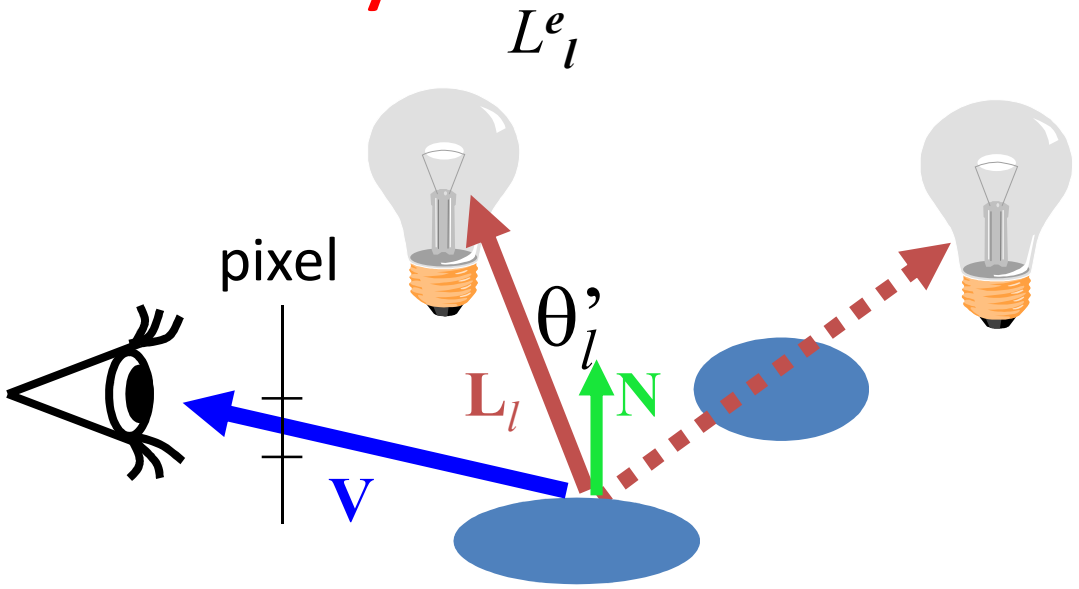
Sugárkövetés:
ray-casting, ray-tracing

Szirmay-Kalos László

Lokális illumináció: rücskös felületek, absztrakt fényforrások



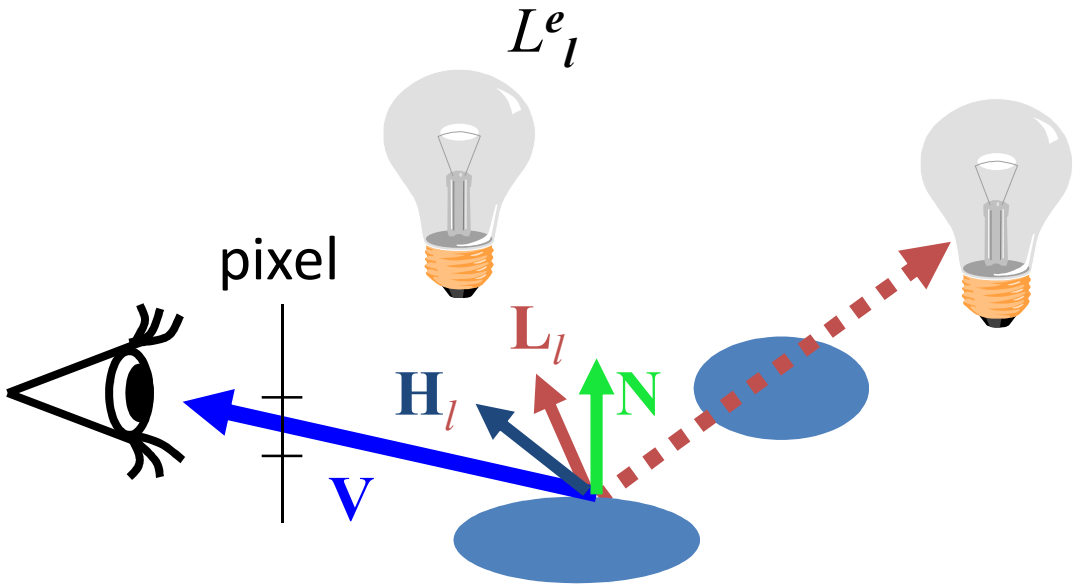
Csak absztrakt fényforrások direkt megvilágítása

$$L(\mathbf{V}) \approx \sum_l L^e_l(\mathbf{L}_l) * f_r(\mathbf{L}_l, \mathbf{N}, \mathbf{V}) \cdot \cos\theta'_l$$

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

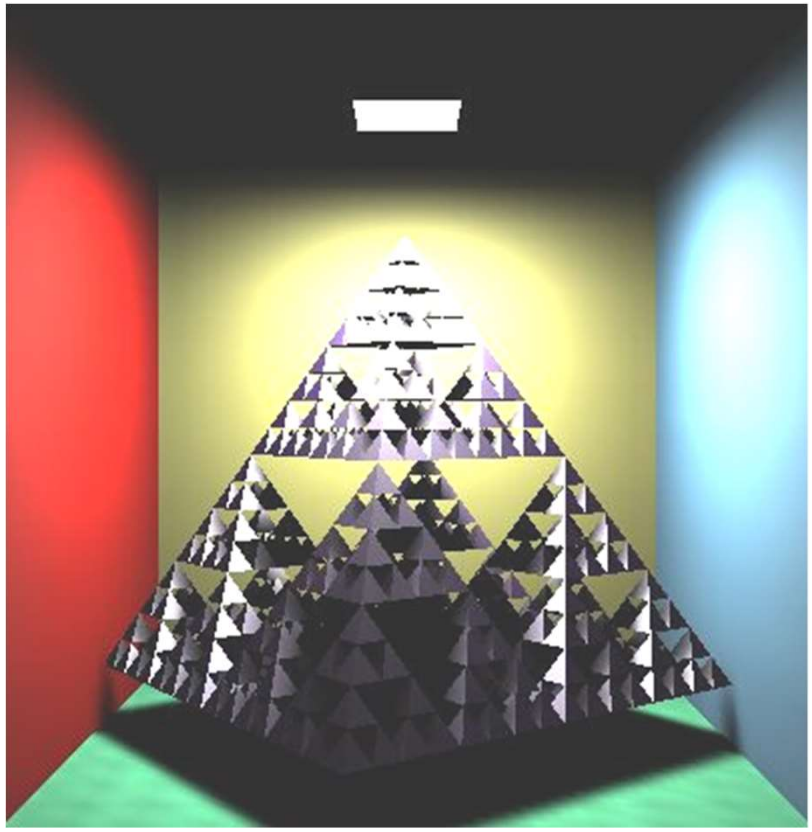


$$L(\mathbf{V}) \approx \sum_l L^e_l(\mathbf{L}_l) * \{k_d \cdot (\mathbf{L}_l \bullet \mathbf{N})^+ + k_s \cdot ((\mathbf{H}_l \bullet \mathbf{N})^+)^{shine}\}$$

