Eliminarea partilor nevizibile ale scenelor 3D din imagini -2

Prof. univ. dr. ing. Florica Moldoveanu

Algoritmul BSP pentru eliminarea partilor nevizibile ale primitivelor

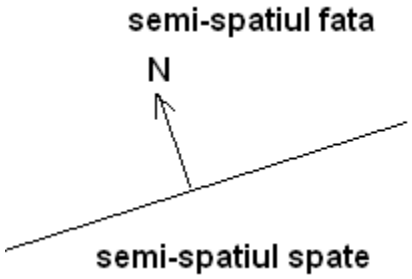
- Scena 3D este reprezentata printr-un arbore binar, denumit sugestiv arbore BSP (**Binary Space Partitioning)**
- Initial, arborele BSP a fost folosit pentru operatia de *frustum culling*.
- Ulterior a fost folosit şi în algoritmul cu acelaşi nume, pentru eliminarea părților nevizibile ale poligoanelor (hidden surface removal), algoritm utilizat în jocuri ca *Doom* şi *Quake* (aparute in anii '90), pentru a creste viteza de redare a scenelor 3D.
- Arborele BSP al scenei se foloseste şi în alti algoritmi grafici: algoritmul Ray-tracing, calcule de iluminare, generarea umbrelor, detectia coliziunilor, ş.a.
- **Arborele BSP al unei scene este independent de pozitia observatorului (camerei)- reprezinta scena în sistemul coordonatelor globale.**

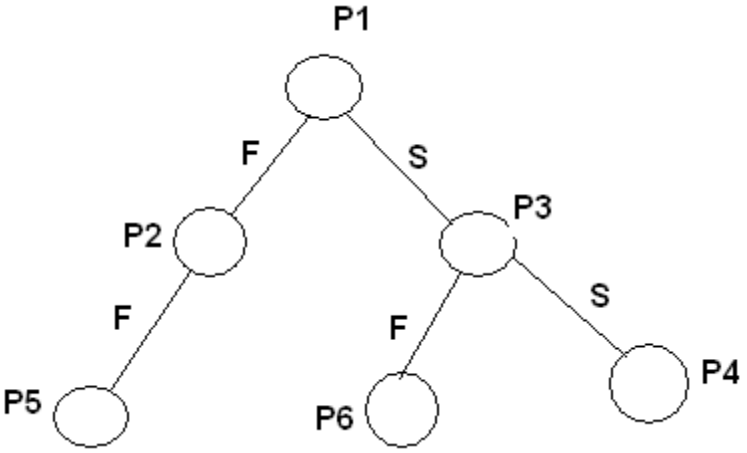
Intrarea algoritmului de construire a arborelui: lista poligoanelor care compun scena 3D, în orice ordine; nu are importanta din care obiect face parte fiecare poligon.

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: planele 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

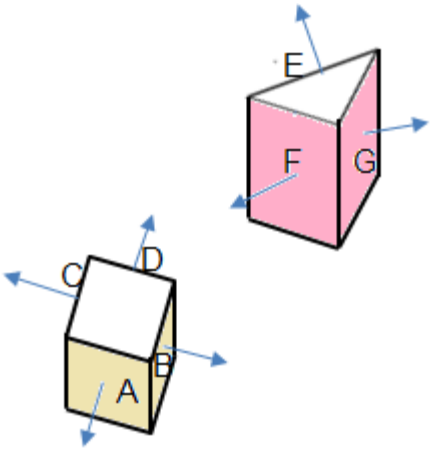




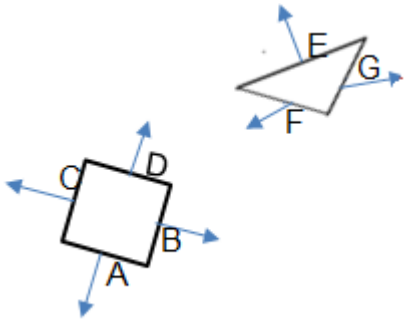
1. Se începe cu un poligon oarecare din lista (de regula prim pentru care se creaza nodul radacina al arborelui
   - Poligoanele aflate in semi-spatiul “faţă” formeaza “lista-faţă”, care va genera subarborele “faţă” al nodului
   - Primitivele din semi-spatiul “spate” formeaza “lista-spate”, care va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

