# GLOBALNO OSVETLJEVANJE

# Upodabljanje

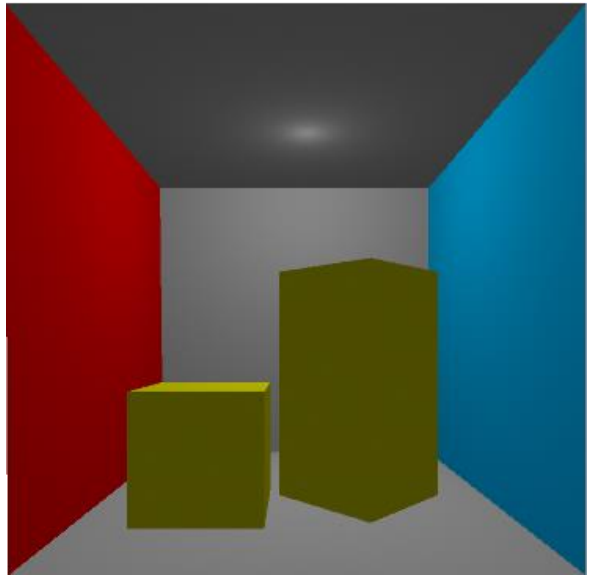
- Slika nastane kot interakcija med
  - **objekti** v sceni
    - imajo položaj, material, ki določa kako interaktirajo s svetlobo
  - **lučmi**
    - imajo položaj, barvo svetlobe, obliko, smer širjenja svetlobe
  - **gledalcem** – položaj gledalca oz. kamere določa kaj vidimo oz. kaj je na sliki

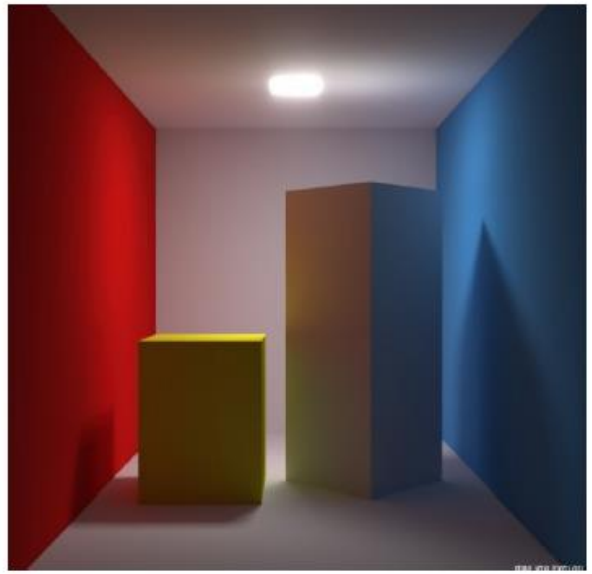
*Maverick Render*

*"Chado" by Norbert Kern (2001) – POV Ray*

# Lokalno in globalno osvetljevanje

- **Lokalno** osvetljevanje
  - poenostavljeno, hitro
  - le en odboj med izvorom in gledalcem
  - računanje osvetlitve ene ploskve je neodvisno od ostalih
  - **rasterizacija** bazira na lokalnem osvetljevanju
- **Globalno** osvetljevanje:
  - upoštevamo več odbojev svetlobe od predmetov
  - računsko zahtevno – simulacija fizike
  - metode **sledenja žarka** – sledimo odbojem svetlobe v sceni
    - vse večja podpora za izvajanje v realnem času





# Globalno osvetljevanje

- Metode večinoma temeljijo na **sledenju žarkom** svetlobe
- Sledenje žarku (*ray tracing*)
  - osnovna metoda
  - zrcalni odboji, ni difuznih odbojev
  - Whitted 1980
- Sledenje poti (*path tracing*)
  - stohastičen algoritem
  - difuzni in zrcalni odboji
  - Kajiya 1986
- Ne-realnočasovna uporaba v animaciji, filmih, arhitekturi …
  - v zadnjem času prehod v realnočasovno sledenje s strojnim pospeševanjem in AI modeli za odstranjevanje šuma

