TRANSPARENTA, CEATA SI UMBRE IN MODELELE DE ILUMINARE LOCALA

Prof. univ. dr. ing. Florica Moldoveanu

MODELAREA TRANSPARENTEI (1)

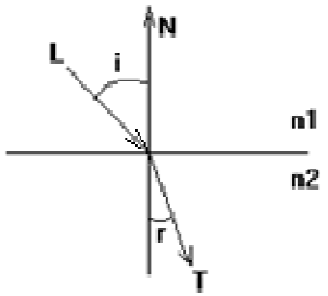
Unele obiecte ale scenei sintetizate pot fi construite din materiale transparente sau translucide.

Transmisia luminii prin obiectele transparente este specular, în timp ce prin cele translucide este difuză.

- Atunci când lumina trece dintr-un mediu într-altul (de exemplu, din aer în apă), direcția sa se modifică datorită refracției.
- Relația dintre unghiul razei incidente, i, şi cel al razei refractate, r, este dată de legea lui Snell:

sin(i)/sin(r) = n1/n2

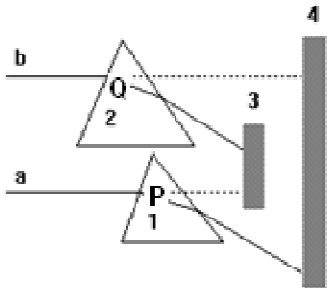
n1 şi n2 sunt indicii de refracție ai celor două medii (materiale) traversate de lumină.



MODELAREA TRANSPARENTEI (2)

Indicele de refracție al unui material este dependent de lungimea de undă a luminii incidente şi chiar de temperatură, dar în modelele de iluminare el este considerat constant.

Culoarea vizibila intr-un punct al unei suprafete transparente provine de la obiectul aflat pe directia razei transmise.



3 si 4 sunt poligoane opace

1 si 2 sunt poligoane transparente, aflate in fata poligoanelor 3 si 4

a si b sunt raze de lumina provenind de la surse, incidente in P si Q

Tinand cont de refractie:

- Culoarea in P este data de poligonul 4
- Culoarea in Q este data de poligonul 3

Neglijand refractia

- Culoarea in P este data de poligonul 3
- Culoarea in Q este data de poligonul 4

MODELAREA TRANSPARENTEI (3)

- Refracția produce, de asemenea, o distorsionare a obiectelor, asemănătoare cu aceea produsă de o proiecție perspectivă: → dacă se doreşte obținerea de imagini realiste, trebuie să se țină cont de refracție.

- Multe metode practice de modelare a transparenței ignoră refracția, astfel încât obiectele vizibile printr-o suprafață transparentă sunt cele aflate pe direcția razei incidente.

Motivul ignorării:

- reducerea volumului de calcule;
- obținerea realismului fotografic în totalitate (fara deformare)

Atunci când suprafața vizibilă într-un pixel este transparentă, culoarea în care va afişat pixelul se poate obține combinând culoarea suprafeței vizibile cu aceea a suprafeței aflată imediat în spatele său, folosind următoarea formulă de interpolare:

MODELAREA TRANSPARENTEI (4)

$I_\lambda = (1-kt_1)*I_{\lambda 1} + kt_1*I_{\lambda 2}$

Coeficientul de transmisie, kt_1 măsoară transparența suprafeței vizibile în pixel,

$0 <= kt_1 <= 1$.

$kt_1 = 0$ → suprafața vizibilă este opacă şi deci pixelul va fi afişat în culoarea sa, $I_{\lambda 1}$;

$k_{t1} = 1$ → suprafața vizibilă este perfect transparentă şi nu contribuie la culoarea pixelului. Pixelul va fi afişat în culoarea suprafeței din spate, $I_{\lambda 2}$

Dacă $k_{t1}=1$ şi suprafața din spatele celei vizibile este la rândul său transparentă, metoda de calcul se aplică recursiv, până când se întâlneşte o suprafață opacă sau fondul.

