Sugárkövetés a GPU-n

Fény-felület kölcsönhatás

- Árnyalási egyenlet

$$L(\vec{x},\omega) = L^e(\vec{x},\omega) + \int_{\Omega} L(h(\vec{x},-\omega'),\omega') \cdot f_r(\omega',\vec{x},\omega) \cdot \cos\theta' d\omega'$$

  - Saját emisszió
  - Adott irányú visszaverődés

Fény-felület kölcsönhatás

- Rekurzív sugárkövetés

$$L(\vec{x},\omega) = L^e(\vec{x},\omega) + k_a \cdot L^a + \sum_l f_{ri}(\omega_l', \vec{x}, \omega) \cdot \cos\theta_l' \cdot L^{in}(\vec{x},\omega_l') +$$

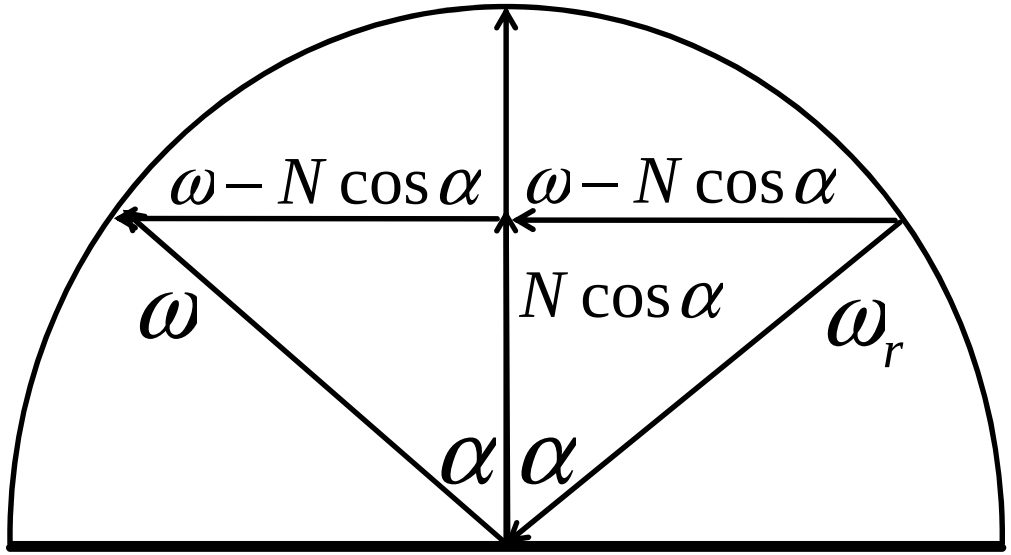
$$k_r \cdot L^{in}(\vec{x},\omega_r) + k_t \cdot L^{in}(\vec{x},\omega_t)$$

  - Saját sugárzás és ambiens visszaverődés
  - Tükörirányból jövő radiancia
  - Törési irányból jövő radiancia
  - Fényforrás láthatósága

Tükörirány számítása

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

$$\omega_r = \omega - 2\cos\alpha \cdot \vec{N}$$

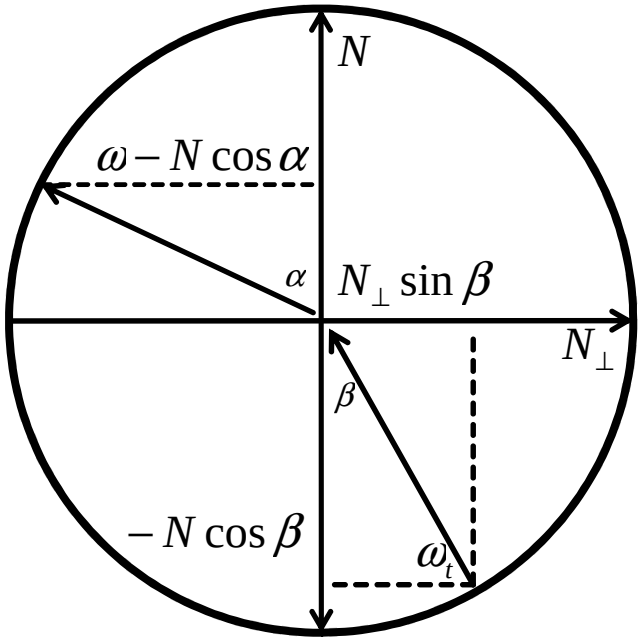


Törési irány számítása

$$-\omega_t = -\cos\beta \cdot \vec{N} + \sin\beta \cdot \vec{N}_\perp$$

$$\vec{N}_\perp = \frac{\cos\alpha \cdot \vec{N} - \omega}{\left|\cos\alpha \cdot \vec{N} - \omega\right|} = \frac{\cos\alpha \cdot \vec{N} - \omega}{\sin\alpha}$$