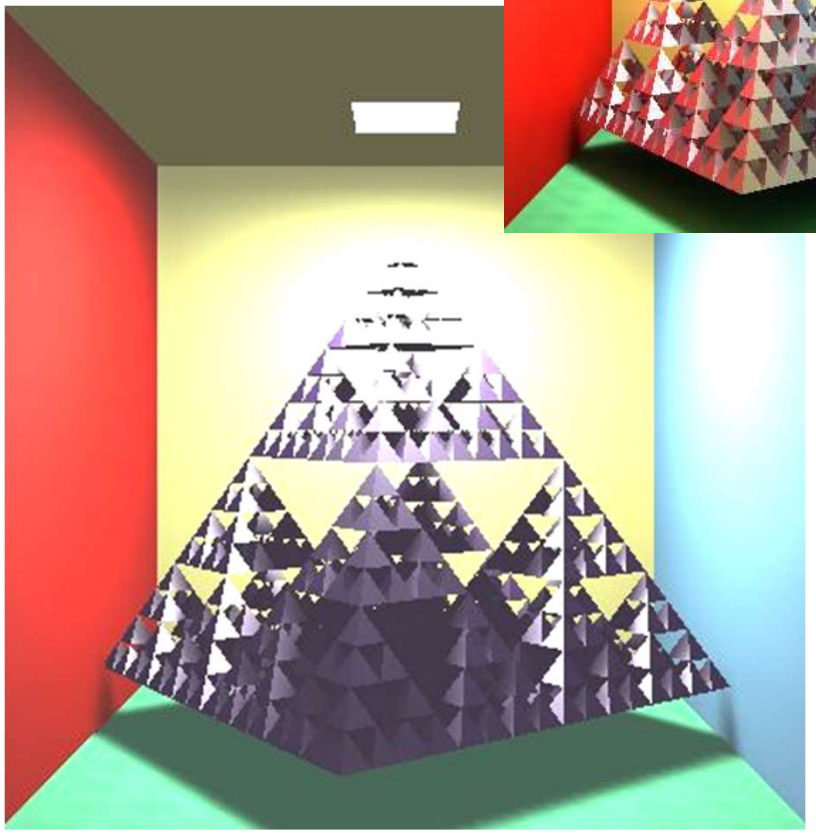
Absztrakt fényforrásokból származó megvilágítás.
(*Irányforrás = konstans; Pontforrás = távolság négyzetével csökken*
Ha takart, akkor zérus)

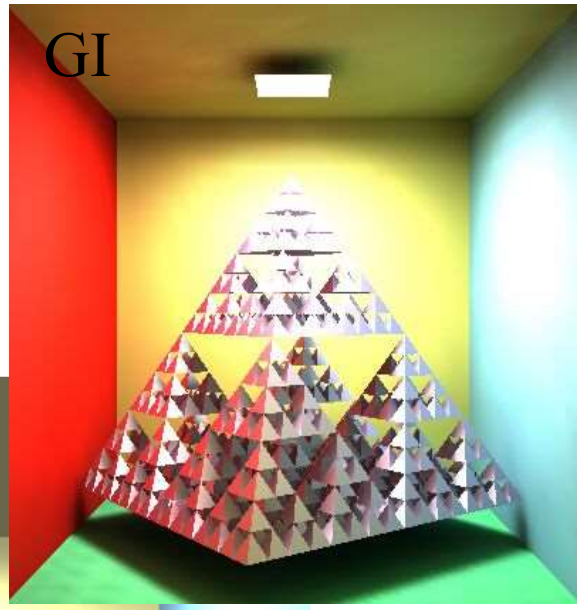
Ambiens tag

Lokális illumináció



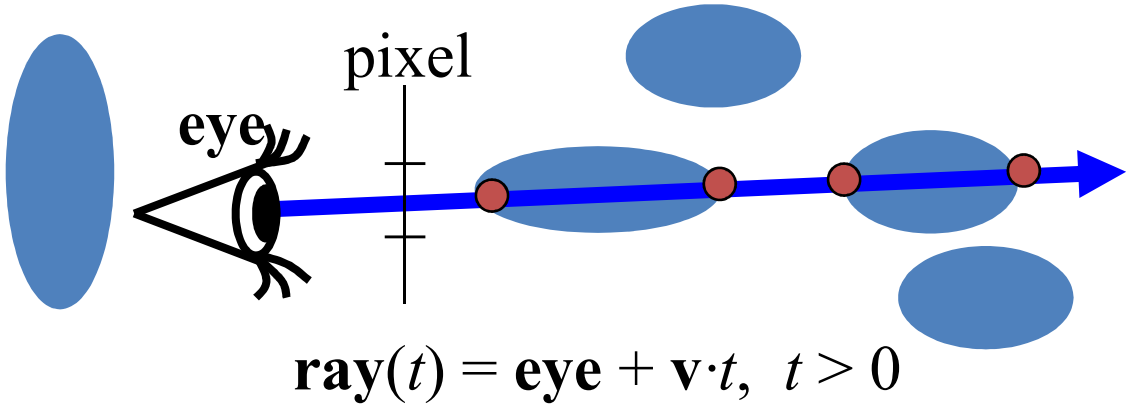
+ ambiens tag





$$L(\mathbf{V}) \approx \sum_l L^e{}_l(\mathbf{L}_l) * f_r(\mathbf{L}_l, \mathbf{N}, \mathbf{V}) \cdot \cos\theta_l' + k_a * L_a$$

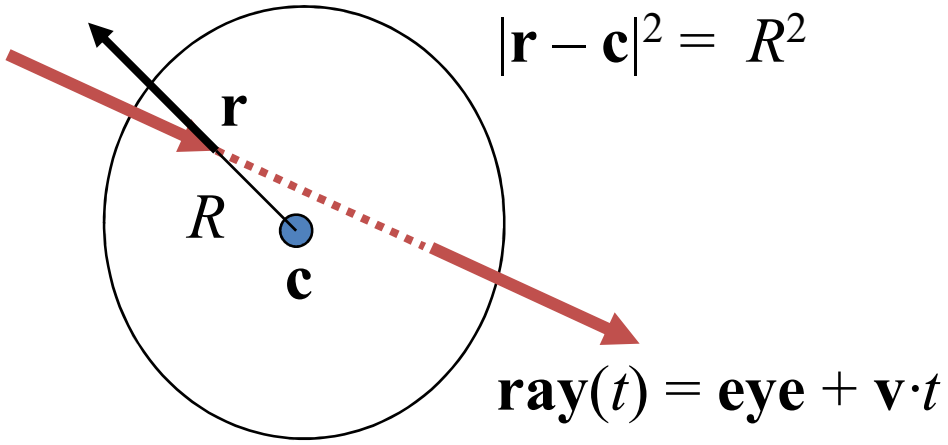
Láthatóság: trace a ray




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

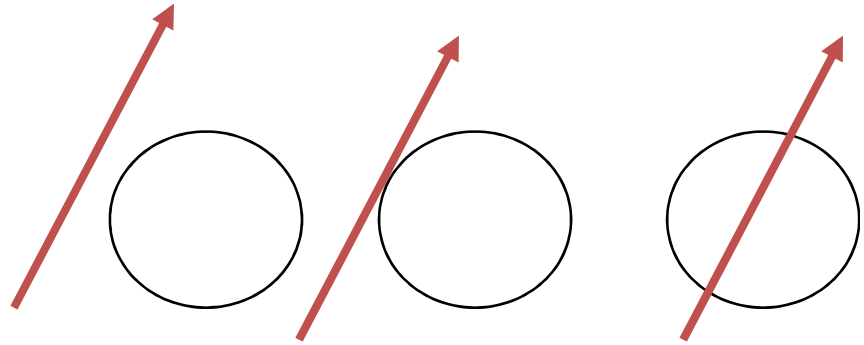
```
Hit firstIntersect(Ray ray) {
  Hit bestHit;
  for(Intersectable * obj : objects) {
    Hit hit = obj->intersect(ray); // hit.t < 0 if no intersection
    if(hit.t > 0 && (bestHit.t < 0 || hit.t < bestHit.t))
      bestHit = hit;
  }
  return bestHit;
}
```

Metszéspont számítás gömbbel



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

$$\mathbf{ray}(t) = \mathbf{eye} + \mathbf{v} \cdot t$$



Nincs gyök 1 gyök 2 gyök

$$|\mathbf{ray}(t) - \mathbf{c}|^2 = (\mathbf{ray}(t) - \mathbf{c}) \bullet (\mathbf{ray}(t) - \mathbf{c}) = R^2$$

$$(\mathbf{v} \bullet \mathbf{v})t^2 + 2((\mathbf{eye} - \mathbf{c}) \bullet \mathbf{v})t + ((\mathbf{eye} - \mathbf{c}) \bullet (\mathbf{eye} - \mathbf{c})) - R^2 = 0$$