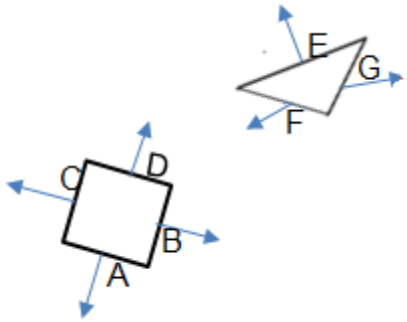


Vederea de sus a scenei

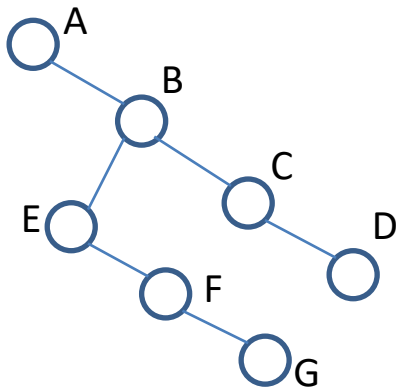
Consideram ca la construirea arborelui poligoanele sunt tratate în ordinea din lista:

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

Construirea arborelui BSP – exemplu (2)



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



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 (camerei)**

- **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_back_to_front(A->spate, Observator); *afisare poligoane din nodul radacina;
      afisareBSP_back_to_front(A->fata, Observator);}
  altfel {afisareBSP_back_to_front(A->fata, Observator); *afisare poligoane din nodul radacina;
          afisareBSP_back_to_front(A->spate, Observator);}
}
```

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).

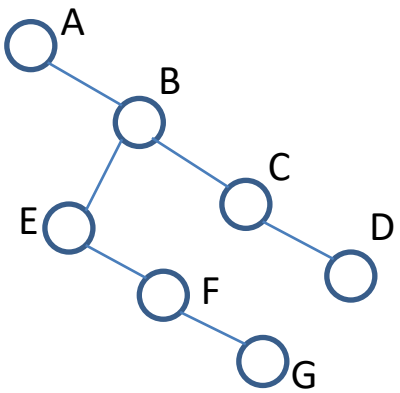
- Se porneste din radacina arborelui si se avanseaza in arbore pana la frunze, in fiecare nod coborandu-se in subarborele nodului care se afla de aceeasi parte cu observatorul fata de planul nodului.

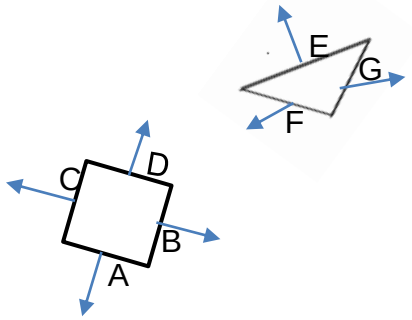
```
void afisareBSP_front_to_back(arbore * A, Pozitie Observator)
{ daca (! A) return;
  daca (Observator este in semispatiul fata al planului radacinii arborelui A) atunci
    { afisareBSP_front_to_back(A->fata, Observator); *afisare poligoane din nodul radacina;
     afisareBSP_front_to_back(A->spate,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

Afisarea arborelui BSP – exemplu (1)

Afişare “din față în spate”

Se tine cont de pozitia observatorului (camerei) în scena vizualizata.