Aproximarea liniară din model nu dă rezultate bune pentru suprafețele curbe: în apropierea laturilor siluetei unei suprafețe curbe (de exemplu, o vază sau o sticlă) grosimea materialului reduce transparența.

Solutia propusa de Kay: kt se calculeaza in functie de normala la suprafata in punctul considerat.

MODELAREA TRANSPARENTEI (5)

$k_t = kt_{min} + (kt_{max} - kt_{min})(1-(1-N_z)^m)$

unde

kt_{min} şi kt_{max} reprezintă transparența minimă şi cea maximă a suprafeței,

N_z este componenta z a normalei normalizate la suprafață în punctul pentru care se calculează kt, iar

m este un exponent ce caracterizează transparența. Valorile uzuale pentru m sunt 2 şi 3.

- **Majoritatea algoritmilor de determinare a vizibilității suprafetelor la afişarea scenelor 3D pot fi adaptați pentru a îngloba transparența.**
  - In algoritmii care afişează poligoanele scenei 3D în ordinea "din spate în față" (back to front), de ex. algoritmul Pictorului şi BSP, $\mathbf{I_{\lambda 1}}$ corespunde poligonului care se rasterizeaza la un moment dat iar $\mathbf{I_{\lambda 2}}$ este valoarea existentă în memoria imagine pentru pixelul considerat.

MODELAREA TRANSPARENTEI (6)

- Adăugarea efectului de transparență în algoritmul Z-Buffer este mai dificilă, deoarece poligoanele sunt rasterizate în ordinea în care sunt transmise în banda grafică, neținându-se cont de distanța lor față de observator.

-Se poate realiza folosind mai multe memorii buffer-imagine si generand imaginea in mai multe etape, imaginea finala obtinandu-se prin combinarea imaginilor poligoanelor opace cu cele ale poligoanelor transparente

In OpenGL, transparenta poate fi simulata prin amestecul dintre culoarea fragmentului curent si cea a pixelului in care se afiseaza, folosind modelul (R, G, B, A);

A – opacitatea: 0 – transparent; 1 – opac

Se folosesc functiile:

glEnable(GL_ALPHA_TEST) – activeaza testul alfa

MODELAREA TRANSPARENTEI (7)

glAlphaFunc(func, ref) – specifica functia de comparare a valorii A a fragmentului cu valoarea de referinta; exemplu:

func = GL_GREATER – fragmentul este afisat daca opacitatea sa este > decat valoarea ref

glEnable(GL_BLEND) - activeaza amestecul culorilor

glBlendFunc(..) – specifica modul de amestec al culorilor. Exemplu:

glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA):

noua_culoare_pixel = Culoare_fragment * A_fragment + Culoare_pixel * (1 – A_fragment)

Pentru obtinerea unui efect corect ar trebui ca poligoanele sa fie transmise in banda grafica in ordinea descrescatoare a opacitatii lor.

MODELAREA CEȚII (1)

Ceața adaugă realism imaginii şi ajută în eliminarea defectelor produse de decuparea la nivelul planului din spate al volumului vizual:

- Fără ceață, pe măsură ce observatorul se îndepărtează de un obiect, obiectul se aproprie de planul din spate şi deci este decupat atunci când planul din spate îl intersectează.
- Cu ceață, dacă intensitatea ceții creşte proporțional cu distanța de la poziția observatorului, se crează efectul de distanță pentru obiectele îndepărtate.
- Dacă densitatea ceții creşte până la opacitate la nivelul planului din spate, decuparea față de planul din spate nu mai este necesară căci obiectele dispar datorită opacității.

