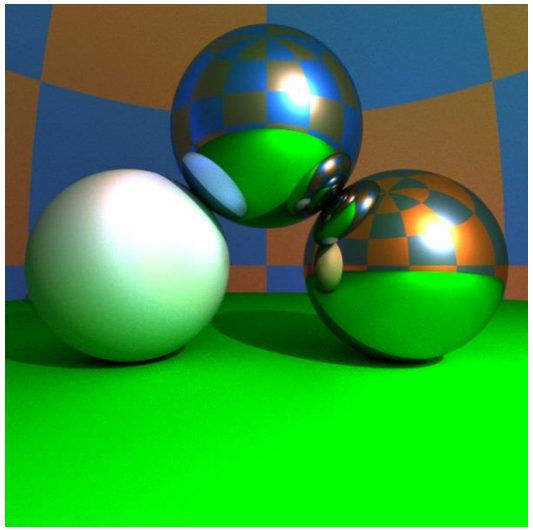
"As technology advances, the rendering time remains constant."

Jim Blinn

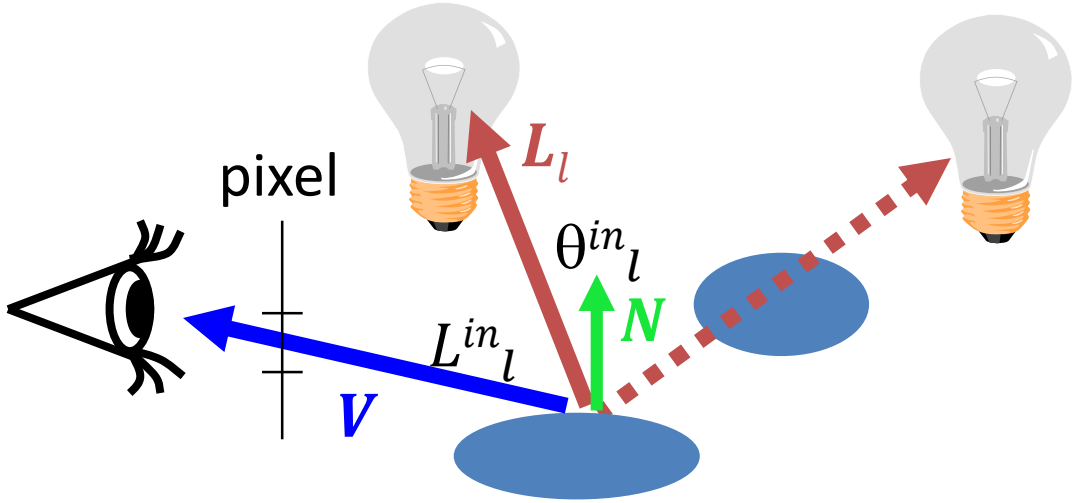
Sugárkövetés: ray-casting, ray-tracing, path-tracing

Szirmay-Kalos László



Lokális illumináció: rücskös felületek, absztrakt fényforrások

Csak absztrakt fényforrások direkt megvilágítása

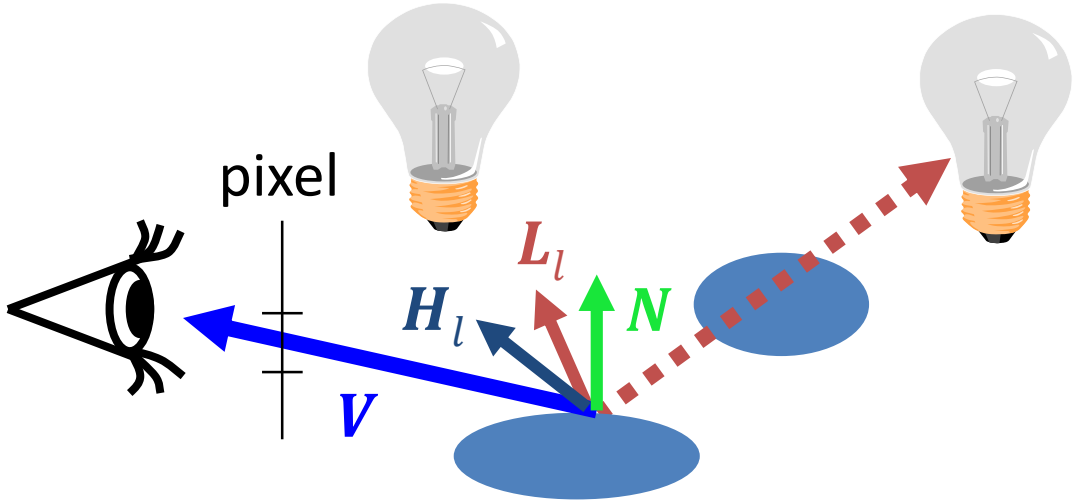


$$L(\boldsymbol{V}) \approx \sum_l L^{in}{}_l * f_r(\boldsymbol{L}_l, \boldsymbol{N}, \boldsymbol{V}) \cdot \cos^{+}\theta^{in}{}_l$$

Absztrakt fényforrásokból származó megvilágítás.
(*Irányforrás = konstans; Pontforrás = távolság négyzetével csökken*
Ha takart, akkor zérus)

Lokális illumináció: rücskös felületek, absztrakt fényforrások

Csak absztrakt fényforrások direkt megvilágítása

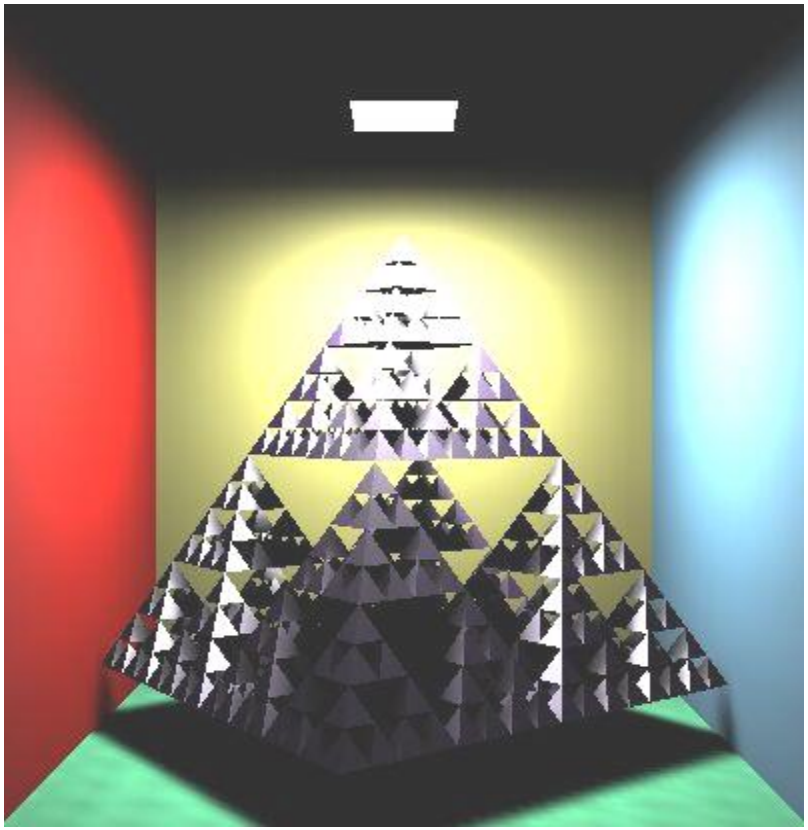


$$L(\boldsymbol{V}) \approx \sum_l L^{in}{}_l * \{k_d \cdot (\boldsymbol{L}_l \bullet \boldsymbol{N})^+ + k_s \cdot ((\boldsymbol{H}_l \bullet \boldsymbol{N})^+)^{shine}\}$$

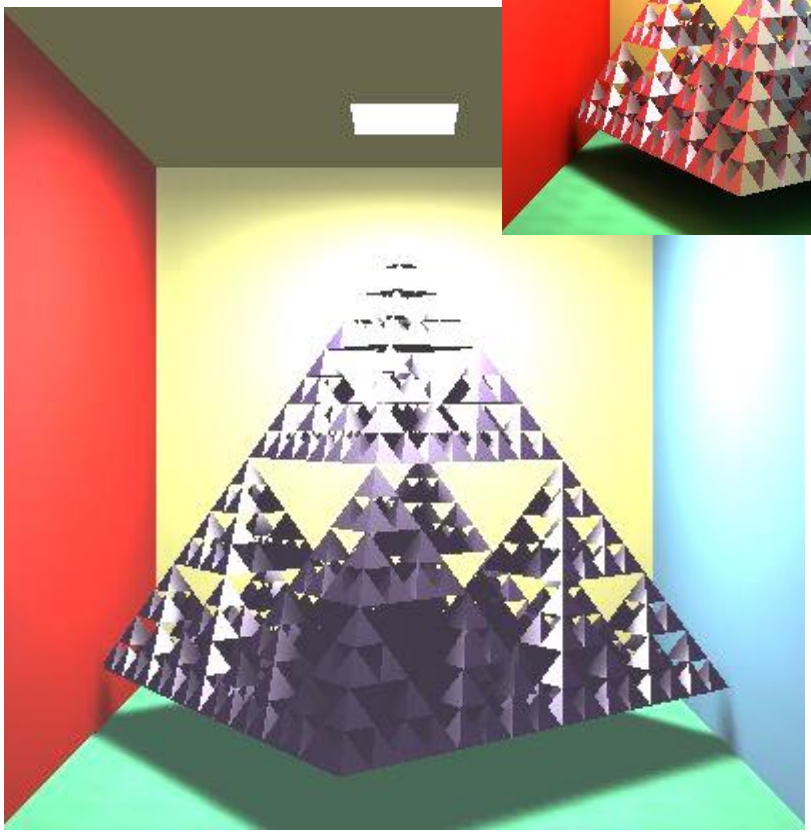
Absztrakt fényforrásokból származó megvilágítás.
(*Irányforrás = konstans; Pontforrás = távolság négyzetével csökken*
Ha takart, akkor zérus)

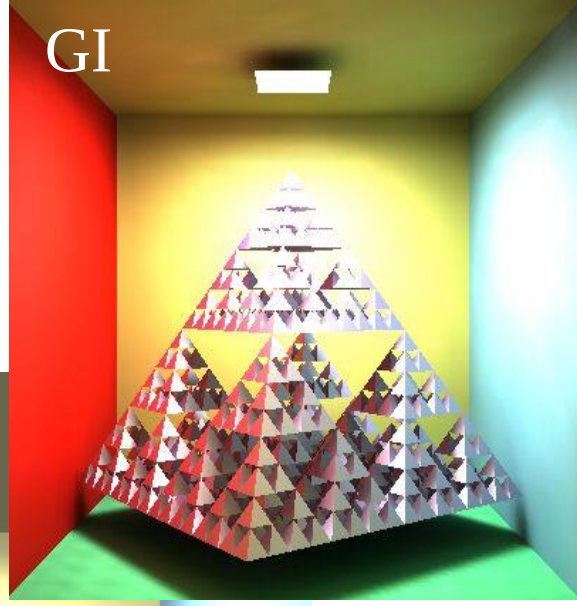
Ambiens tag

Lokális illumináció



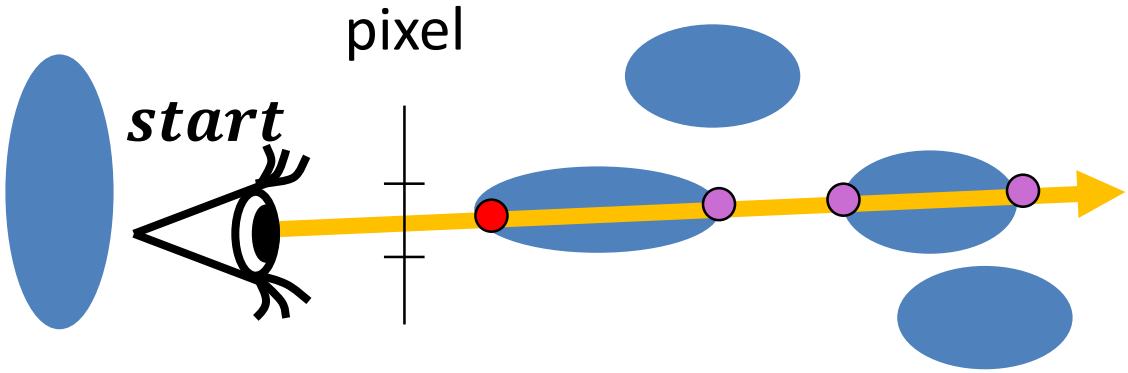
+ ambiens tag





$$L(\boldsymbol{V}) \approx \sum_l {L^{in}}_l * f_r(\boldsymbol{L}_l, \boldsymbol{N}, \boldsymbol{V}) \cdot \cos^+ {\theta^{in}}_l + \boxed{k_a * L_a}$$

Láthatóság



$$ray(t) = start + dir \cdot t, \quad t > 0$$

```
struct Ray {
   vec3 start;
   vec3 dir; // egységvektor
   bool out; // kívül? vagy ior
};

struct Hit {
   float t;
   vec3 position;
   vec3 normal;
   Material* material;
   Hit() { t = -1; }
};
```

```
Hit firstIntersect(Ray ray) {
   Hit bestHit;
   for(Intersectable * obj : objects) {
      Hit hit = obj->intersect(ray); // hit.t < 0 ha nincs metszés
      if(hit.t > 0 && (bestHit.t < 0 || hit.t < bestHit.t))
         bestHit = hit;
   }
   if (dot(ray.dir, bestHit.normal) > 0) bestHit.normal *= -1;
   return bestHit;
}
```

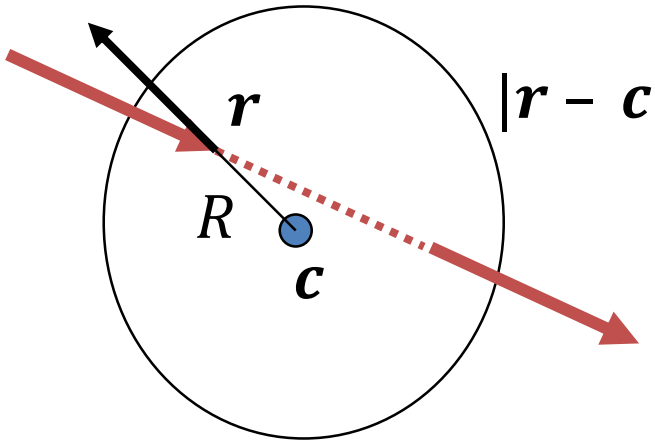
Nézzen felénk!

if $(N \bullet dir > 0)\ \{\ N = -N;\ \}$

Objektum = Intersectable

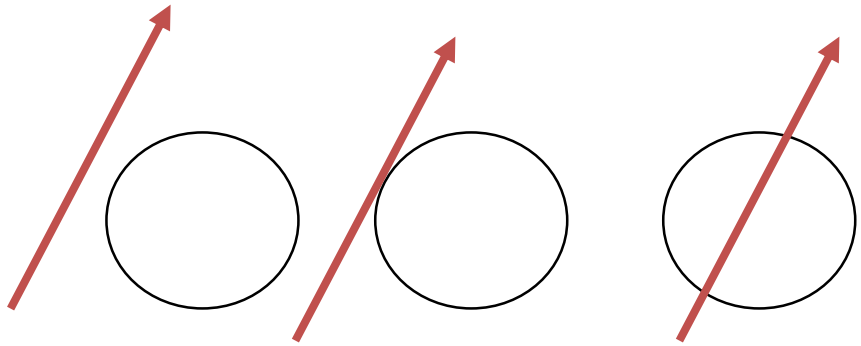
```
struct Intersectable {
  Material* material;
  virtual Hit intersect(const Ray& ray) = 0;
};
```

Metszéspont számítás gömbbel



$$|\boldsymbol{r} - \boldsymbol{c}|^2 = R^2$$

$$\boldsymbol{ray}(t) = \boldsymbol{start} + \boldsymbol{dir} \cdot t$$



Nincs gyök 1 gyök 2 gyök

$$|\boldsymbol{ray}(t) - \boldsymbol{c}|^2 = (\boldsymbol{start} + \boldsymbol{dir} \cdot t - \boldsymbol{c}) \cdot (\boldsymbol{start} + \boldsymbol{dir} \cdot t - \boldsymbol{c}) = R^2$$

$$(\boldsymbol{dir} \cdot \boldsymbol{dir})t^2 + 2((\boldsymbol{start} - \boldsymbol{c}) \cdot \boldsymbol{dir})t + (\boldsymbol{start} - \boldsymbol{c}) \cdot (\boldsymbol{start} - \boldsymbol{c}) - R^2 = 0$$

Wanted: a pozitív megoldások közül a kisebb

Felületi normális: $\boldsymbol{N} = (\boldsymbol{ray}(t) - \boldsymbol{c})/R$

Sphere mint Intersectable

```
class Sphere : public Intersectable {
    vec3 center;
    float radius;
public:
    Hit intersect(const Ray& ray) {
        Hit hit;
        vec3 dist = ray.start - center;
        float a = dot(ray.dir, ray.dir);
        float b = dot(dist, ray.dir) * 2;
        float c = dot(dist, dist) - radius * radius;
        float discr = b * b - 4 * a * c;
        if (discr < 0) return hit; else discr = sqrtf(discr);
        float t1 = (-b + discr)/2/a, t2 = (-b - discr)/2/a;
        if (t1 <= 0) return hit; // t1 >= t2 for sure
        hit.t = (t2 > 0) ? t2 : t1;
        hit.position = ray.start + ray.dir * hit.t;
        hit.normal = (hit.position - center)/radius;
        hit.material = material;
        return hit;
    }
};
```

Implicit felületek

- A felület pontjai: $f(\boldsymbol{r}) = 0$
- Sugár: $\boldsymbol{ray}(t) = \boldsymbol{s} + \boldsymbol{d} \cdot t$
- Metszés paraméter t^*: $f(\boldsymbol{s} + \boldsymbol{d} \cdot t^*) = 0$
- Metszéspont: $\boldsymbol{r}^* = \boldsymbol{ray}(t^*) = \boldsymbol{s} + \boldsymbol{d} \cdot t^*$
- Normálvektor $= \nabla f(\boldsymbol{r}^*)$

- **Kvadratikus felületek** példa: $f(\boldsymbol{r}) = [\boldsymbol{r}, 1] \cdot \boldsymbol{Q} \cdot [\boldsymbol{r}, 1]^T = 0$
- Metszés paraméter:

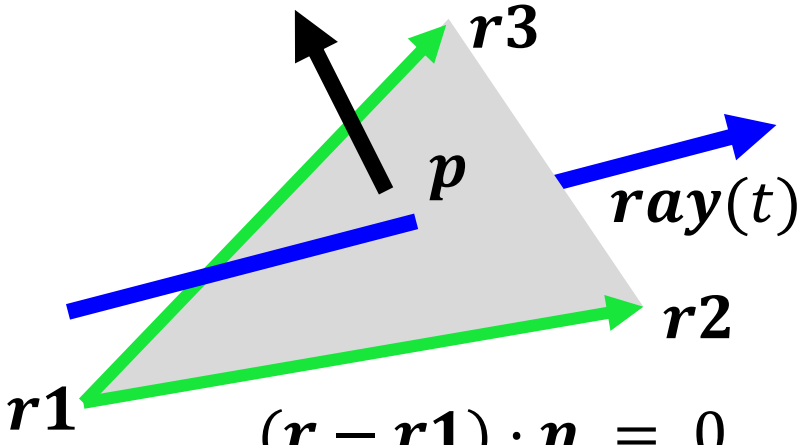
$$[\boldsymbol{s} + \boldsymbol{d} \cdot t^*, 1] \cdot \boldsymbol{Q} \cdot [\boldsymbol{s} + \boldsymbol{d} \cdot t^*, 1]^T = 0$$

$$[\boldsymbol{d}, 0] \cdot \boldsymbol{Q} \cdot [\boldsymbol{d}, 0]^T (t^*)^2 + 2[\boldsymbol{s}, 1] \cdot \boldsymbol{Q} \cdot [\boldsymbol{d}, 0]^T t^* + [\boldsymbol{s}, 1] \cdot \boldsymbol{Q} \cdot [\boldsymbol{s}, 1]^T = 0$$

- Normálvektor: $\nabla f(\boldsymbol{r}^*) = \boldsymbol{Q} \cdot [\boldsymbol{r}^*, 1]^T$ első három koord.