$$\frac{\sin\alpha}{\sin\beta} = n$$



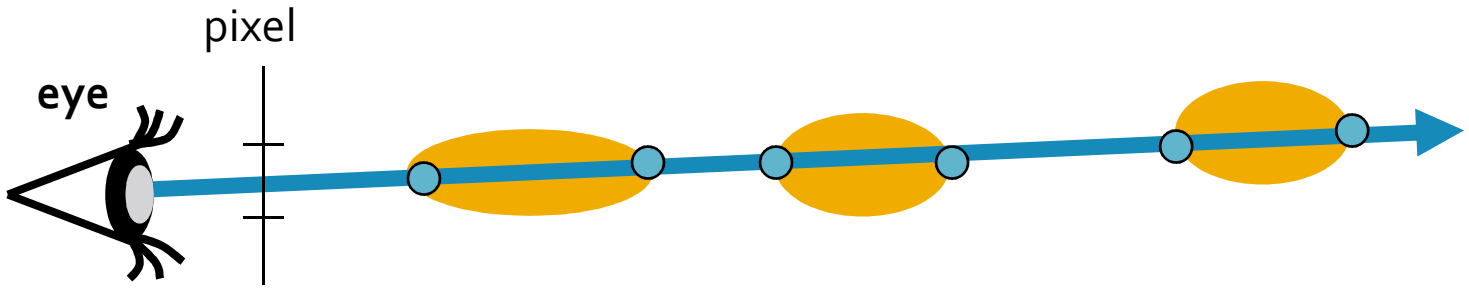
Törési irány számítása

$$\omega_t = \cos\beta \cdot \vec{N} - \frac{\sin\beta}{\sin\alpha}(\cos\alpha \cdot \vec{N} - \omega) =$$

$$= \frac{\omega}{n} - \vec{N}\left(\frac{\cos\alpha}{n} - \cos\beta\right) = \frac{\omega}{n} - \vec{N}\left(\frac{\cos\alpha}{n} - \sqrt{1-\sin^2\beta}\right) =$$

$$= \frac{\omega}{n} - \vec{N}\left(\frac{\cos\alpha}{n} - \sqrt{1-\frac{(1-\cos^2\alpha)}{n}}\right)$$

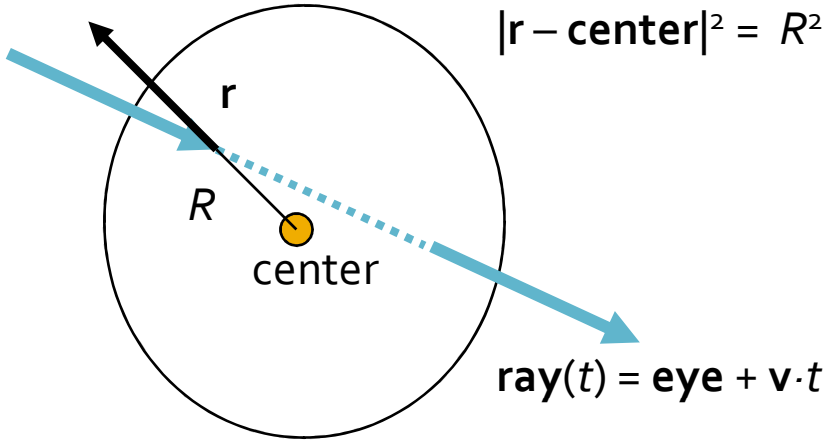
Láthatóság vizsgálat



ray(t) = **eye** + **v**·t, $t > 0$

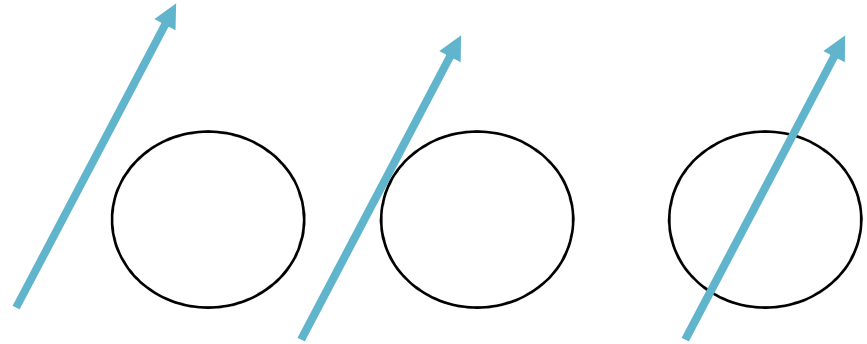
```
FirstIntersect(ray ⇨ t, iobject, x)
   t = FLT_MAX;
   FOR each object
         tnew = Intersect( ray, object );  // < 0 ha nincs metszés
         IF (tnew > 0 && tnew < t )  { t = tnew; iobject = object; }
   ENDFOR
   IF (t < FLT_MAX)  { x = eye + v·t;  RETURN (t, iobject, x); }
   RETURN „no intersection”
END
```