Wanted: a pozitív megoldások közül a kisebb

Felületi normális: $(\mathbf{ray}(t) - \mathbf{c})/R$

Sphere as Intersectable

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

class Sphere : public Intersectable {
  vec3 center;
  float radius;
public:
  Hit intersect(const Ray& ray) {…}
};
```

Implicit felületek

- A felület pontjai: $f(x,y,z) = 0$ vagy $f(\mathbf{r}) = 0$
- A sugár pontjai: $\mathbf{ray}(t) = \mathbf{eye} + \mathbf{v} \cdot t$
- A metszéspont: $f(\ \mathbf{ray}(t)\ ) = 0$,
  - 1 ismeretlenes, ált. nemlineáris egyenlet: t^*
  - $(x^*,y^*,z^*) = \mathbf{eye} + \mathbf{v} \cdot t^*$
- Normálvektor = $\operatorname{grad} f \Big|_{x^*,y^*,z^*}$
  - $0 = f(x,y,z) = f(x^* + (x-x^*), y^* + (y-y^*), z^* + (z-z^*)) \approx$
    $f(x^*,y^*,z^*) + \frac{\partial f}{\partial x}(x-x^*) + \frac{\partial f}{\partial y}(y-y^*) + \frac{\partial f}{\partial z}(z-z^*)$

Az érintősík egyenlete:

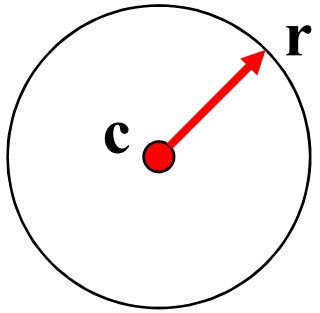
$$\left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}\right) \bullet (x-x^*, y-y^*, z-z^*) = 0$$

Konkrét példa: gömb normálvektora

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

$$|\mathbf{r} - \mathbf{c}|^2 - R^2 = 0$$

$$f(\mathbf{r}) = 0$$



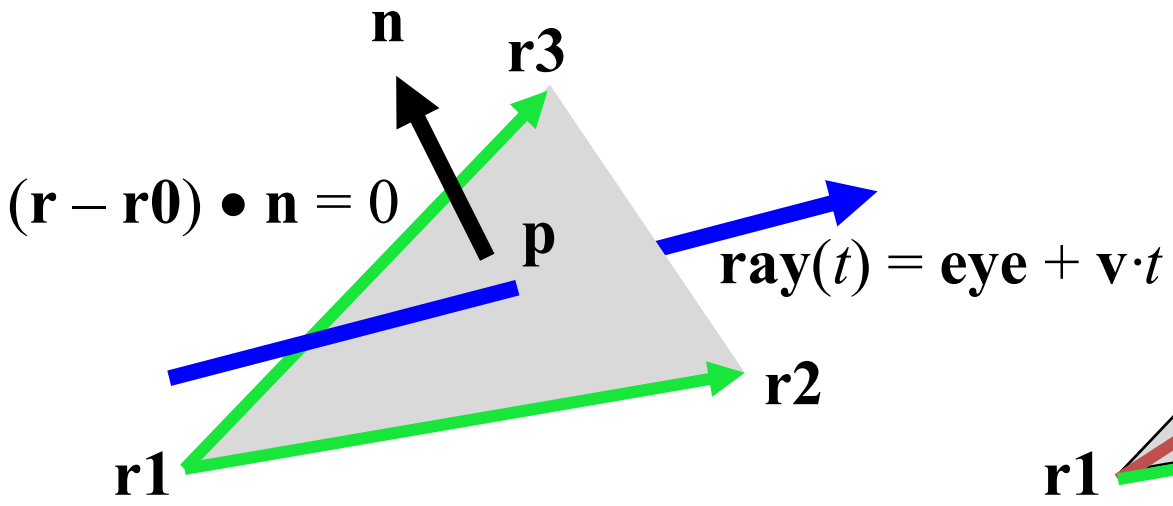
$$f(x,y,z) = (x-c_x)^2 + (y-c_y)^2 + (z-c_z)^2 - R^2 = 0$$

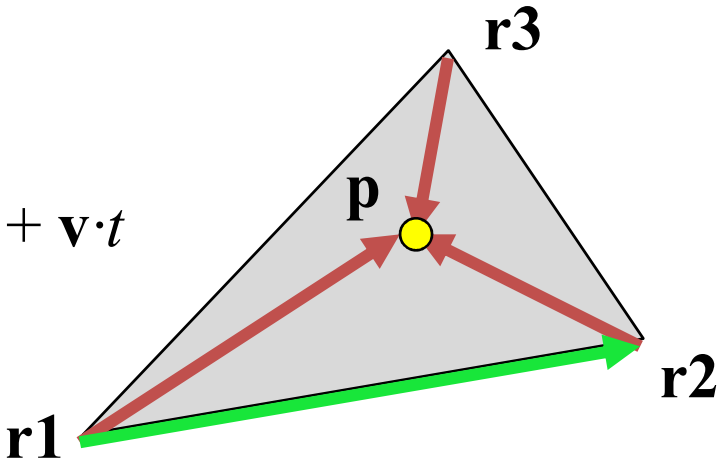
$$\frac{\partial f}{\partial x} = 2(x - c_x) + 0 + 0 - 0$$

$$\frac{\partial f}{\partial x} = 2(x - c_x) \qquad \frac{\partial f}{\partial y} = 2(y - c_y) \qquad \frac{\partial f}{\partial z} = 2(z - c_z)$$

