Transparenta
în modelul de iluminare locala

Prof. univ. dr. ing. Florica Moldoveanu

Curs *Elemente de Grafică pe Calculator* – UPB, Automatică şi Calculatoare
2021-2022

MODELAREA TRANSPARENTEI (1)

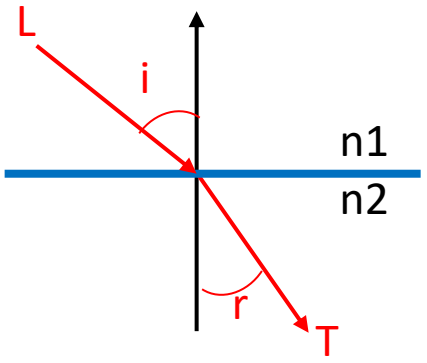
Unele obiecte ale scenei sintetizate pot fi construite din materiale transparente sau translucide.

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), direcţia sa se modifică datorită refracţiei.
- Relaţia dintre unghiul razei incidente, i, şi cel al razei refractate, r, este dată de legea lui Snell:

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

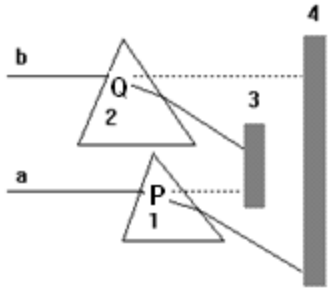
n1 şi n2 sunt indicii de refracţie ai celor două medii (materiale) traversate de lumină.



MODELAREA TRANSPARENTEI (2)

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

Culoarea vizibila intr-un punct al unei suprafete transparente provine de la suprafata din spate aflata 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 (T are directia razei incidente)

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

MODELAREA TRANSPARENTEI (3)

- Refracţia produce, de asemenea, o distorsionare a obiectelor, asemănătoare cu aceea produsă de o proiecţie perspectivă→ dacă se doreşte obţinerea de imagini realiste, trebuie să se ţină cont de refracţie.

- Multe metode practice de modelare a transparenţei ignoră refracţia, astfel încât obiectele vizibile printr-o suprafaţă transparentă sunt cele aflate pe direcţia razei incidente.

Motivul ignorării:

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

Atunci când suprafaţa vizibilă într-un pixel este transparentă, culoarea în care va afişat pixelul se poate obţine combinând culoarea suprafeţei vizibile cu aceea a suprafeţei aflată imediat în spatele său (sau a fondului), folosind următoarea formulă de interpolare:

MODELAREA TRANSPARENTEI (4)

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

$I_{\lambda 1}$ este culoarea suprafetei vizibile în pixel

$I_{\lambda 2}$ este culoarea suprafetei din spatele celei vizibile, care se proiecteaza in acelasi pixel

Coeficientul de transmisie, kt_1, este o măsură a transparenţei suprafeţei vizibile în pixel,

$0 <= kt_1 <= 1$

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

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

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

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

MODELAREA TRANSPARENTEI (5)

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

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

unde

kt_{min} şi kt_{max} reprezintă transparenţa minimă şi cea maximă a suprafeţei

N_z este componenta z a normalei normalizate la suprafaţă în punctul pentru care se calculează kt (normala fragmentului),

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

$N_z = 1 \rightarrow k_t = kt_{max}$ $\quad I_\lambda = (1-kt_{max})*I_{\lambda 1} + kt_{max}*I_{\lambda 2}$

$N_z = 0 \rightarrow k_t = kt_{min}$ $\quad I_\lambda = (1-kt_{min})*I_{\lambda 1} + kt_{min}*I_{\lambda 2}$

Majoritatea algoritmilor de eliminare a partilor nevizibile la afişarea scenelor 3D pot fi adaptaţi pentru a îngloba transparenţa.

