Iluminarea globala
-Ray Tracing-

Prof. univ. dr. ing. Florica Moldoveanu

Curs *Elemente de Grafică pe Calculator* – UPB, Automatică şi Calculatoare
2020-2021

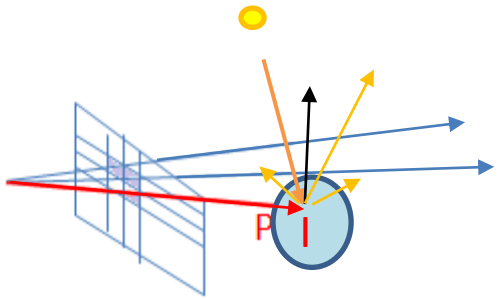
Ray-casting vs Ray-tracing

- In algoritmul Ray Tracing:

❖ Imaginea este calculata pixel cu pixel, intersectand scena 3D cu raze care pleaca din pozitia observatorului si intersecteaza planul imaginii in puncte ce corespund pixelilor imaginii.

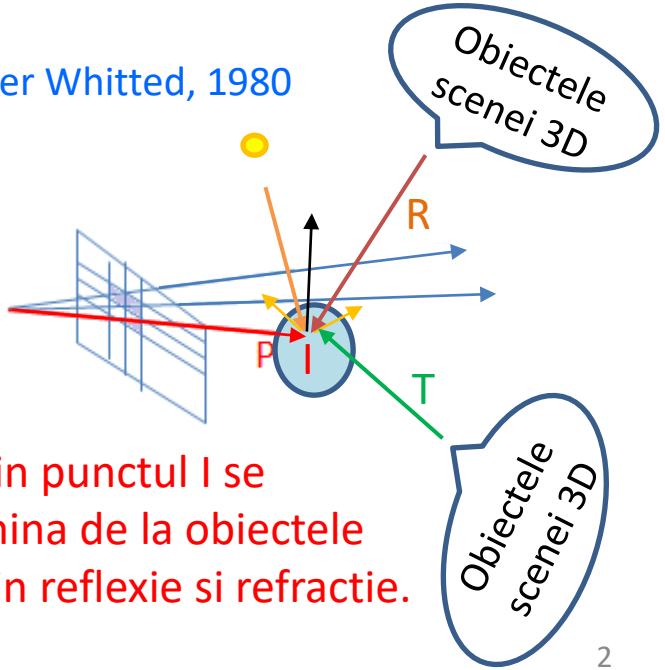
- Culorile pixelilor sunt calculate in punctele de intersectie ale razelor cu scena 3D.

Ideea calculării imaginii prin intersectia razelor vederii cu scena a aparut in algoritmul Ray-casting, predecesorul algoritmului Ray-tracing actual, care este numit si Ray-tracing recursiv.



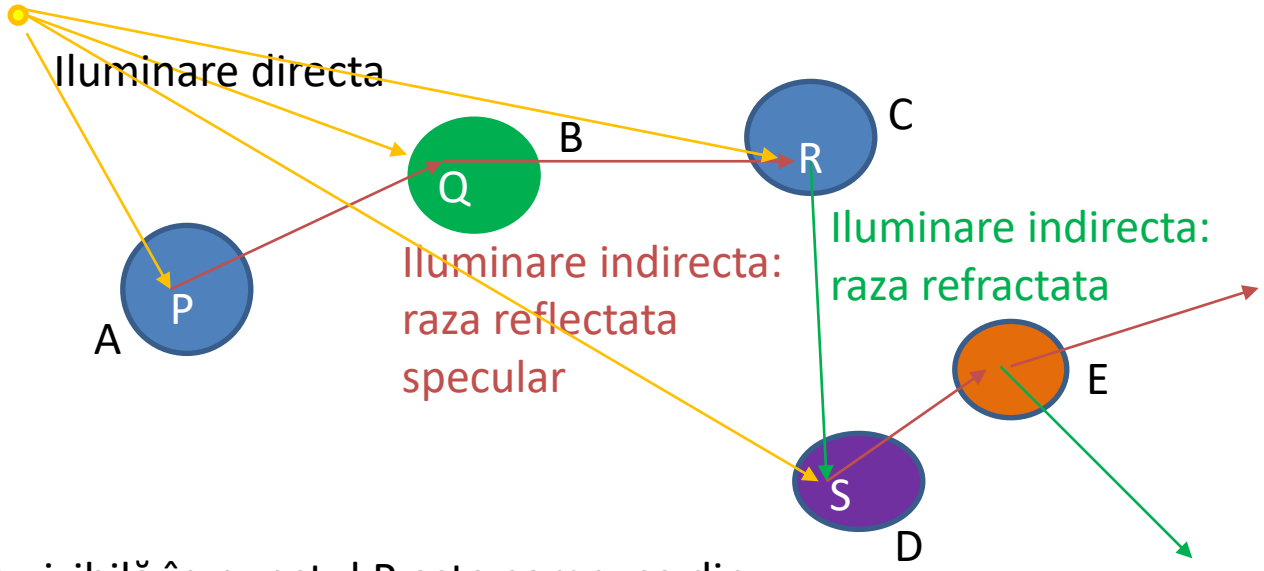
Ray-casting: culoarea pixelului este calculata in I tinand cont de lumina provenita de la surse, aplicand modelul de iluminare locala in I. Sunt eliminate partile nevizibile ale scenei 3D.

Ray-tracing: Turner Whitted, 1980



Ray-tracing: in punctul I se primeste lumina de la obiectele scenei 3D prin reflexie si refractie.

Iluminarea globala în Ray tracing



Culoarea vizibilă în punctul P este compusa din:
reflexia luminii de la sursă + lumina provenita din punctul Q (compusă din reflexia luminii de la sursă + lumina provenita din punctul R (compusa din reflexia luminii de la sursă + lumina provenita din punctul S))

- Culoarea vizibilă într-un punct S al unei suprafețe:

lumina reflectată datorită iluminarii directe (iluminarea locală) +

lumina provenită prin iluminare indirectă (pe directia razei speculare si a celei transmise)

$I_\lambda(S) = I_{local\lambda}(S) + Ks*R_\lambda(S) + Kt*T_\lambda(S)$ unde $R_\lambda(S)$ şi $T_\lambda(S)$ se calculeaza recursiv

Ray tracing

Imbina:

- Eliminarea partilor nevizibile ale scenelor 3D
- Calculul reflexiei luminii
- Calculul refractiei luminii (transparenta)
- Calculul umbrelor
- Interactiunea globala a reflexiei si refractiei luminii la nivelul intregii scene 3D

➢ In algoritmul Ray Tracing:

❖ Imaginea unei scene 3D NU se obtine prin rasterizarea de primitive grafice.

