CURS 9 -- Algortim Z-Buffer. Bandă grafică programabilă.

Când vedem o imagine care conține obiecte și suprafețe netransparente, atunci nu putem vedea obiectele care sunt în spate. Trebuie să înlăturăm aceste suprafețe ascunse pentru a obține o imagine realistă pe ecran. Identificarea și îndepărtarea acestor suprafețe se numește problemă cu suprafața ascunsă.

Există două abordări pentru eliminarea problemelor de suprafață ascunsă - metoda Object-Space și metoda Image-space. Metoda obiect-spațiu este implementată în sistemul de coordonate fizic și metoda imagine-spațiu este implementată în sistemul de coordonate ecran.

Când dorim să afișăm un obiect 3D pe un ecran 2D, trebuie să identificăm acele părți ale unui ecran care sunt vizibile dintr-o poziție de vizualizare aleasă.

1 Algoritmul Z-Buffer

Este o abordare imagine-spațiu. Ideea de bază este de a testa adâncimea Z a fiecărei suprafețe pentru a determina cel mai apropiat vizibil de pe suprafaţă.

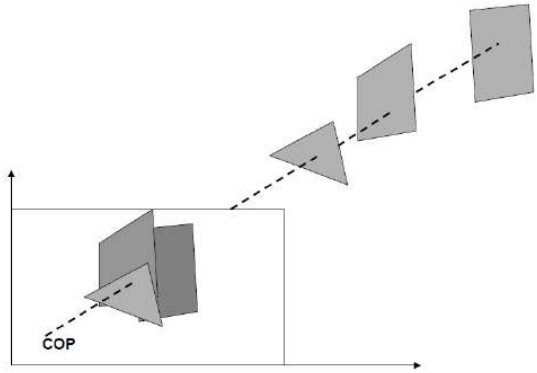
În această metodă, fiecare suprafață este procesată separat câte o poziție de pixel la un moment dat pe suprafață. Se compară valorile adâncimii pentru un pixel și se compară cea mai mica suprafață determină culoarea care urmează să fie afișată în tamponul de cadru.

Se aplica foarte eficient pe suprafetele poligonului. Suprafețele pot fi prelucrate în orice ordine. Pentru a suprascrie poligoane mai apropiate de cele îndepărtate, sunt utilizate două buffer-uri numite frame buffer și depth buffer.

Bufferul de adâncime este utilizat pentru a stoca valorile adâncimii pentru variabilele de poziție x, y, deoarece suprafețele sunt prelucrate astfel

$$0 \leq depth \leq 1$$

Buffer-ul de cadru este utilizat pentru a stoca valoarea intensității valorii culorii la fiecare poziție x,y.
Coordonatele z sunt de obicei normalizate la intervalul [0, 1]. Valoarea 0 pentru coordonatele z indică panoul de tăiere din spate, iar valoarea 1 pentru coordonatele z indică panoul de tăiere din față.



Algoritmul este urmatorul:

Algoritmul Z-Buffer

Algorithm
Pas 1 Set the buffer values
Depthbuffer $x, y = 0$

Framebuffer x,y= background color
Pas 2 Process each polygon One at a time
For each projected x,y pixel position of a polygon, calculate depth z.

If Z >depthbuffer x,y
Compute surface color,

set depthbuffer x,y= z,

framebuffer x,y= surface color

Avantaje

- Este ușor de implementat.
- Reduce problema vitezei dacă este implementată în hardware.
- Procesează câte un obiect.

Dezavantaje

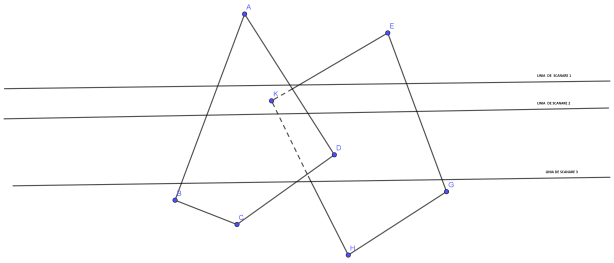
- Necesită memorie mare.
- Este un proces consumator de timp.