*PBRT Gallery*

# Sledenje žarku – ray tracing

# Sledenje žarku

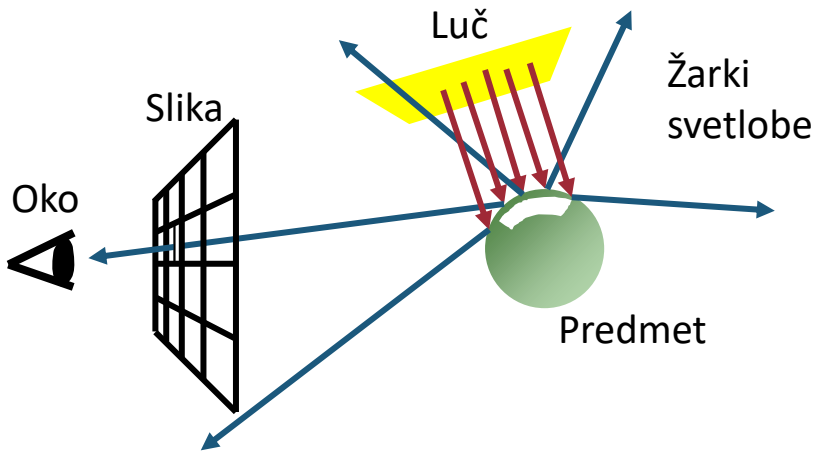
- Metoda **globalnega** osvetljevanja
- Simulacija svetlobnih žarkov
  - “naravna” osvetlitev
  - odboji
  - lom svetlobe
  - mehke sence
  - …