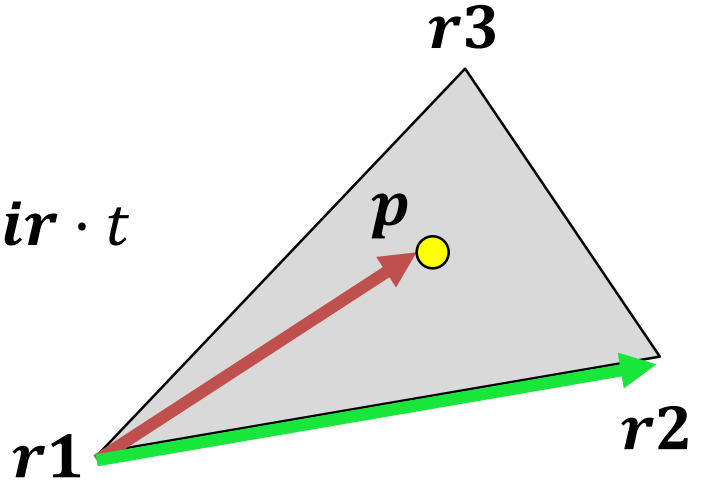
Háromszög

$$\boldsymbol{n} = (\boldsymbol{r2} - \boldsymbol{r1}) \times (\boldsymbol{r3} - \boldsymbol{r1})$$



$$\boldsymbol{ray}(t) = \boldsymbol{start} + \boldsymbol{dir} \cdot t$$

$$(\boldsymbol{r} - \boldsymbol{r1}) \cdot \boldsymbol{n} = 0$$



1. Síkmetszés: $(\boldsymbol{ray}(t) - \boldsymbol{r1}) \cdot \boldsymbol{n} = 0,$
$t > 0$

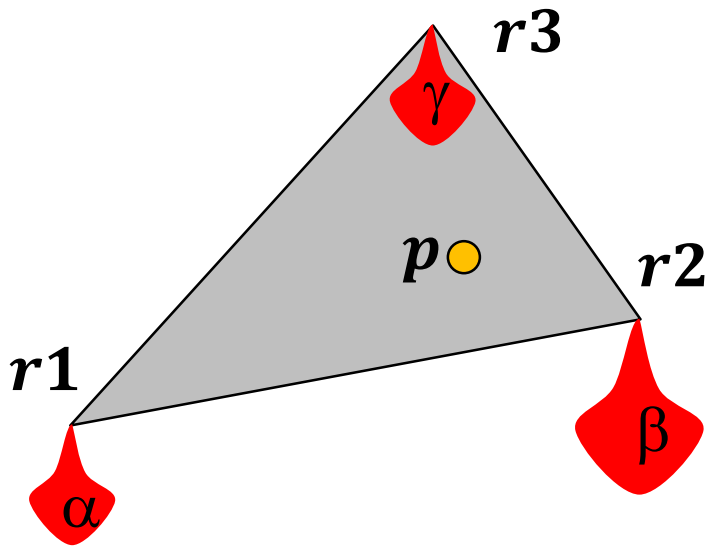
$$t = \frac{(\boldsymbol{r1} - \boldsymbol{start}) \cdot \boldsymbol{n}}{\boldsymbol{dir} \cdot \boldsymbol{n}}$$

2. A metszéspont a háromszögön belül van-e?

$$((\boldsymbol{r2} - \boldsymbol{r1}) \times (\boldsymbol{p} - \boldsymbol{r1})) \cdot \boldsymbol{n} > 0$$
$$((\boldsymbol{r3} - \boldsymbol{r2}) \times (\boldsymbol{p} - \boldsymbol{r2})) \cdot \boldsymbol{n} > 0$$
$$((\boldsymbol{r1} - \boldsymbol{r3}) \times (\boldsymbol{p} - \boldsymbol{r3})) \cdot \boldsymbol{n} > 0$$

Felületi normális: $\boldsymbol{n}$
vagy árnyaló normálok
(shading normals)

Háromszög metszés baricentrikus koordinátákban



$$\boldsymbol{p}(\alpha,\beta,\gamma) = \alpha \cdot \boldsymbol{r1} + \beta \cdot \boldsymbol{r2} + \gamma \cdot \boldsymbol{r3}$$

$\alpha + \beta + \gamma = 1$ 1 skalár egyenlet

$\alpha, \beta, \gamma \geq 0$

$$\boldsymbol{ray}(t) = \boldsymbol{p}(\alpha,\beta,\gamma)$$

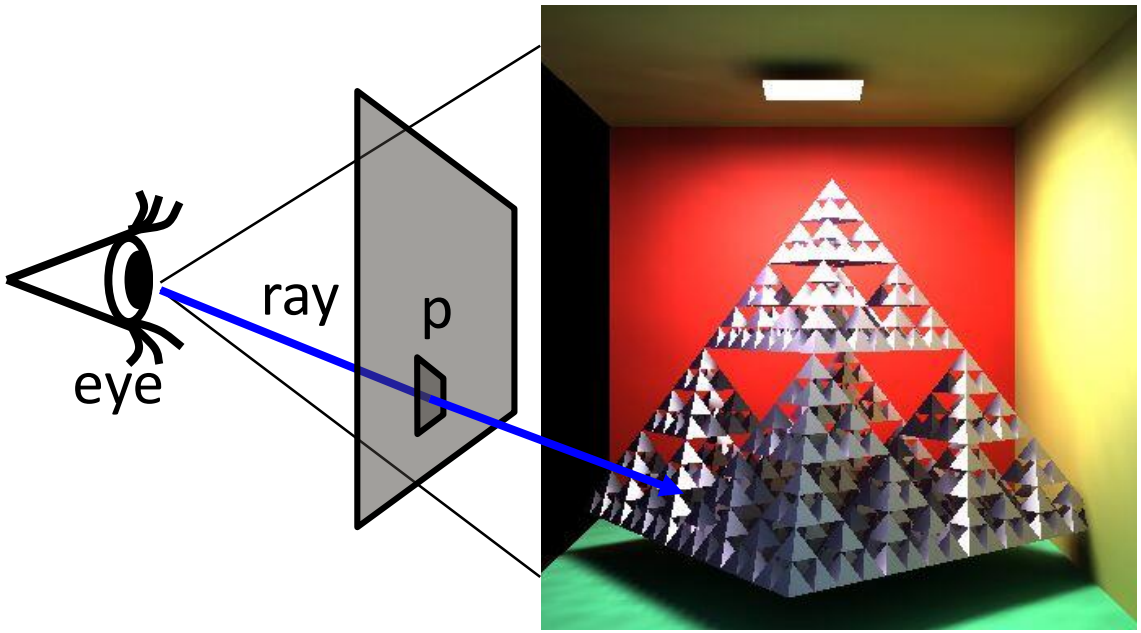
$$\boldsymbol{start} + \boldsymbol{dir} \cdot t = \alpha \cdot \boldsymbol{r1} + \beta \cdot \boldsymbol{r2} + \gamma \cdot \boldsymbol{r3}$$ 3 skalár egyenlet

4 ismeretlen: α, β, γ, t

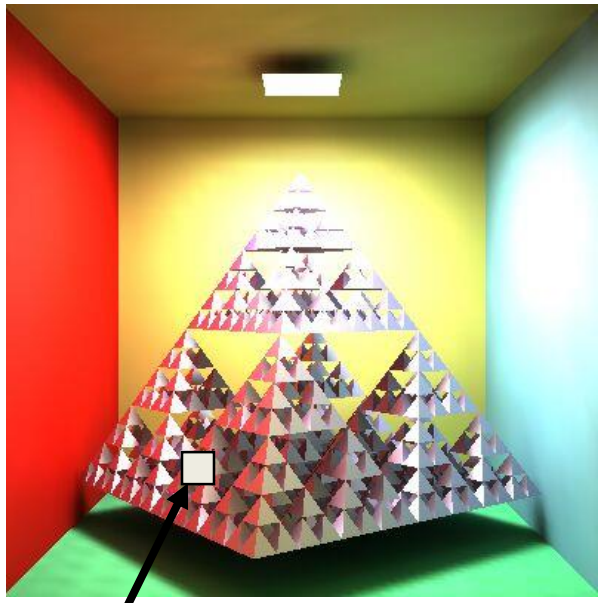
A megoldás után ellenőrzés, hogy mind nem negatív-e

Sugárkövetés: Render

Virtuális világ: szem+ablak



Valós világ: néző+képernyő

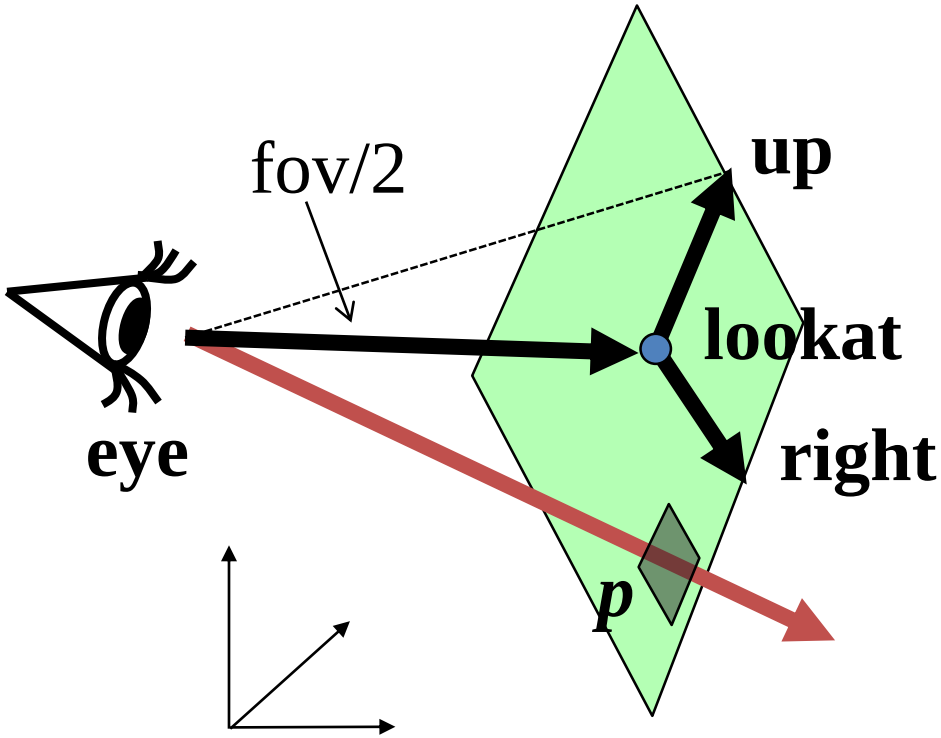


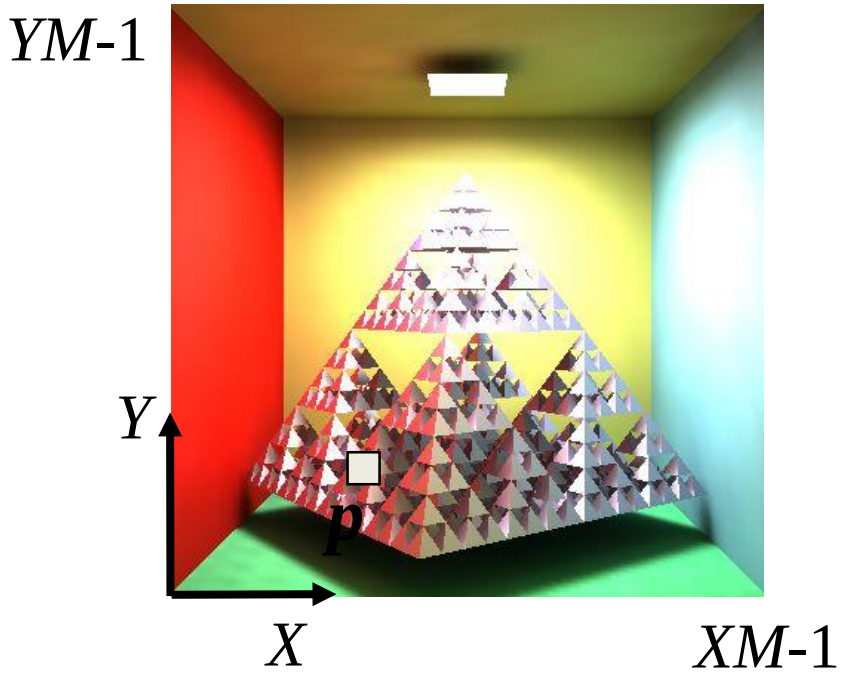
p color

Render( )

```
    for each pixel p
        Ray r = getRay( eye ➔ pixel p )
        color = trace(ray)
        WritePixel(p, color)
    endfor
end
```

Kamera: getRay



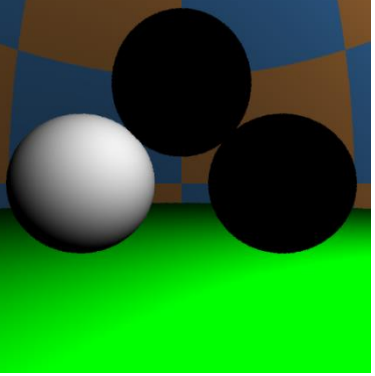


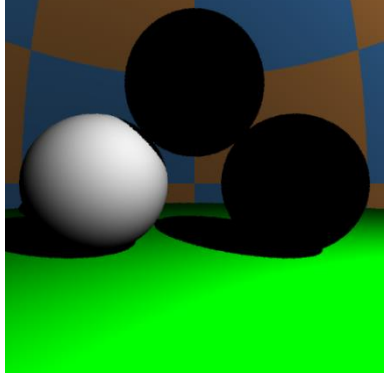
Normalizált eszköz koordináták

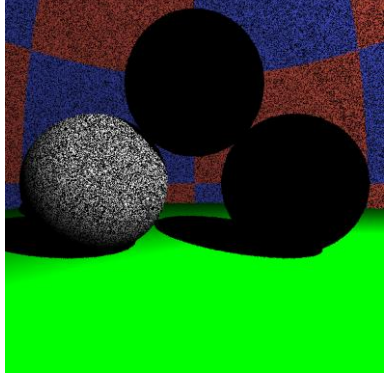
$\boldsymbol{p}$ = **lookat** + $\alpha \cdot$**right** + $\beta \cdot$**up**, α, β in [-1,1]

= **lookat** + $(2(X+0.5)/XM-1) \cdot$**right** + $(2(Y+0.5)/YM-1) \cdot$**up**

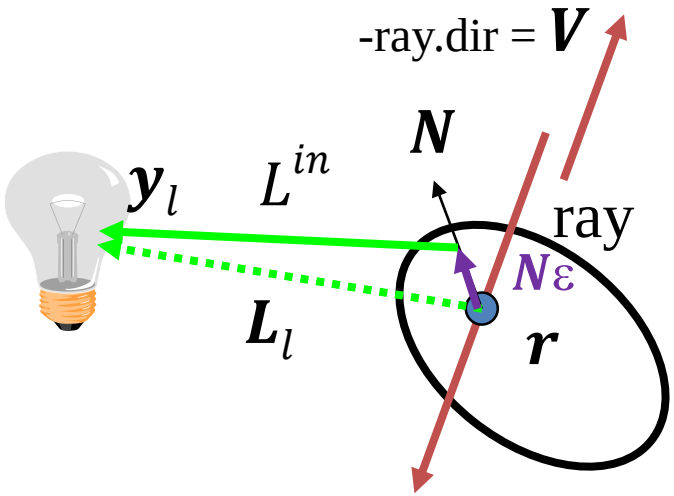
Ray: start = eye, dir =p – eye







No Shadow Shadow ε = 0: acne

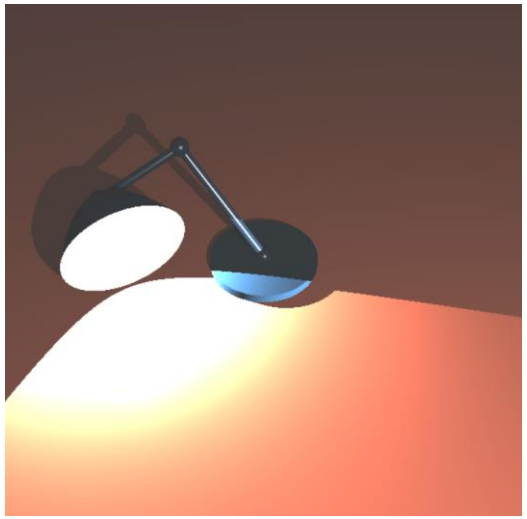



```
vec3 trace(Ray ray) {
    Hit hit = firstIntersect(ray);
    if(hit.t < 0) return La; // nothing
    [r, N, ka, kd, ks, shine] ← hit;
    vec3 outRad = ka * La;
    for(each light source l) {
        Ray shadowRay(r + Nε, Ll);
        Hit shadowHit = firstIntersect(shadowRay);
        if(shadowHit.t < 0 || shadowHit.t > |r − yl|)
            outRad += L^in_l * {kd·(Ll•N)+ + ks·((Hl•N)+)^shine}
    }
    return outRad;
}
```

DirectLight()

Shadow

2. házi: Luxo Grandpa



Luxo Junior-ból *Luxo Grandpa (LG)* lett,

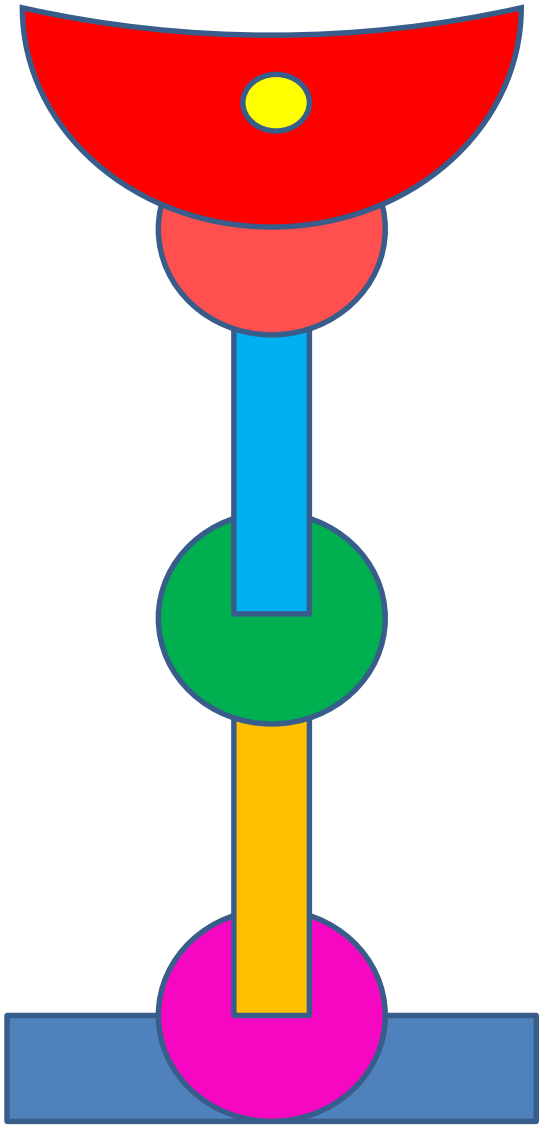
ezért új programot szentelünk neki.