2 Metoda liniei de scanare

Este o metodă imagine-spațiu pentru a identifica suprafața vizibilă. Prin această metodă obținem doar o informație de profunzime doar pentru o singură linie de scanare. Pentru a solicita o linie de scanare a valorilor de adâncime, trebuie să grupăm și să procesăm toate poligoanele care intersectează o anumită linie de scanare în același timp înainte de a procesa următoarea linie de scanare. Două liste importante, lista de margini și lista de poligoane, sunt menținute pentru aceasta.

Lista de margini - Conține punctele finale de coordonate ale fiecărei linii, panta inversă a fiecărei linii și indicatori în tabelul poligoanelor pentru a conecta marginile la suprafețe.

Lista poligoanelor - Conține coeficienții plani, proprietățile materialului de suprafață, alte date despre suprafață și poate fi un indicator al listei de margini.



Pentru a facilita căutarea suprafețelor care traversează o anumită linie de scanare, se formează o listă activă de margini. Lista activă stochează numai acele margini care traversează linia de scanare în ordinea creșterii variabilei x. De asemenea, este setat un steag pentru fiecare suprafață pentru a indica dacă o poziție de-a lungul unei linii de scanare este fie în interiorul, fie în exteriorul suprafeței.

Pozițiile pixelilor pe fiecare linie de scanare sunt procesate de la stânga la dreapta. La intersecția din stânga cu o suprafață, steagul de suprafață este pornit, iar în dreapta, steagul este stins. Trebuie să efectuați calcule de adâncime doar atunci când mai multe suprafețe au steagurile activate la o anumită poziție a liniei de scanare.

3 Metoda zonei de subdiviziune

Metoda zonei de subdiviziune se referă la localizarea acelor zone de vedere care reprezintă o parte dintr-o singură suprafață. Împărțim suprafața totală de vizualizare în dreptunghiuri din ce în ce mai mici până când fiecare zonă mică este proiecția unei părți dintr-o singură suprafață vizibilă sau a nici unei suprafețe.

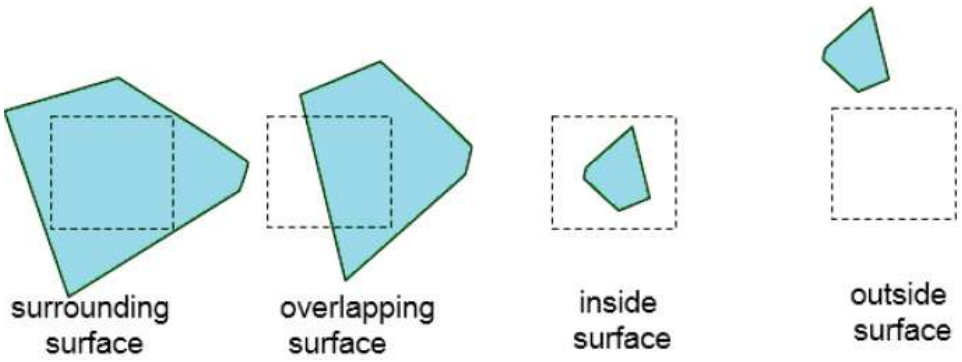
Continuăm acest proces până când subdiviziunile sunt ușor de analizat ca aparținând unei singure suprafețe sau până când sunt reduse la dimensiunea unui singur pixel. O modalitate ușoară de a face acest lucru este să împărțim succesiv zona în patru părți egale la fiecare pas. Există patru relații posibile pe care o suprafață le poate avea cu o limită de zonă specificată.

Suprafața înconjurătoare - Una care înconjoară complet zona.

Suprafață suprapusă - Una care este parțial în interiorul și parțial în afara zonei.

Suprafața interioară - Una care se află complet în interiorul zonei.

Suprafața exterioară - Una care se află complet în afara zonei.