Cfog – culoarea de ceață

Cfragment – culoarea calculată pentru un fragment

$0 <= \Phi <= 1$ – factorul de ceață, proporțional cu distanța de la observator la punctul din spațiul 3D vizibil in pixel (Φ=1 : opacitate)

Atunci, Cpixel, culoarea de afişare a pixelului corespunzator fragmentului este:

Cpixel = (1-Φ)*Cfragment + Φ*Cfog

MODELAREA CEȚII (2)

- Factorul de ceata, Φ, se calculeaza, in mod uzual, folosind coordonata z a fragmentului, necesara si in algoritmul Z-buffer (interpolata de GPU).
- Factorul de ceață liniar (obtinut prin interpolare liniara) actioneaza intre 2 distante definite in sistemul coordonatelor observator, zo si z1, masurate pe directia de observare.
- Factorul de ceata este calculat in timpul rasterizarii, pentru fiecare fragment, folosind coordonata z a fragmentului si distantele z0 si z1 transformate in coordonate ale spatiului de afisare:

  z: coordonata z a fragmentului

  $\Phi = 0$ pentru $z < z0$

  $\Phi = (z-z0) / (z1- z0)$ pentru $z0 <= z <= z1$

  $\Phi = 1$ pentru $z > z1$
- Valorile implicite pentru z0 si z1 sunt 0 si 1, in spatiul de afisare: coordonatele z ale planului din fata si planului din spate al vederii, dupa transformarea in poarta de afisare.
- Factorul de ceata moduleaza culoarea fragmentului dupa texturare.

MODELAREA CEȚII (3)

In OpenGL, factorul de ceata poate avea o crestere liniara sau exponentiala.

- Factorul de ceata exponential este calculat in functie de coordonata z a fragmentului si de densitatea cetii astfel:

  $\Phi = e^f$

  f = (- densitate*z) - in modul GL_EXP

  $f = (- densitate*z)^2$ - in modul GL_EXP2

```
glEnable(GL_FOG); // activeaza modul in care se calculeaza ceata
float FogCol[3]={0.5f,0.5f,0.5f, 1}; // Se defineste culoarea de ceata
glFogfv(GL_FOG_COLOR,FogCol); // Se seteaza culoarea de ceata
glFogi(GL_FOG_MODE, GL_LINEAR); // Seteaza factorul de ceata liniar
glFogf(GL_FOG_START, 10.f); //seteaza distanta  z0
glFogf(GL_FOG_END, 20.f); // seteaza distanta z1
```

MODELAREA CEȚII (4)

Pentru modurile GL_EXP si GL_EXP2:

GLfloat density = 0.3;

glFogf (GL_FOG_DENSITY, density);

Se poate selecta precizia calculelor de ceata:

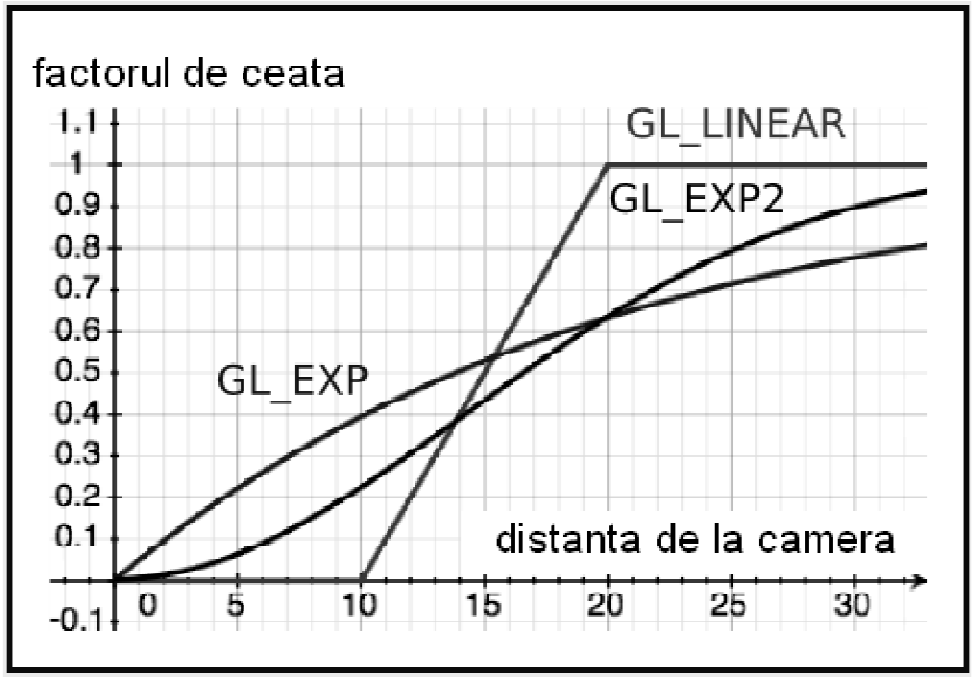
la nivel de varf sau la nivel de fragment.