Metszéspont számítás gömbbel



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

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



Nincs gyök 1 gyök 2 gyök

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

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

Wanted: a pozitív megoldások közül a kisebb

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

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 = 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:

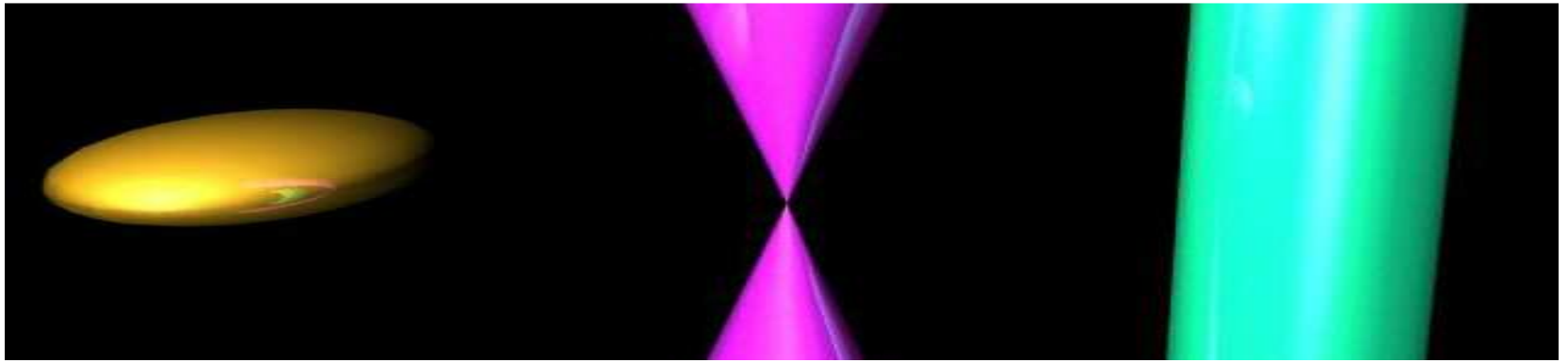
$$\left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}\right)\bullet(x-x^*,\ y-y^*,\ z-z^*) = 0$$

Kvadratikus felületek

Kvadratikus felület:

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

Másodfokú egyenlet



Ellipszoid

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

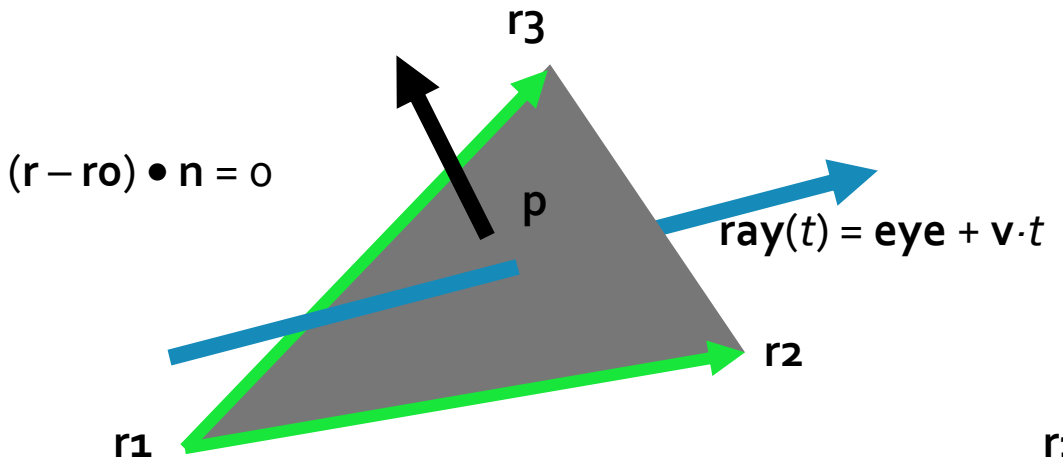
Végtelen kúp

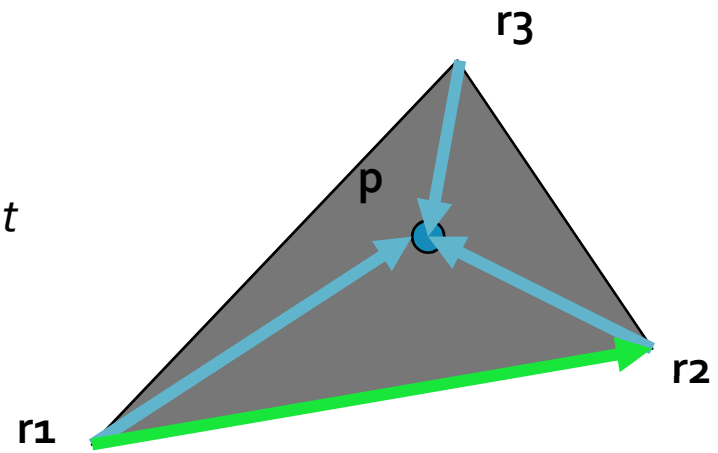
$$\frac{x^2}{a^2}+\frac{y^2}{b^2}-z^2=0$$

Végtelen henger

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

Hárömszög





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?