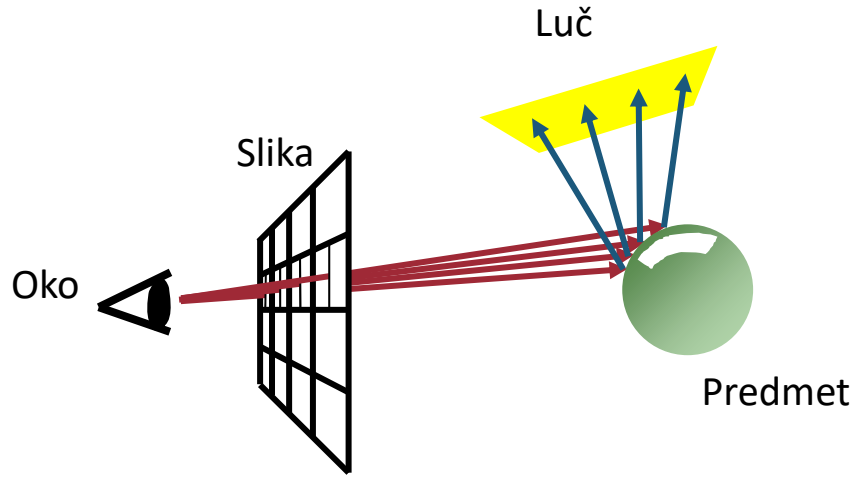

*Disney/Pixar: Cars 2*


*Control*

# Sledenje žarku – kako

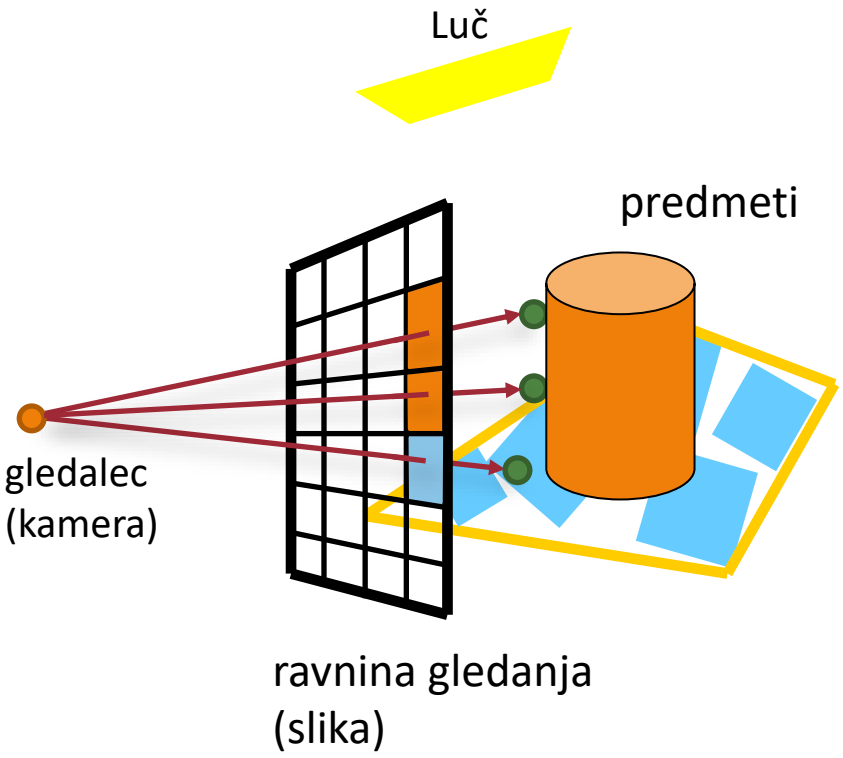
- Realnost: žarke generirajo svetlobni viri in potujejo do očesa
  - za tovrsten izračun moramo slediti veliko žarkom, le malo pa jih pride do očesa
- **Obrnemo** situacijo:
  - sledimo žarkom od očesa preko vseh pikslov v sliki in gledamo kam se zaletijo