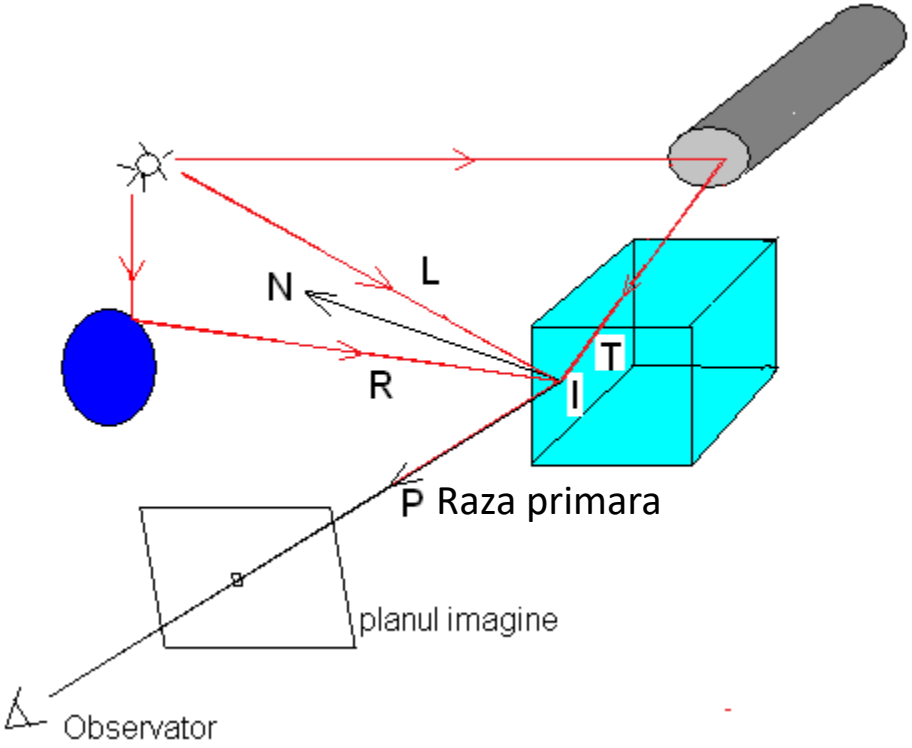
❖ Obiectele 3D (ex. un cub, o sfera) nu trebuie sa fie descompuse in primitive grafice

❖ Imaginea unei scene 3D NU se obtine prin executia operatiilor din banda grafica OpenGL

❖ O imagine a scenei 3D este calculata intersectand scena 3D cu raze care pleaca din pozitia observatorului si trec prin puncte din planul imaginii ce corespund pixelilor imaginii.

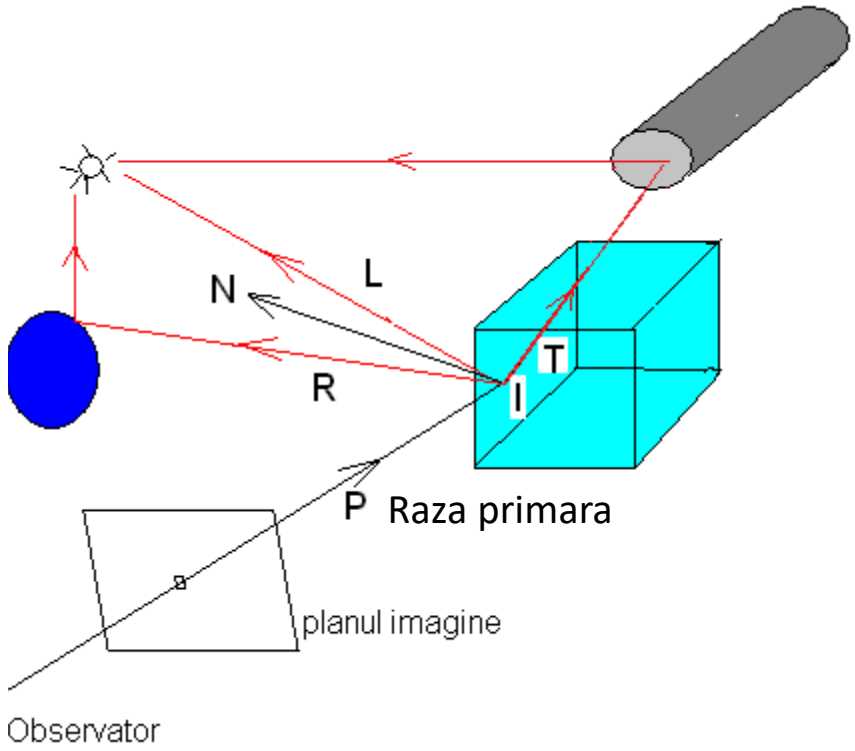
➢ Culorile pixelilor sunt calculate in punctele de intersectie ale razelor cu scena 3D.

Razele folosite in Ray tracing(3)



Directia razelor de lumina **in realitate**.

Pixelul este colorat in culoarea punctului I.
P este directia pe care lumina din punctul I ajunge la observator.



In algoritmul Ray-tracing,

pentru calculul culorii pixelului se considera raze cu directia inversa celor din realitate

Calculul razelor primare (1)

❖ Se stabileste sistemul de coordonate al camerei virtuale, folosind o functie similara cu lookAt:

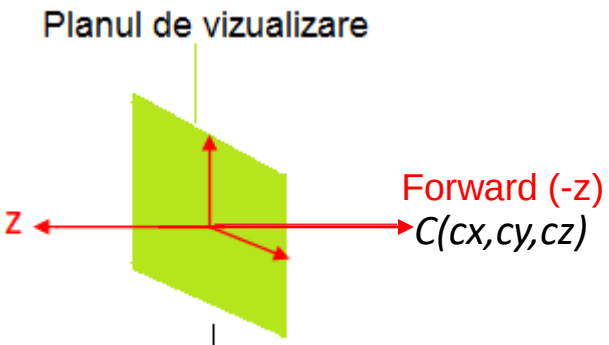
glm::mat4 **V** = glm::lookAt (glm::vec3 (xo,yo,zo), glm::vec3 (cx,cy,cz), glm::vec3 (Upx, Upy, Upz))

O*(xo, yo, zo):* **poziţia observatorului;**

C(cx,cy,cz): **centrul de interes** - punctul din scena 3D catre care priveste observatorul

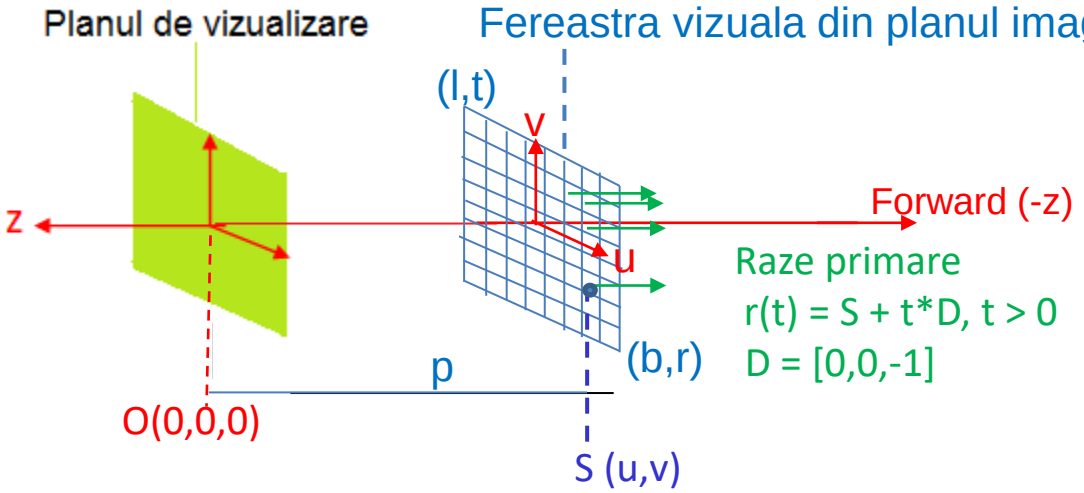
(Upx, Upy, Upz): **vectorul "sus" al camerei (Up)**

- Folosind parametrii functiei lookAt se determina sistemul de cordonate observator (al camerei virtuale) din planul de vizualizare.
- Obiectele scenei se transforma in sistemul coordonatelor observator folosind matricea V.