$$\text{grad}\, f(x,y,z) = 2\,(x-c_x\,,\, y-c_y\,,\, z-c_z)$$

Háromszög





1. Síkmetszés: $(\mathbf{ray}(t) - \mathbf{r1}) \bullet \mathbf{n} = 0, \quad t > 0$

   normál: $\mathbf{n} = (\mathbf{r2} - \mathbf{r1}) \times (\mathbf{r3} - \mathbf{r1})$

$$t = \frac{(\mathbf{r1} - \mathbf{eye}) \bullet \mathbf{n}}{\mathbf{v} \bullet \mathbf{n}}$$

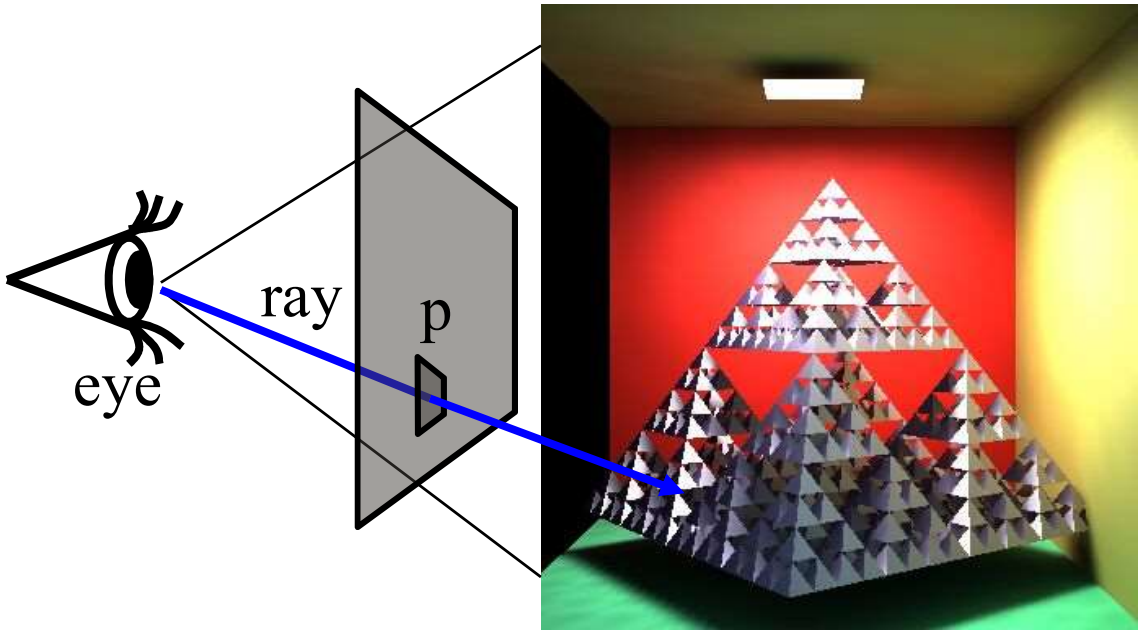
2. A metszéspont a háromszögön belül van-e?

$$((\mathbf{r2} - \mathbf{r1}) \times (\mathbf{p} - \mathbf{r1})) \bullet \mathbf{n} > 0$$
$$((\mathbf{r3} - \mathbf{r2}) \times (\mathbf{p} - \mathbf{r2})) \bullet \mathbf{n} > 0$$
$$((\mathbf{r1} - \mathbf{r3}) \times (\mathbf{p} - \mathbf{r3})) \bullet \mathbf{n} > 0$$

Felületi normális: **n**
vagy árnyaló normálok
(shading normals)

Ray tracing: Render

Virtual world: eye+window



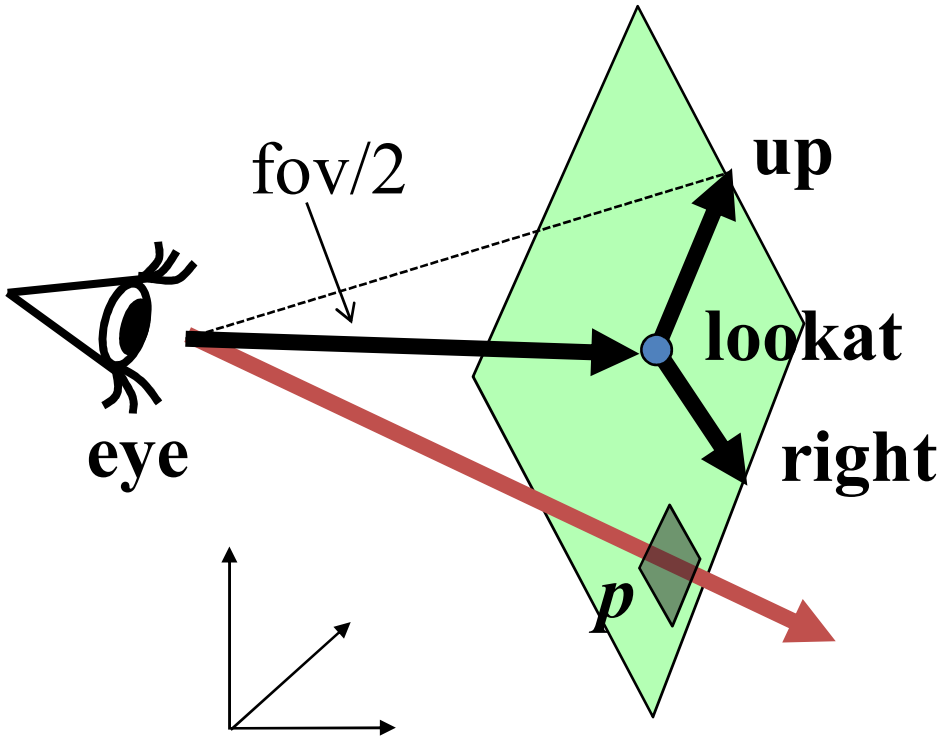
Real world: user+screen

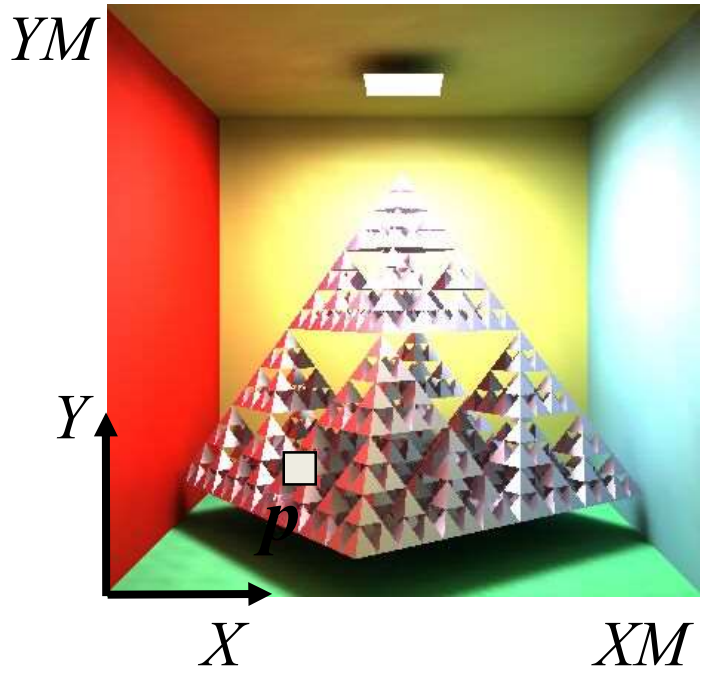
p color

Render( )

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

Kamera: GetRay



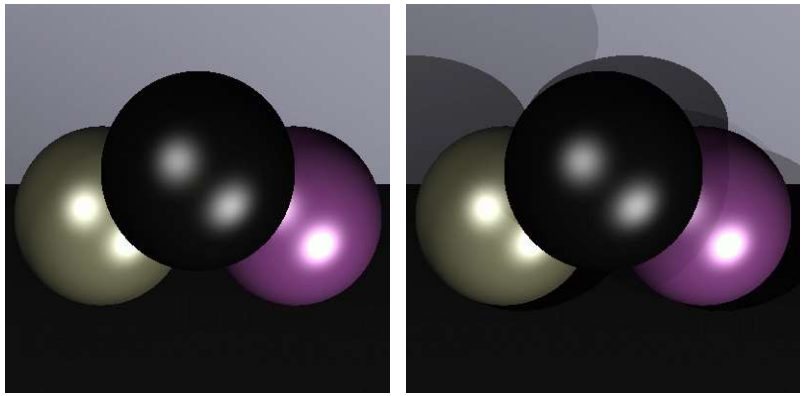


Normalizált eszköz koordináták

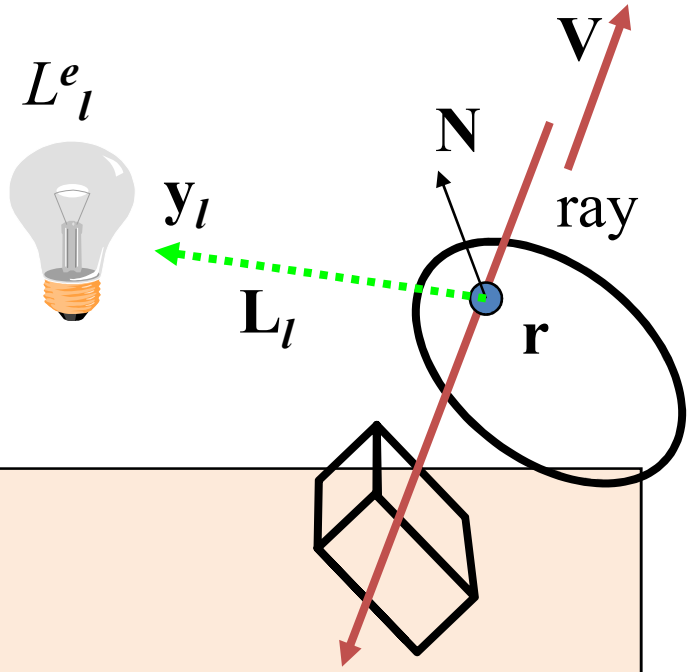
$\boldsymbol{p} = \mathbf{lookat} + \alpha \cdot \mathbf{right} + \beta \cdot \mathbf{up}, \qquad \alpha, \beta \text{ in } [-1,1]$

$= \mathbf{lookat} + (2X/XM - 1) \cdot \mathbf{right} + (2Y/YM - 1) \cdot \mathbf{up}$

Ray dir =p – eye



trace

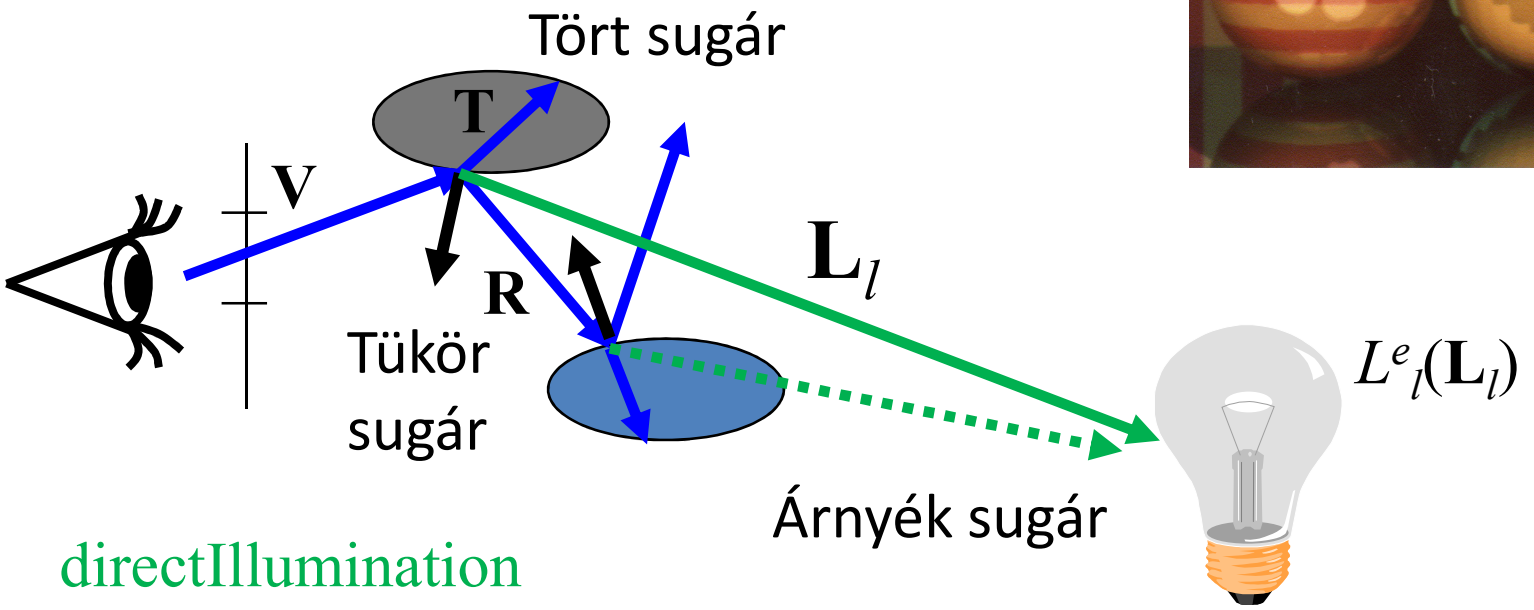