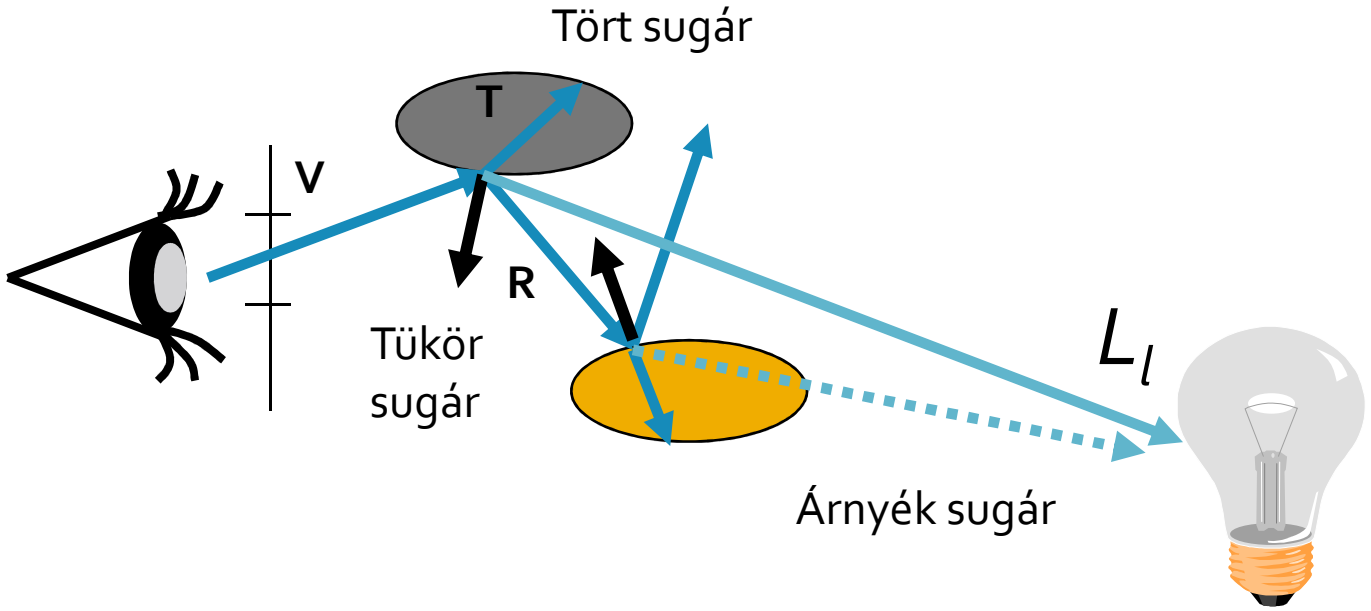
$((\mathbf{r2} - \mathbf{r1}) \times (\mathbf{p} - \mathbf{r1})) \bullet \mathbf{n} > 0$

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

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

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

Rekurzív sugárkövetés



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

$$+ k_r * L^{\text{in}}(\mathbf{R}) \quad + \quad k_t * L^{\text{in}}(\mathbf{T})$$