O(xo, yo, zo) →(0,0,0) in sistemul de coordonate observator

Calculul razelor primare (2)
Proiectia ortografica



Obiectele redate in imagine sunt cele aflate la o distanta fata de O mai mare decat p.

Planul imagine: paralel cu planul de vizualizare

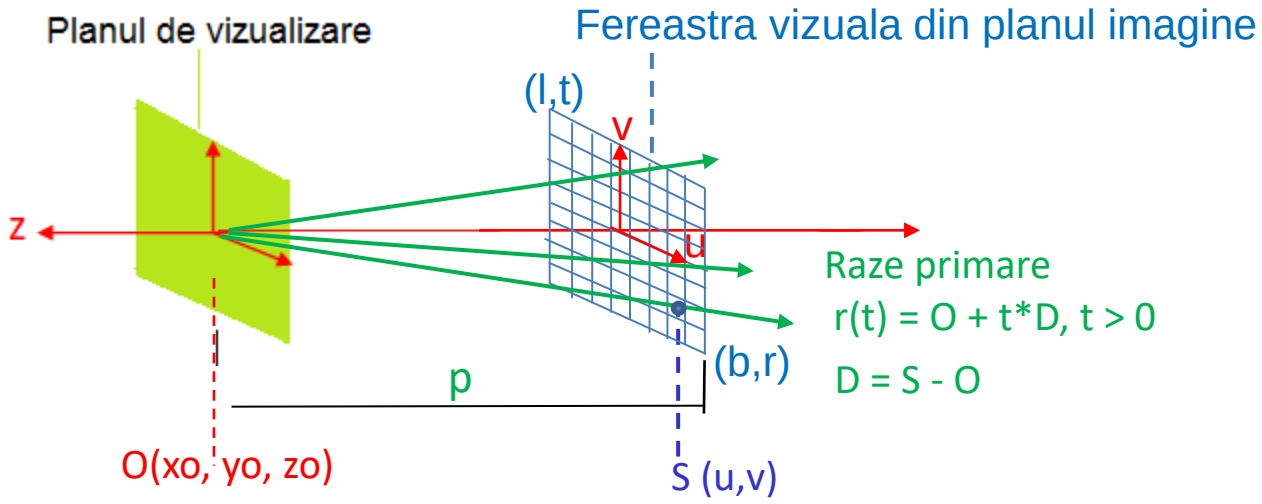
p: distanta de la planul de vizualizare la planul imagine

(u,v): axele atasate planului imagine, paralele cu axele OX, OY din planul de vizualizare

(l,t) – (b,r): sunt colturile stanga-sus si dreapta-jos ale ferestrei din planul imagine in care se calculeaza imaginea; are laturile paralele cu axele (u,v)

- Fiecare rază primară, r(t), are originea intr-un punct din fereastra vizuala, S, ce corespunde unui pixel şi direcția D = [0,0,-1] → razele primare sunt perpendiculare pe planul imagine.

Calculul razelor primare (3)
Proiectia perspectiva



- Fiecare rază primară, r(t), are originea in pozitia observatorului, O, si trece printr-un punct al planului imagine, S, ce corespunde unui pixel.

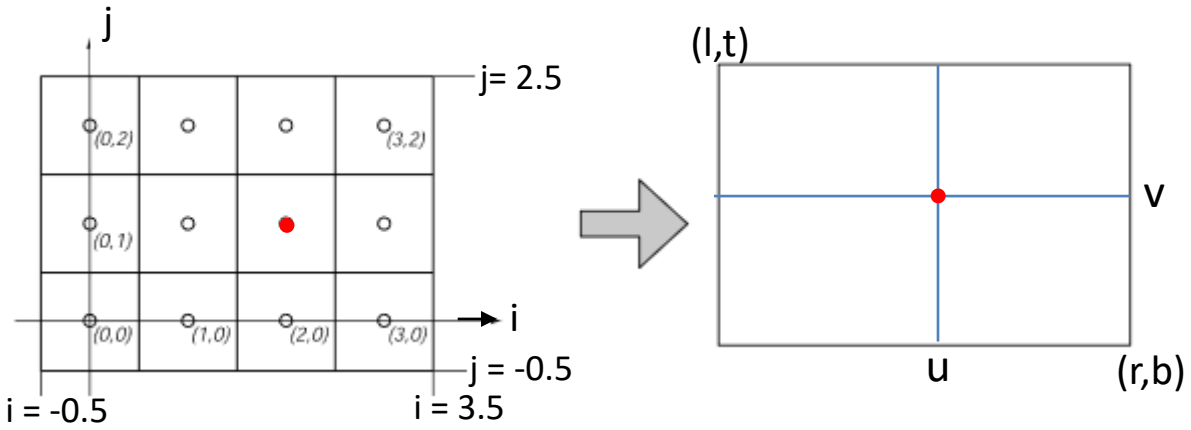
  Direcția razei este D = S - O

p: distanta de la observator la planul imagine controleaza distanta focala (analog cu distanta la planul din fata al proiectiei perspectiva in OpenGL);

- împreuna cu fereastra vizuală determină câmpul de vizualizare (field of view) al camerei.

Corespondenta imagine – fereastra vizuala

Calculul coordonatelor (u,v) corespunzatoare unui pixel



Coordonatele (u,v) se obtin printr-o transformare fereastra – poarta.