```
vec3 trace(Ray ray) {

  Hit hit = firstIntersect(ray);
  if(hit.t < 0) return La; // nothing
  vec3 outRadiance = hit.material->ka * La;
  for(each light source l){
    Ray shadowRay(r + Nε, Ll);
    Hit shadowHit = firstIntersect(shadowRay);
    if(shadowHit.t < 0 || shadowHit.t > |r - yl| )
        outRadiance += hit.material->shade(N, V, Ll, Le_l);
  }
  return outRadiance;
}
```

Shadow

Rekurzív sugárkövetés



directIllumination

$$L(\mathbf{V}) \approx \sum_l L^e{}_l(\mathbf{L}_l) * (k_d \cdot (\mathbf{L}_l \bullet \mathbf{N})^+ + k_s \cdot ((\mathbf{H}_l \bullet \mathbf{N})^+)^{shine}) + k_a * L_a$$

$$+ F(\mathbf{V} \bullet \mathbf{N}) * L^{\text{in}}(\mathbf{R}) + (1 - F(\mathbf{V} \bullet \mathbf{N})) * L^{\text{in}}(\mathbf{T})$$

Fresnel

Tükör irányból érkező fény

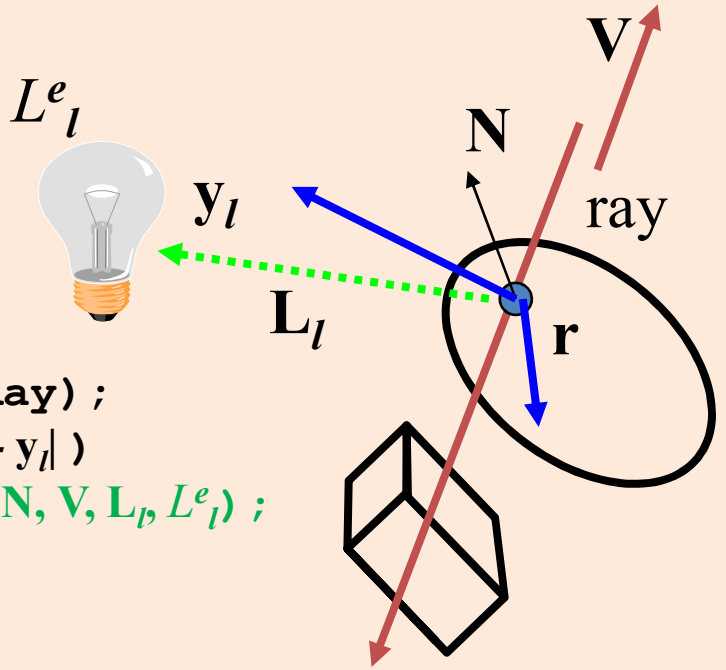
1-Fresnel

Törési irányból érkező fény

trace

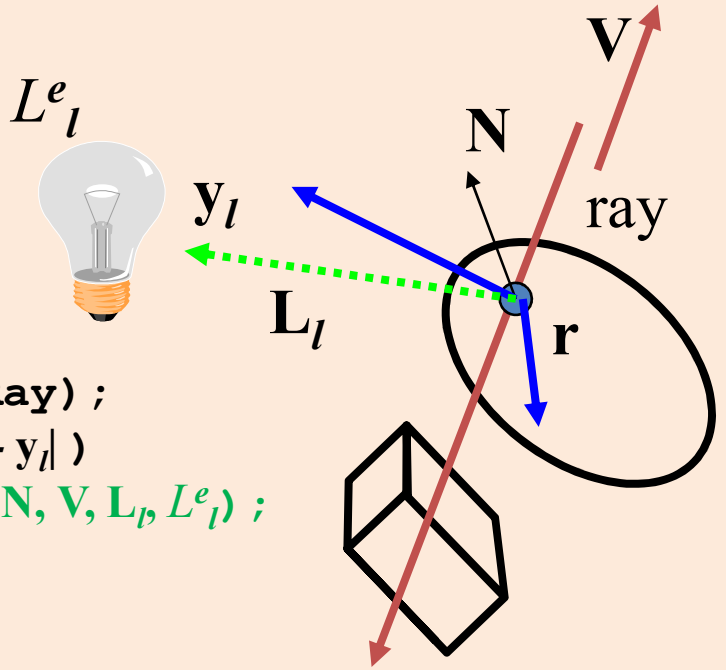
```
vec3 trace(Ray ray) {

  Hit hit = firstIntersect(ray);
  if(hit.t < 0) return La; // nothing
  vec3 outRadiance = hit.material->ka * La;
  for(each light source l){
    Ray shadowRay(r + Nε sign(N·V), Ll);
    Hit shadowHit = firstIntersect(shadowRay);
    if(shadowHit.t < 0 || shadowHit.t > |r - yl| )
       outRadiance += hit.material->shade(N, V, Ll, Le l);
  }
  if(hit.material->reflective){
    vec3 reflectionDir = reflect(-V,N);
    Ray reflectedRay(r + Nε sign(N·V), reflectionDir);
    outRadiance += trace(reflectedRay)*F(V,N);
  }
  if(hit.material->refractive) {
    vec3 refractionDir = refract(-V,N);
    Ray refractedRay(r - Nε sign(N·V), refractionDir);
    outRadiance += trace(refractedRay)*(vec3(1,1,1)-F(V,N));
  }
  return outRadiance;
}
```

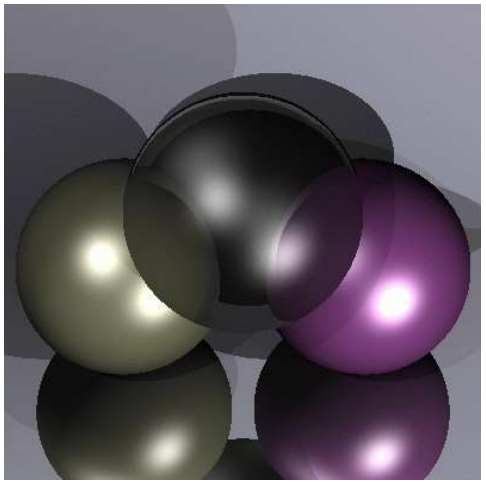


trace