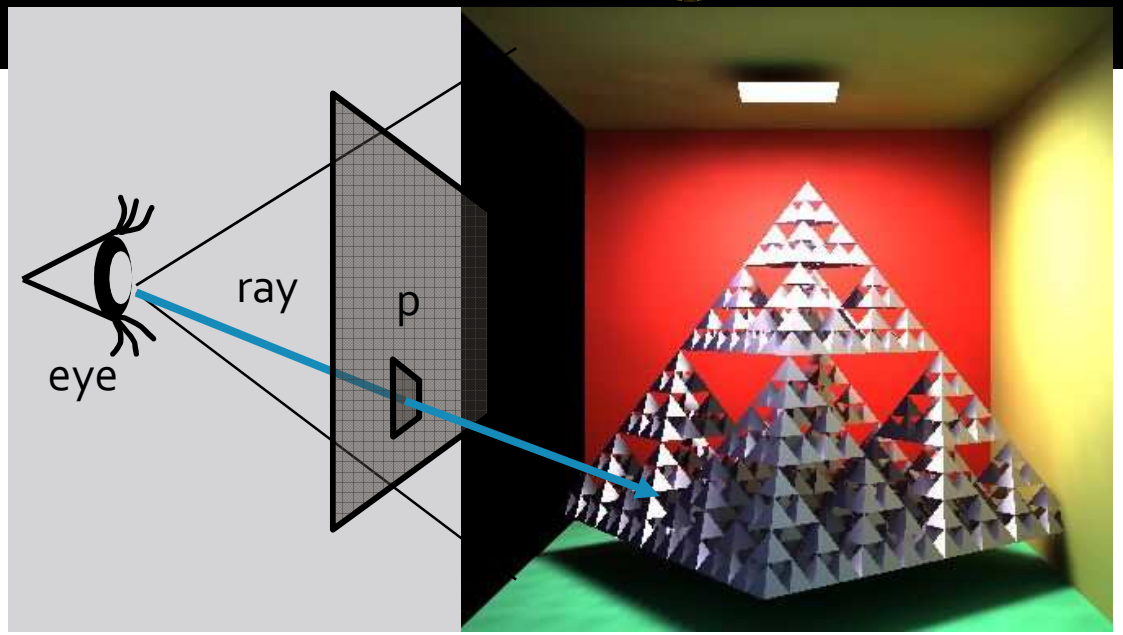
Fresnel

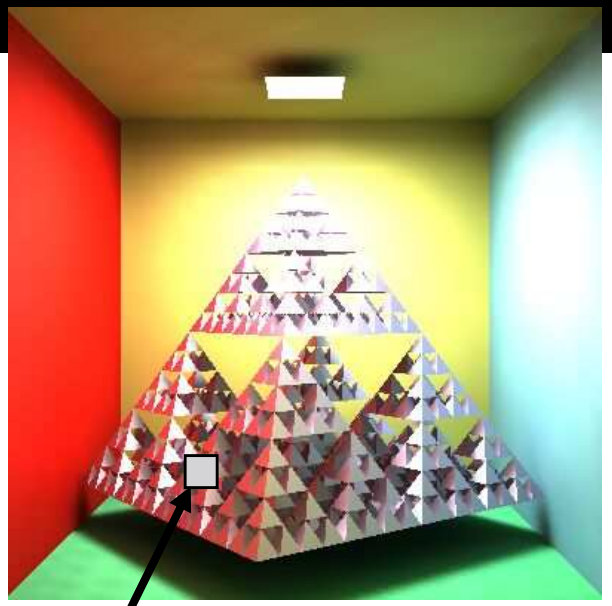
Tükör irányból érkező fény

1-Fresnel

Törési irányból érkező fény

Rekurzív sugárkövetés





p color

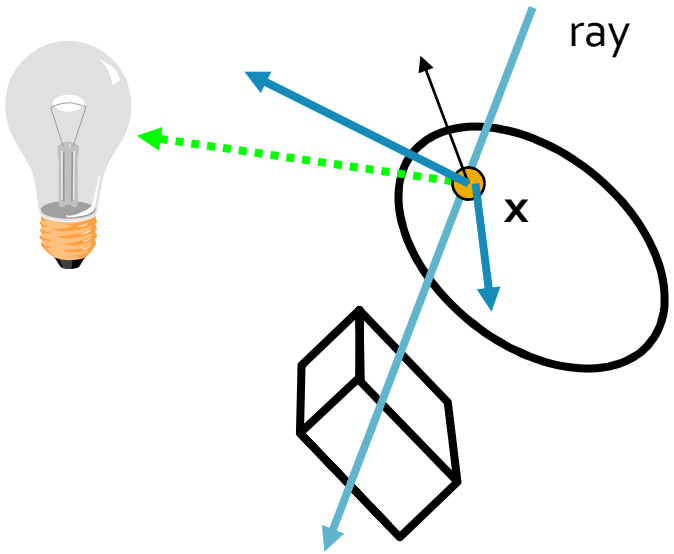
Render( )