Considerand centrul pixelului (j,i):

$(u-l)/(r-l) = (i-(-0.5))/Nx$

$(v-b)/(t-b) = (j-(-0.5))/Ny$

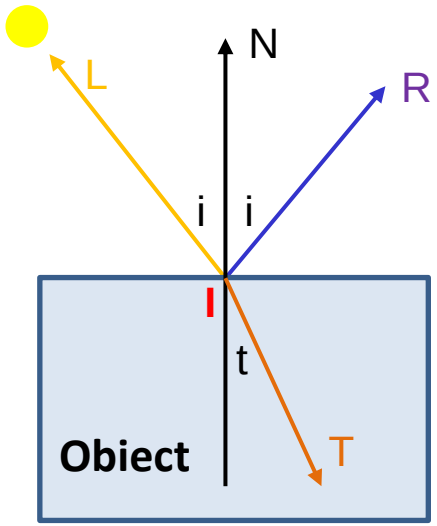
$u = l + (r - l)*(i + 0.5)/Nx$

$v = b + (t - b)*(j + 0.5)/Ny$

Nx, Ny – rezolutia imaginii pe axele X si Y: dimensiunile viewport-ului

Calculul razelor secundare

I: originea razei secundare



L: raza din I catre sursa de lumina

R: raza reflexiei speculare, simetrica cu L, fata de N,

$$R = 2(N \cdot L) \cdot N - L$$

Ecuatia razei: r(t) = I +R*t, t>0

T: raza refractiei luminii in I, calculata pe baza legii lui Snell:

$n1/n2 = \sin(t)/\sin(i)$, n1,n2: indicii de refractie

$T = L*(n1/n2) - (\cos(t) + (n1/n2)*(L \cdot N))*N$

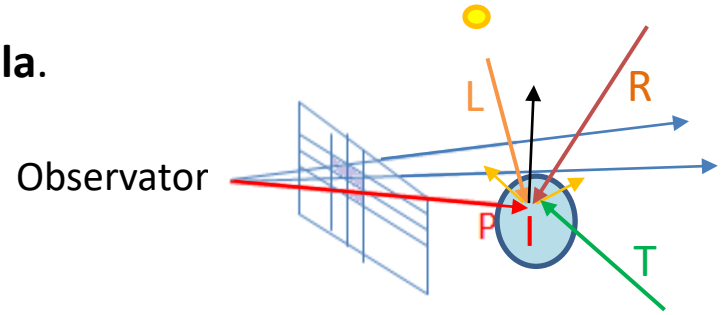
Ecuatia razei: r(t) = I + T*t, t>0

✓*Reflexia speculara si refractia au loc fara împrastiere (sunt perfect focalizate):* **reflexia speculara este considerata strict pe directia R şi nu într-un con de raze, ca in modelul Phong.**

➢**Efect**: obiectele din imaginea produsa sunt de regula stralucitoare, producand reflexii multiple focalizate.

Algoritmul Ray tracing

Imaginea se calculeaza pixel cu pixel, în fereastra vizuala.



Considerand o singura sursa de lumina:

Pentru fiecare pixel al imaginii:

- Se calculeaza raza primara, P.
- Se determina intersectiile razei primare cu scena si punctul de intersectie**, I**, cel mai apropiat de originea razei.
- **Daca** raza nu intersecteaza nici un obiect al scenei,
  - Se afiseaza pixelul in culoarea fondului

  altfel
  - Se calculeaza culoarea in punctul I, tinand cont de:
    - Reflexia luminii provenita direct de la sursa de lumina, raza L
    - Lumina provenita in I de la alte obiecte pe directia reflexiei speculare, raza R
    - Lumina provenita in I de la alte obiecte pe directia razei transmise, raza T
  - Se afiseaza pixelul in culoarea obtinuta prin combinarea contributiei celor trei raze

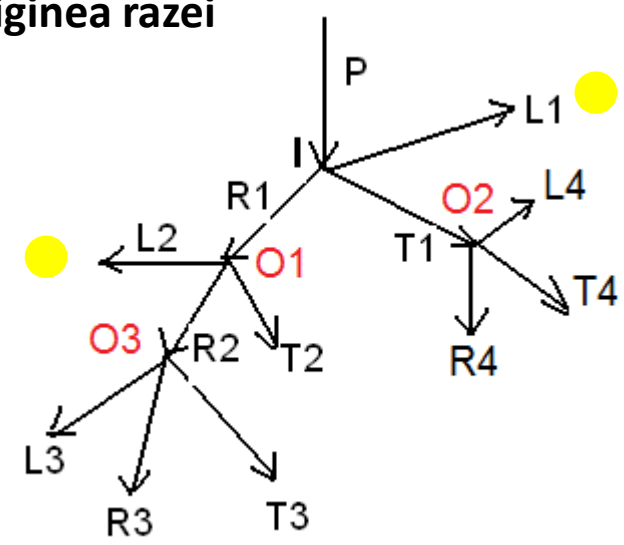
Ray tracing – arborele de raze

Algoritmul Ray tracing este recursiv:

Lumina provenita in punctul **I** pe directia razei speculare de la un obiect O1 (de-a lungul razei R1) sau prin transmisie de la un obiect O2 (de-a lungul razei T1), poate fi compusa din:

- Reflexia luminii provenita direct de la o sursa, de catre O1/O2 – raza L2/L4
- Reflexia speculara a altor obiecte de catre O1/O2 – raza R2/R4
- Transmisia luminii provenita de la alte obiecte ale scenei prin O1/O2 – raza T2/T4

➢ **Pentru fiecare raza se considera punctul de intersectie cu scena, care este cel mai apropiat de originea razei**



Arborele de raze

➢*Pentru obtinerea culorii in punctul I, se evalueaza arborele de raze de la frunze catre radacina*

Ray tracing- calculul culorii unui pixel (1)

Calculul culorii în punctul de intersectie al razei primare cu scena:

$$I_\lambda(I) = I_{local\lambda}(I) + Ks*R_\lambda(I) + Kt*T_\lambda(I)$$

unde:

λ – reprezinta lungimea de unda: expresia se evalueaza pentru R,G,B

$I_{local\lambda}(I)$ – reprezinta reflexia luminii provenite direct de la sursele de lumina din scena 3D (calculata folosind modelul de iluminare locala)

Ks – este coeficientul de reflexie speculara al materialului obiectului

$R_\lambda(I)$ – reprezinta lumina provenita de la alte obiecte ale scenei in punctul I, pe directia razei speculare: **se obtine prin evaluarea arborelui de raze**

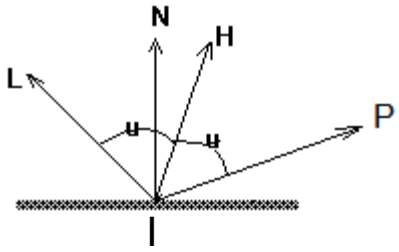
Kt – este coeficientul de transmisie, specific materialului obiectului

$T_\lambda(I)$ - reprezinta lumina provenita prin transmisie (refractie) in punctul I de la alte obiecte ale scenei: **se obtine prin evaluarea arborelui de raze**

0<= ks, kt <= 1

Ray tracing- calculul culorii unui pixel (2)

Modelul de iluminare locala (reflexia lumii in punctul I) :



$I_{local\lambda}(I) = Ia_\lambda * Ka + Isursa_\lambda * fat * s * lum * [kd*(N \cdot L) + Ks*(N \cdot H)^n]$

Iaλ – reprezinta intensitatea luminii ambiante

Ka – este coeficientul de difuzie a luminii ambiante, specific materialului obiectului, 0<= ka <= 1

$Isursa_\lambda$ – reprezinta intensitatea luminii provenite de la sursa

L – este versorul directiei din punctul I catre pozitia sursei de lumina

N – este normala in punctul I (versor)