```
vec3 trace(Ray ray, int depth) {
  if (depth > maxdepth) return La;
  Hit hit = firstIntersect(ray);
  if(hit.t < 0) return La; // nothing
  vec3 outRadiance = hit.material->ka * La;
  for(each light source l){
    Ray shadowRay(r + Nε sign(N·V), Ll);
    Hit shadowHit = firstIntersect(shadowRay);
    if(shadowHit.t < 0 || shadowHit.t > |r - yl| )
       outRadiance += hit.material->shade(N, V, Ll, Le l);
  }
  if(hit.material->reflective){
    vec3 reflectionDir = reflect(-V,N);
    Ray reflectedRay(r + Nε sign(N·V), reflectionDir);
    outRadiance += trace(reflectedRay,depth+1)*F(V,N);
  }
  if(hit.material->refractive) {
    vec3 refractionDir = refract(-V,N);
    Ray refractedRay(r - Nε sign(N·V), refractionDir);
    outRadiance += trace(refractedRay,depth+1)*(vec3(1,1,1)-F(V,N));
  }
  return outRadiance;
}
```



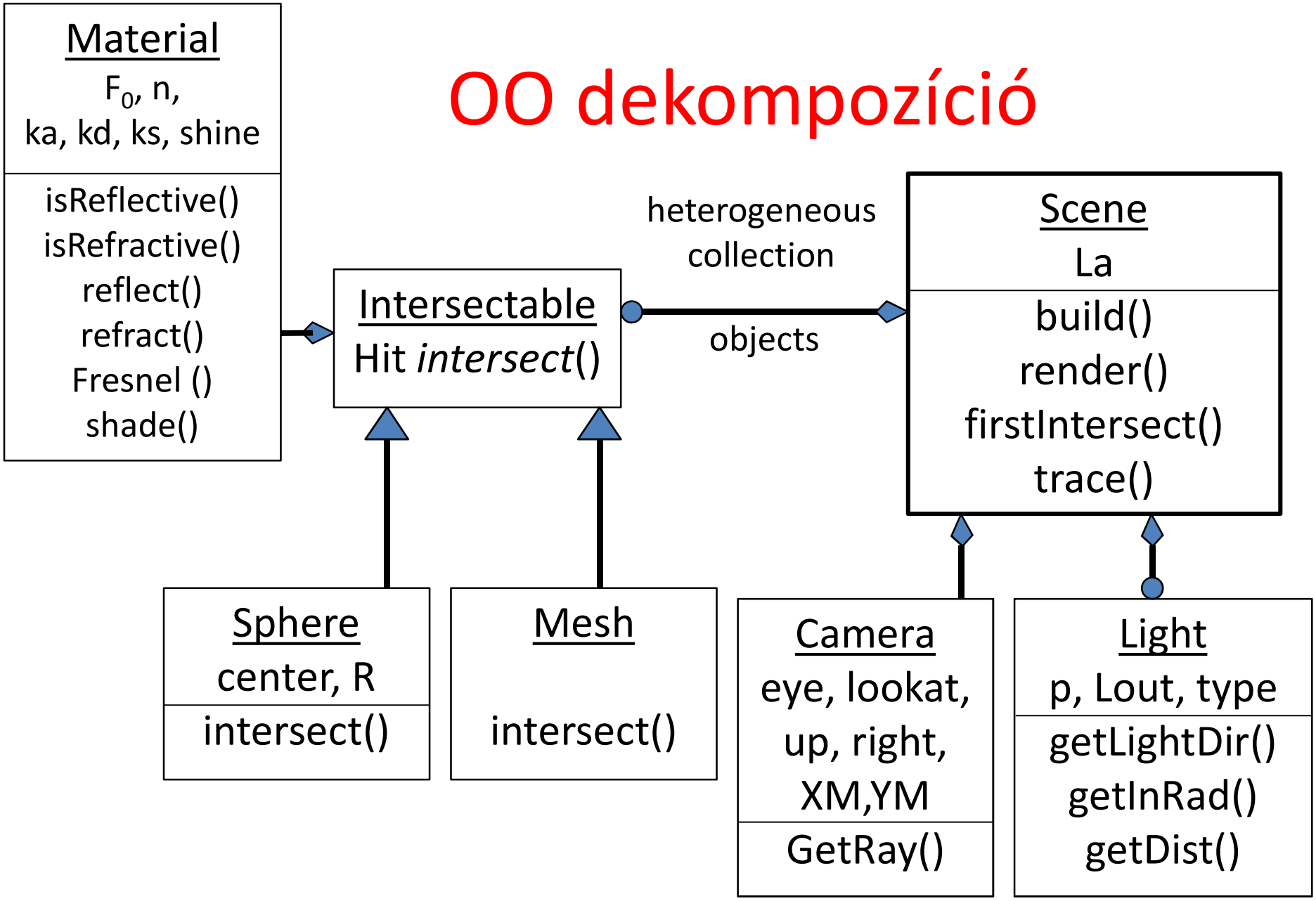
Heckbert Palika házija a névjegyén



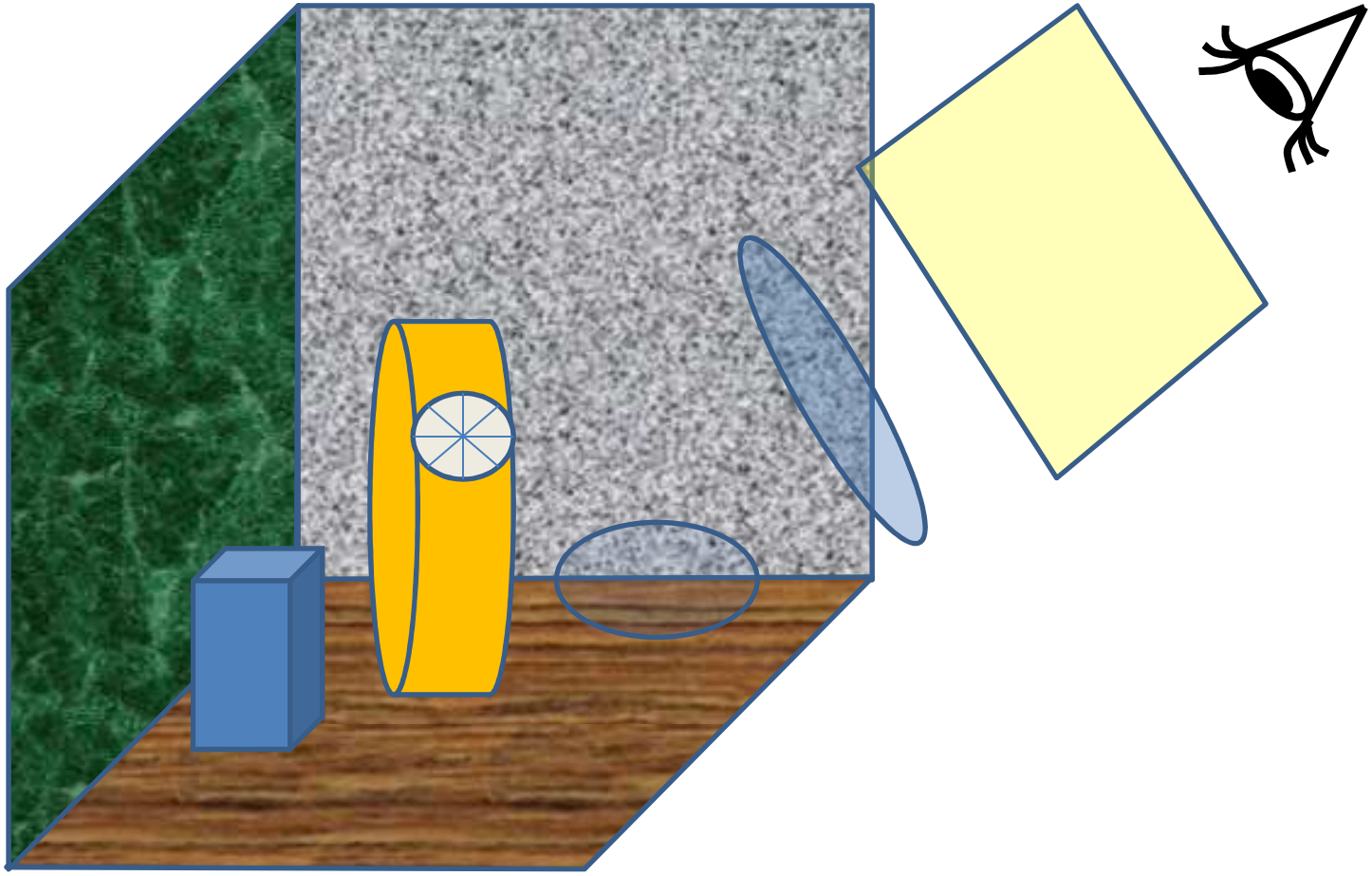
```
typedef struct{double x,y,z}vec;vec U,black,amb={.02,.02,.02}; struct sphere{ vec cen,color;double rad,kd,ks,kt,kl,ir}*s,