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

Rekurzív sugárkövetés

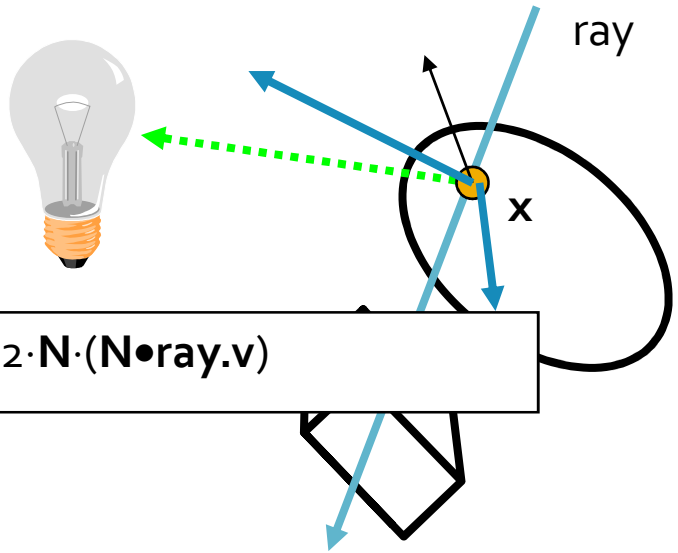
Color Trace(ray)

```
IF (FirstIntersect(ray ⇨ obj, x) < o)
        RETURN La
ENDIF
color = DirectLightSource(x, ray.v, obj)
IF ( obj.mirror )
        ReflectDir( ray, reflected ray)
        color += obj.kr * Trace( reflected ray )
ENDIF
IF ( obj.refractive && RefractDir( ray, refracted ray ) )
        color += obj.kt * Trace( refracted ray )
ENDIF
RETURN color
```



Rekurzív sugárkövetés

```
Color Trace( ray, d )
    IF (d > dmax) RETURN La
    IF (FirstIntersect(ray ⇨ obj, x) < 0)
            RETURN La
    ENDIF
    color = DirectLightSource(x, ray.v, obj)
    IF ( obj.mirror )
            ReflectDir( ray, reflected ray)
            color += obj.kr * Trace( reflected ray, d+1 )
    ENDIF
    IF ( obj.refractive && RefractDir( ray, refracted ray ) )
            color += obj.kt * Trace( refracted ray, d+1 )
    ENDIF
RETURN color
```



ray.v-2·**N**·(**N**•**ray.v**)

ray.v/*n*+**N**·((**N**•**ray.v**)/*n*-√1-(1-(**N**•**ray.v**)²)/*n*²)

Rekurzív sugárkövetés

DirectLightSource(x, v, obj)

color = obj.k_a*L_a

FOR each lightsource l DO

árnyék

shadowray.eye = x; shadowray.v = light[l].pos – x;
(t,y) = FirstIntersect(shadowray);
IF (t < o || |x-y| > |x-light[l].pos|)

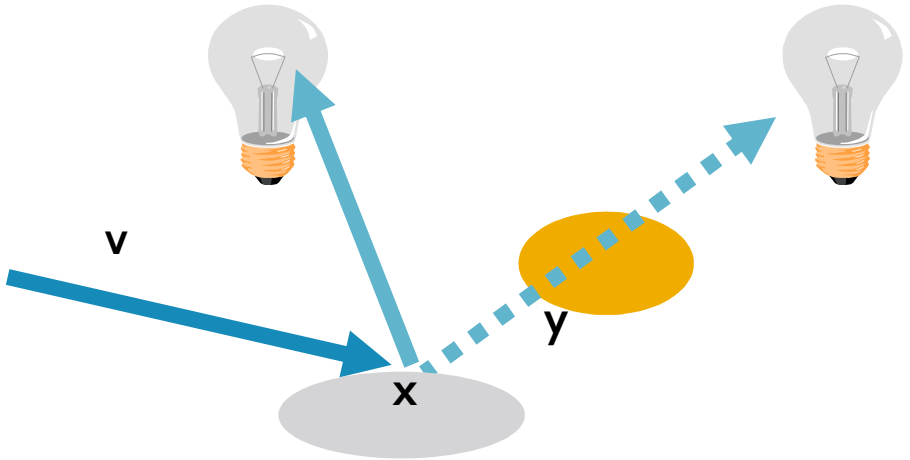
color += light[l].Intensity *

(obj.k_d ·(**L_l**•**N**)$^+$ + obj.k_s ·((**H_l**•**N**)$^+$)$^{obj.shine}$)

ENDIF

ENDFOR

RETURN color



Sugárkövetés a GPU-n

- Sugár

  $\mathbf{p} = \mathbf{o} + t\mathbf{d}$

  - $\mathbf{p}$ – pont $[x\ y\ z\ 1]$
  - $\mathbf{o}$ – sugár kezdőpont $[o_x\ \ o_y\ \ o_z\ \ 1]$
  - $\mathbf{d}$ – sugár irány $[d_x\ \ d_y\ \ d_z\ \ 0]$
  - t – sugárparaméter
- Kvadratikus felület (quadric)