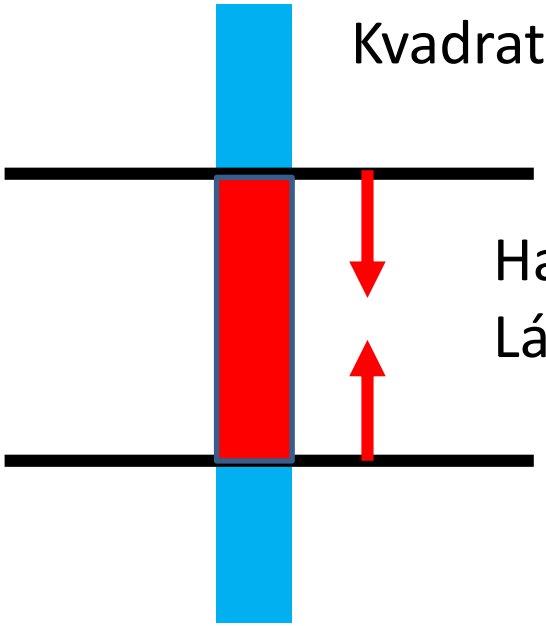
LG egy síkon áll, a szerkezete (alulról felfelé): henger talp, gömbcsukló1, henger rúd1, gömbcsukló2, henger rúd2, gömbcsukló3, paraboloid, amelynek fókuszpontjában egy pontfényforrás ül.

A gömbcsuklók a koordinátatengelyektől eltérő tengelyek mentén folyamatosan elfordulnak. A szereplőket még egy pontfényforrás és ambiens fény világítja meg. A kamera forog LG körül.

A szereplők diffúz-spekuláris típusú rücskös anyagúak.

Luxo Grandpa





Végtelen
Kvadratikus felület

Határoló síkok
Látható?

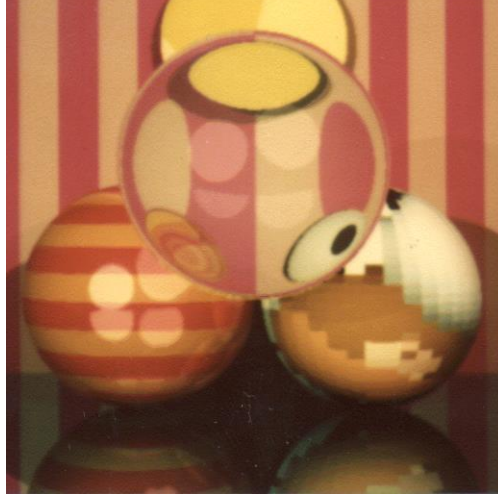
Feladatok:

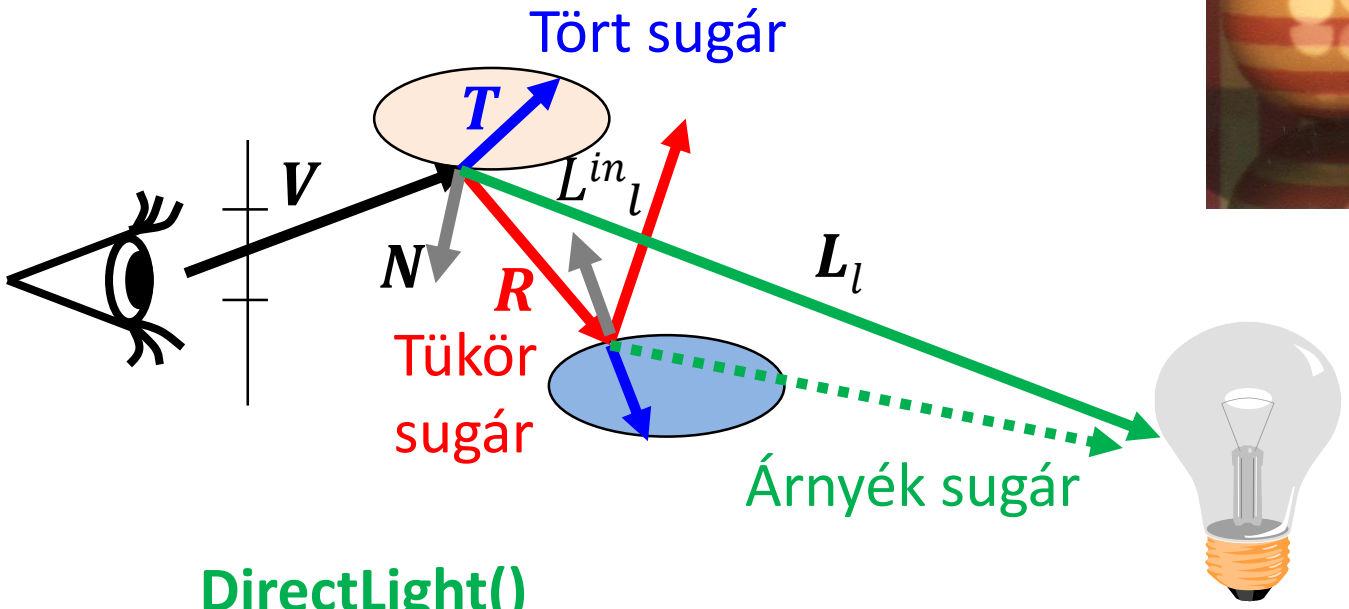
Kvadratikus felület és síkok

- Hierarchikus transzformálása
- Metszése
- Normálvektora

Fókuszpont hierarchikus transzformálása

Rekurzív sugárkövetés





DirectLight()

$$L(\boldsymbol{V}) \approx \left\{ k_a * L_a + \sum_l L^{in}{}_l \; * \{k_d \cdot (\boldsymbol{L}_l \bullet \boldsymbol{N})^+ + k_s \cdot ((\boldsymbol{H}_l \bullet \boldsymbol{N})^+)^{shine}\} \right.$$
$$\left. F(\boldsymbol{V} \bullet \boldsymbol{N}) * L^{in}(\boldsymbol{R}) + (1 - F(\boldsymbol{V} \bullet \boldsymbol{N})) * L^{in}(\boldsymbol{T}) \right\}$$

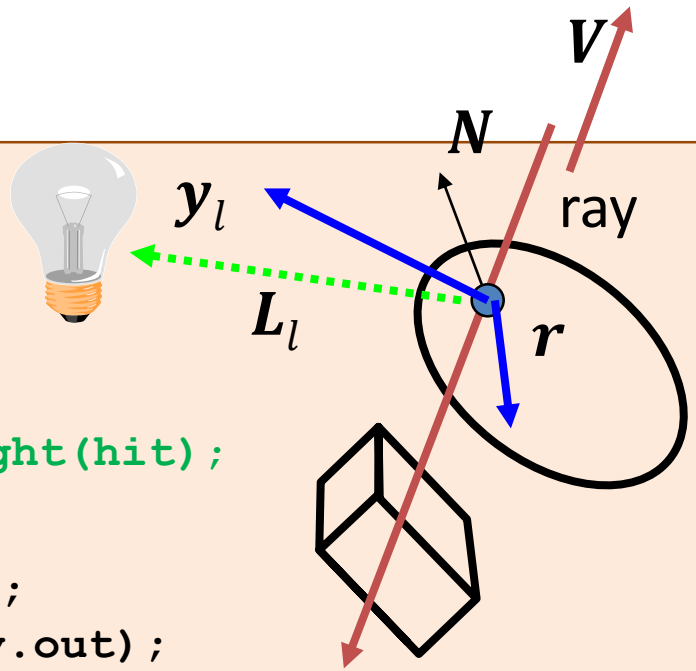
Fresnel

Tükör irányból érkező fény

1-Fresnel

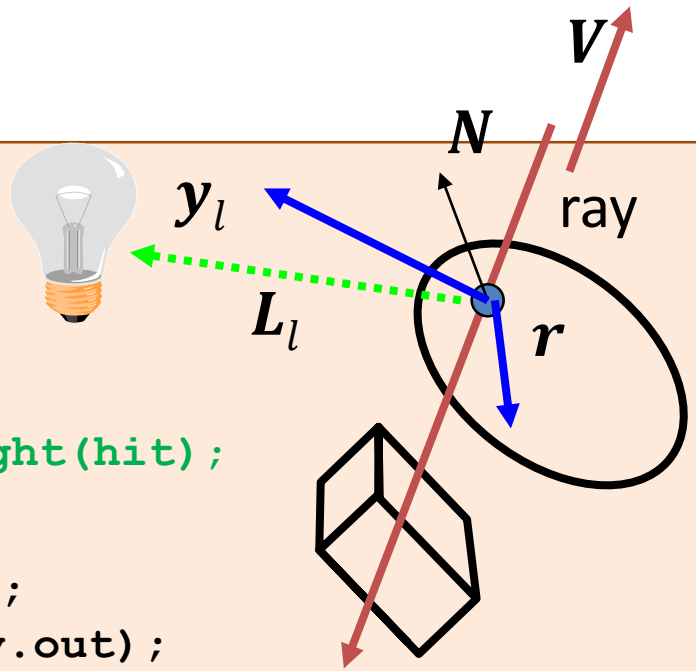
Törési irányból érkező fény

trace



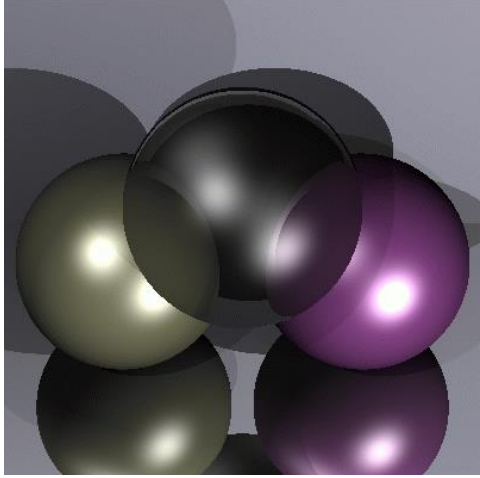

```
vec3 trace(Ray ray) {

  Hit hit = firstIntersect(ray);
  if(hit.t < 0) return La; // nothing
  vec3 outRad(0, 0, 0);
  if(hit.material->rough) outRad = DirectLight(hit);

  if(hit.material->reflective){
    vec3 reflectionDir = reflect(ray.dir,N);
    Ray reflectRay(r + Nε, reflectionDir, ray.out);
    outRad += trace(reflectRay)*Fresnel(ray.dir,N);
  }
  if(hit.material->refractive) {
    ior = (ray.out) ? n.x : 1/n.x;
    vec3 refractionDir = refract(ray.dir,N,ior);
    if (length(refractionDir) > 0) {
      Ray refractRay(r - Nε, refractionDir, !ray.out);
      outRad += trace(refractRay)*(vec3(1,1,1)-Fresnel(ray.dir,N));
    }
  }
  return outRad;
}
```

trace



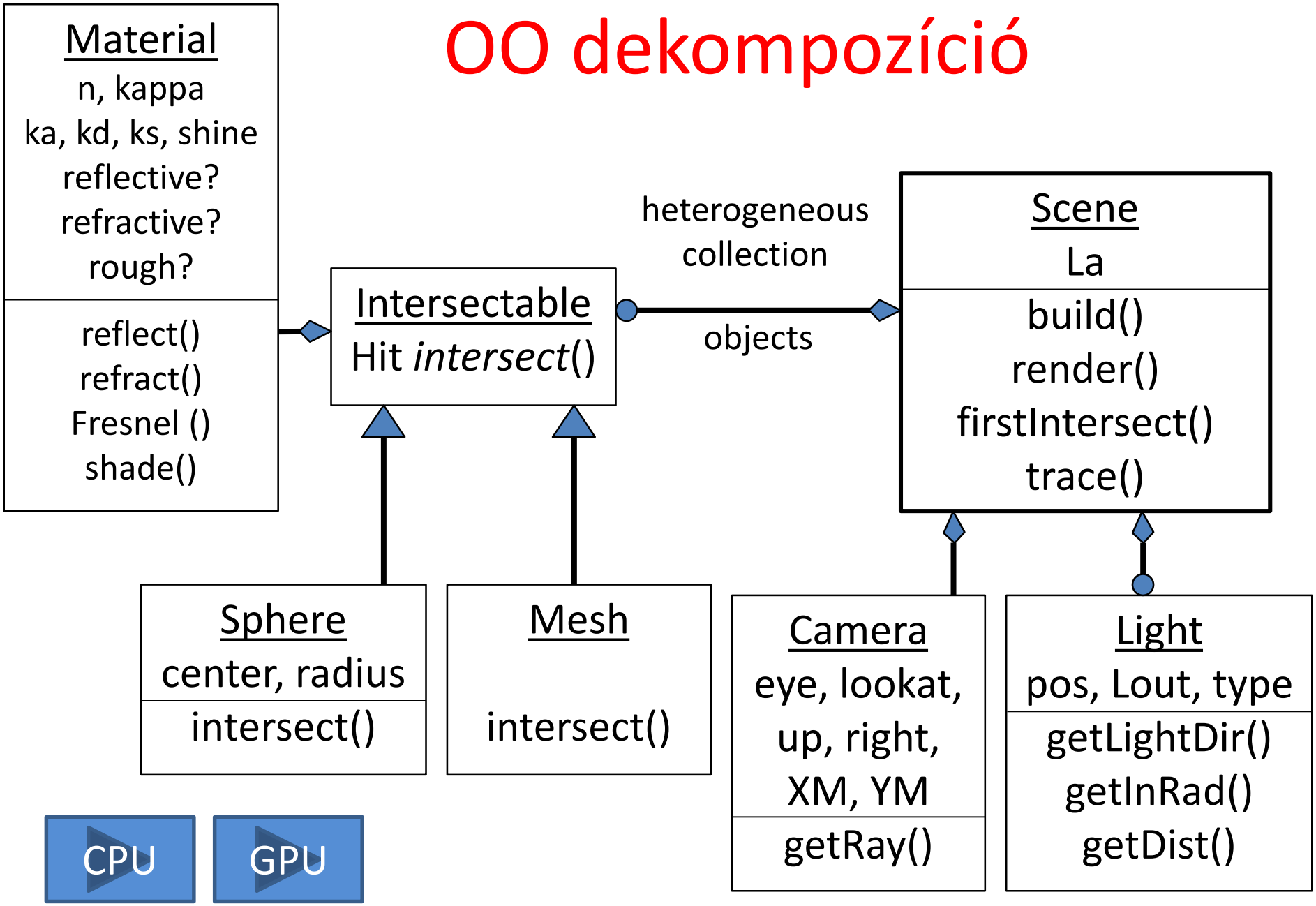

```
vec3 trace(Ray ray, int d=0) {
  if (d > maxdepth) return La;
  Hit hit = firstIntersect(ray);
  if(hit.t < 0) return La; // nothing
  vec3 outRad(0, 0, 0);
  if(hit.material->rough) outRad = DirectLight(hit);

  if(hit.material->reflective){
    vec3 reflectionDir = reflect(ray.dir,N);
    Ray reflectRay(r + Nε, reflectionDir, ray.out);
    outRad += trace(reflectRay,d+1)*Fresnel(ray.dir,N);
  }
  if(hit.material->refractive) {
    ior = (ray.out) ? n.x : 1/n.x;
    vec3 refractionDir = refract(ray.dir,N,ior);
    if (length(refractionDir) > 0) {
      Ray refractRay(r - Nε, refractionDir, !ray.out);
      outRad += trace(refractRay,d+1)*(vec3(1,1,1)-Fresnel(ray.dir,N))
    }
  }
  return outRad;
}
```

Paul Heckbert névjegye



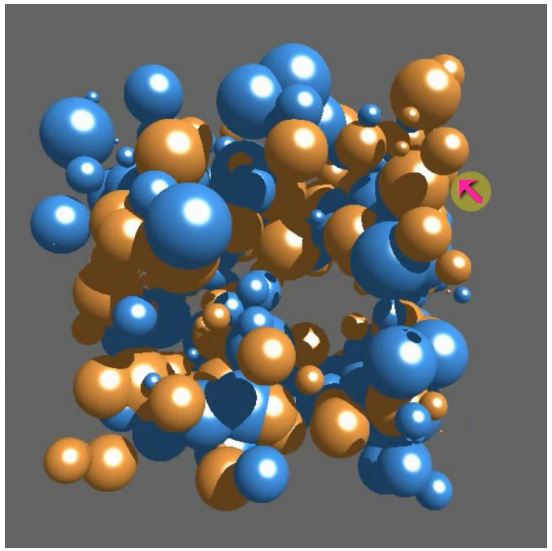
```
typedef struct{double x,y,z}vec;vec U,black,amb={.02,.02,.02}; struct sphere{ vec cen,color;double rad,kd,ks,kt,kl,ir}*s,
*best,sph[]={0.,6.,.5,1.,1.,1.,.9, .05,.2,.85,0.,1.7,-1.,8.,-.5,1.,.5,.2,1.,.7,.3,0.,.05,1.2,1.,8.,-.5,.1,.8,.8, 1.,.3,.7,0.,0.,1.2,3.,-6.,15.,1.,
.8,1.,7.,0.,0.,0.,.6,1.5,-3.,-3.,12.,.8,1., 1.,5.,0.,0.,0.,.5,1.5,};yx; double u,b,tmin,sqrt(),tan();double vdot(A,B)vec A ,B;
{return A.x*B.x+A.y*B.y+A.z*B.z;}vec vcomb(a,A,B)double a;vec A,B; {B.x+=a* A.x;B.y+=a*A.y;B.z+=a*A.z;return B;}
vec vunit(A)vec A;{return vcomb(1./sqrt( vdot(A,A)),A,black);}struct sphere *intersect(P,D)vec P,D;{best=0;tmin=1e30;
s= sph+5;while(s-->sph)b=vdot(D,U=vcomb(-1.,P,s->cen)),u=b*b-vdot(U,U)+s->rad*s ->rad,u=u>0?sqrt(u):1e31,u=b-u>
1e-7?b-u:b+u,tmin=u>=1e-7&&u<tmin?best=s,u: tmin;return best;}vec trace(level,P,D)vec P,D;{double d,eta,e;vec N,color;
struct sphere*s,*l;if(!level--)return black;if(s=intersect(P,D));else return amb;color=amb;eta=s->ir;d= -vdot(D,N=vunit(vcomb
(-1.,P=vcomb(tmin,D,P),s->cen )));if(d<0)N=vcomb(-1.,N,black),eta=1/eta,d= -d;l=sph+5;while(l-->sph)if((e=l ->kl*vdot(N,
U=vunit(vcomb(-1.,P,l->cen))))>0&&intersect(P,U)==l)color=vcomb(e ,l->color,color);U=s->color;color.x*=U.x;color.y*=
U.y;color.z*=U.z;e=1-eta* eta*(1-d*d);return vcomb(s->kt,e>0?trace(level,P,vcomb(eta,D,vcomb(eta*d-sqrt (e),N,black))):
black,vcomb(s->ks,trace(level,P,vcomb(2*d,N,D)),vcomb(s->kd, color,vcomb(s->kl,U,black))));}

main(){printf("%d %d\n",32,32);while(yx<32*32) U.x=yx%32-32/2,U.z=32/2-yx++/32,U.y=32/2/tan(25/114.5915590261),
U=vcomb(255., trace(3,black,vunit(U)),black),printf("%.0f %.0f %.0f\n",U);}/*minray!*/
```

OO dekompozíció



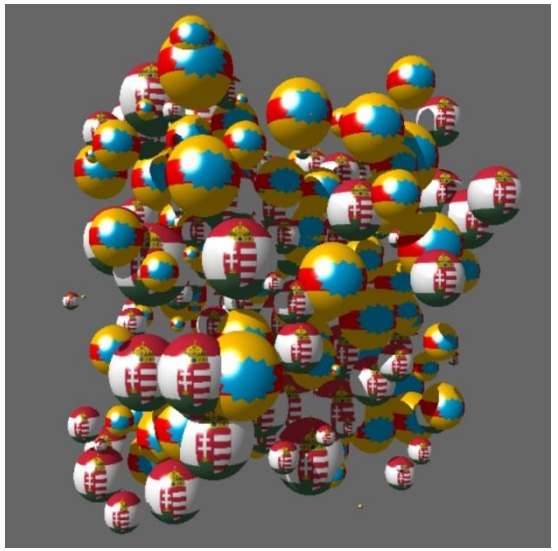
Sugárkövetés
1. Program: Ray casting

Szirmay-Kalos László



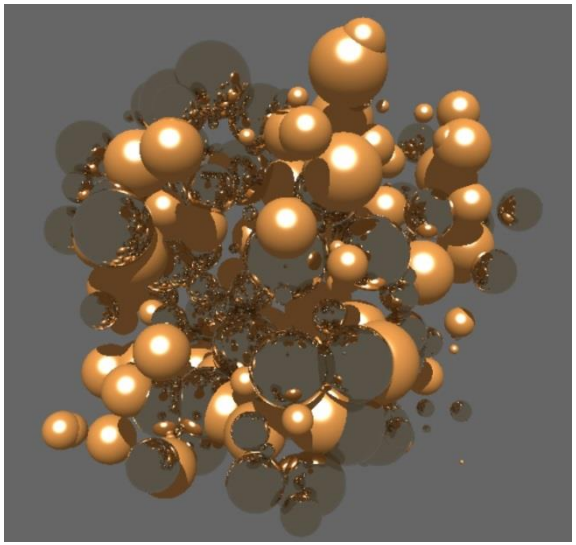
Sugárkövetés
2. Program: Ray casting textúrázással