*best,sph[]={0.,6.,.5,1.,1.,1.,.9, .05,.2,.85,0.,1.7,-1.,8.,-.5,1.,.5,.2,1.,.7,.3,0.,.05,1.2,1.,8.,-.5,.1,.8,.8, 1.,.3,.7,0.,0.,1.2,3.,-6.,15.,1.,
.8,1.,7.,0.,0.,0.,.6,1.5,-3.,-3.,12.,.8,1., 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!*/
```

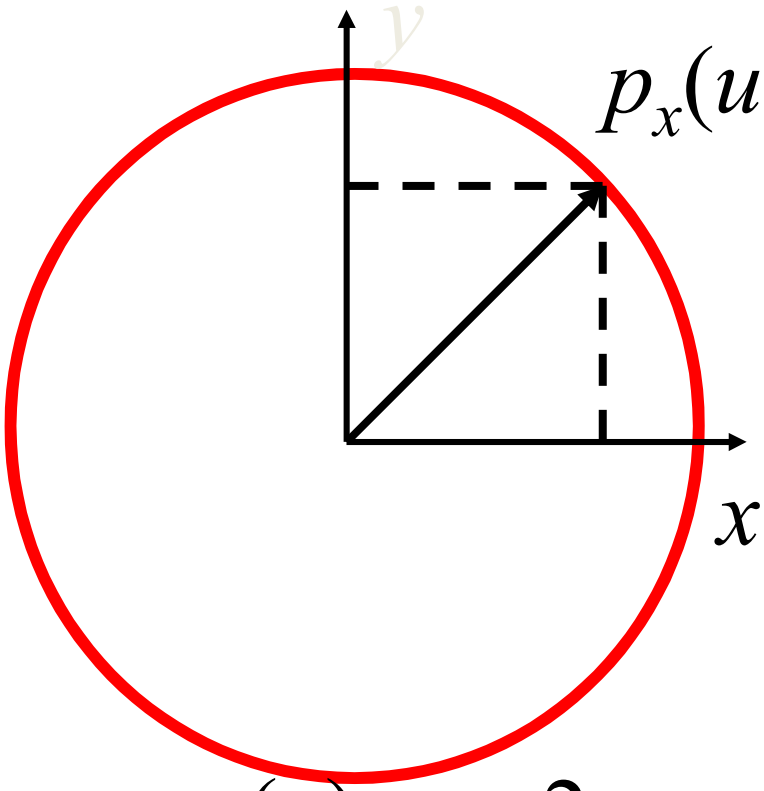
OO dekompozíció



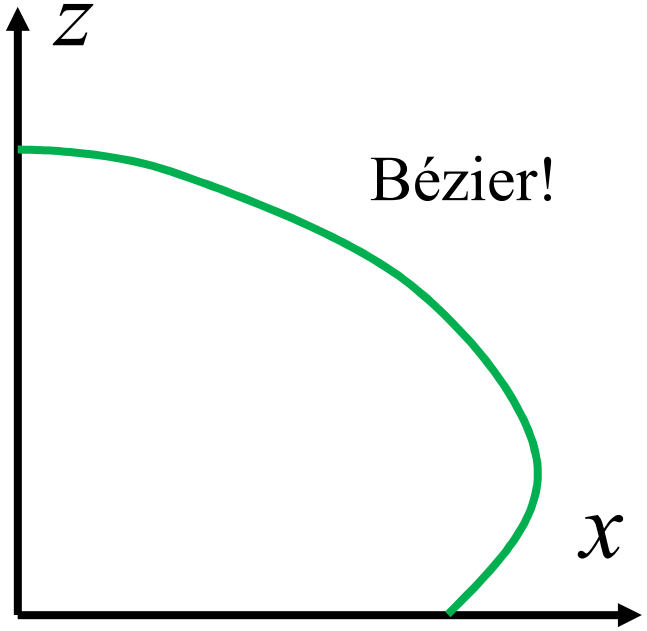
2. házi



Forgatás



$$x = p_x(u)$$
$$z = p_z(u)$$



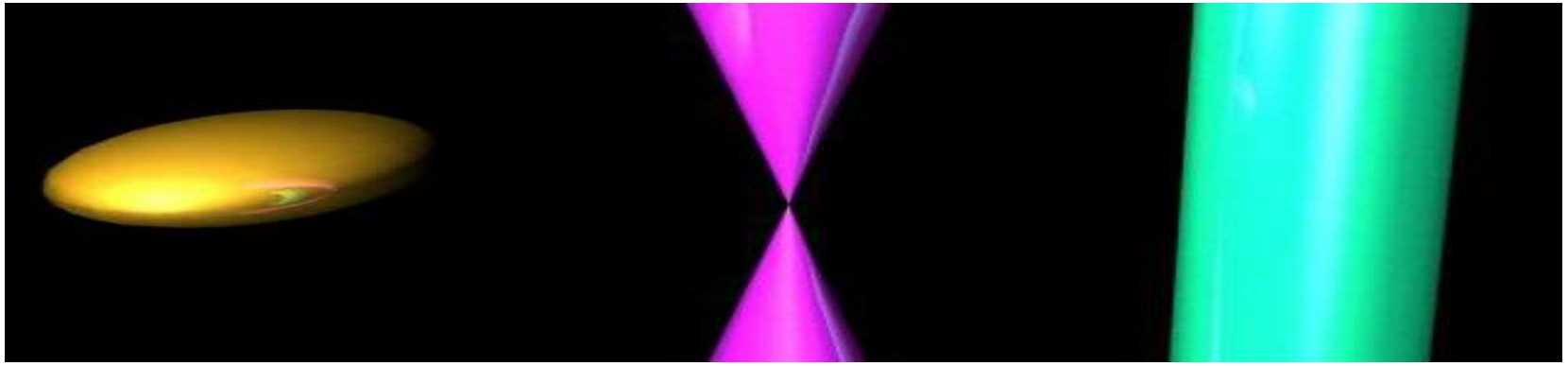
$$x = p_x(u) \cdot \cos 2\pi v$$
$$y = p_x(u) \cdot \sin 2\pi v$$
$$z = p_z(u)$$

Kvadratikus felületek

$$[x,y,z,1]\mathbf{A}\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = 0 \Rightarrow$$

Másodfokú egyenlet

$$\mathbf{[eye} + \mathbf{v}\cdot t,1]\cdot\mathbf{A}\cdot\mathbf{[eye} + \mathbf{v}\cdot t,1]^T =$$
$$(\mathbf{[eye},1]+[\mathbf{v},0]\cdot t)\cdot\mathbf{A}\cdot(\mathbf{[eye},1]^T+[\mathbf{v},0]^T\cdot t) =$$
$$[\mathbf{v},0]\cdot\mathbf{A}\cdot[\mathbf{v},0]^T\cdot t^2 +$$