Testele pentru determinarea vizibilității la suprafață într-o zonă pot fi enunțate în termenii acestor patru clasificări. Nu sunt necesare alte subdiviziuni ale unei zone specificate dacă una dintre următoarele condiții este adevărată:

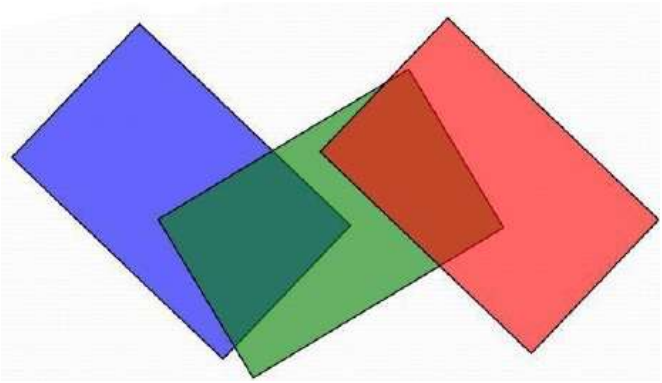
- Toate suprafețele sunt suprafețe exterioare în raport cu zona.

- Doar o suprafață interioară, suprapusă sau înconjurătoare se află în zonă.
- O suprafață înconjurătoare ascunde toate celelalte suprafețe din limitele zonei.

4 A-buffer

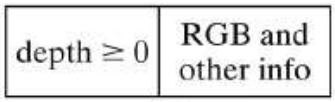
Metoda A-buffer este o extensie a metodei depth-buffer. Metoda A-buffer este o metodă de detectare a vizibilității dezvoltată de Lucas film Studios pentru sistemul de randare Renders Everything You Ever Saw REYES.

Tamponul (bufferul) A extinde metoda bufferului de adâncime pentru a permite transparențe. Structura cheie a datelor din bufferul A este bufferul de acumulare.

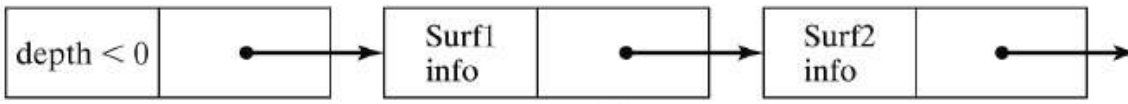


Fiecare poziție din A-buffer are două câmpuri:

- Câmp de adâncime - Stochează un număr real pozitiv sau negativ
- Câmp de intensitate - Stochează informații despre intensitatea suprafeței sau o valoare de indicator



(a)



(b)

Dacă adâncimea ≥ 0, numărul stocat în acea poziție este adâncimea unei singure suprafețe care se suprapune pe zona pixelilor corespunzătoare. Câmpul de intensitate stochează apoi componentele RGB ale culorii suprafeței în acel punct și procentul de acoperire a pixelilor.

Dacă adâncimea < 0, aceasta indică contribuții de mai multe suprafețe la intensitatea pixelului. Câmpul de intensitate stochează apoi un indicator către o listă legată de date de suprafață. Tamponul de suprafață din tamponul A include:

- Componente de intensitate RGB
- Parametru de opacitate
- Adâncime
- Procentul de acoperire a zonei
- Identificator de suprafață.

Algoritmul procedează la fel ca algoritmul tamponului de adâncime. Valorile de adâncime și opacitate sunt utilizate pentru a determina culoarea finală a unui pixel.