Szirmay-Kalos László



Sugárkövetés
3. Program: Rekurzív sugárkövetés

Szirmay-Kalos László

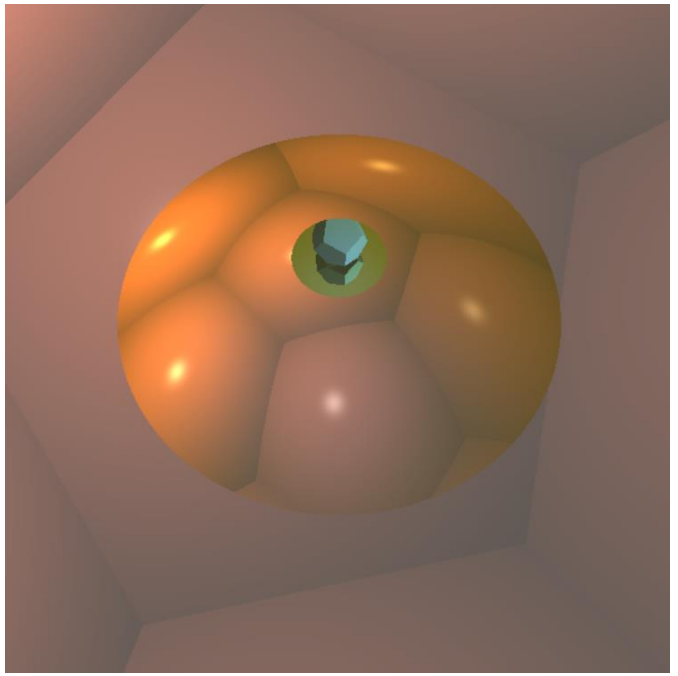


Sugárkövetés
4. Program: Napfénycső szimulátor

Szirmay-Kalos László

GPU sugárkövetés
5. Program: Mirascope

Szirmay-Kalos László



Specifikáció

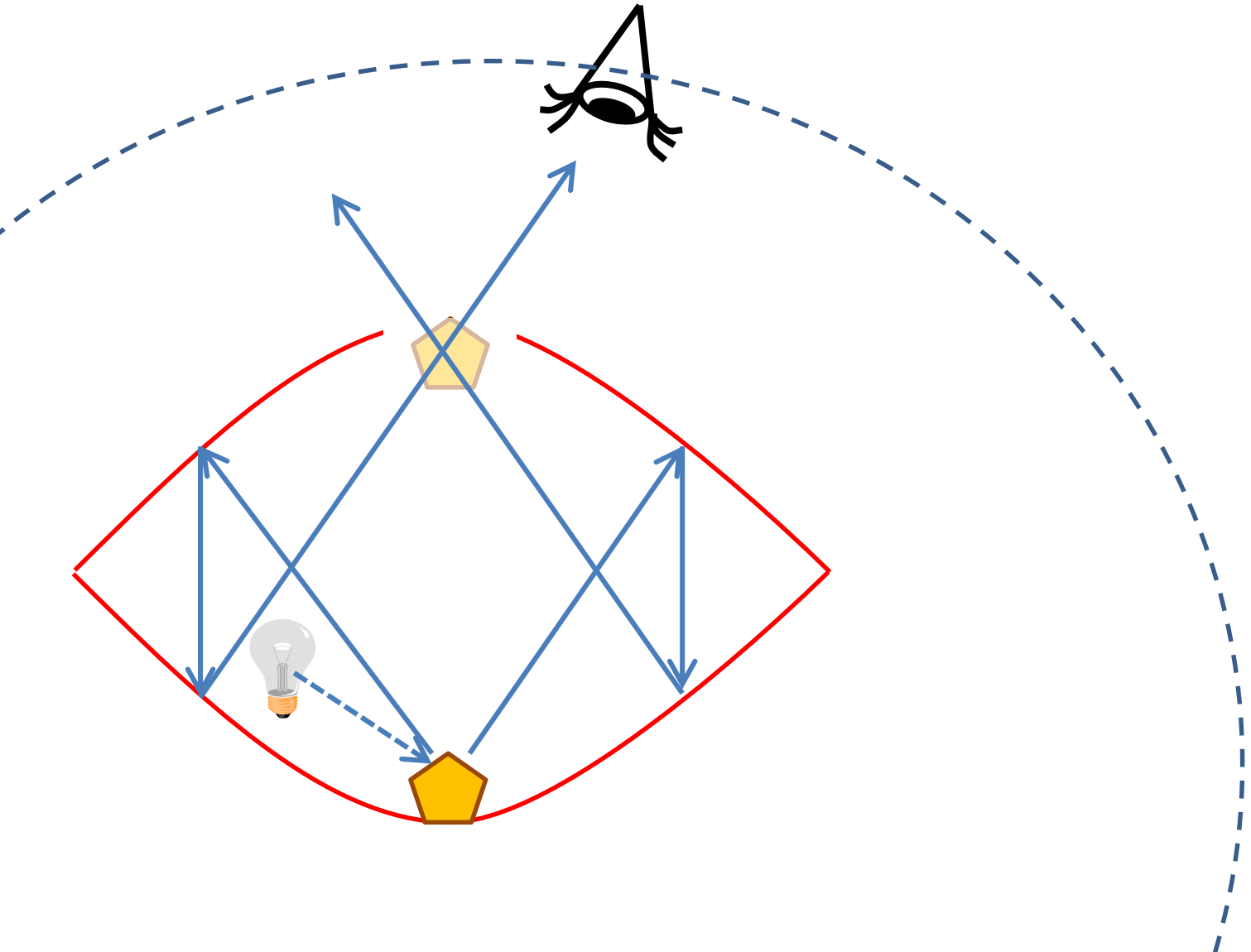
Készítsen **mirascope szimulátort**. A mirascope két ugyanolyan, egymásra tett és szembe fordított paraboloid tükör, ahol az egyik fókuszpontja a másik aljára esik és a felsőn lyuk van. A mi mirascope-unk aranyból van és tükör lévén optikailag sima. A mirascope az alsó aljára tett tárgyat a felső lyukban megjeleníti. A szimulátorban a tárgy diffúz-spekuláris anyagú dodekaéder, amit a mirascope belsejében lévő pont fényforrás és az egész térben jelen lévő ambiens forrás világít meg. A néző és a mirascope egy Platon-i szabályos test geometriájú szoba belsejében szemlélődik.

A virtuális kamerán kívülről, felülről és oldalról néz rá a mirascope-ra. Az ‘f’ lenyomásával a szempozíció feljebb kerül, az ‘F’ hatására lejjebb úgy, hogy a távolsága a mirascope aljától állandó legyen.

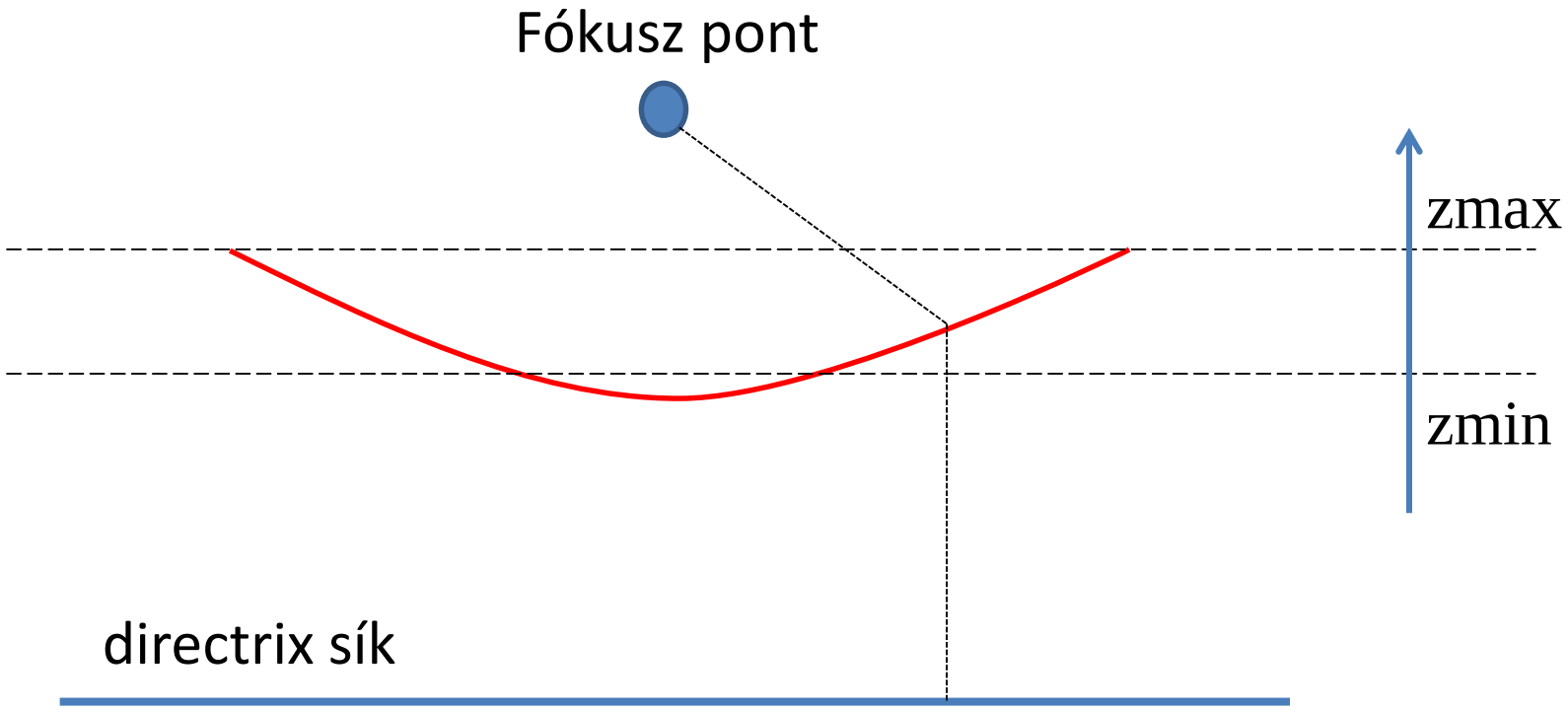
Az arany törésmutatója és kioltási tényezője az r,g,b hullámhosszokon:

n/k: 0.17/3.1, 0.35/2.7, 1.5/1.9

Mirascope



Paraboloid

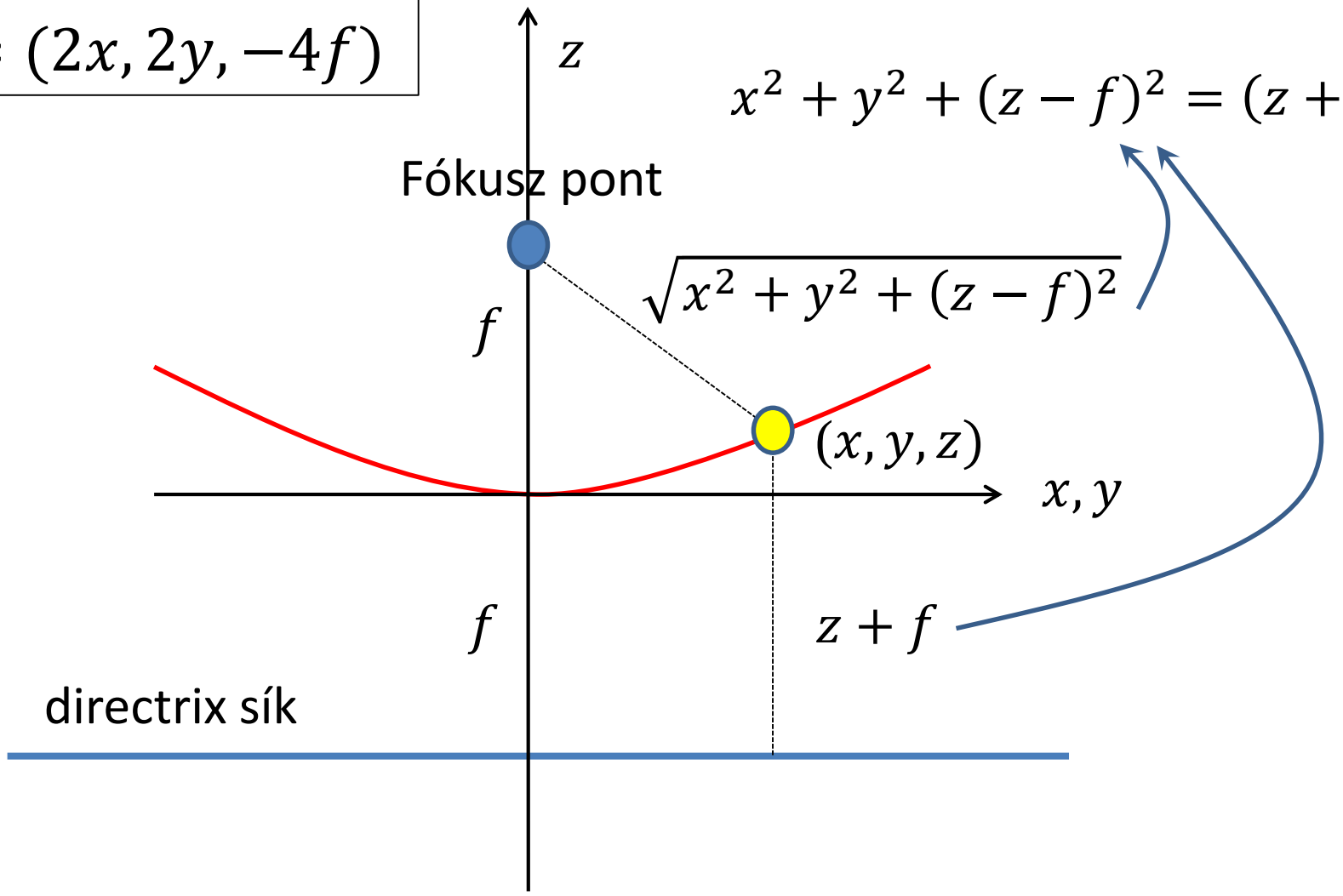


Paraboloid

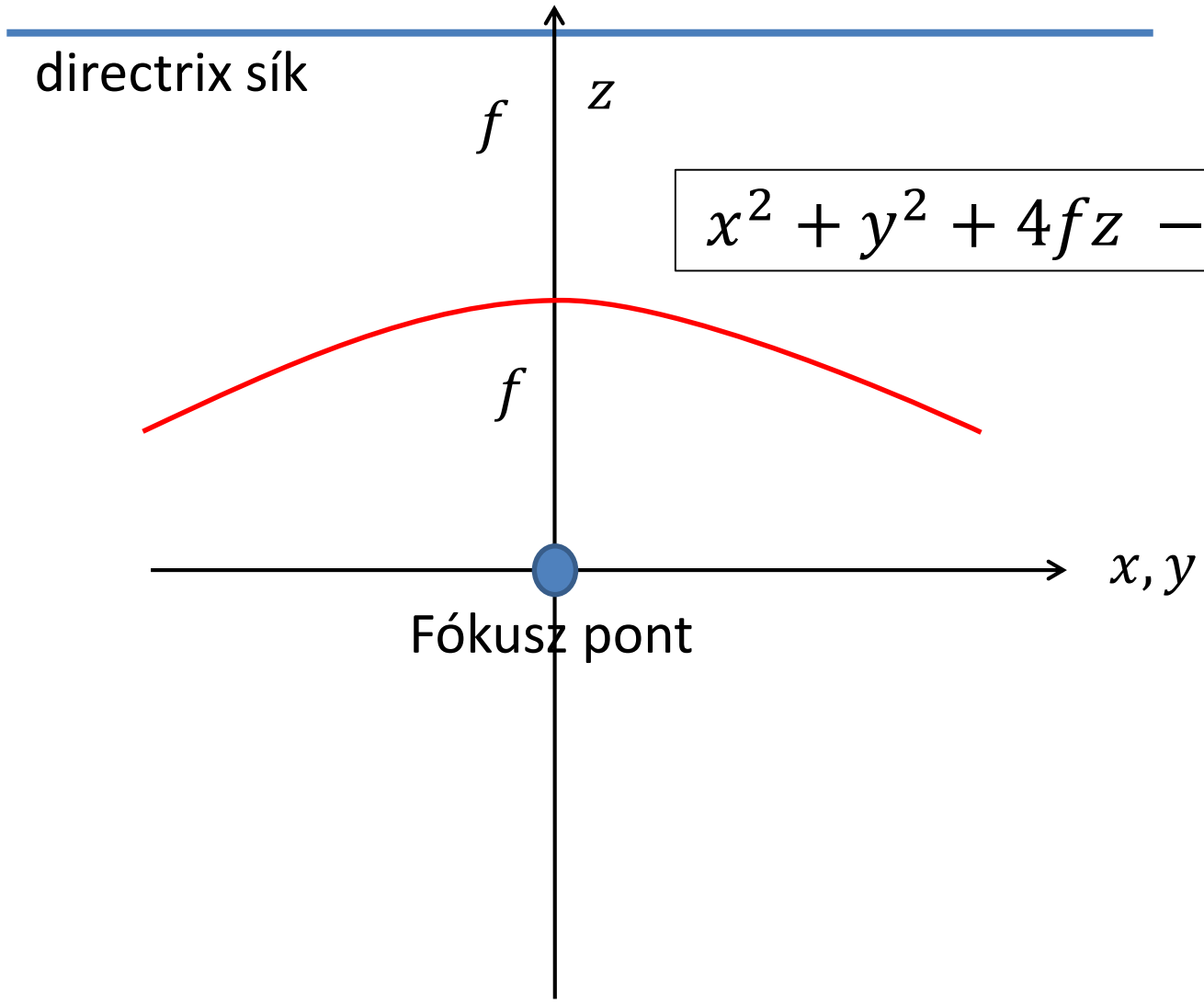
$x^2 + y^2 - 4fz = 0$

Gradiens:

$\boldsymbol{n} = (2x, 2y, -4f)$



Paraboloid

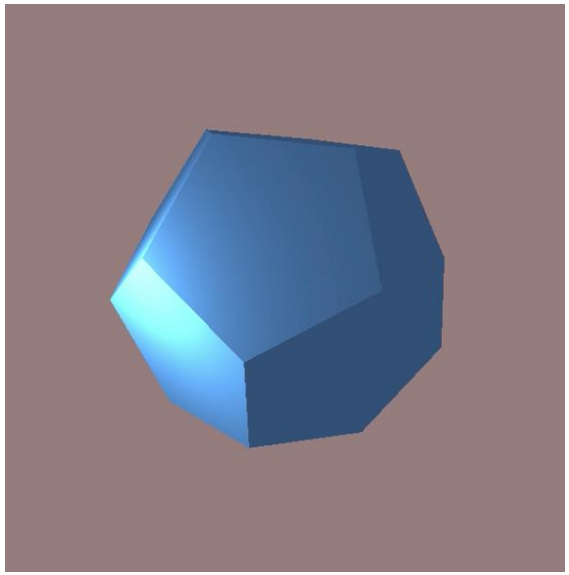


$$x^2 + y^2 + 4fz - 4f^2 = 0$$

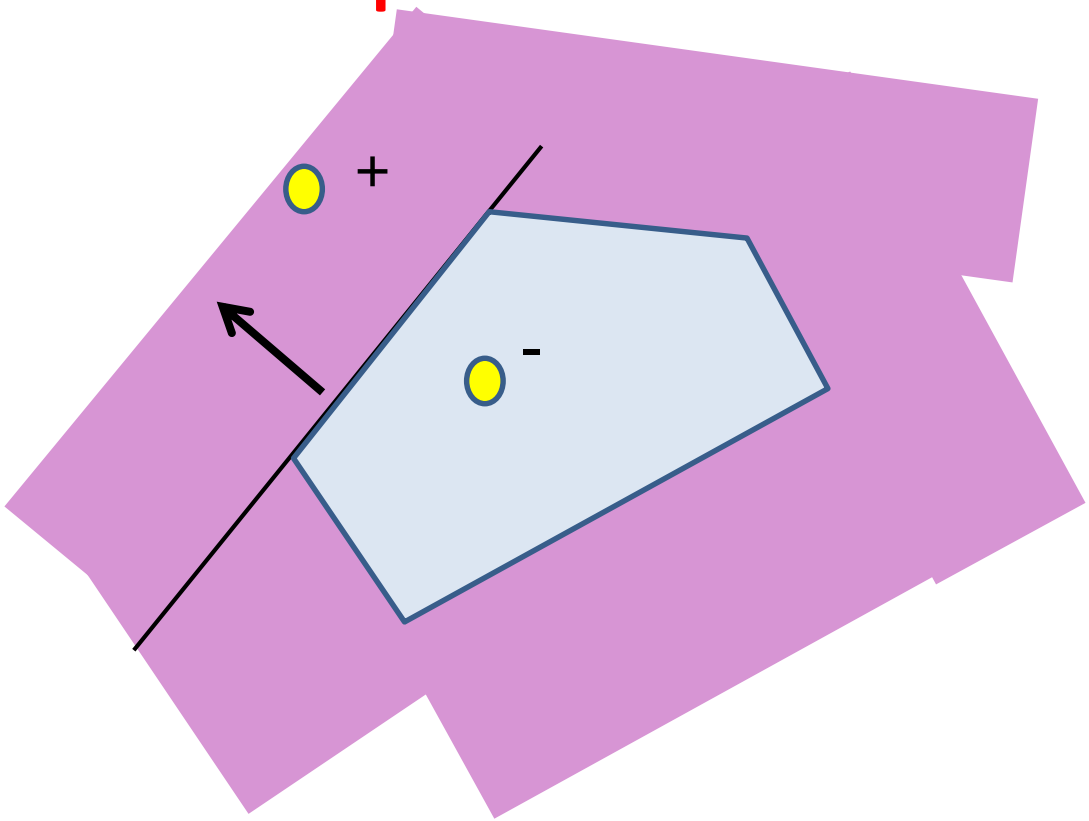
Dodekaéder: OBJ

```
v 0 0.618 1.618
v 0 -0.618 1.618
v 0 -0.618 -1.618
v 0 0.618 -1.618
v 1.618 0 0.618
v -1.618 0 0.618
v -1.618 0 -0.618
v 1.618 0 -0.618
v 0.618 1.618 0
v -0.618 1.618 0
v -0.618 -1.618 0
v 0.618 -1.618 0
v 1 1 1
v -1 1 1
v -1 -1 1
v 1 -1 1
v 1 -1 -1
v 1 1 -1
v -1 1 -1
v -1 -1 -1
```

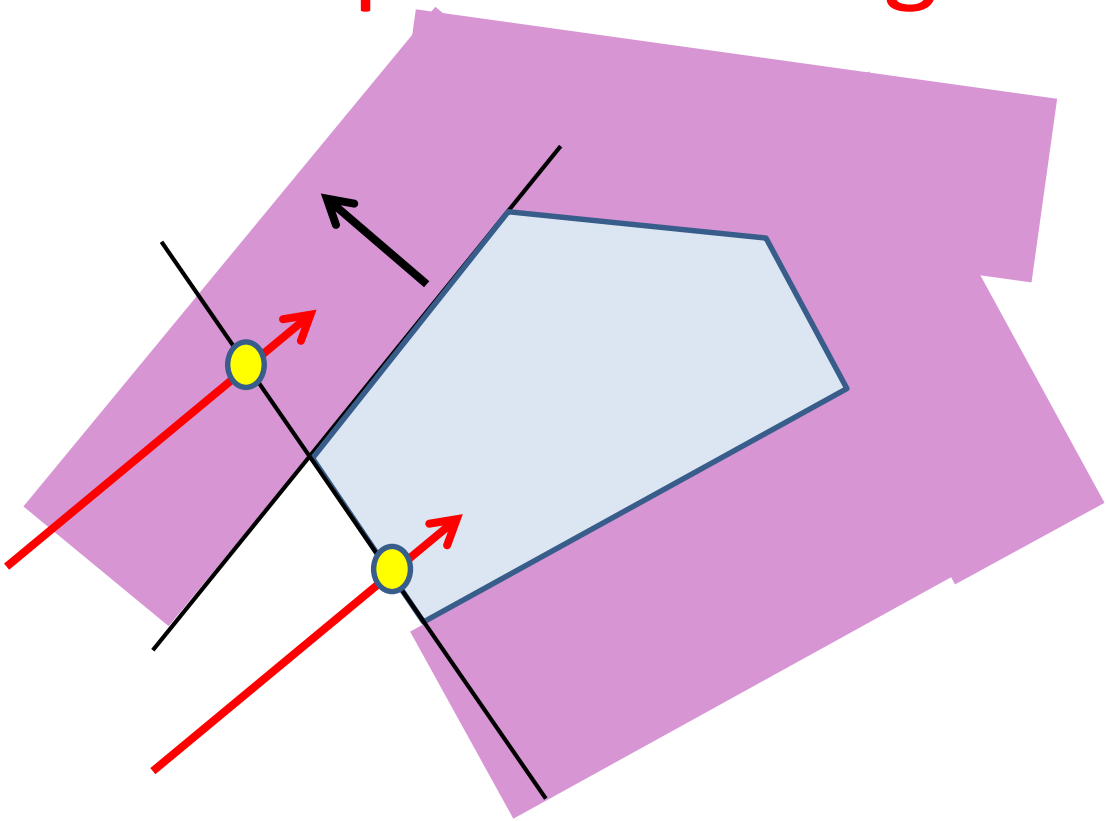
```
f 1 2 16 5 13
f 1 13 9 10 14
f 1 14 6 15 2
f 2 15 11 12 16
f 3 4 18 8 17
f 3 17 12 11 20
f 3 20 7 19 4
f 19 10 9 18 4
f 16 12 17 8 5
f 5 8 18 9 13
f 14 10 19 7 6
f 6 7 20 11 15
```



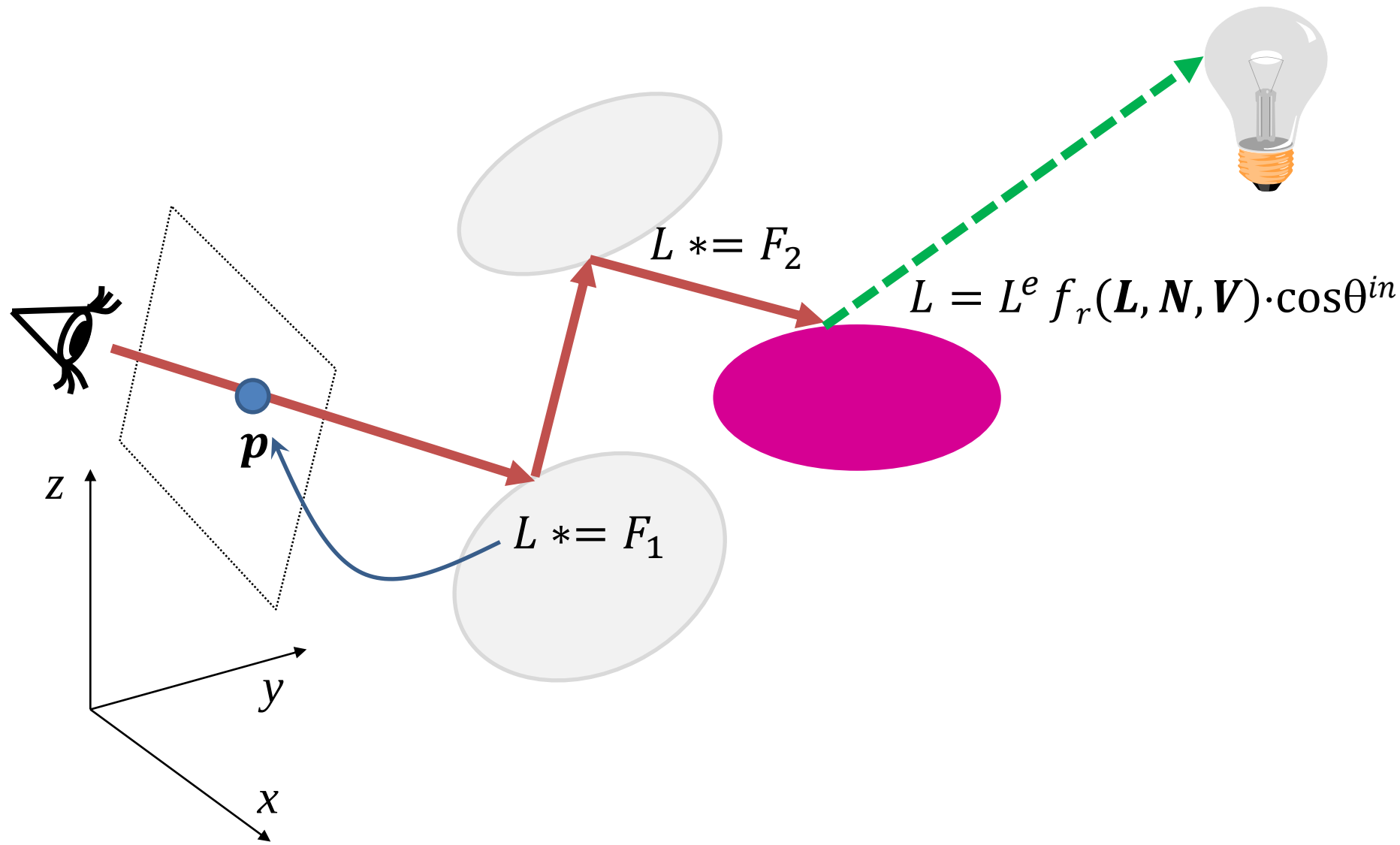
Konvex poliéder tartalmazás



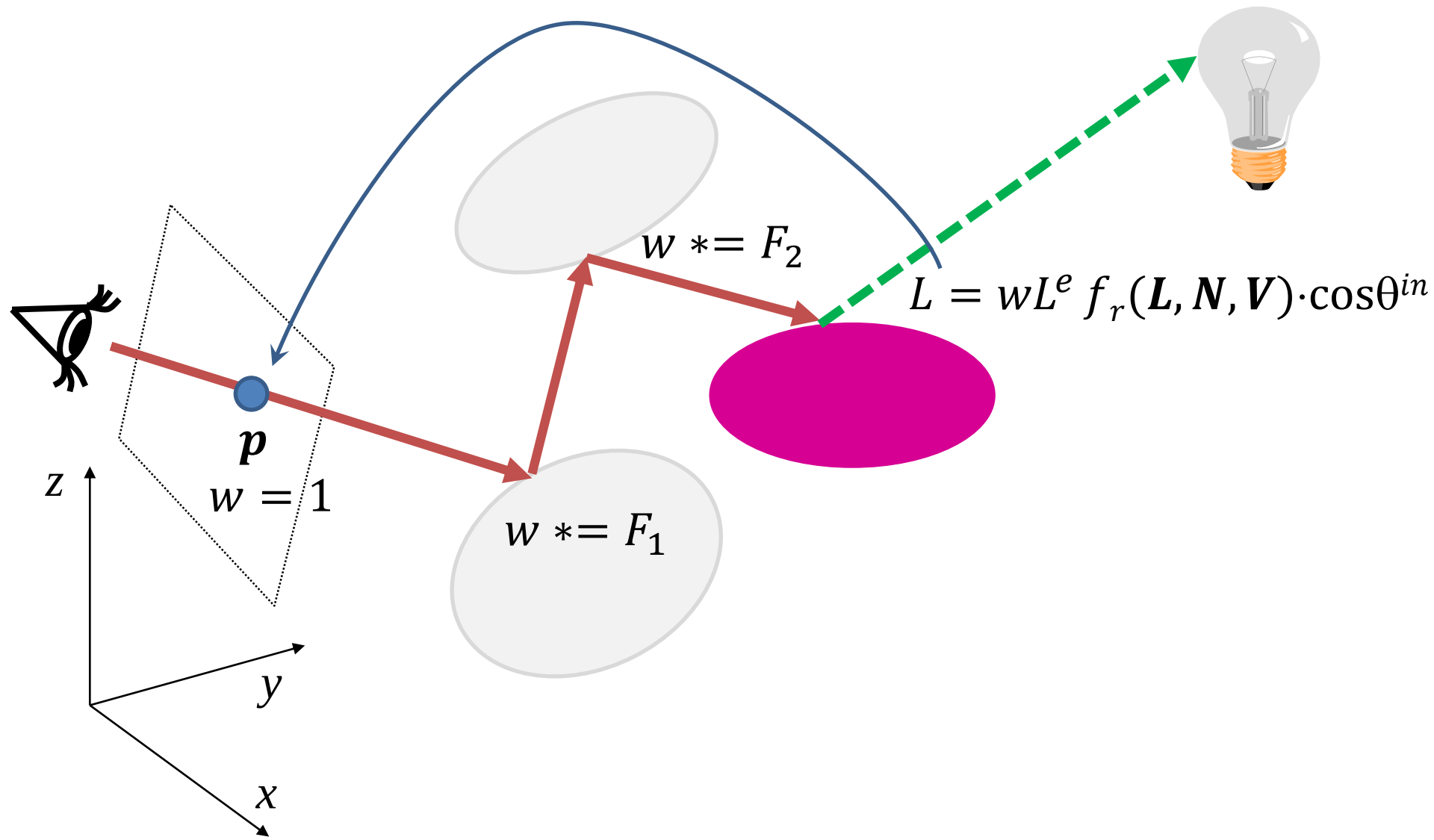
Konvex poliéder sugárkövetés



CPU sugárkövetés



GPU sugárkövetés



Feladatok

- Sugár-paraboloid metszés, figyelembe véve a korlátozást
- Sugár-dodekaéder, sugár-platoni metszés (poligon vagy sík)
- Tükörirány számítás, Fresnel
- Diffúz-spekuláris sugársűrűség számítás
- Kamera vezérlés

“Bad programmers worry about the code. Good programmes worry about data structures.”

Linus Torvalds

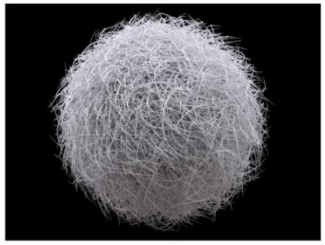
Térpartícionáló adatstruktúrák

Szirmay-Kalos László

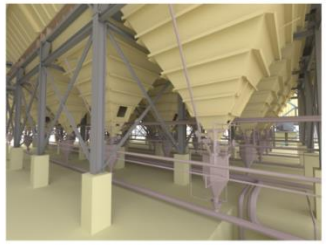
(a) CONFERENCE (282K triangles)

(b) CRYTEK SPONZA (262K triangles)

(c) FAIRY (174K triangles)



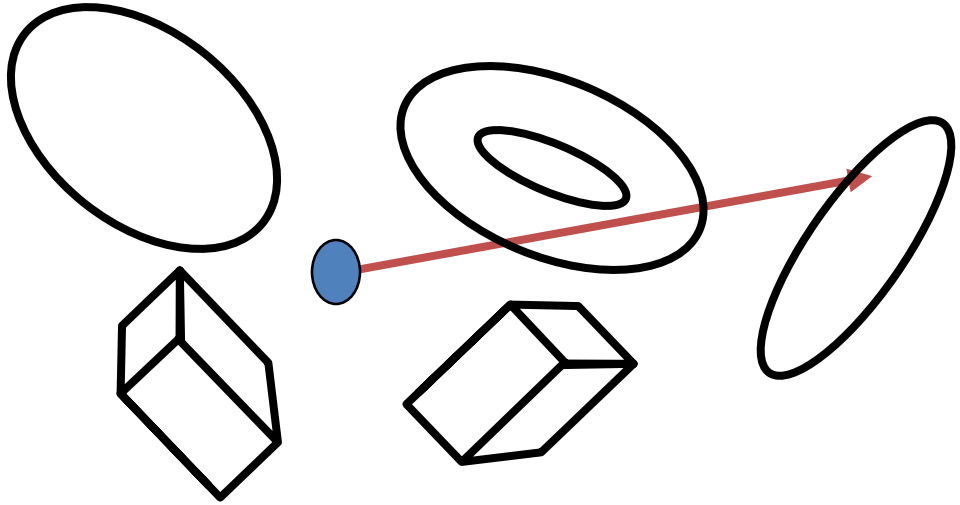
(d) HAIRBALL (2.9M triangles)

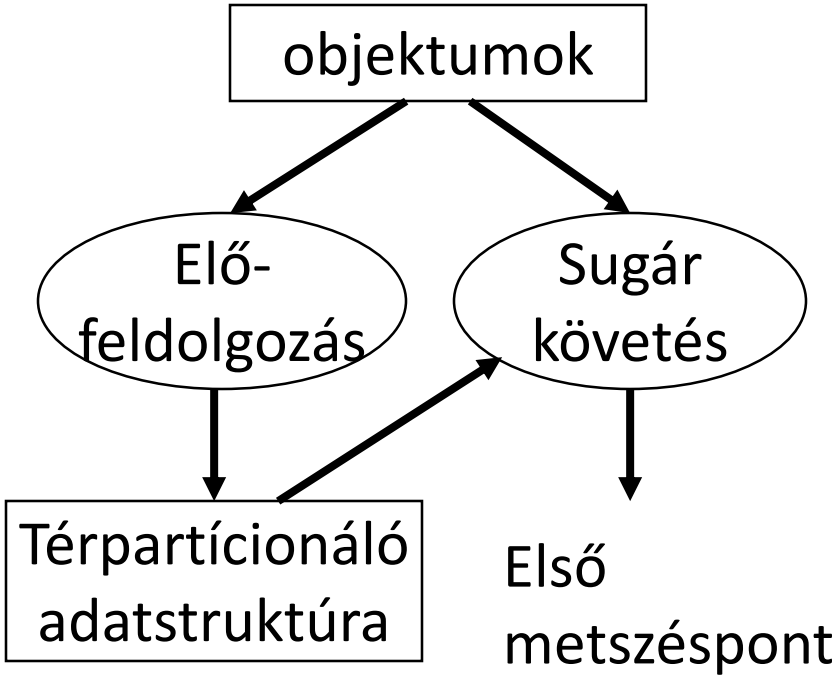


(e) POWER PLANT (12.7M triangles)

(f) SAN MIGUEL (10.5M triangles)