gl.glHint(GL_FOG_HINT, GL_DONT_CARE);

GL_DONT_CARE,

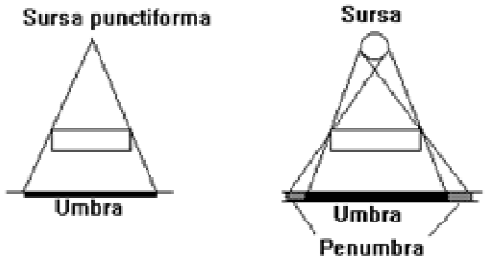
GL_NICEST (la nivel de fragment) sau

GL_FASTEST (la nivel de varf)



INTRODUCEREA UMBRELOR IN IMAGINI(1)

- Atunci când un observator priveşte o scenă 3D luminată de o sursă de lumină dintr-o poziție diferită de aceea a sursei de lumină, va vedea umbrele produse de obiectele scenei.
- Umbrele au o contribuție însemnată la realismul imaginii, îmbunătățind percepția profunzimii.



In general, se considera numai umbra pura: calcule mai simple.

Volumul de calcule depinde şi de poziția sursei de lumină:

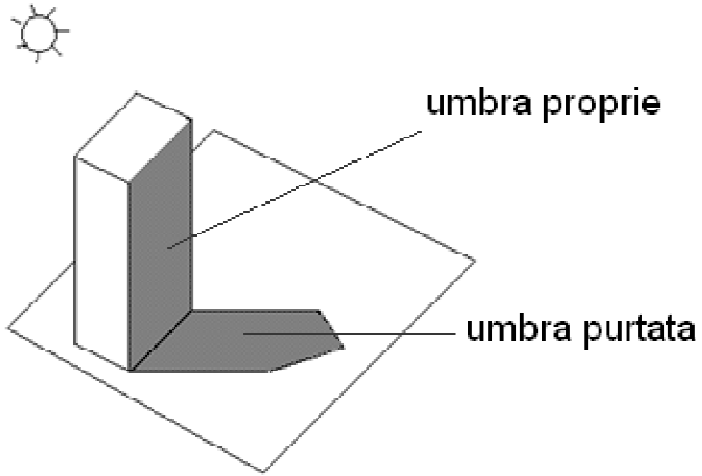
- Dacă sursa este la infinit calculele sunt mai simple.
- In cazul unei surse situate la distanță finită dar în afara câmpului vizual, este necesară o proiecție perspectivă din poziția sursei.
- Cazul cel mai dificil este acela în care sursa este situată în câmpul vizual.

INTRODUCEREA UMBRELOR IN IMAGINI(2)

- Problema determinării umbrelor este similară aceleia de determinare a vizibilității obiectelor: suprafețele care nu sunt vizibile din poziția sursei de lumină sunt în umbră.

  → Crearea unei imagini cu umbre presupune rezolvarea de două ori a problemei suprafețelor nevizibile: o dată privind scena din poziția fiecărei surse de lumină, a doua oară privind-o din poziția observatorului.

- Sunt două tipuri de umbre: **umbre proprii** şi **umbre purtate**.



INTRODUCEREA UMBRELOR IN IMAGINI(3)- POLIGOANE DETALIU

–Umbrele proprii sunt generate de obiectul însuşi, care împiedică lumina să ajungă la unele dintre fețele sale: fețele umbrite de umbra proprie sunt fețele auto-obturate atunci când scena este văzută din poziția sursei de lumină.

–Umbra purtată este umbra pe care un obiect o produce pe alte părți ale scenei, la care lumina nu ajunge din cauza obiectului.

