-o structura de date spatiala.

**Structuri de date spatiale pentru managementul scenei 3D**

- Ierarhie de volume incadratoare (BVH)
- Arbore PA-BSP
- Arbore AA-BSP
- Arbore octal
- ş.a.

- Managementul scenei este realizat in cadrul unui motor grafic 3D, implementat peste OpenGL / Direct 3D/Vulkan.

- Structurile de date sunt utile si pentru alte operatii: mai sunt numite "**structuri de date de accelerare**".

# 1. Scena- Ierarhie de volume incadratoare (1) (Bounding Volume Hierarchy - BVH)

- **Scena este reprezentata printr-un arbore care are in fiecare nod un volum încadrator pentru un obiect sau un grup de obiecte**: sfera, paralelipipedul aliniat cu axele (Axis-Aligned minimum Bounding Box: AABB), poliedrul convex minimal incadrator, cilindrul, elipsoidul.

**Construirea arborelui:**

1. **Top-down** – Se incepe cu nodul radacina, avand asociat volumul incadrator al scenei

- Se partitioneaza setul de obiecte in 2 sau mai multe subseturi si se aloca fiecare subset unui volum incadrator – rezulta primul nivel al arborelui. Volumele incadratoare trebuie sa fie disjuncte sau minimal suprapuse.
- Se continua procedura recursiv la nivelul fiecarui grup de pe nivelul anterior pana cand se ajunge la un singur obiect (sau primitiva) pe nod – se creaza un nod frunza, avand asociat volumul incadrator al obiectului.
- Metoda este simpla dar, in general, nu produce arbori optimi.

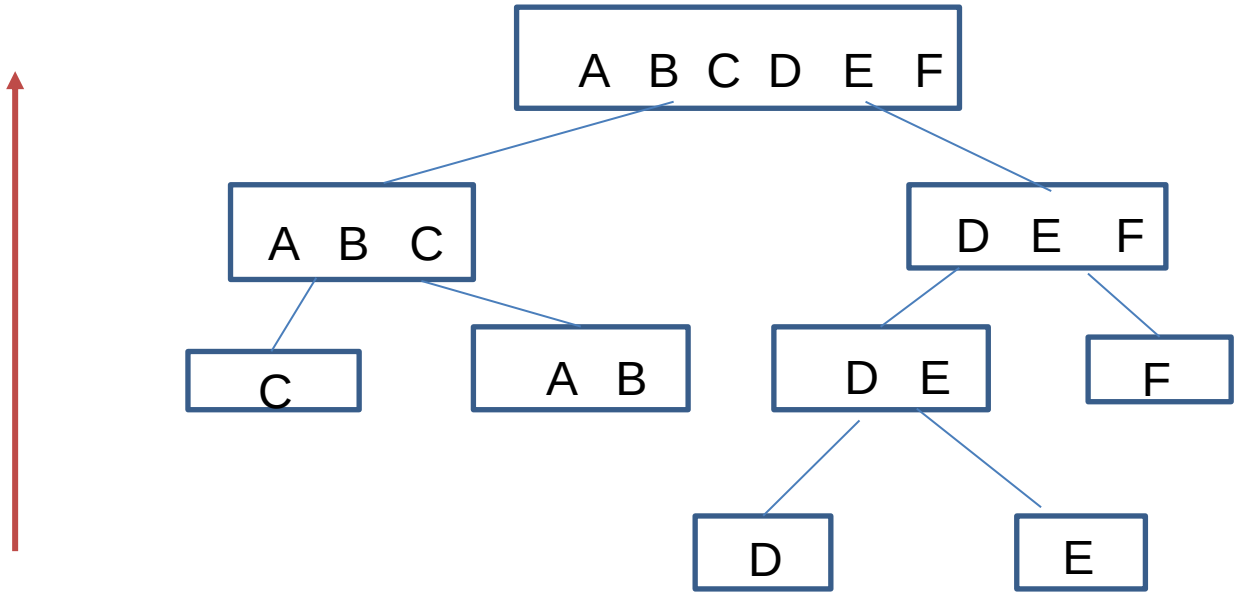
# *Ierarhie de volume incadratoare (2)*

Construirea arborelui – cont.