H – este versorul directiei bisectoare a unghiului dintre L si vectorul din I catre observator (raza P)

fat – este factorul de atenuare a luminii de la sursa, proportional cu distanta de la sursa la punctul I

0<= s <= 1,

s = 1, daca vectorul L nu intersecteaza un alt obiect al scenei

s = 0, daca lumina de la sursa nu ajunge in punctul I: L intersecteaza un obiect opac al scenei

0 < s <1, daca vectorul L intersecteaza un obiect transparent (sau mai multe)

Punctul I primeste lumina de la sursa daca:

- produsul scalar (N·L)>0 (lum =1, altfel =0)
- raza L nu intersecteaza un obiect opac al scenei

➢ **Daca I nu primeste lumina de la sursa, iluminarea locala se reduce la componenta ambianta.**

Ray tracing- calculul culorii unui pixel (3)

- Daca in scena 3D exista mai multe surse de lumina, fiecare poate contribui in mod diferit la $I_{local\lambda}(I)$:

$$I_{local\lambda}(I) = Ia_{\lambda} * Ka + \sum_{i=1,n} Isursa_{i,\lambda} * fat_i * s_i * lum_i * [kd * (N \cdot L_i) + Ks * (N \cdot H_i)^n]$$

- Pentru evaluarea componentelor $R_{\lambda}(I)$ si $T_{\lambda}(I)$ din calculul culorii in I,

$$I_{\lambda}(I) = I_{local\lambda}(I) + Ks * R_{\lambda}(I) + Kt * T_{\lambda}(I)$$

se coboara in arborele de raze pana la un numar pre-specificat de nivele (energia luminoasa scade destul de repede!).

Algoritmul Ray tracing

Algoritmul Ray tracing

Pentru fiecare pixel al imaginii

```
{    *calculeaza raza primara, P;
     *culoare_pixel = TraseuRaza(P, 1); // culoarea in punctul de intersectie al razei P cu scena
     *afiseaza pixelul in culoare_pixel;
}
```

Culoare TraseuRaza**(Raza R, int n)** //apelata pt. toate razele care contribuie la cul. unui pixel

```
{    // n este nivelul in arborele de raze
     // intoarce culoarea in punctul de intersectie al razei R cu scena
     *calculeaza intersectiile razei R cu obiectele scenei;
     daca (nu exista intersectii), atunci
               return  (culoare_fond);
     altfel
        *fie I punctul de intersectie cel mai apropiat de originea razei si O obiectul intersectat;
        *calculeaza normala, N, in punctul I;
         return  CuloarePunct(O, R, I, N, n); // culoarea in I
}
```

Algoritmul Ray tracing

```
Culoare CuloarePunct(Obiect O, Raza R, Punct I, Normala N, adancime_arbore n)
{    Culoare  culoare;
     culoare = culoare_ambianta;
     pentru fiecare sursa de lumina S executa //calculeaza iluminarea locala
          *calculeaza vectorul L, din I catre S
           daca ((Nu·Lu) > 0 si vectorul L nu intersecteaza un obiect opac al scenei) atunci
             *calculeaza contributia sursei S la culoare, CS, folosind modelul de iluminare locala
              culoare = culoare + CS
     daca (n<=nivel_max) atunci // n>nivel_max: iesirea din recursivitate
          daca (obiectul O produce reflexii speculare) atunci
               *calculeaza raza reflectata in punctul I, RS;
                culoare = culoare + Ks* TraseuRaza(RS, n+1); // se coboara in arbore
          daca (obiectul O este transparent) atunci
               *calculeaza raza transmisa (refractata) in punctul I, RT;
                culoare = culoare + Kt* TraseuRaza(RT, n+1); //se coboara in arborele de raze
     return (culoare)
}
```

Algoritmul Ray tracing

Deficienţele algoritmului

1. **Efecte de aliasing**:

- In cazul proiectiei perspectiva, razele sunt convergente: esantionarea spatiului 3D este neuniforma.
- Se produc defecte la intersectiile razelor cu marginile suprafetelor (aliasing: marginile au aspect zimţat)
- Obiectele mici si subtiri pot sa apara si sa dispara din imagine, la schimbarea pozitiei observatorului

Imbunatatire:

– marirea rezolutiei esantionarii spatiului 3D; ex. 4 raze primare/pixel → creste complexitatea

2. **Complexitatea computationala**

- Buna din punct de vedere teoretic:

  - pentru o raza/pixel este de O(p*n), unde p este nr. de pixeli - constant (independent de scena) = O(n), unde n este nr. de obiecte din scena
- Creste liniar cu numarul de raze/pixel
- Complexitatea calculelor de intersectie depinde de geometria obiectelor intersectate

Imbunatatire:

– Reducerea calculelor de intersectie, prin folosirea de volume incadratoare ale obiectelor si gruparea obiectelor din scena 3D

– Complexitatea se poate reduce la O(log (n))

Imbunatatirea calitatii imaginii(1)

Marirea numarului de raze primare

1) Uniforma

4 raze primare/pixel, corespunzatoare colturilor suprafetei pixelului

- Culoarea pixelului se determina ca medie a culorilor obtinute cu cele 4 raze primare
- Razele primare utilizate pentru un pixel contribuie si la culoarea pixelilor adiacenti
  - Pentru o imagine de mxn pixeli: (m+1)x(n+1) raze→ numarul de raze creste cu (m+n+1)

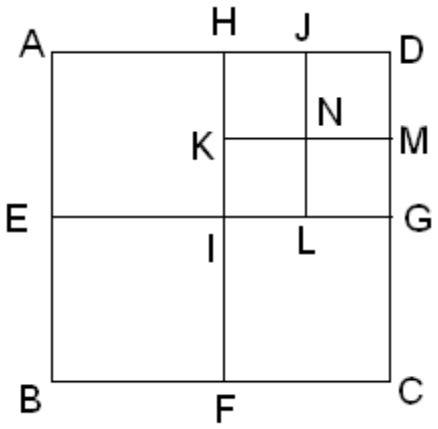
2) **Adaptiva** (Whitted) – subdivizarea adaptiva a suprafetei pixelului

- Se creste rezolutia eşantionarii spatiale numai in zonele in care este necesara operatia de anti-aliasing:
- Daca diferenta dintre culorile celor 4 raze primare pentru un pixel este mare, se subdivizeaza suprafata pixelului in 4 subzone si se duc raze prin colturile subzonelor
- Se aplica acelasi criteriu de comparatie intre culorile celor 4 raze primare corespunzatoare unei subzone
- Subdivizarea se continua recursiv pana la un nivel maxim prestabilit sau pana cand diferenta dintre cele 4 culori scade sub un prag dat
- Culoarea pixelului se obtine ca o medie ponderata a culorilor subzonelor in care a fost divizata suprafata pixelului

Imbunatatirea calitatii imaginii(2)

Imbunatatirea calitatii imaginii(2)

Exemplu de subdivizare adaptiva



CA, CB, CC, CD, CE, CF, CG, CH, CI, CJ, CK, CL, CM, CN

- sunt culorile obtinute cu razele primare care trec prin punctele respective

Culoare pixel (A-B-C-D) =

¼((CA+CE+CI+CH)/4 + (CE+CB+CF+CI)/4 + (CI+CF+CC+CG)/4 +

¼((CH+CK+CN+CJ)/4 + (CK+CI+CL+CN)/4 + (CJ+CN+CM+CD)/4 +

(CN+CL+CG+CM)/4))

Reducerea complexitatii computationale

Reducerea calculelor de intersectie

1. **Utilizarea de volume incadratoare la nivel de obiect**

   Se testeaza intersectia raza-volum incadrator in loc de raza-obiect:

   - Daca raza nu intersecteaza volumul incadrator → nu se va calcula intersectia cu obiectul

2. **Divizarea scenei in volume incadratoare**

   Se divizeaza scena in volume egale pana la o anumita rezolutie spatiala sau se divizeaza scena adaptiv.