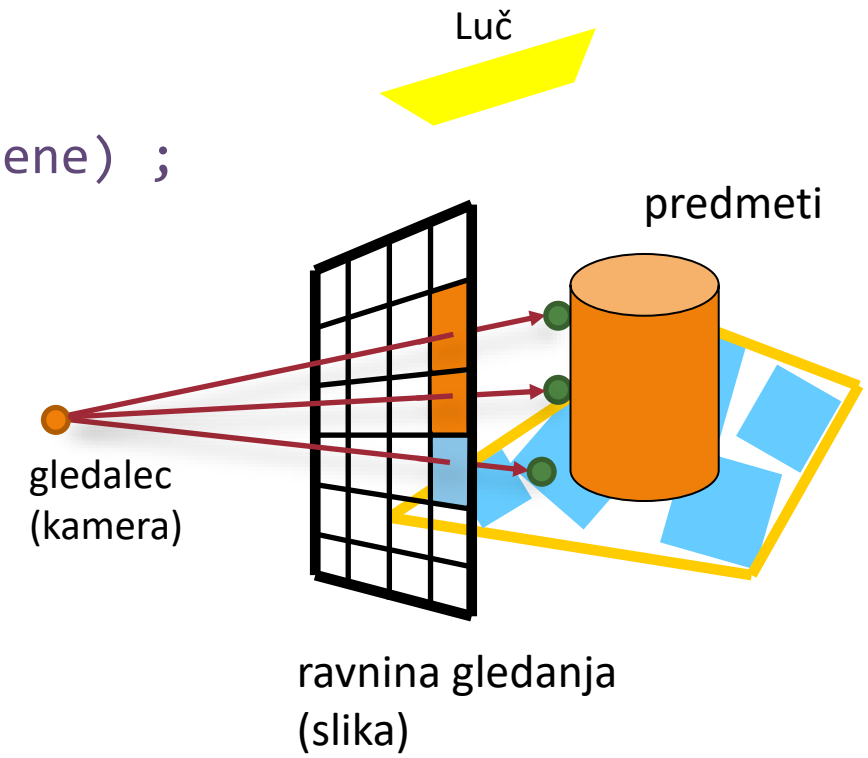
# Metanje žarka – ray casting

- **Metanje žarka**: je prvi del algoritma sledenja žarku
  - Appel, 1968
- Uporablja se tudi pri
  - upodabljanju volumetričnih podatkov
  - prikazu polnih teles (CSG)
  - odstranjevanju zakritih površin
  - …
- Koncept:
  - sledimo žarku svetlobe od očesa do presečišča s prvim predmetom
  - en žarek skozi vsak piksel v končni sliki
  - v presečišču s predmetom izračunamo barvo z osvetlitvenim modelom
  - če žarek ne seka nobenega predmeta, je piksel črn



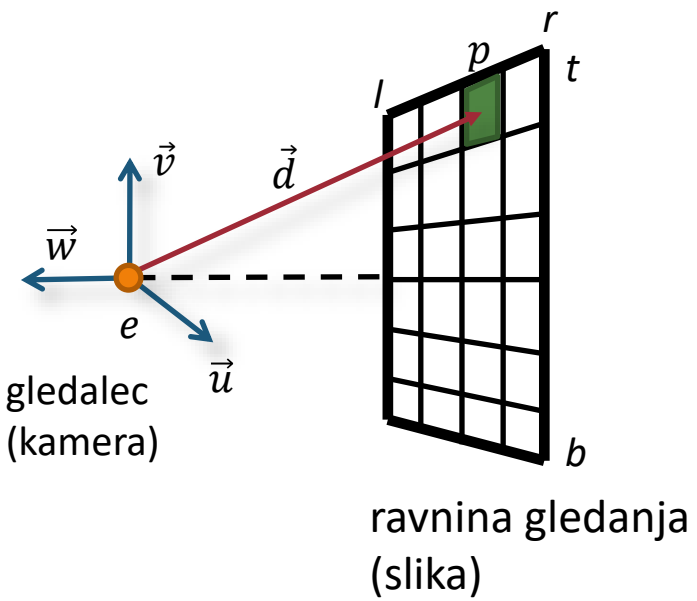
# Metanje žarka – ray casting

```
// okvirni algoritem
Image image = new Image (width, height) ;
for (int i = 0 ; i < height ; i++)
    for (int j = 0 ; j < width ; j++) {
            Ray ray = RayThruPixel (cam, i, j) ;
            Intersection hit = Intersect (ray, scene) ;
            image[i][j] = FindColor (hit) ;
    }
return image ;
```