Térpartícionáló módszerek

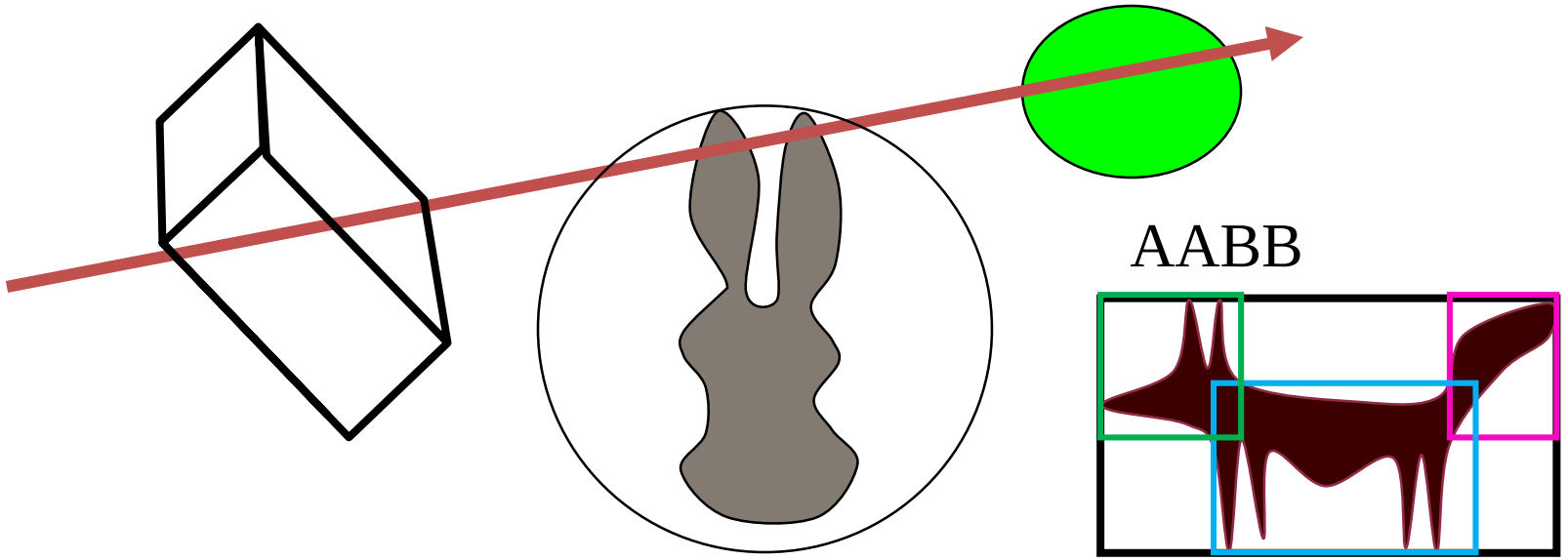




Adatstruktúra:

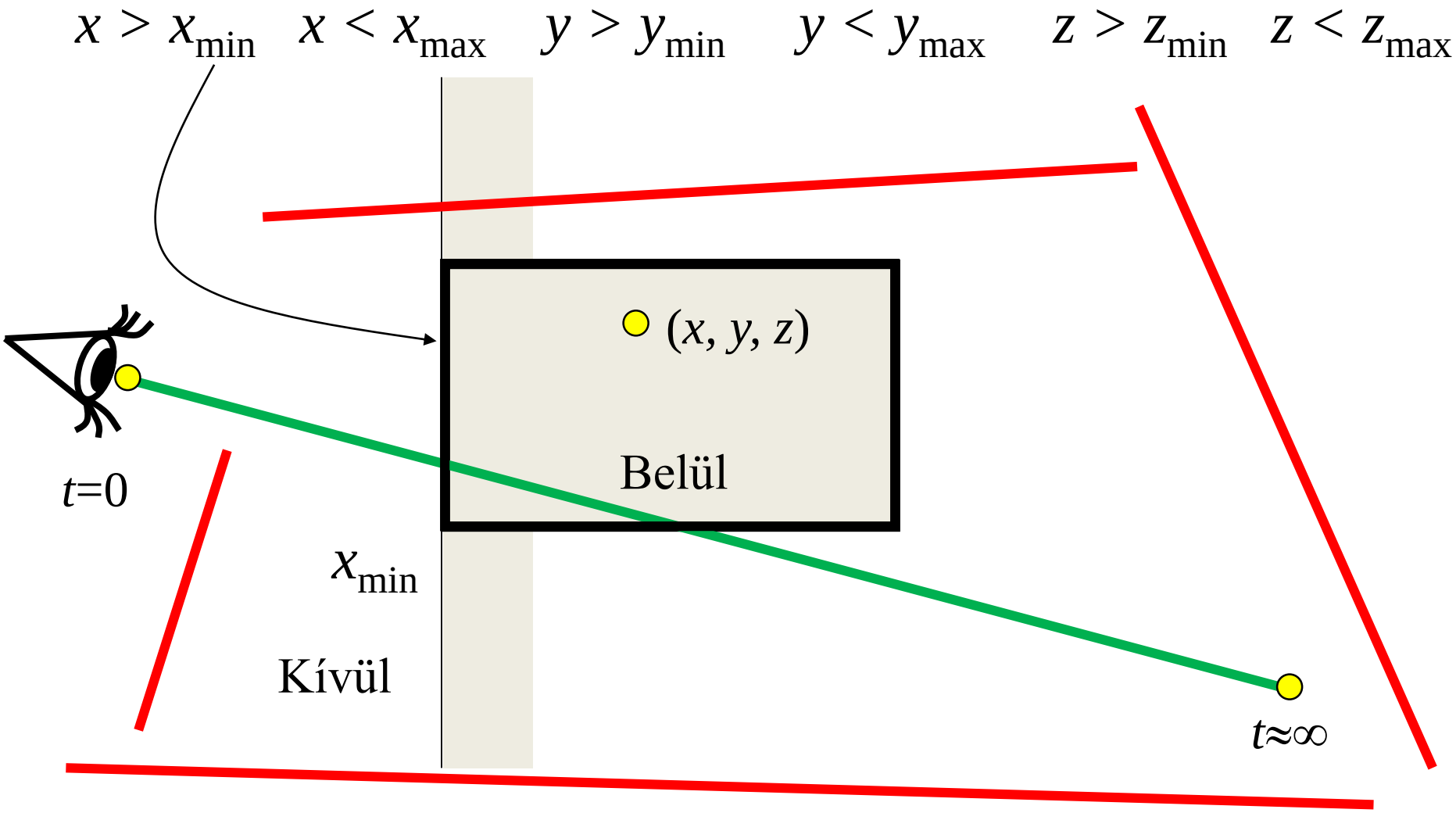
- Ha ismert a sugár, akkor a potenciális metszett objektumok számát csökkenti
- Ha a potenciálisak közül találunk egyet, akkor a többi nem lehet közelebb

Befoglaló térfogat (Bounding Volume)

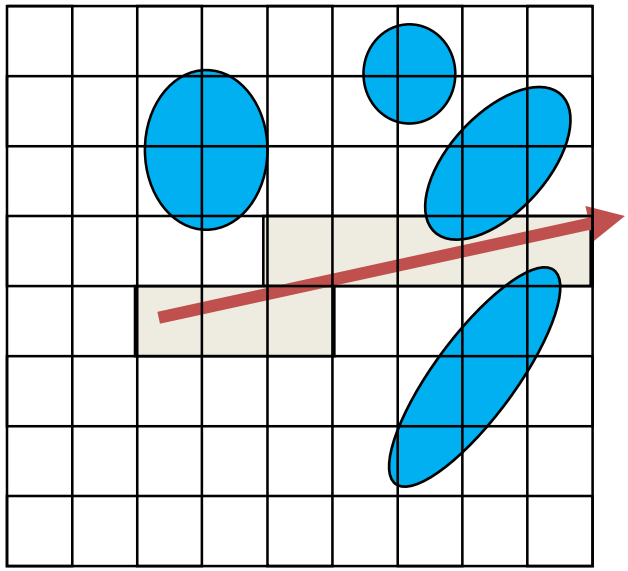


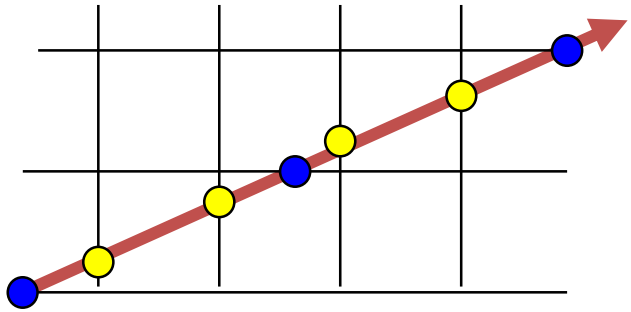

```
double IntersectBV( ray, object )          //  < 0 ha nincs
    IF ( Intersect( ray, bounding volume of object) < 0) RETURN -1;
    RETURN Intersect( ray, object );
END
```

AABB metszés vágással



Reguláris térháló





Előfeldolgozás:

Minden cellára a metszett objektumok

komplexitás: $O(n \cdot c) = O(n^2)$

Sugárkövetés:

FOR each cell of the line // line drawing

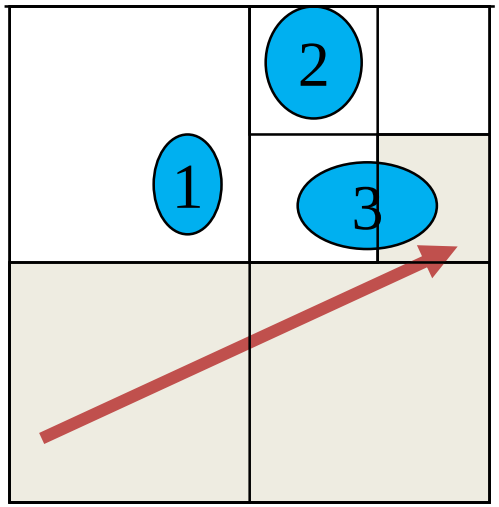
Metszés a cellában lévőkkel

IF van metszés RETURN

ENDFOR

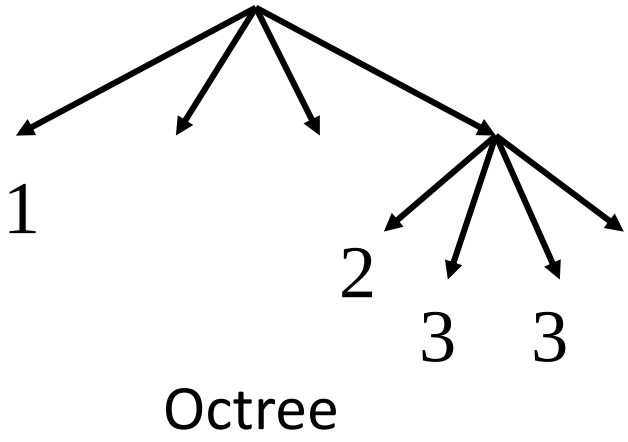
átlagos eset komplexitás: $O(1)$

Nyolcas (oktális) fa



Faépítés(cella):

IF a cellában kevés objektum van
cellában regisztráld az objektumokat
ELSE
cellafelezés: c1, c2, …, c8
Faépítés(c1); … Faépítés(c8);
ENDIF

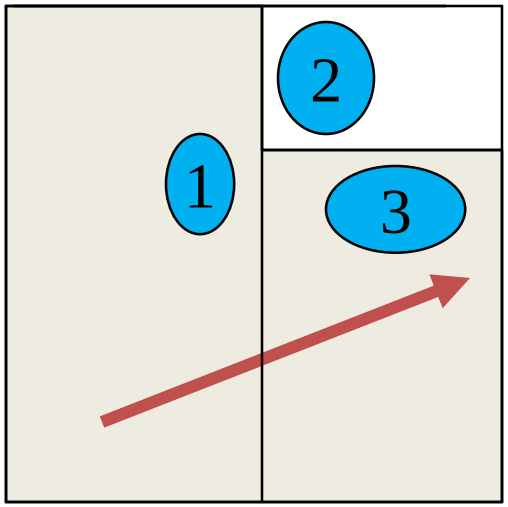


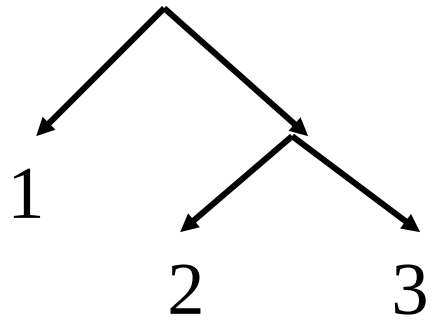
Octree

Sugárkövetés:

FOR összes sugár által metszett cellára
Metszés a cellában lévőkkel
IF van metszés RETURN
ENDFOR

Binary Space Partitioning fa (kd-fa)





kd-tree

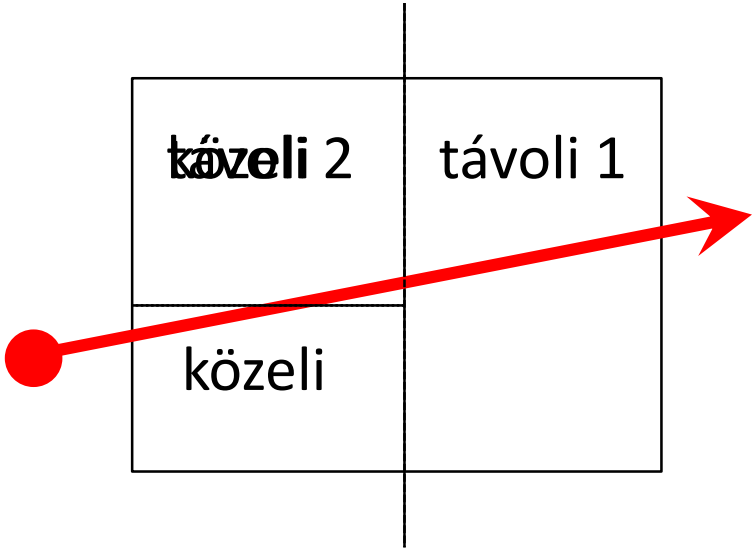
Faépítés(cella):

IF a cellában kevés objektum van
cellában regisztráld az objektumokat
ELSE
sík keresés
cella felezés a síkkal: c1, c2
Faépítés(c1); Faépítés(c2);
ENDIF

Sugárkövetés:

FOR each cell intersecting the line
Metszés a cellában lévőkkel
IF van metszés RETURN
ENDFOR

kd-fa bejárása




```
while (stack nem üres) {
    Pop(AABB)
    while (nem levél) {
        Ha van AABB-nek távoli
            Push(távoli)
        AABB = közeli
    }
    Regisztrált objektumok metszése
    Ha van metszéspont return
}
```