   Se testeaza intersectia razei cu volumele in care a fost divizata scena.

3. **Reprezentarea scenei printr-o ierarhie de volume incadratoare**

   Scena este reprezentata printr-un arbore in care frunzele sunt obiectele scenei iar nodurile interne sunt grupari logice de obiecte ale scenei.

   Fiecare nod are un volum incadrator.

Utilizarea de volume incadratoare (1)

Utilizarea de volume incadratoare la nivel de obiect/ grup de obiecte

- Permite evitarea calculelor de intersectie cu obiecte ale scenei care pot avea geometrie complexa
- Permite eliminarea din calculele de intersectie a unui intreg grup de obiecte care nu este intersectat de raza

Volume incadratoare:

- Sfera, paralelipipedul cu fetele paralele cu planele principale, elipsoidul, cilindrul.
- Calculul intersectiei raza-volum incadrator trebuie sa fie mai simplu decat calculul intersectiei cu obiectul (de ex. o retea poligonala)
- Volumul incadrator al unui obiect se alege in functie de forma obiectului

Utilizarea de volume incadratoare (2)

Calcule de intersectie raza-volum incadrator

Ecuatia razei: r(t) = P0 + t*D ,

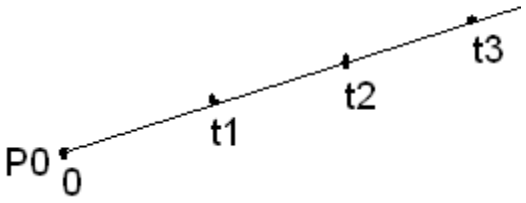
P0 – este originea razei

D – este directia razei

$x = x0 + t*dx$

$y = y0 + t*dy$

$z = zo + t*dz$



Punctul cel mai apropiat de originea razei este r(tmin)

1) Intersectia cu sfera

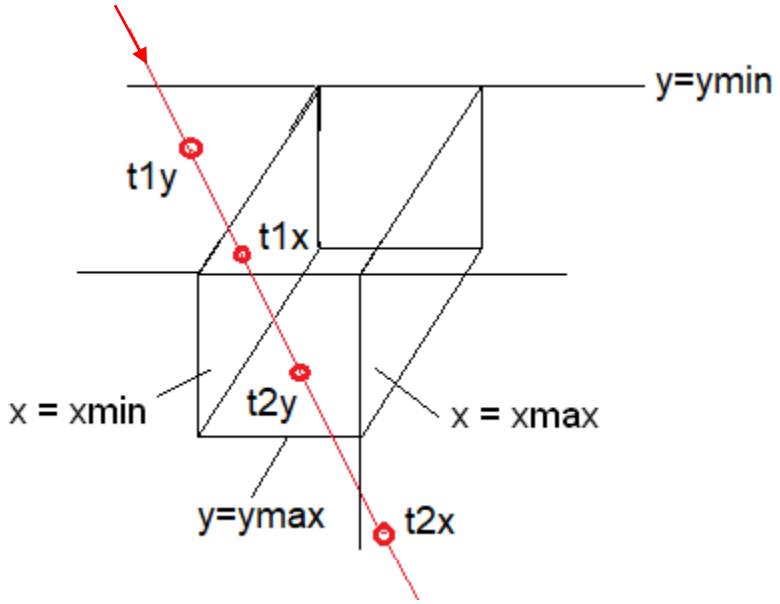
$(x-a)^2 + (y-b)^2 + (z-c)^2 = r^2$

- Se inlocuiesc in ec sferei x, y, z, cu cele din ecuatia razei:

  $(x0+t*dx-a)^2 + (y0+t*dy-b)^2 + (z0+t*dz-c)^2 = r^2$
- Rezulta o ecuatie de grad 2 in t
- Se calculeaza discriminantul Δ, al ecuatiei:
  - Δ <0 – raza nu intersecteaza sfera
  - Daca sfera este obiect al scenei (intereseaza punctul de intersectie):
    - Δ =0 – raza este tangenta la sfera
    - Δ >0 – se calculeaza radacinile, t1, t2
      - punctul de intersectie mai apropiat de originea razei este r(tmin(t1,t2))

Utilizarea de volume incadratoare (3)

2) Intersectia cu un paralelipiped cu fetele paralele cu planele principale



Planele care delimiteaza volumul:

x= xmin, x = xmax, y=ymin, y = ymax, z= zmin, z= zmax

Intersectia razei cu planele x=xmin si x=xmax

xmin = x0 + t*dx → t1x = (xmin – x0)/dx

xmax = x0 + t*dx →t2x = (xmax – x0)/dx

tix = min(t1x, t2x), tex = max(t1x, t2x)

Analog pentru intersectia cu planele y=ymin si y=ymax:

tiy = min(t1y, t2y), tey = max(t1y, t2y)

Raza intersecteaza volumul daca: max(tix, tiy) < min (tex, tey) si
coordonatele punctelor de intersectie sunt intre zmin – zmax

- Se testeaza intersectiile cu planele zmin, zmax

tix=min(t1x,t2x) = t1x

tex=max(t1x,t2x) = t2x

tiy=min(t1y,t2y) = t1y

tey=max(t1y,t2y) = t2y

max(tix,tiy) = max(t1x,t1y) = t1x

min(tex,tey) = min(t2x,t2y) = t2y

t1x<t2y

Utilizarea de volume incadratoare (4)

daca max(tix,tiy) > min(tex,tey) → raza nu intersecteaza volumul

altfel, se calculeaza intersectia cu z=zmin si z= zmax

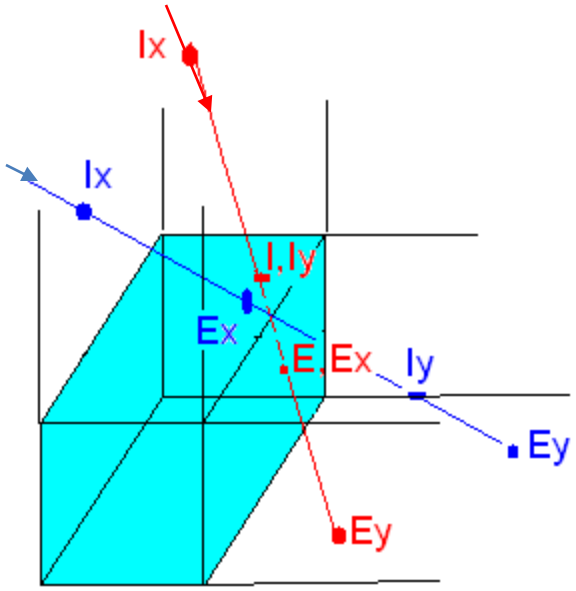
daca max(tix, tiy, tiz) < min(tex, tey, tez)

atunci raza intersecteaza volumul

Punctele de intrare si iesire din volum sunt:

I = r(max(tix, tiy, tiz))

E = r(min(tex,tey,tez)



Raza albastra intersecteaza prelungirile fețelor x=xmin si x=xmax

Ix, Ex – intersectiile razei albastre cu x=xmin, x=xmax

Iy, Ey – intersectiile razei albastre cu y=ymin, y=ymax

Raza rosie intra in volum prin fața y=ymin si iese prin fața x=xmax

Ix, Ex – intersectiile razei rosii cu x=xmin, x=xmax

Iy, Ey – intersectiile razei rosii cu y=ymin, y=ymax

max(tix, tiy) = tiy , min(tex,tey)=tex

max(tix,tiy) > min(tex,tey)

- Raza albastra nu intersecteaza volumul

max(tix, tiy) = tiy, min(tex,tey) = tex

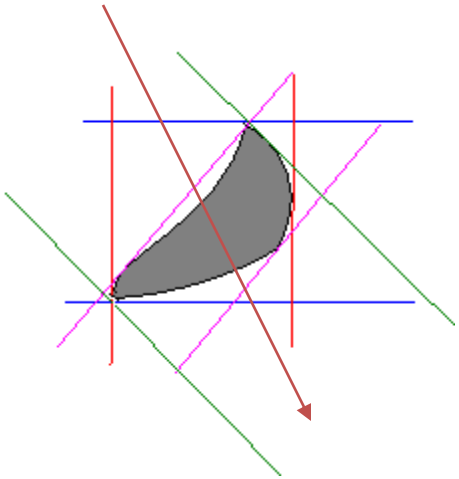
max(tix,tiy) < min(tex,tey)

- Se calculeaza tiz, tez

Utilizarea de volume incadratoare (5)

3) Intersectia cu un volum incadrator poliedru convex

- Volumul incadrator (Kay, Kajiya[1986]) este un poliedru convex format din intersectiile a 4 perechi de plane paralele, inclinate la 0, 45, 90, 135 grade fata de planul orizontal.
- Planele incadreaza minimal obiectul



Ec. unui plan: A*x + B*y + C*z + D =0

- Fie t11 si t12 - valorile lui t pentru intersectiile cu o pereche de plane paralele

  t1min = min(t11,t12) corespunde punctului de intersectie mai apropiat de originea razei
- Fie t21, t22 - valorile lui t pentru intersectiile cu urmatoarea pereche de plane paralele

  t2min = min (t21,t22) corespunde punctului de intersectie mai apropiat de originea razei
- daca max (t1min, t2min) > min (t1max, t2max) raza nu intersecteaza volumul

  altfel, se continua cu intersectia urmatoarei perechi de plane

➢ **Punctele de intersectie cu volumul sunt**: I = r(max($t_i$min)), E = r(min($t_i$max))

Scena reprezentata printr-o ierarhie de volume incadratoare

Ierarhie de volume incadratoare de grupuri de obiecte (Bounding Volume Hierarchy – BVH)

▪ Scena 3D este reprezentata printr-un arbore (arborele scenei) in care fiecare nod are atasat un volum incadrator pentru un grup de obiecte: vezi cursul 8 (BVH).

▪ Radacina contine volumul incadrator al intregii scene

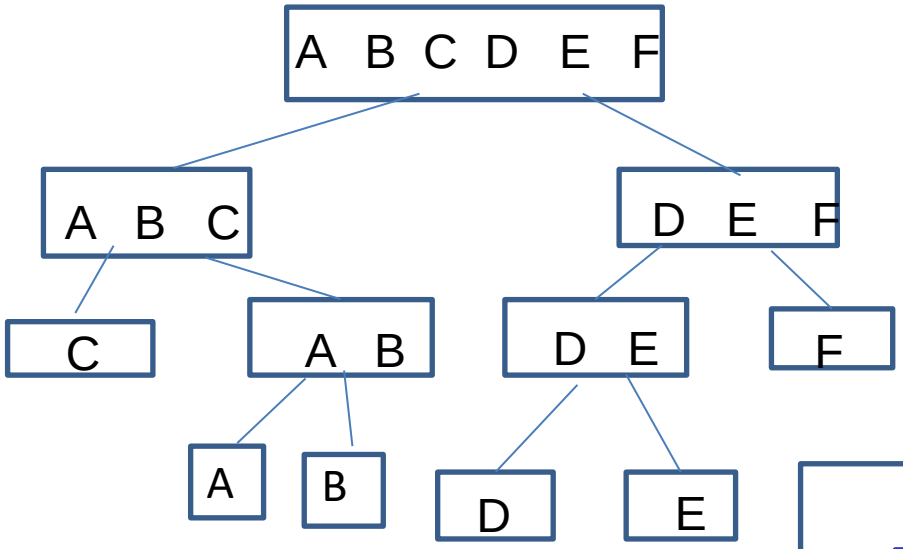
▪ Fiecare frunza contine un obiect al scenei (sau mai multe)