$$(\mathbf{[eye},1]\cdot\mathbf{A}\cdot[\mathbf{v},0]^T+[\mathbf{v},0]\cdot\mathbf{A}\cdot\mathbf{[eye},1]^T)\cdot t +$$
$$\mathbf{[eye},1]\cdot\mathbf{A}\cdot\mathbf{[eye},1]^T = 0$$



Ellipszoid

$$\frac{x^2}{a^2}+\frac{y^2}{b^2}+\frac{z^2}{c^2}-1=0$$

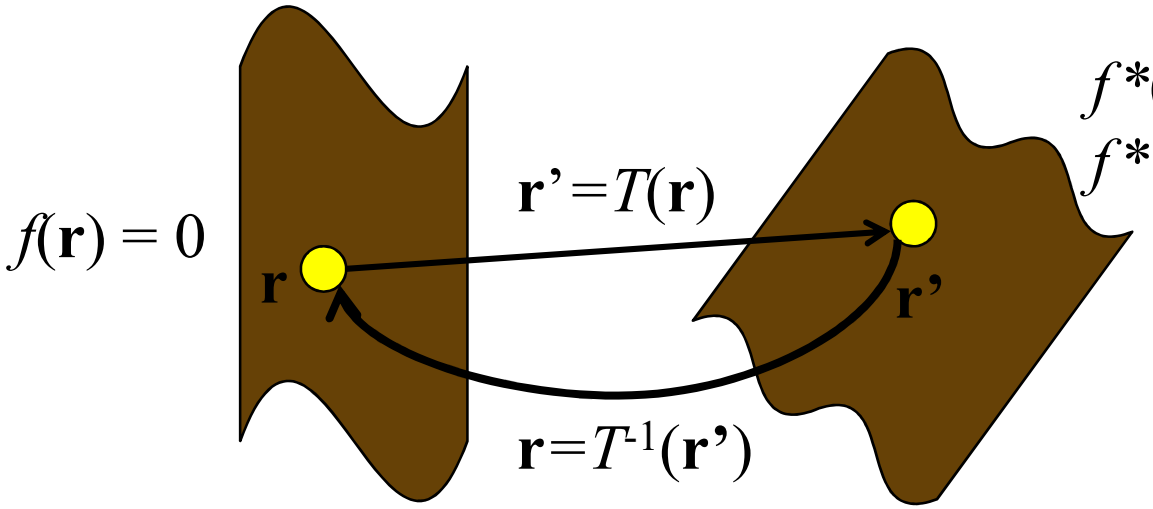
Kúp

$$\frac{x^2}{a^2}+\frac{y^2}{a^2}-z^2=0$$

Henger

$$\frac{x^2}{R^2}+\frac{y^2}{R^2}-1=0$$

Implicit felületek transzformációja



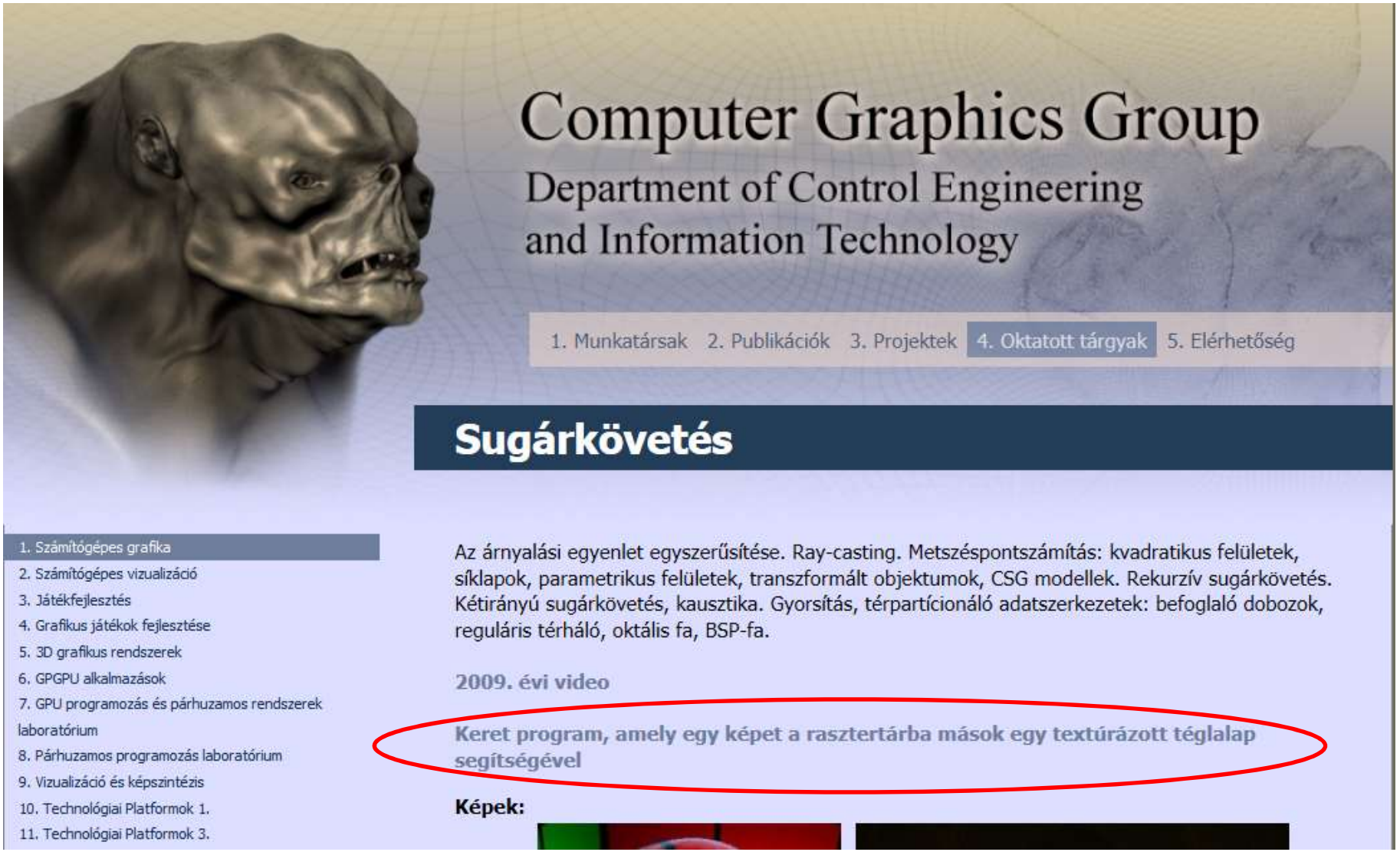
$$f^*(\mathbf{r}') = f(T^{-1}(\mathbf{r}')) = 0$$

Kvadratikus felületek és homogén lineáris transzformációk:

$$[x,y,z,1]\cdot\mathbf{A}\cdot[x,y,z,1]^T = 0,$$
$$[x',y',z',1] = [x,y,z,1]\cdot\mathbf{T} \quad \Rightarrow \quad [x,y,z,1] = [x',y',z',1]\cdot\mathbf{T}^{-1}$$
$$[x',y',z',1]\cdot\mathbf{T}^{-1}\cdot\mathbf{A}\cdot([x',y',z',1]\cdot\mathbf{T}^{-1})^T =$$
$$[x',y',z',1]\cdot\mathbf{T}^{-1}\cdot\mathbf{A}\cdot(\mathbf{T}^{-1})^T\cdot[x',y',z',1]^T = 0$$

$$\mathbf{A}^* = \mathbf{T}^{-1}\cdot\mathbf{A}\cdot(\mathbf{T}^{-1})^T$$

CPU-n számított kép másolása a rasztertárba



Valami ilyesmi, de ez még nem tökéletes