Poligoane detaliu

❑ Umbrele purtate se pot determina proiectând în scenă, din poziția sursei, toate poligoanele (fețele) neumbrite de umbra proprie.

- Rezultă un set de **poligoane de umbră**, care se adaugă la reprezentarea scenei.

❑ Atat fetele umbrite de umbra proprie cat si poligoanele de umbra se folosesc ca poligoane-detaliu la redarea scenei 3D.

❑ Poligoanele detaliu sunt poligoane coplanare cu poligoanele scenei, care se folosesc in calculul culorii fragmentelor.

INTRODUCEREA UMBRELOR IN IMAGINI(4)- POLIGOANE DETALIU

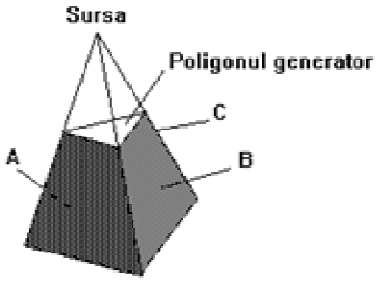
- Numărul de poligoane de umbră este mai mic, dacă în loc să se proiecteze fiecare față luminată de sursă, se proiectează silueta fiecărui obiect (văzută din poziția sursei de lumină).
- După adăugarea poligoanelor de umbră la reprezentarea scenei, se generează imaginea văzută din poziția observatorului.
- Pot fi generate mai multe vederi fără a recalcula umbrele, deoarece umbrele depind numai de poziția sursei (surselor) de lumină.
- Netinand cont de umbre, un punct P al unei suprafețe care este vizibil din poziția observatorului dar nu şi din poziția unei surse se afişează cu intensitatea rezultata din reflexia luminii ambiante sau cu o intensitate care rezultă din iluminarea sa de către alte surse existente în scenă:

$$I_\lambda = Ia_\lambda * ka_\lambda + \sum_{1<=i<=m} S_i * fat_i * Il_{\lambda i} * [kd_\lambda * (L_{ui} . Nu) + ks_\lambda * (R_{ui} . V_u)^n]$$

S_i= 0 dacă lumina de la sursa i nu ajunge în punctul P;
= 1 dacă lumina de la sursa i ajunge în punctul P: $(L_{ui} . Nu) > 0$

INTRODUCEREA UMBRELOR IN IMAGINI(5) – VOLUME DE UMBRE

- Sursa de lumină este considerată punctiformă iar obiectele ca având fațete poligonale.
- Un **volum de umbră** este definit de o sursă de lumină şi un poligon luminat (vizibil din poziția sursei de lumină), pe care-l vom numi **poligonul generator**.



- Fiecare față laterală a volumului este numită **poligon de umbră**. Ea este determinată de o latură a poligonului generator şi de cele două drepte care pleacă din sursa de lumină, fiecare trecând printr-un vârf al laturii.
- Normalele la fețele laterale punctează înspre exteriorul volumului.

INTRODUCEREA UMBRELOR IN IMAGINI(6) – VOLUME DE UMBRE

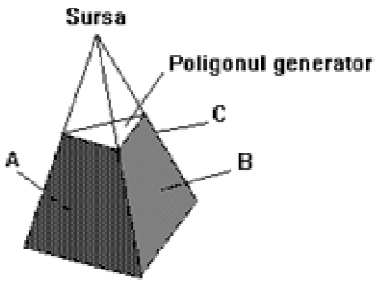
- Volumul infinit determinat de o sursă şi un poligon generator este delimitat de o față care reprezintă poligonul generator scalat. Această față este situată la o distanță față de sursă dincolo de care intensitatea luminii sursei este neglijabilă, deci orice punct aflat dincolo de această limită este umbrit.
- Volumul de umbră poate fi decupat la marginile volumului vizual.
- Poligoanele de umbră se folosesc pentru determinarea umbririi produse de poligonul generator în scenă.