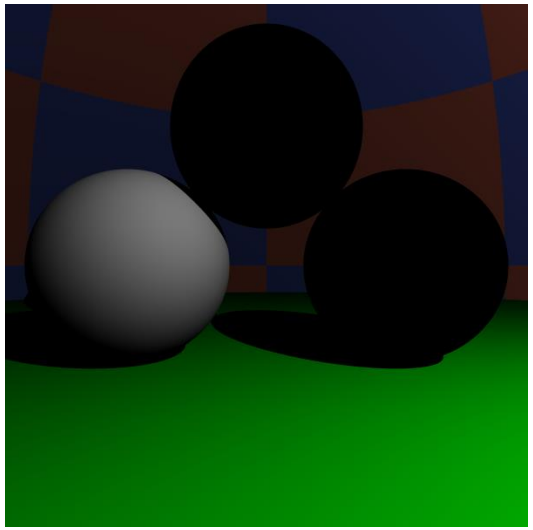
Távolabbi AABB
távoli 1

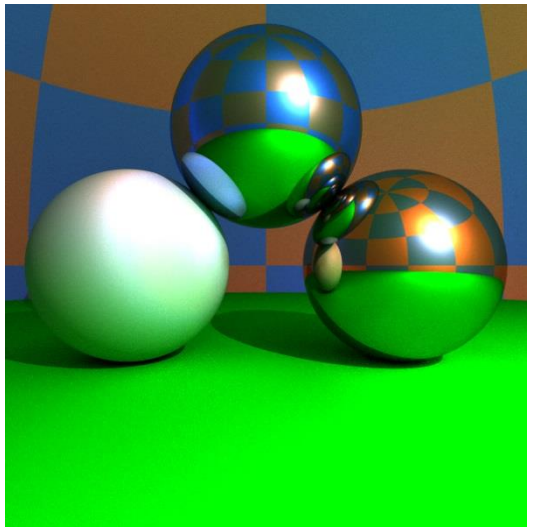
“Úgy szeretnék integrálni.”
Kockaéder

Globális illumináció: path tracing

Szirmay-Kalos László

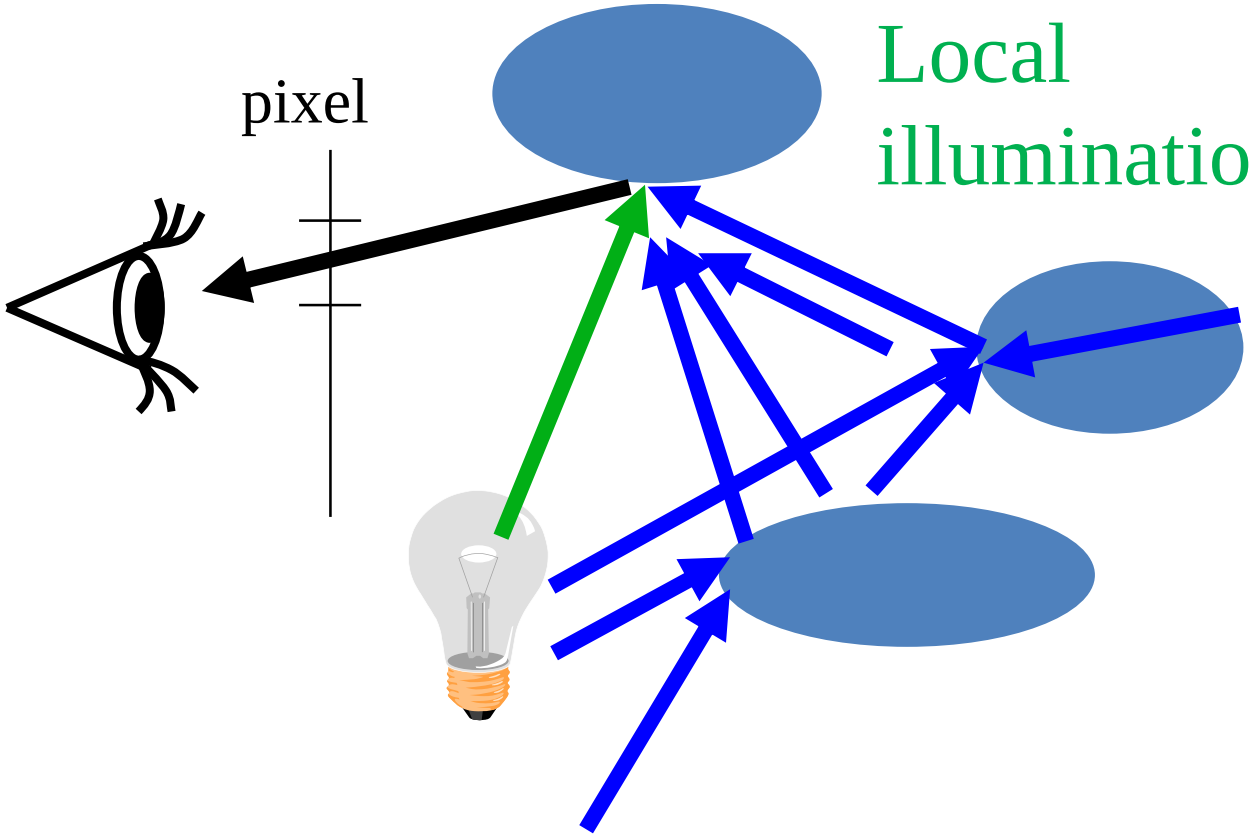
Képszintézis





Local illumination

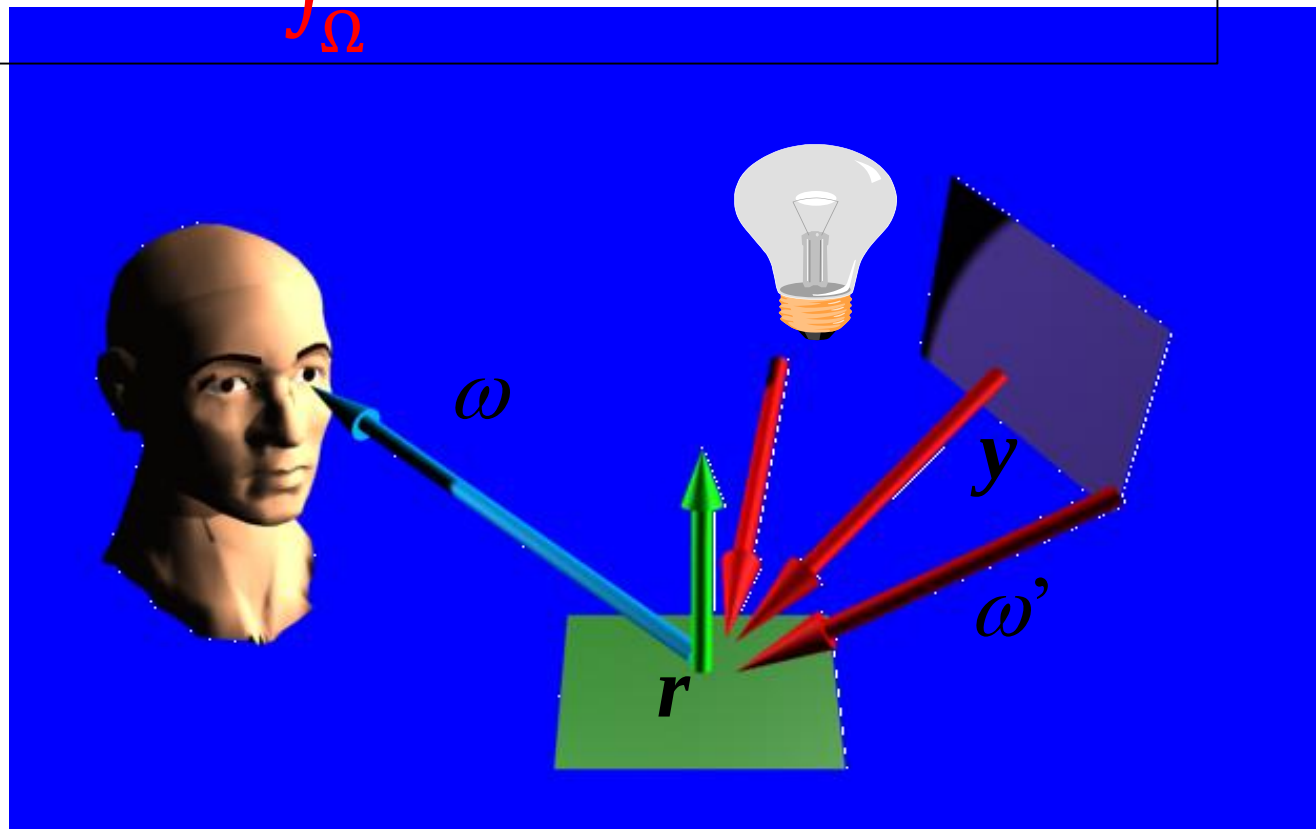
Global illumination



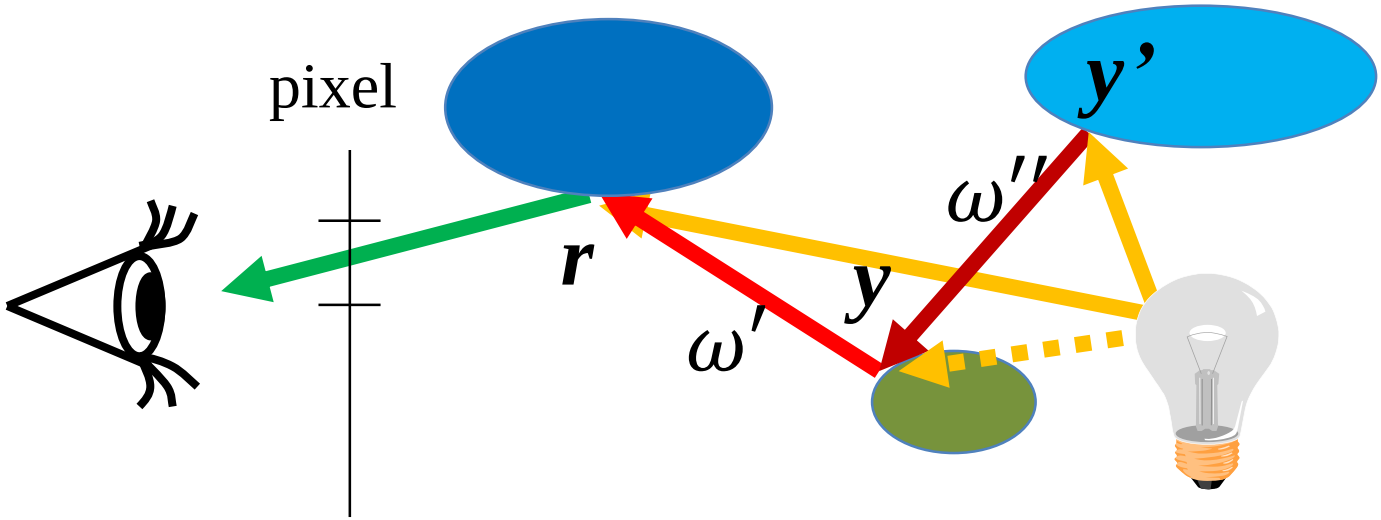
Rendering equation

OutRad = DirectLight + $\sum$ InRad * Reflection

$$L(\boldsymbol{r},\omega) = D(\boldsymbol{r},\omega) + \int_{\Omega} L(\boldsymbol{y},\omega')R(\omega,\omega')\,\mathrm{d}\omega'$$



A megoldás integrálok sorozata

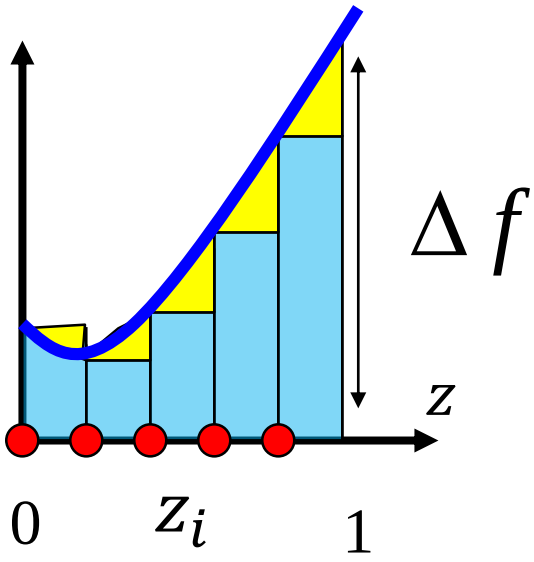


$$L(\boldsymbol{y}', \omega'') = D(\boldsymbol{y}', \omega'') + \int_{\Omega''} L(\boldsymbol{y}'', \omega''') R(\omega'', \omega''') \, \mathrm{d}\omega'''$$

$$L(\boldsymbol{y}, \omega') = D(\boldsymbol{y}, \omega') + \int_{\Omega'} L(\boldsymbol{y}', \omega'') R(\omega', \omega'') \, \mathrm{d}\omega''$$

$$L(\boldsymbol{r}, \omega) = D(\boldsymbol{r}, \omega) + \int_{\Omega} L(\boldsymbol{y}, \omega') R(\omega, \omega') \, \mathrm{d}\omega'$$

Numerikus integrálás



$$\int_0^1 f(z)\mathrm{d}z \approx \frac{1}{M}\sum_{i=1}^{M} f(z_i)$$

M minta

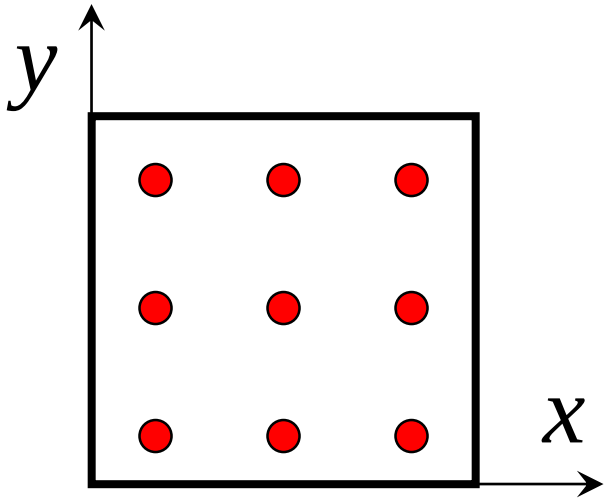
Átlagos magasság

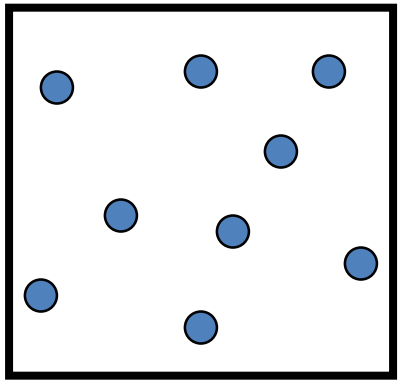
Alap

Háromszögek száma

$$\text{Error} = \frac{\Delta f}{2M} \cdot \frac{1}{M} \cdot M = \frac{\Delta f}{2M} = O(M^{-1})$$

Magasabb dimenziók: Megátkozva





Véletlen minták

$n = \sqrt{M}$ minta egy dimenzióban

Hiba 2-dim $= O(n^{-1}) = O(M^{-0.5})$

Hiba D-dim $= O(n^{-1}) = O(M^{-1/D})$

Monte Carlo integrálás:
Integrál = várható érték

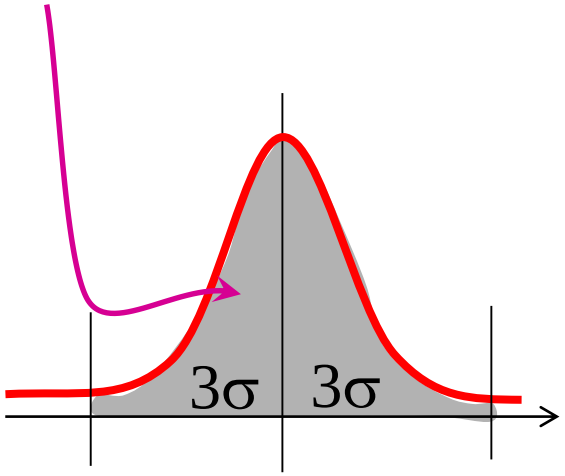
$$\int f(z)\mathrm{d}z = \int \frac{f(z)}{p(z)} p(z)\mathrm{d}z = \mathbf{E}\left[\frac{f(z)}{p(z)}\right] \approx \frac{1}{M}\sum_{i=1}^{M} \frac{f(z_i)}{p(z_i)}$$

A becslő egy valószínűségi változó:

$$\text{Variancia} = \sigma^2 = \mathbf{D}^2\left[\frac{f(z)}{p(z)}\right]\frac{1}{M}$$

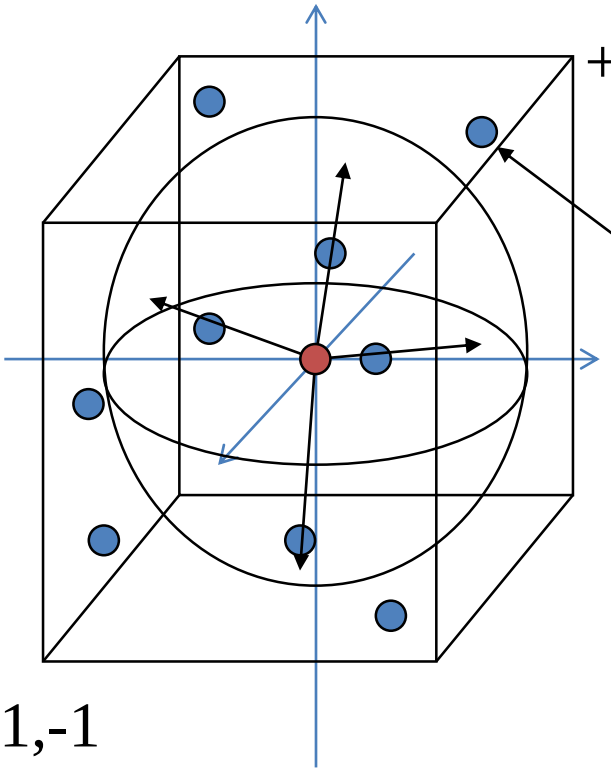
$$\text{Error} < 3\sigma = \frac{3}{\sqrt{M}}\mathbf{D}\left[\frac{f(z)}{p(z)}\right]$$

3 × szóráson belül
99.7% konfidenciával



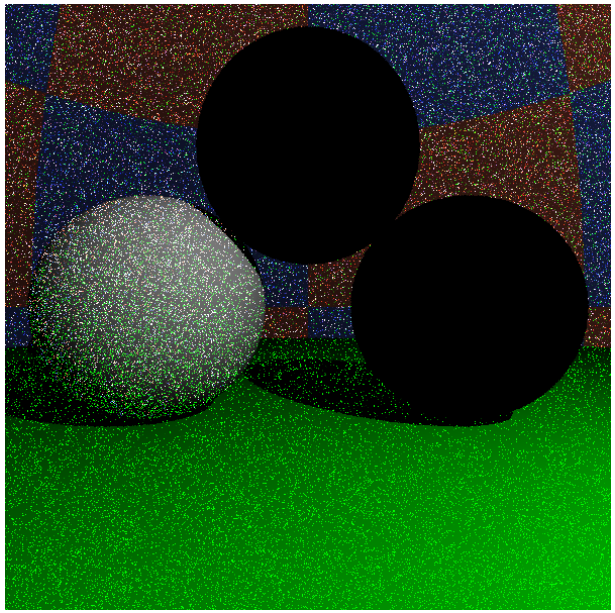
Irányok generálása egyenletes valószínűséggel

```
float random() { return (float)rand()/RAND_MAX; }
```

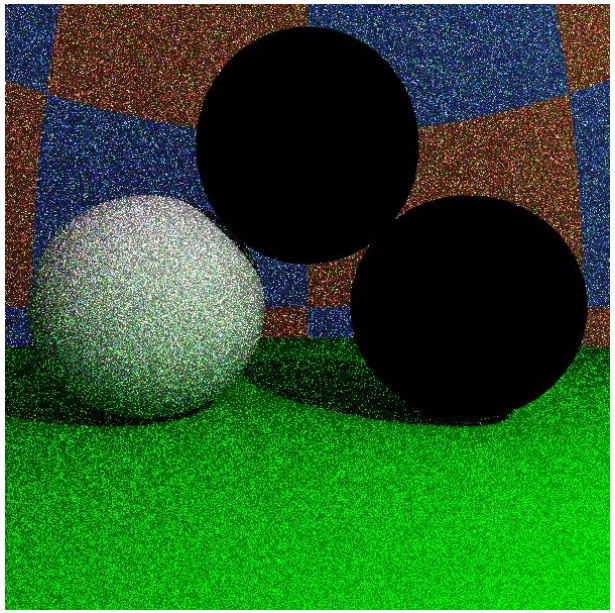



```
vec3 p;
do {
   p.x = 2*random()-1;
   p.y = 2*random()-1;
   p.z = 2*random()-1;
} while(dot(p, p) > 1);

outDir = normalize(p);
pdf = 1.0/4.0/M_PI;
```