  $\mathbf{p}\mathbf{A}\mathbf{p}^T = 0$

  $a\,x^2 + b\,xy + c\,xz + d\,x + e\,yx + f\,y^2 + \ldots = 0$

Sugár-quadric metszés

$$(\mathbf{o} + t\mathbf{d})\mathbf{A}(\mathbf{o} + t\mathbf{d})^T = 0$$

$$(\mathbf{oA} + t\mathbf{dA})(\mathbf{o} + t\mathbf{d})^T = 0$$

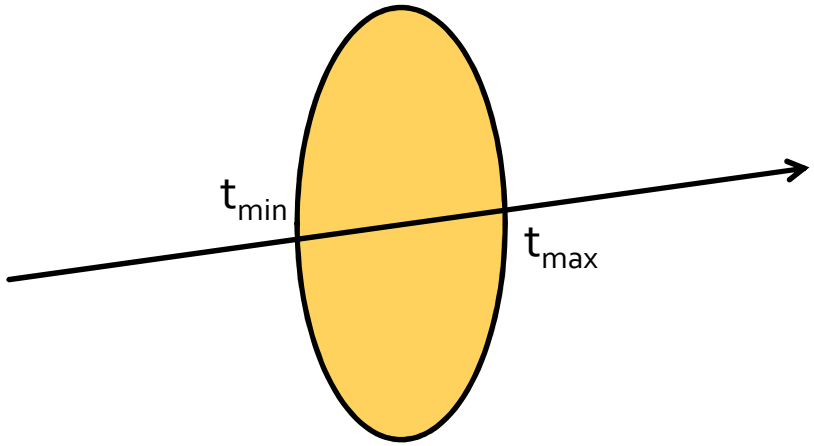
$$\mathbf{oAo}^T + \mathbf{oAtd}^T + t\mathbf{dAo}^T + t\mathbf{dA}t\mathbf{d}^T = 0$$

$$\mathbf{dAd}^T t^2 + (\mathbf{oAd}^T + \mathbf{dAo}^T)\, t + \mathbf{oAo}^T = 0$$

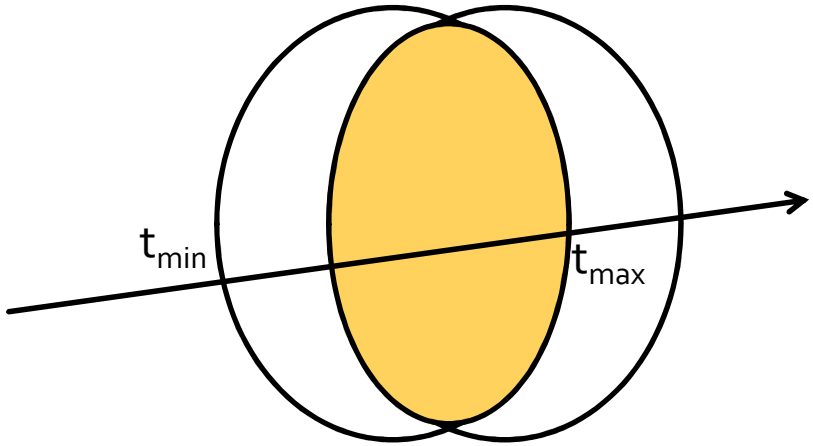
másodfokú egyenlet t-re

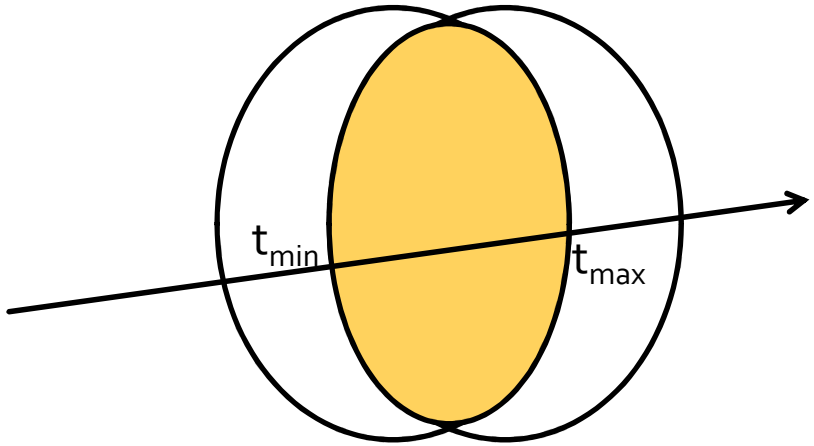
Sugár-quadric metszés

- Kvadratikus felület



- Felületek metszete





Sugár-quadric metszés

```
vec2 intersectQuadric( mat4 A, vec4 o, vec4 d, vec2 tMinMax, out bvec2 visible)
```

- visszaadja a két metszéspont t-jét sorrendben
- ha kilóg a tMinMax.x és tMinMax.y közötti tartományból akkor levágja
  - visible true, ha nem kellett levágni