2. **Bottom-up**

- Se defineste un volum incadrator pentru fiecare obiect sau primitiva – nodurile frunza.
- Se grupeaza 2 sau mai multe volume nod frunza, rezultand noduri de pe nivelul urmator
- Se continua gruparea pana cand totul este grupat intr-un singur volum – nodul radacina

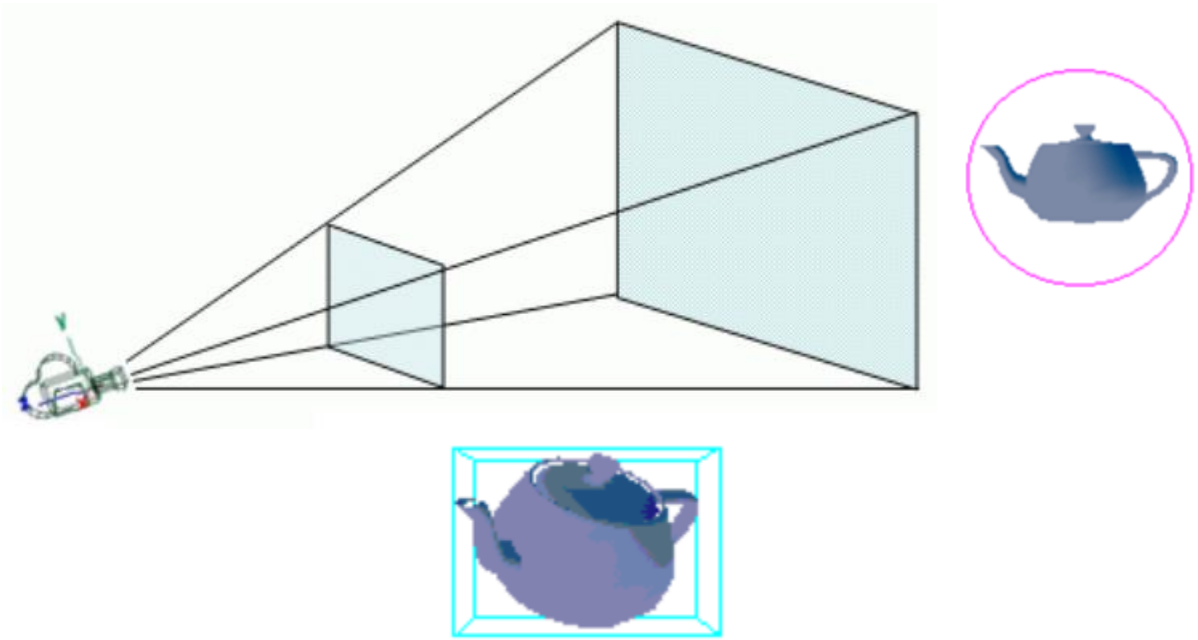
❖Construirea bottom-up produce arbori mai eficienti decat cea top-down.



# Ierarhie de volume incadratoare (3)

- **Afisarea scenei 3D folosind arborele BVH**

Daca volumul incadrator al unui nod este in afara volumului vizual atunci obiectele din subarborele său **NU** sunt trimise in banda grafica.



# Ierarhie de volume incadratoare (4)

❖ **Afisarea scenei 3D folosind arborele BVH**

- testele de vizibilitate incep cu radacina arborelui;

- **daca** volumul incadrator al nodului curent este complet in afara volumului vizual

**atunci** se revine pe nivelul anterior al arborelui (intregul grup de obiecte din subarborele nodului curent este exclus din banda grafica)

**altfel** se coboara in arbore;

- **daca** se ajunge la o frunza al carei volum incadrator nu este in afara volumului vizual,

**atunci** - grupul de primitive (obiectul) asociat frunzei este trimis in banda grafica;

- se revine pe nivelul anterior al arborelui;

➢ **Pentru simplitate si eficienţă, testele de vizibilitate se pot face in spatiul coordonatelor de decupare, transformand volumele incadratoare prin matricea MVP si testand intersectia cu volumul vizual canonic (cub cu laturile paralele cu axele)**

❖ Structura BVH este frecvent utilizata pentru detectia coliziunilor dintre obiectele unei scene dinamice.

# *Ierarhie de volume incadratoare (5)*

❖ **Eficienta structurii BVH privind eliminarea obiectelor nevizibile din banda grafica**

**-** depinde mult de felul in care sunt grupate obiectele in volume incadratoare; poate fi eficienta pentru anumite pozitii ale observatorului si ineficienta pentru alte pozitii.

Alegerea tipului de volum incadrator pentru un nod:

- Sa aiba o forma simpla, a.i. testele de intersectie cu volumul vizual sa fie simple si rapide
- Sa incadreze minimal obiectul/obiectele

**Proprietati dorite ale ierarhiei de volume incadratoare:**

- Volumele nodurilor dintr-un subarbore trebuie sa fie apropriate spatial, cu atat mai apropiate cu cat sunt situate mai jos in arbore
- Volumul fiecarui nod sa incadreze minimal obiectul/grupul de obiecte corespunzator
- Volumul incadrator al intregii scene trebuie sa fie minimal
- Volumele nodurilor vecine de pe acelasi nivel sa fie disjuncte sau minimal suprapuse.