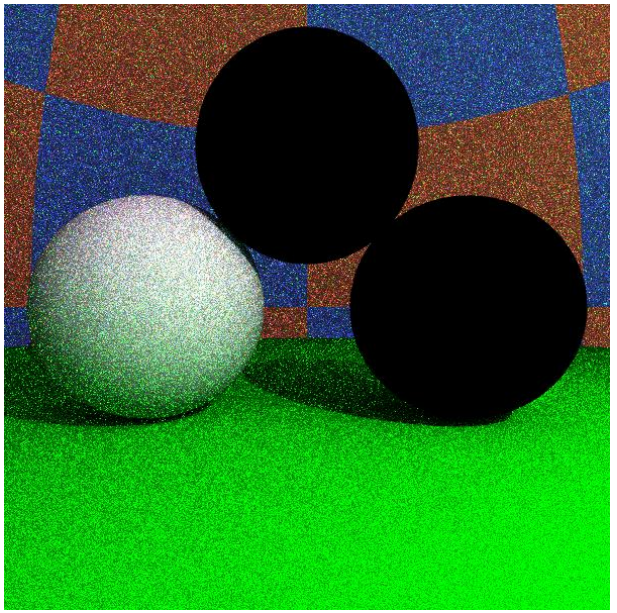
Uniform



1 sample/pixel

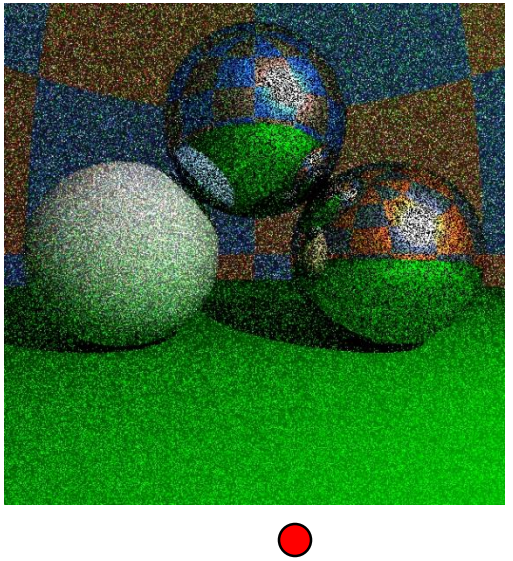


10 samples/pixel



100 samples/pixel

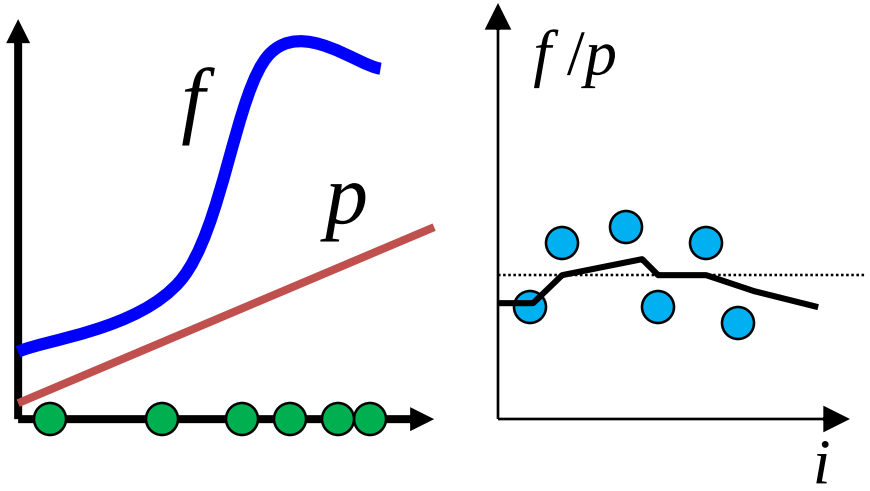
Fontosság szerinti mintavétel



Becslő: $\frac{1}{M}\sum_{i=1}^{M}\frac{f(z_i)}{p(z_i)}$

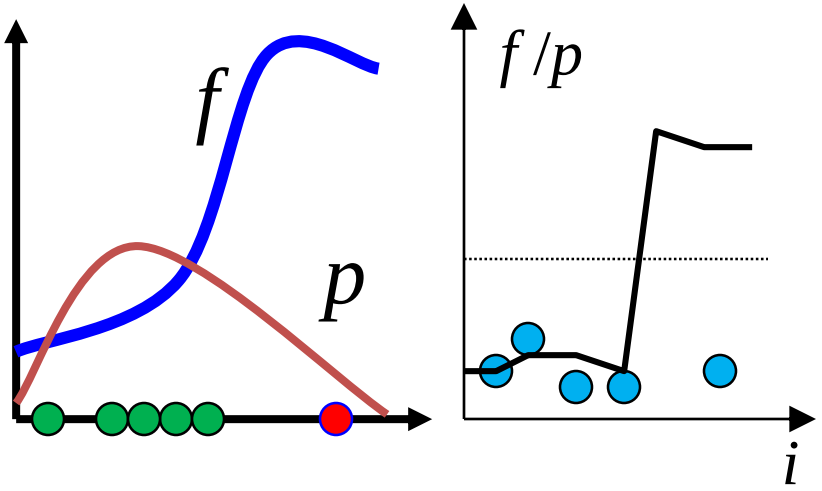
$f(z)/p(z)$ legyen lapos

Ahol f nagy, p is legyen nagy



jó

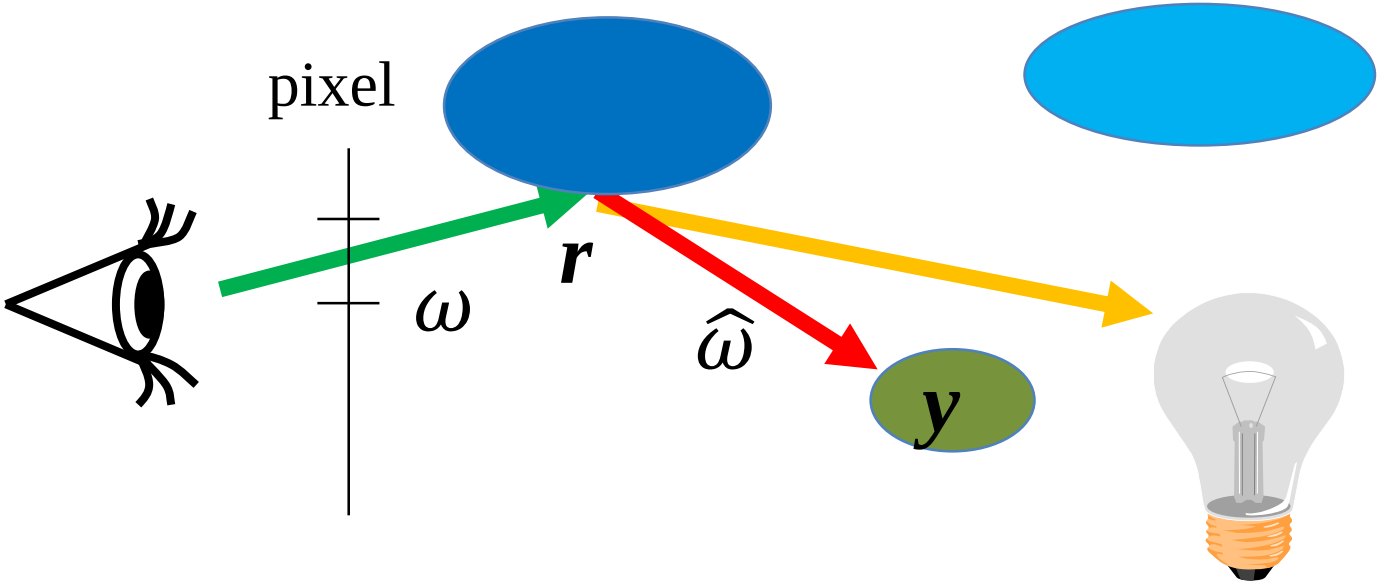
hasonló f/p hozzájárulások



rossz

Ritka, nagy f/p hozzájárulások

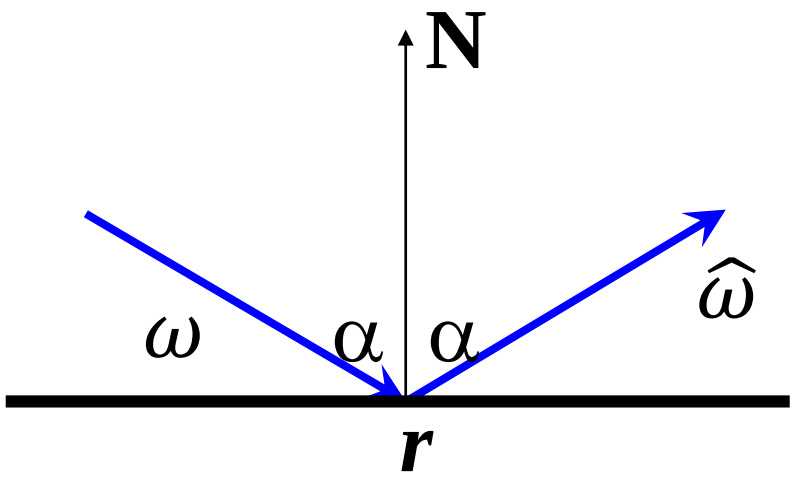
Véletlen bolyongás



$$\int_{\Omega} L(\boldsymbol{y}, \omega') R(\omega, \omega') \, \mathrm{d}\omega' \approx \frac{L(\boldsymbol{y}, \hat{\omega}) R(\omega, \hat{\omega})}{p(\hat{\omega})}$$

- $p(\hat{\omega})$ –nak az integrandusra kell hasonlítania (fontosság)
- A $L(\boldsymbol{y}, \omega')$ nem ismerjük, mert éppen számoljuk
- $p(\hat{\omega})$ legyen arányos $R(\omega, \hat{\omega})$

Tükör: Irány diszkrét eloszlású (Diract-delta)

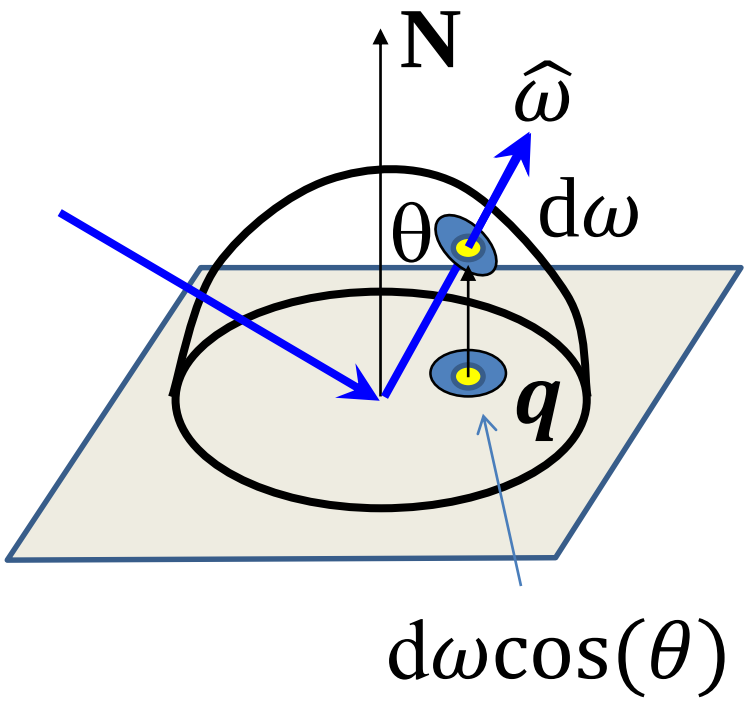


$$\cos\alpha = -(\omega \cdot \mathbf{N})$$

$$\hat{\omega} = \omega + 2\mathbf{N}\cos\alpha$$

```
float SampleMirror(vec3 N, vec3 inDir, vec3& outDir) {
    outDir = inDir - N * dot(N, inDir) * 2.0f;
    return 1; // pdf
}
```

Diffúz: Irány cos eloszlással



$$p(\hat{\omega})\,\mathrm{d}\omega = \mathrm{Prob}\{\hat{\omega} \text{ in } \mathrm{d}\omega\}$$

$$= \mathrm{Prob}\{\boldsymbol{q} \text{ in } \mathrm{d}\omega\cos(\theta)\}$$

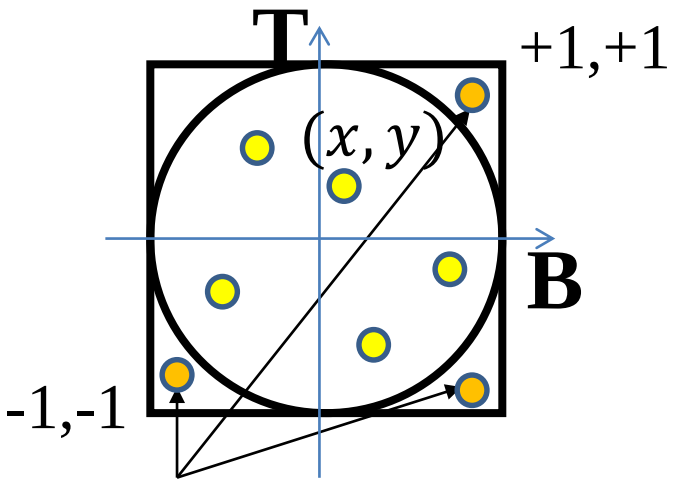
$$= \frac{\mathrm{d}\omega\,\cos(\theta)}{\text{kör területe}}$$

Ha $\boldsymbol{q}$ egyenletes eloszlású

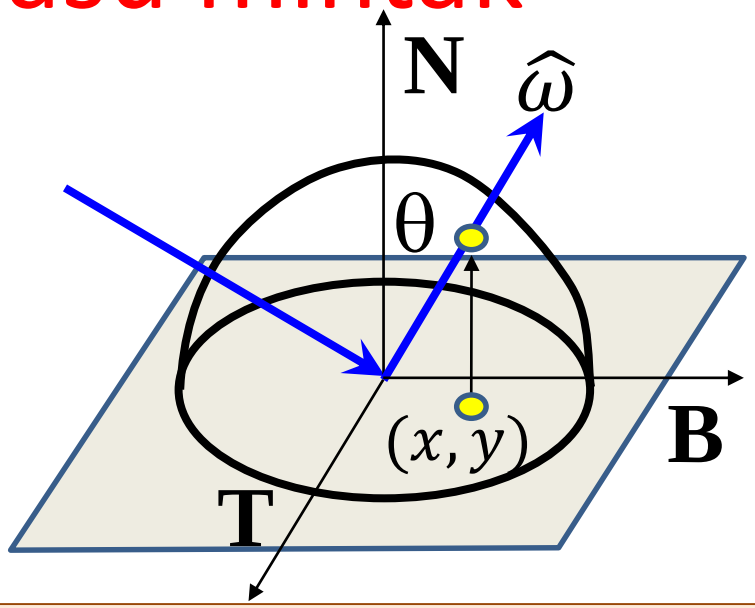
$$= \frac{\mathrm{d}\omega\,\cos(\theta)}{\pi}$$

$$\boxed{p(\hat{\omega}) = \frac{\cos(\theta)}{\pi}}$$

Cos eloszlású minták



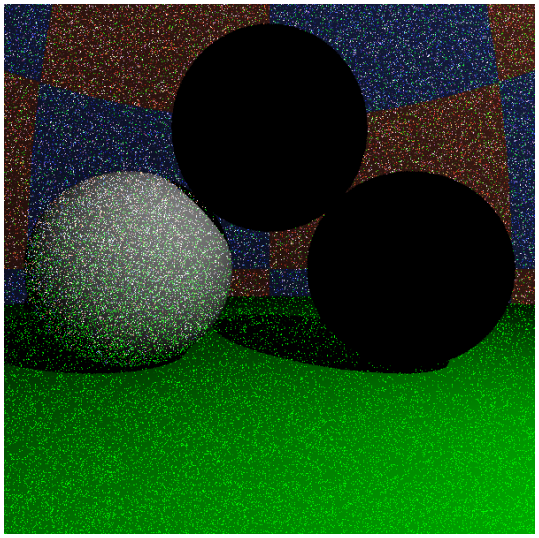
Elvetett minták

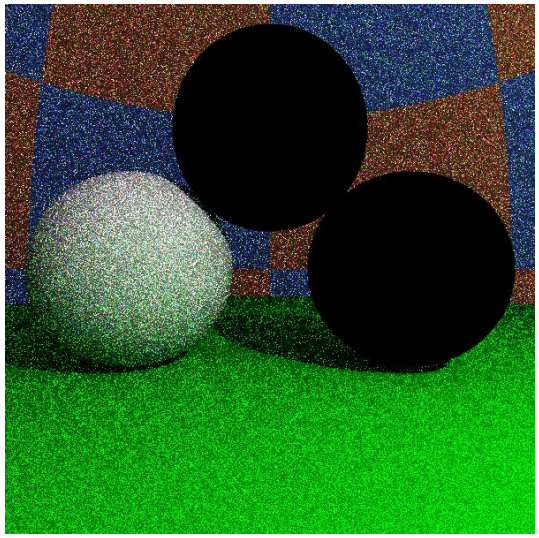


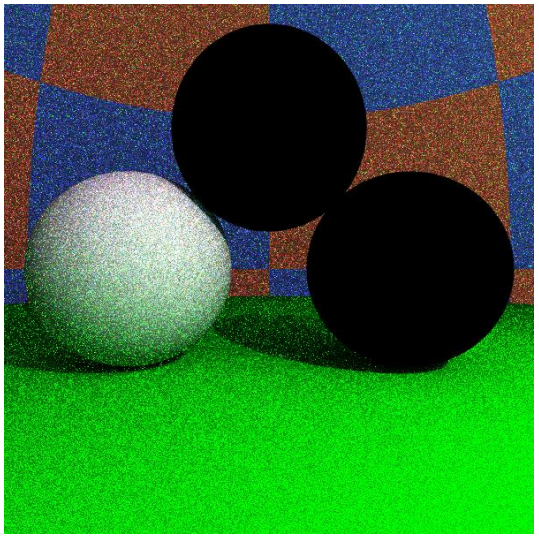

```
float SampleDiffuse(vec3 N, vec3 inDir, vec3& outDir) {
   vec3 T = normalize(cross(N, vec3(1,0,0)));
   vec3 B = cross(N, T);
   float x, y, z;
   do { x = 2*random()-1; y = 2*random()-1; }
      while(x*x + y*y > 1);  // reject if not in circle

   z = sqrtf(1 - x*x - y*y); // project to hemisphere
   outDir = T * x + B * y + N * z;
   return z/M_PI;  // pdf
}
```

Fontosság szerinti mintavétel

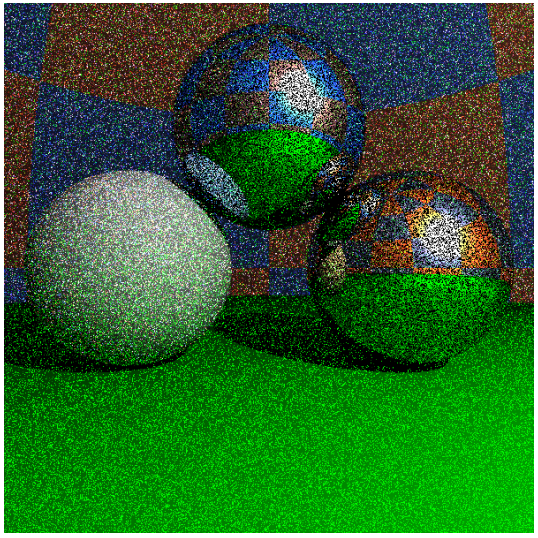
egyenletes

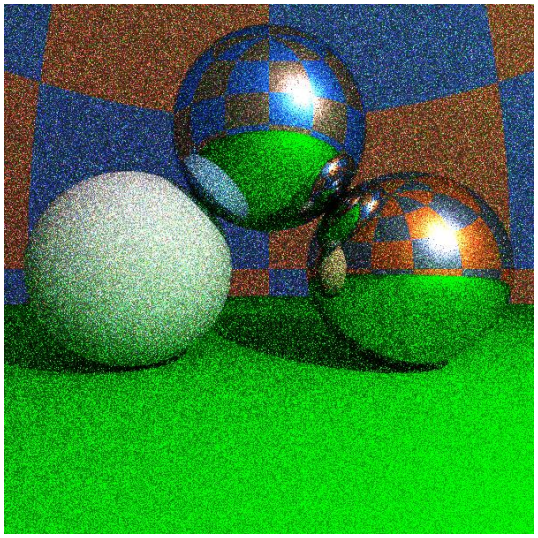


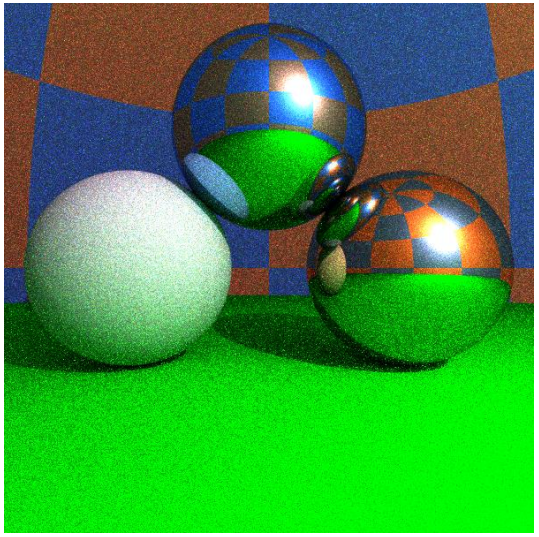




cos + diszkrét







1 sample/pixel

10 samples/pixel

100 samples/pixel

Megállás és visszaverődés típus: Orosz rulett

1. Monte Carlo integrál:

$$\int_{\Omega} L(\boldsymbol{y}, \omega') R(\omega, \omega') \, \mathrm{d}\omega' = \mathbf{E}\left[\frac{L(\boldsymbol{y}, \hat{\omega}) k_d \cos^{+}(\theta)}{p_d(\hat{\omega})}\right] + \mathbf{E}\left[\frac{L(\boldsymbol{y}, \hat{\omega}) F \delta(\omega_m, \hat{\omega})}{p_m(\hat{\omega})}\right]$$

$R_d + R_m$ $\qquad$ $\mathbf{E}[L_d]$ $\qquad$ $\mathbf{E}[L_m]$

2. Számold $\mathbf{E}[L_d]$ -t s_d valószínűséggel, $\mathbf{E}[L_m]$ -t s_m valószínűséggel, egyébként 0
3. Kompenzálj s-sel osztással

<u>Várható érték OK:</u>

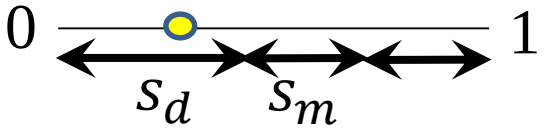
$$s_d \mathbf{E}[L_d / s_d] + s_m \mathbf{E}[L_m / s_m] + (1 - s_d - s_m) 0 = \mathbf{E}[L_d] + \mathbf{E}[L_m]$$

A visszaverődés típusának kiválasztása

$$\frac{L_d}{s_d} = \frac{L(\boldsymbol{y}, \hat{\omega}) k_d \cos^+(\theta)}{p_d(\hat{\omega}) s_d} = L(\boldsymbol{y}, \hat{\omega}) \frac{k_d \cos^+(\theta)}{\frac{\cos(\theta)}{\pi} s_d} = L(\boldsymbol{y}, \hat{\omega}) \frac{k_d \pi}{s_d}$$

$$\frac{L_m}{s_m} = \frac{L(\boldsymbol{y}, \hat{\omega}) F \delta(\omega_m, \hat{\omega})}{p_m(\hat{\omega}) s_m} = L(\boldsymbol{y}, \hat{\omega}) \frac{F}{s_m}$$

- Tükörirány valószínűsége: $s_m = F$ luminance
- Diffúz irány valószínűsége: $s_d = k_d \pi$ luminance
- Megállás valószínűsége: $1 - s_m - s_d$



Path Tracer

```
vec3 trace(Ray ray, int depth = 0) {
   Hit hit = firstIntersect(ray);
   if (hit.t < 0 || depth >= maxdepth) return vec3(0,0,0);
   [r, N, k_d, n, κ] ← hit;
   vec3 outRad = DirectLight(hit);
   rnd = random();          // Russian roulette
   s_d = Luminance(k_d π); s_m = Luminance(Fresnel(ray.dir,N));
   if (rnd < s_d) { // diffuse
      pdf = SampleDiffuse(N, ray.dir, outDir);
      inRad = trace(Ray(r + Nε, outDir), depth+1);
      outRad += inRad * k_d * dot(N, outDir) / pdf / s_d;
   } else if (rnd < s_d + s_m) { // mirror
      pdf = SampleMirror(N, ray.dir, outDir);
      inRad = trace(Ray(r + Nε, outDir), depth+1);
      outRad += inRad * Fresnel(ray.dir,N) / pdf / s_m;
   }
   return outRad;
}
```