Notăm cu :

– PUV poligoanele de umbră care sunt vizibile din poziția observatorului (A şi B în figura) şi cu

– PUN poligoanele de umbră care nu sunt vizibile din poziția observatorului (de exemplu, poligonul C).

Clasificarea poligoanelor de umbra in PUV si PUN se poate face pe baza normalelor la poligoane (back face culling).



INTRODUCEREA UMBRELOR IN IMAGINI(7) – VOLUME DE UMBRE

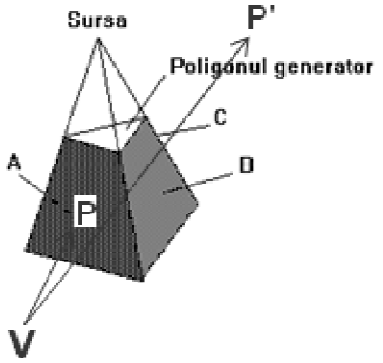
Fie

– un punct P al unei suprafețe şi

– VP vectorul din poziția observatorului (V) în punctul P.

Punctul P este umbrit dacă numărul de poligoane de tip PUV intersectate de vectorul VP este mai mare decât numărul de poligoane de tip PUN intersectate de vector.

Acesta este singurul caz, atunci când punctul V nu este în umbră.



❑ In general, pentru a determina dacă un punct este în umbră, se poate folosi un contor în care inițial se memorează numărul de volume de umbră care conțin poziția observatorului.

– Se asociază poligoanelor de tip PUV valoarea +1 iar celor de tip PUN valoarea -1.

– Atunci când vectorul VP traversează un poligon de umbră, se adună la contor valoarea asociată poligonului.

– Punctul P este umbrit dacă valoarea contorului este pozitivă în P.

INTRODUCEREA UMBRELOR IN IMAGINI(8) – VOLUME DE UMBRE

➢ Volumul de calcul presupus de acest algoritm poate fi redus dacă în loc să se calculeze volumul de umbră pentru fiecare poligon vizibil din poziția sursei, se calculează un singur volum de umbră pentru o suprafață poliedrală. In acest scop, se determină poligoanele de umbră numai pentru laturile care fac parte din silueta suprafeței, văzută din poziția sursei.

❑ Silueta unei suprafețe, corespunzătoare unui punct de observare, este un set conectat de laturi care aparțin poligoanelor vizibile din punctul de observare.

❑ O latură de siluetă este fie o latură de margine a unei suprafețe deschise, fie o latură care separă un poligon vizibil de unul nevizibil.

➢Pentru determinarea laturilor de siluetă, este necesar să se folosească o structură de date care reflectă adiacența poligoanelor. Cunoscându-se poligonul adiacent pe fiecare latură a fiecărui poligon vizibil din poziția observatorului, se pot determina rapid laturile de siluetă.

INTRODUCEREA UMBRELOR IN IMAGINI(9) – VOLUME DE UMBRE

Implementarea metodei folosind stencil-buffer(1)

- Buffer-ul de marcaje (stencil-buffer) este prezent pe majoritatea placilor grafice actuale, in memoria placii, alaturi de buffer-ul imagine (frame-buffer) si buffer-ul de adancime (z-buffer).
  – este o memorie de numere intregi asociata pixelilor imaginii, de regula un byte/pixel;
  – poate fi utilizat pentru a implementa diferite operatii grafice, de regula in combinatie cu z-buffer, cum este si cazul implementarii volumelor de umbra:
  – valorile din stencil-buffer pot fi incrementate/ decrementate automat pentru fiecare fragment care trece/ nu trece testul de adancime.

- Exista mai multe metode de a utiliza stencil-buffer pentru implementarea metodei volumelor de umbre.

INTRODUCEREA UMBRELOR IN IMAGINI(10) – VOLUME DE UMBRE

Implementarea metodei folosind stencil-buffer(2)

- In metoda prezentata in continuare, se creaza in stencil-buffer o masca a umbrelor din imagine: orice pixel de umbra, (x,y), va fi reprezentat in stencil-buffer printr-o valoare #0.