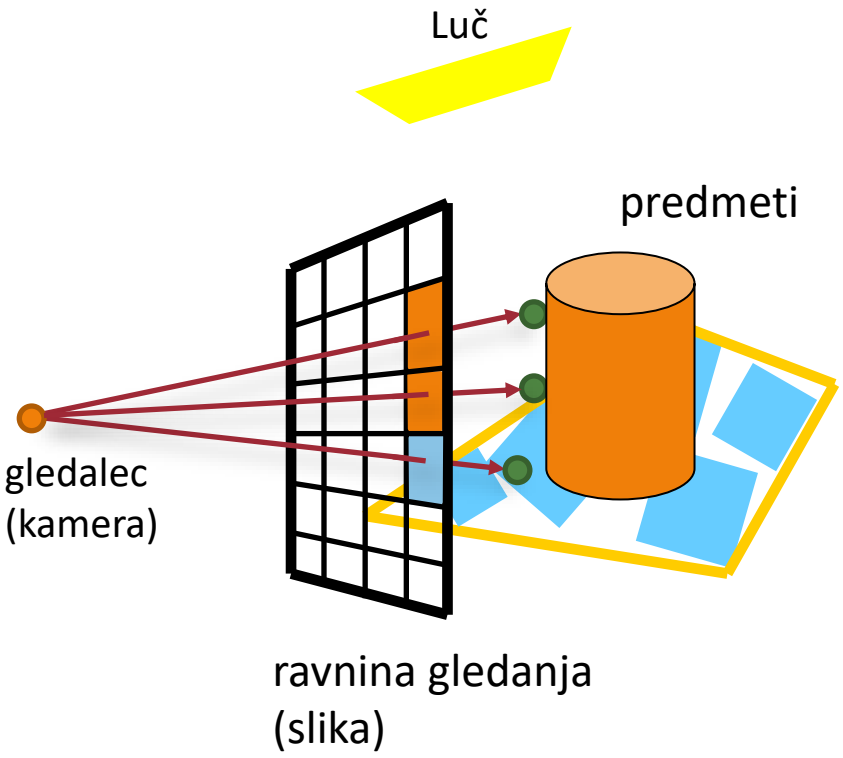
# Konstrukcija žarka

- Kako za nek piksel slike izračunati **smer žarka**
- Konstrukcija žarka:
  - koordinatni system kamere je $[\vec{u}, \vec{v}, \vec{w}]$
  - slika (velikosti $n_x \times n_y$) je pravokotna na $\vec{w}$ kamere in na razdalji $d$ od kamere
  - koordinate $(u, v, w)$ piksla $(i, j)$ v koordinatah kamere so:
    - $u = l + (r - l)\frac{i+0.5}{n_x}$
    - $v = b + (t - b)\frac{j+0.5}{n_y}$
    - $w = -d$
  - izvor žarka je $e$
  - smer žarka: $\vec{d} = u\vec{u} + v\vec{v} - d\vec{w}$
  - piksel na sliki: $p = e + \vec{d}$
- Parametrična **enačba žarka**:
  - $r(t) = e + t(p - e) = e + t\vec{d}$



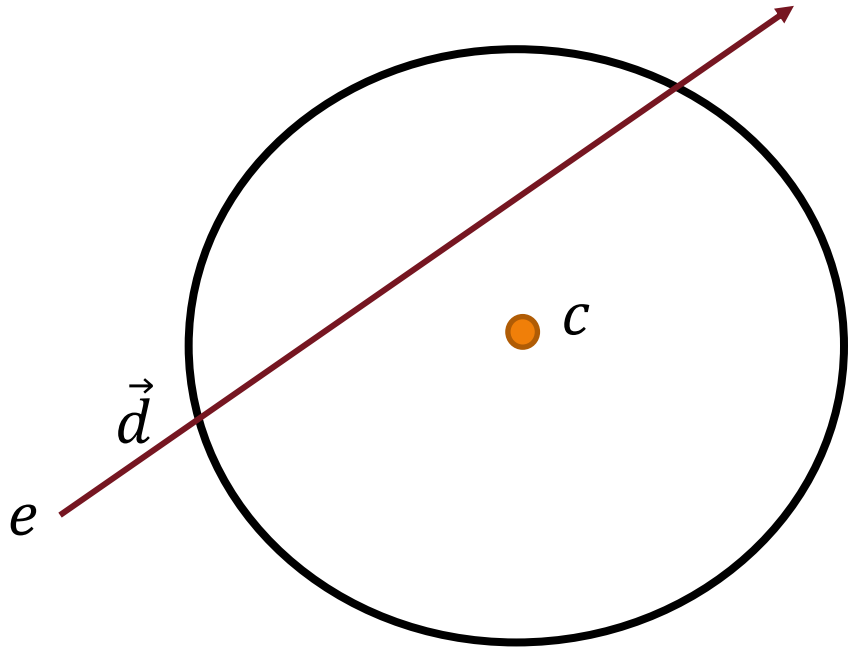
# Preseki

- Kako ugotovimo ali in kje žarek **preseka** predmet?
  - odvisno od prestavitve predmeta – poligoni, parametrična itn.
- Žarek je predstavljen parametrično kot
  - $r(t) = e + t\vec{d}$
- Iščemo torej vrednost $t$ pri kateri žarek preseka nek predmet
  - najmanjši $t > 0$ bo najbližji presek
- Iskanje presekov je najbolj časovno zahteven del pri tej metodi in iz nje izpeljanih (npr. sledenje žarku)