Sugár-quadric metszés

```
vec2 intersectQuadric(mat4 A, vec4 o, vec4 d, vec2 tMinMax, out bvec2 visible)
 float a = dot(d * A,  d);
 float b = dot(d * A, o) + dot(o * A, d);
 float c = dot(o * A, o);
 float det = b*b - 4 * a * c;
 if(det < 0)
        return tMinMax.yx;
 vec2 t = (-b.xx + sqrt(det) * vec2(1, -1)) / (2 * a);
 if(t.x > t.y) t = t.yx;
        visible = bvec2(true, true);
 if(t.x < tMinMax.x) { t.x = tMinMax.x; visible.x = false; }
 if(t.y > tMinMax.y) { t.y = tMinMax.y; visible.y = false; }
 return t;
}
```

Quadric normál vektor

- **Kvadratikus felület**
  - **pAp**^{T} = 0 – Izofelület
  - Normálvektor = Gradiens

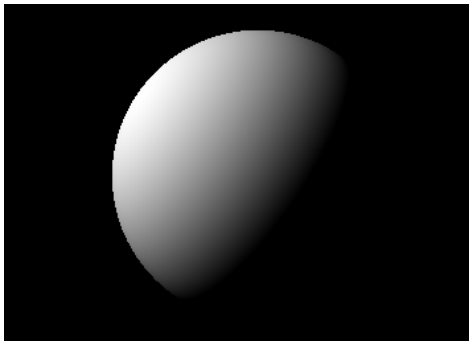
$$pAp^T = 0$$

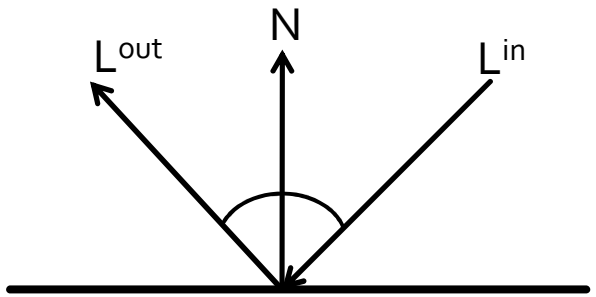
$$\frac{\partial}{\partial x}\left([x,y,z,1]\cdot \underline{\underline{A}} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \right) = [1,0,0,0]\cdot \underline{\underline{A}} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} + [x,y,z,1]\cdot \underline{\underline{A}} \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\frac{\partial}{\partial x} = \vec{p}\,\underline{\underline{A}}\Big|_x + \underline{\underline{A}}\,\vec{p}\Big|_x \qquad \boxed{\vec{p}\,\underline{\underline{A}} + \underline{\underline{A}}\,\vec{p} = \left[\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}, 0\right]}$$

Árnyalás

```
vec3 trace(inout vec4 o, inout vec4 d, inout float contrib) {
  bvec2 visible;
  vec2 t = intersectQuadric(quadrics[0], o, d,
                            vec2(0, 10000), visible);
  if(t.x > t.y)
    return vec3(0, 0, 0);
  vec4 p = o + d * t.x;
  vec3 normal = normalize((p * quadrics[0] +
                           quadrics[0] * p).xyz);
  vec3 lightDir = normalize(vec3(-1, 1, 1));
  return dot(normal, lightDir).xxx;
}
```





Színtérmodell

- Objektumok
  - Quadricok metszetei
  - Feltesszük hogy a két metszéspont között van mindig a belseje (kúpra, hiberboloidra ez hibás eredményre fog vezetni)
- Anyagtulajdonságok minden quadricra

Tükröződés

- Anyagtulajdonság alapján illumináció
- Sugárparaméterek módosítása

```
vec3 result = contrib * texProc(p.xyz, materials[quadric].xyz) *
                        (clamp(dot(normal, lightDir), 0, 1) + 0.3);

float kr = materials[quadric].w;
if(kr >= 0) {
   contrib *= kr;
   o = p;
   o.xyz += normal * 0.01;
   d = vec4(reflect(d.xyz, normal), 0);
}
```

Törő irány számítása

- Anyagtulajdonság alapján illumináció
- Sugárparaméterek módosítása

```
if(kr < 0){
  vec3 rdir = refract(d.xyz, normal, -(backFacing?(kr):(1/kr)));
  if(dot(rdir, rdir) > 0.001) {
   contrib *= 0.99;
   o = p;
   o.xyz -= normal * 0.01;
   d = vec4(rdir, 0);
  } else
    kr = 1;    // total internal relfection
}
```

Rekurzió helyett

```
uniform int nRecursions = 2;

void main() {
  vec4 o = vec4(eye, 1);
  vec4 d = vec4(normalize(viewDir), 0);
  outcolor = vec4(0, 0, 0, 1);

  float contrib = 1;
  for(int iReflection=0;
      iReflection<nRecursions && contrib > 0.01;
      iReflection++)
    outcolor.xyz += trace(o, d, contrib);
}
```

Sugárkövetés a GPU-n