- **In algoritmii care afişează poligoanele scenei 3D în ordinea "din spate în faţă"** (back to front), de ex. algoritmul BSP şi Pictorului, $I_{\lambda 1}$ corespunde fragmentului curent iar $I_{\lambda 2}$ este culoarea pixelului în care se afiseaza fragmentul.

MODELAREA TRANSPARENTEI (6)

- Adăugarea efectului de transparenţă în algoritmul Z-Buffer este mai dificilă, deoarece poligoanele sunt rasterizate în ordinea în care sunt transmise în banda grafică, neţinându-se cont de distanţa lor faţă de observator.

→ Nu intotdeauna fragmentul afisat intr-un pixel (x,y) apartine suprafetei aflate imediat in spatele celei din care face parte fragmentul curent.

OpenGL permite specificarea modului de amestec dintre culoarea fragmentului curent si cea a pixelului in care se afiseaza daca cele 2 culori sunt reprezentate in modelul (R, G, B, A):

A – opacitatea culorii: 0<= A <= 1 0 – transparenta; 1 – opaca

- Amestecul culorilor este o operatie raster efectuata de procesorul de rasterizare dupa testul de adancime, inainte de actualizarea valorii pixelului in bufferul imagine.

MODELAREA TRANSPARENTEI (7)

Functiile OpenGL pentru specificarea modului de amestec dintre culoarea fragmentului curent si cea a pixelului in care se afiseaza:

glEnable(GL_ALPHA_TEST) – activeaza testul alfa

glAlphaFunc (func, ref) – specifica functia “alpha test”

func - functia de comparare a valorii A a fragmentului cu o valoare de referinta, 0<=ref<=1;

Exemplu: glAlphaFunc(GL_GREATER, 0.5)

– fragmentul este afisat daca opacitatea sa este > 0.5

Daca fragmentul curent

- nu “trece” testul alfa, culoarea pixelului nu se modifica
- altfel, el va modifica culoarea pixelului, dacă trece testul stencil si testul de adancime

glEnable(GL_BLEND) - activeaza amestecul culorilor

MODELAREA TRANSPARENTEI (8)

Amestecul culorilor in OpenGL:

Cfinal =Cfrag ∗ sfactor + Cpixel ∗ dfactor

Specificarea modului de amestec (daca fragmentul trece testul alfa):

glBlendFunc(GLenum sfactor, GLenum dfactor)

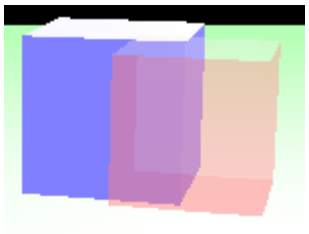
Exemplu:

glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA):

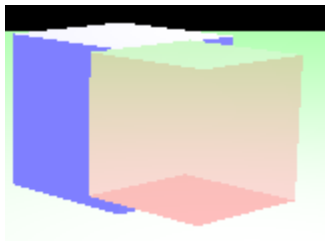
noua_culoare_pixel = Culoare_fragment * A_fragment + Culoare_pixel * (1 – A_fragment)

Daca 0 < A_fragment < 1 noua_culoare_pixel va fi o combinatie intre cele 2 culori.

(https://learnopengl.com/Advanced-OpenGL/Blending)



Rasterizare:
- Cub albastru, opac
- Cub rosu, transparent



Rasterizare:
- Cub rosu, transparent
- Cub albastru, opac

Cubul rosu este pozitionat in fata celui albastru;
A_cub rosu =0.3

Fragment shader

```
in vec3 world_position;
in vec3 world_normal;

uniform vec3 light_position;
uniform vec3 eye_position;
……
uniform vec3 object_color;
uniform float object_alpha;

layout(location = 0) out vec4 out_color;

void main()
{
   // Calculeaza culoarea folosind modelul de iluminare locala
   ……
    vec3 color = object_color + ambient_color + light_color* (diffuse_color + specular_color);

    out_color = vec4(color, object_alpha);
}
```