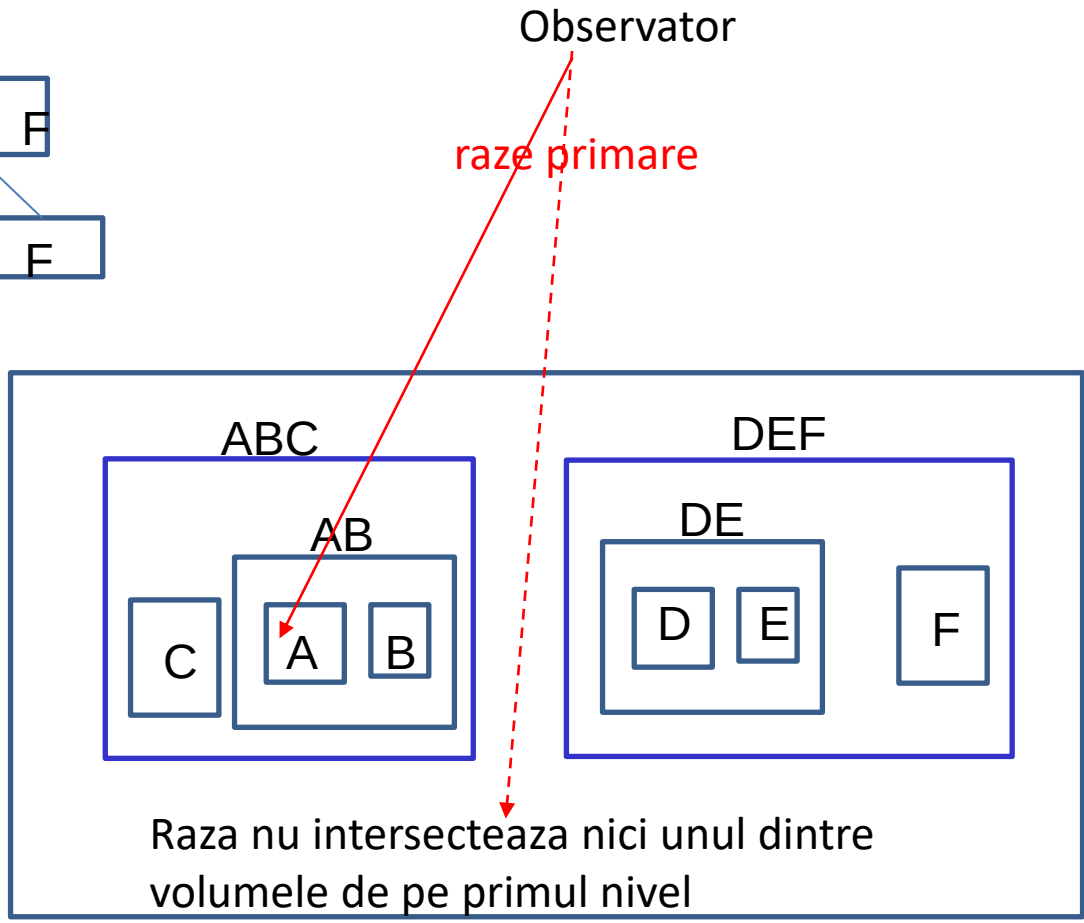
▪ Nodurile interne contin volume incadratoare pentru grupuri de obiecte.

▪ Pentru intersectia razei cu scena este parcurs arborele de la radacina spre frunze

- daca raza nu intersecteaza volumul incadrator al unui nod atunci:
  - nici unul dintre obiectele din volumul nodului nu este intersectat
- daca raza intersecteaza volumul incadrator al unui nod atunci:
  - se testeaza intersectia razei cu volumele incadratoare din subarborele nodului

Scena reprezentata printr-o ierarhie de volume incadratoare





- Raza intersecteaza volumul incadrator al nodului (ABC)

↓

Se coboara in subarborele nodului
- Raza nu intersecteaza volumul C
- Raza intersecteaza volumul (AB)

↓

Se coboara in subarborele nodului
- Raza intersecteaza volumul A

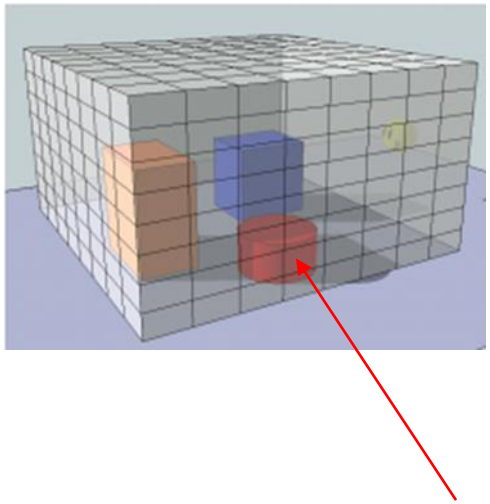
↓

Se intersecteaza raza cu obiectul din A

Scena divizata in volume incadratoare (1)

Divizarea regulata a scenei

- Se porneste de la paralelipipedul incadrator al scenei, care se divizeaza recursiv in opt subvolume egale, pana la o anumita rezolutie.
- Subvolumele finale sunt numite **voxeli**.
- La divizare nu se tine cont de structura scenei
- Fiecarui voxel ii este asociata o lista de obiecte pe care le contine.
- Un obiect se poate afla in mai multi voxeli (liste)



Intersectia unei raze cu scena:

- Raza este intersectata numai cu obiectele din voxelii traversati de raza
- Traseul razei prin volumul de voxeli poate fi calculat eficient printr-un algoritm DDA 3D (extensie pentru 3D a algoritmului DDA pentru rasterizarea vectorilor).

Scena divizata in volume incadratoare (2)

Voxelul de start: prin care raza primara intra in volumul scenei

Voxel_curent = voxel de start; gata = false; in_volum = true;

cât timp (!gata && in_volum)

{ **daca** voxelul curent nu este gol (lista de obiecte nu este vida) **atunci**

{*se intersecteaza raza cu obiectele continute in voxelul current;

daca exista intersectii **atunci**

{ *se retine punctul de intersectie cel mai apropiat de originea razei;

gata = true;

continue;

}

}

// voxelul curent este gol sau nu exista intersectii cu obiectele voxelului curent

*determina urmatorul voxel de pe traseul razei;

daca raza iese din volum **atunci**

in_volum = false;

altfel voxel_curent = urmatorul voxel de pe traseul razei;

}

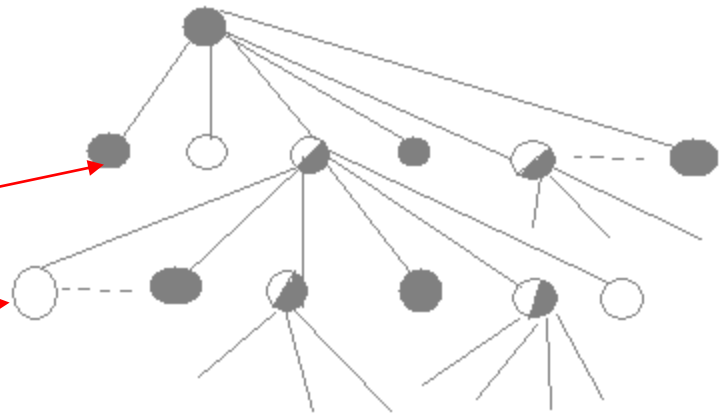
Scena divizata in volume incadratoare (3)

Divizarea adaptiva a scenei (scena reprezentata prin arbore octal)

- Se porneste de la paralelipipedidul incadrator al scenei, care se divide in 8 subvolume egale, in mod recursiv, divizarea unui subvolum terminandu-se daca el nu contine nici un obiect al scenei sau contine un singur obiect.
- Scena se reprezinta printr-un arbore octal, fiecare nod fiind asociat unui subvolum.
- Pentru intersectia razei cu scena se fac teste de intersectie pornind din nodul de start (radacina pentru raza primara) spre frunze.
- Daca subvolumul unui nod nu este intersectat de raza, atunci niciunul dintre subvolumele (obiectele) din subarborele nodului nu va fi intersectat.

Arborele octal al scenei

- subvolumul contine un singur obiect
- subvolum gol



Platforma NVIDIA RTX permite implementarea algoritmului Ray-tracing folosind API-uri si kituri software de dezvoltare dedicate: Nvidia OptiX, Microsoft DirectX si Vulkan.