- **Pixelii de umbra sunt afisati in culorile fragmentelor vizibile, calculate cu formula.**

Cfragment = Cemisiva + Cambianta = ke + ka*globalAmbient

ke, ka – proprietati de material

– Se presupune ca s-au calculat poligoanele de umbra.

– Conturul fiecarui poligon de umbra trebuie sa fie orientat in sens trigonometric atunci cand este privit din exteriorul volumului de umbra. In acest fel, poligoanele PUV sunt cele “din fața” observatorului, iar poligoanele PUN sunt cele “din spate”.

INTRODUCEREA UMBRELOR IN IMAGINI(11) – VOLUME DE UMBRE

Implementarea metodei folosind stencil-buffer(3)

Se considera cazul in care observatorul nu este in umbra.

1.

Initializeaza z-buffer si stencil-buffer; // nu trebuie programate, acestea sunt

// initializate in mod automat pentru fiecare cadru imagine

Activeaza scrierea in z-buffer si dezactiveaza scrierea in stencil-buffer ; //implicite

glDisable(GL_LIGHTING); // Dezactiveaza luminile din scena 3D

*** Rasterizeaza scena (folosindu-se algoritmul z-buffer)**

Efectul:

— in frame-buffer se obtine imaginea scenei in umbra (fara lumini): pentru fiecare fragment vizibil intr-un pixel se memoreaza o culoare:

Cfragm = culoarea emisiva + culoarea ambientala;

— la sfarsitul acestui pas, z-buffer contine coordonatele z ale fragmentelor vizibile in imagine.

INTRODUCEREA UMBRELOR IN IMAGINI(12) – VOLUME DE UMBRE

Implementarea metodei folosind stencil-buffer(4)

2.

```
glDepthMask(GL_FALSE); // Dezactiveaza scrierea in z-buffer
glColorMask(GL_FALSE, GL_FALSE, GL_FALSE, GL_FALSE); // Dezactiveaza scrierea in frame-buffer
glFrontFace(GL_CCW);//fetele "din fata" sunt orientate trigonometric
glCullFace(GL_BACK);// nu sunt rasterizate poligoanele "din spate"
glStencilFunc(GL_ALWAYS, 0x0, 0xff);
glStencilOp(GL_KEEP, GL_INCR, GL_KEEP);
```

*rasterizeaza poligoanele de umbra

Efectul:

Sunt rasterizate poligoanele de umbra de tip PUV. Pentru fiecare fragment rezultat, f, se face testul:

daca f.z < z-buffer[f.y, f.x] // fragmentul f in fata fragmentului din scena 3D vizibil in pixelul (x,y)

atunci stencil-buffer[f.y, f.x]++; // se numara poligoanele PUV aflate intre observator si //obiectele scenei

INTRODUCEREA UMBRELOR IN IMAGINI(13) – VOLUME DE UMBRE

Implementarea metodei folosind stencil-buffer(5)

3.

glCullFace(GL_FRONT);// nu sunt rasterizate poligoanele “din fata”

glStencilFunc(GL_ALWAYS, 0x0, 0xff);

glStencilOp(GL_KEEP, GL_DECR, GL_KEEP);

*rasterizeaza poligoanele de umbra

Efectul:

Sunt rasterizate poligoanele de umbra de tip PUN. Pentru fiecare fragment rezultat, f, se face testul:

daca f.z < z-buffer[f.y, f.x] // f este in fata fragmentului din scena 3D vizibil in pixelul (x,y)

atunci stencil-buffer[f.y, f.x]--;

In stencil-buffer s-a obtinut o masca cu “gauri” in punctele care primesc lumina: valorile din stencil-buffer sunt diferite de zero pentru pixelii care trebuie afisati in umbra.