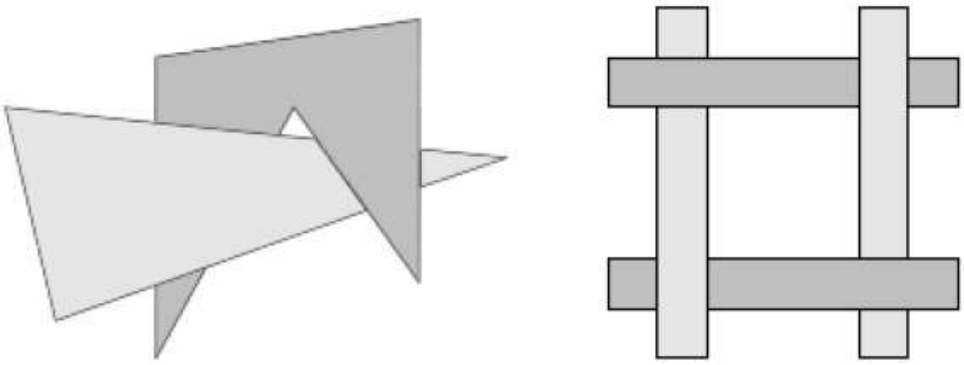
5 Metoda de sortare în adâncime

Metoda de sortare în profunzime folosește atât operații spațiu imagine, cât și spațiu obiect. Metoda de sortare în adâncime îndeplinește două funcții de bază.

În primul rând, suprafețele sunt sortate în ordinea descrescătoare a adâncimii.

În al doilea rând, suprafețele sunt scanate în ordine, începând cu suprafața de cea mai mare adâncime.

Conversia de scanare a suprafețelor poligonului se realizează în spațiul imaginii. Această metodă de rezolvare a problemei suprafețelor ascunse este adesea denumită algoritmul pictorului. Figura următoare arată efectul sortării în adâncime.

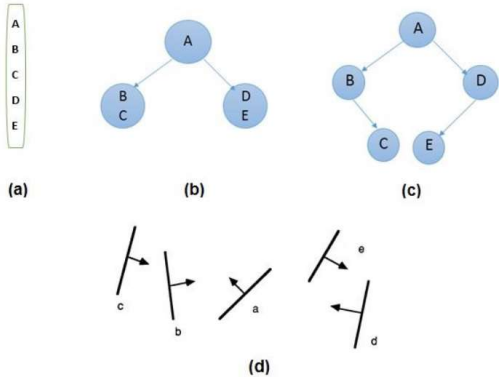


Algoritmul începe prin sortarea după adâncime. De exemplu, estimarea inițială de adâncime a unui poligon poate fi considerată cea mai apropiată valoare z a oricărui vârf al poligonului.

Să luăm poligonul P de la sfârșitul listei. Luați în considerare toate poligoanele Q ale căror extinderi z se suprapun pe P. Înainte de a trasa P, facem următoarele teste. Dacă oricare dintre următoarele teste este pozitiv, atunci putem presupune că P poate fi considerat înainte de Q.
P este în întregime pe partea opusă a planului lui Q din punct de vedere?
Este Q complet pe aceeași parte a planului lui P cu punctul de vedere?
Nu se suprapun proiecțiile poligoanelor? Dacă toate testele eșuează, atunci împărțim fie P, fie Q folosind planul celuilalt. Noile poligoane tăiate sunt introduse în ordinea adâncimii și procesul continuă. Teoretic, această partiționare ar putea genera $O(n^2)$ poligoane individuale, dar în practică, numărul de poligoane este mult mai mic.

6 Partiția spațiului binar BSP

Arbori
Partiționarea spațiului binar este utilizată pentru a calcula vizibilitatea. Pentru a construi arborii BSP, ar trebui să începeți cu poligoane și să etichetați toate marginile. Tratând cu o singură muchie la un moment dat, extindeți fiecare muchie astfel încât să împartă planul în două. Așezați prima margine în arbore ca rădăcină. Adăugați marginile ulterioare în funcție de faptul că sunt în interior sau în exterior. Muchiile care se întind pe extensia unei margini care se află deja în arbore sunt împărțite în două și ambele sunt adăugate în arbore.



- Din figura de mai sus, luați mai întâi A ca rădăcină.
- Faceți o listă cu toate nodurile din figura a.
- Puneți toate nodurile care sunt în fața rădăcinii A în partea stângă a nodului A și puneți toate acele noduri care sunt în spatele rădăcinii A în partea dreaptă, așa cum se arată în figura b.
- Procesați mai întâi toate nodurile din față și apoi nodurile din spate.