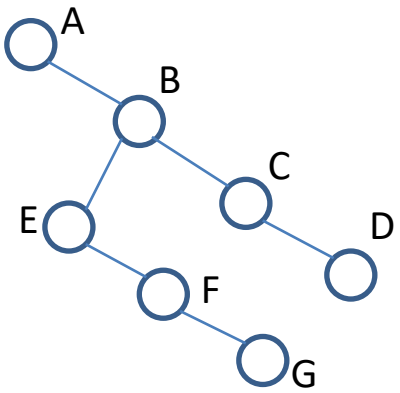


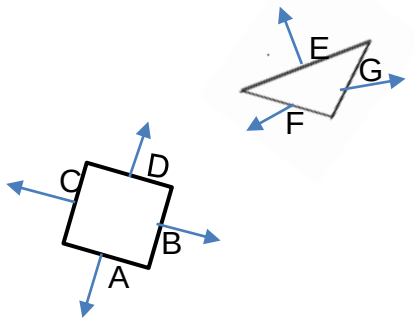


1. Fata de planul nodului A observatorul se afla in semispatiul spate.
   Se coboara in subarborele spate al nodului A.
2. Fata de planul nodului B observatorul se afla in semispatiul față.
   Se coboara in subarborele față al nodului B.
3. Fata de planul nodului E observatorul se afla in semispatiul 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 camerei → 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 ce corespund 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-o scena de exterior, 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)

- Forward rendering (implicit in banda grafica). La rasterizare, pentru fiecare fragment:
  - calcul culoare fragment
  - test de adancime
- Afisarea scenei in 2 pasi, prin tehnica “Deffered rendering (Deffered shading)”, reduce consumul de timp pentru 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 pentru calculul culorilor si in acest caz

  Prin Forward rendering 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.

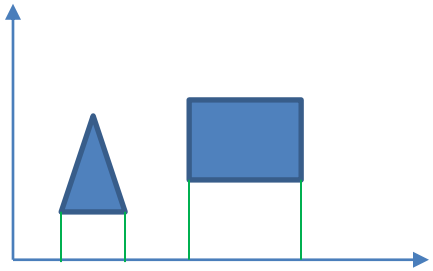
4. Se transmit in banda grafica poligoanele incepand cu ultimul din lista (cel mai indepartat de observator).

Algoritmul pictorului(2)

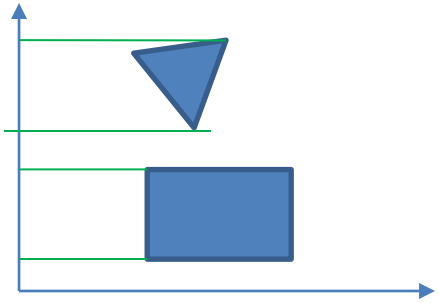
Pasul 3. Se descompun poligoanele ale caror extensii pe axa OZ se suprapun

- Sunt multe aplicatii in care acest pas nu este necesar, poligoanele fiind amplasate in plane de Z-constant: cartografie, generarea straturilor circuitelor imprimate, ş.a.
- Poate fi optimizat, stiind ca nu intotdeauna atunci cand extensiile pe axa OZ se suprapun este necesara descompunerea poligoanelor; descompunerea nu este necesara daca:

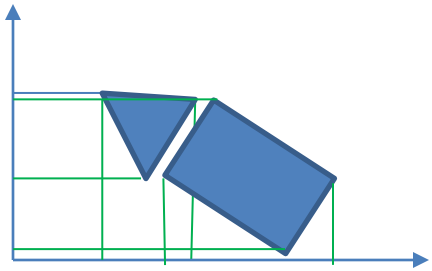
1) extensiile pe axa OX nu se suprapun



2) extensiile pe axa OY nu se suprapun



3) proiectiile poligoanelor nu se suprapun in imagine



Testele se efectueaza progresiv, in functie de complexitatea calculelor presupuse.

Algoritmul pictorului(2)

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

- sortarea este efectuata pentru fiecare cadru imagine

- multe dintre fragmentele poligoanelor sunt suprascrise in etapa z-buffer

- nu este eficient pentru hardware-ul grafic actual: a fost introdus ca algoritm de eliminare a suprafetelor ascuse in 1972

- exista variante ale algoritmului de baza:

Reverse painter's algorithm – trimite poligoanele la afisare in ordinea “din fata in spate”; mai eficient deoarece nu sunt suprascrisi pixelii in care deja s-au afisat fragmente ale poligoanelor