INTRODUCEREA UMBRELOR IN IMAGINI(14) – VOLUME DE UMBRE

Implementarea metodei folosind stencil-buffer(6)

4.

```
glEnable(GL_STENCIL_TEST);// Activeaza testul stencil
glStencilOp(GL_KEEP, GL_KEEP, KEEP); // Dezactiveaza scrierea in stencil-buffer
glStencilFunc(GL_EQUAL, 0x0, 0xff); // Specifica testul stencil
glDepthMask(GL_TRUE); // Activeaza scrierea in z-buffer
glColorMask(GL_TRUE, GL_TRUE, GL_TRUE, GL_TRUE); // Activeaza scrierea in frame buffer
glEnable(GL_LIGHTING); // Activeaza toate luminile:
```

*rasterizeaza scena (folosindu-se algoritmul zbuffer)

Efectul:

Se rasterizeaza scena modificandu-se in frame-buffer numai culorile pixelilor (x,y) pentru care stencil-buffer[y, x] =0 (fragmentul nu este in umbra)

INTRODUCEREA UMBRELOR IN IMAGINI(15) – VOLUME DE UMBRE

Implementarea metodei folosind stencil-buffer(7)

Cazul general:

Se initializeaza stencil-buffer cu o valoare reprezentand numarul de volume de umbra care contin pozitia observatorului.

Aprecieri asupra metodei volumelor de umbra

Pozitive:

- Este generala si poate fi aplicata pentru orice scena, pentru oricate surse de lumina.
- Produce forme de umbre exacte.

Negative:

- Complexa din punct de vedere al timpului de calcul.
- Sursele de lumina sunt considerate punctiforme, de aceea nu pot fi obtinute penumbre.

INTRODUCEREA UMBRELOR IN IMAGINI(16)- Z-BUFFER

Williams [WiLL78] a propus o metodă de generare a umbrelor bazată pe execuția de două ori a algoritmului Z-Buffer:

- In prima etapă se construieşte buffer-ul Z al scenei văzută din poziția sursei (în transformarea de proiecție a scenei se consideră poziția sursei).

  Vom nota acest buffer cu ZS. El mai este numit si « **shadow map** », de unde si denumirea sub care este cunoscuta metoda in prezent.

- In etapa a doua se construieşte imaginea văzută din poziția observatorului, utilizand ZS pentru determinarea punctelor aflate in umbra.

Implementarea moderna a metodei este numita “shadow mapping”.

INTRODUCEREA UMBRELOR IN IMAGINI(17)-Z-BUFFER

Etapa I.

Se transforma varfurile poligoanelor cu o matrice « model-view-projection », MS, calculata cu observatorul in pozitia sursei ;

- Se activeaza scrierea in z-buffer si dezactiveaza scrierea in frame-buffer
- Se rasterizeaza scena:

```
Pentru fiecare poligon al scenei
        Pentru fiecare fragment f(x,y,z) al unui poligon
            Dacă z < ZS[y][x] atunci
            //fragmentul este vizibil în pixelul (x,y) din poziția sursei
              ZS[y][x] = z
```

La sfarsitul acestei etape, in ZS sunt memorate coordonatele z ale fragmentelor care primesc lumina de la sursa.

INTRODUCEREA UMBRELOR IN IMAGINI(18)-Z-BUFFER

Etapa II.

Se calculeaza matricea « model-view-projection» cu observatorul in pozitia sa: matricea MO ;

Se asociaza fiecarui varf: pozitia sa in spatiul obiect si pozitia sa transformata cu MS ;

Se rasterizeaza scena:

Pentru fiecare poligon al scenei

Pentru fiecare fragment de coordonate (x,y,z), al unui poligon

Dacă z < Z-Buffer[y][x] atunci //fragmentul este vizibil în pixelul (x,y), din poziția observatorului

(a) Z-Buffer[y][x] = z

(b) Se calculează cordonatele (x',y',z') ale fragmentului in vederea din pozitia sursei (interpoland intre cordonatele varfurilor transformate cu matricea MS)

(c) Dacă ZS[y'][x'] < z' atunci //fragmentul (x,y,z) este în umbră

afişează pixelul (x,y) în intensitatea corespunzătoare umbrei

altfel

afişează pixelul (x,y) în culoarea fragmentului (calculata folosind un model de iluminare)

In cazul mai multor surse de lumină, se utilizează câte un buffer ZS pentru fiecare sursă.