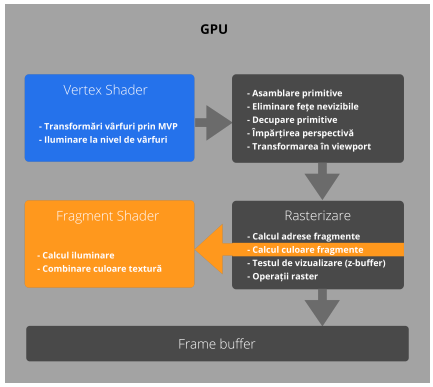
După cum se arată în figura c, vom procesa mai întâi nodul B. Deoarece nu există nimic în fața nodului B, am pus NIL. Cu toate acestea, avem nodul C în spatele nodului B, așa că nodul C va merge în partea dreaptă a nodului B. Repetați același proces pentru nodul D.

7 Banda grafica programabila

Banda Grafica este un lant de operatii executate de procesoarele GPU. Unele dintre aceste operatii sunt descrise in programe numite shadere (eng. shaders),

care sunt scrise de programator si transmise la GPU pentru a fi executate de procesoarele acestuia. Pentru a le deosebi de alte operatii executate in banda grafica, pe care programatorul nu le poate modifica, shaderele sunt numite „etape programabile”. Ele dau o mare flexibilitate in crearea de imagini statice sau dinamice cu efecte complexe redate in timp real (de ex. generarea de apa, nori, foc etc prin functii matematice).

Folosind OpenGL sunt transmise la GPU: coordonatele varfurilor, matricile de transformare a varfurilor (M: modelare, V: vizualizare, P: proiectie, MV: modelare-vizualizare, MVP: modelare-vizualizare-proiectie), topologia primitivelor, texturi si ale date.



1. In etapa programabila VERTEX SHADER se transforma coordonatele unui varf, folosind matricea MVP, din coordonate obiect in coordonate de decupare (eng. clip coordinates). De asemenea, pot fi efectuate si calcule de iluminare la nivel de varf. Programul VERTEX SHADER este executat in paralel pentru un numar foarte mare de varfuri.

2. Urmeaza o etapa fixa, in care sunt efectuate urmatoarele operatii:

asamblarea primitivelor folosind varfurile transformate in vertex shader si topologia primitivelor;
eliminarea fetelor nevizibile; decuparea primitivelor la frontiera volumului canonic de vizualizare (ce inseamna?);
impartirea perspectiva, prin care se calculeaza coordonatele dispozitiv normalizate ale varfurilor: xd = xc/w; yd = yc/w;zd = zc/w, unde [xc,yc,zc,w] reprezinta coordonatele unui varf in sistemul coordonatelor de decupare;
transformarea fereastra–poarta: din fereastra (-1, -1) – (1, 1) in viewport-ul

definit de programator.
3. Urmatoarea etapa este Rasterizarea. Aceasta include:

calculul adreselor pixelilor in care se afiseaza fragmentele primitivelor (bucatele de primitive de dimensiune egala cu a unui pixel); calculul culorii fiecarui fragment, pentru care este apelat programul FRAGMENT SHADER in etapa programabila FRAGMENT SHADER se calculeaza culoarea unui fragment pe baza geometriei si a texturilor; programul FRAGMENT SHADER este executat in paralel pentru un numar mare de fragmente. testul de vizibilitate la nivel de fragment (algoritmul z-buffer); operatii raster, de exemplu pentru combinarea culorii fragmentului cu aceea existenta pentru pixelul in care se afiseaza fragmentul. Rezultatul etapei de rasterizare este o imagine memorata intr-un tablou de pixeli ce va fi afisat pe ecran, numit frame buffer.

8 BIBLIOGRAFIE

1. https://www.tutorialspoint.com/computer graphics/visible surface detection.html.