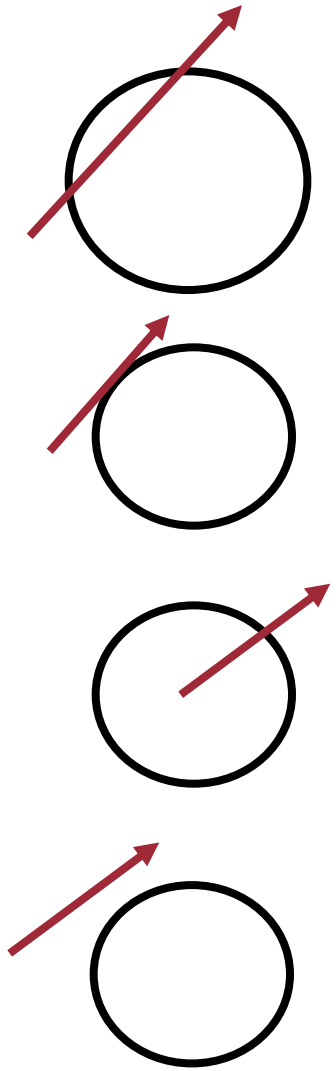
# Presek s kroglo

- Enostaven primer
  - se večkrat uporablja, saj krogla pogosto predstavlja očrtano telo (*bounding ball*) predmeta
- Kroglo implicitno predstavimo kot:
  - $(p - c) \cdot (p - c) - r^2 = 0$
- Vstavimo enačbo žarka
  - $\left(e + t\vec{d} - c\right) \cdot \left(e + t\vec{d} - c\right) - r^2 = 0$
  - $t^2\vec{d} \cdot \vec{d} + 2t\vec{d} \cdot (e - c) + (e - c) \cdot (e - c) - r^2 = 0$



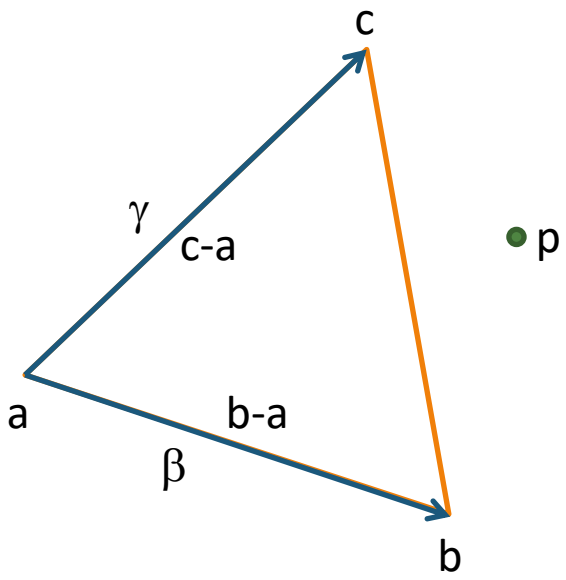
# Presek s kroglo

- Rešimo kvadratno enačbo
  - $$\frac{-\vec{d}\cdot(e-c)\pm\sqrt{\left(\vec{d}\cdot(e-c)\right)^2-(\vec{d}\cdot\vec{d})((e-c)\cdot(e-c)-r^2)}}{\vec{d}\cdot\vec{d}}$$
- 2 realni pozitivni ničli: manjša je prvi presek
- dvojna ničla: tangenta
- ena pozitivna ena negativna ničla: žarek se začne v krogli in gre ven
- kompleksni ničli: žarek ne seka krogle
  - dovolj je, da pogledamo, če je izraz pod korenom negativen, da vemo ali žarek seka kroglo ali ne



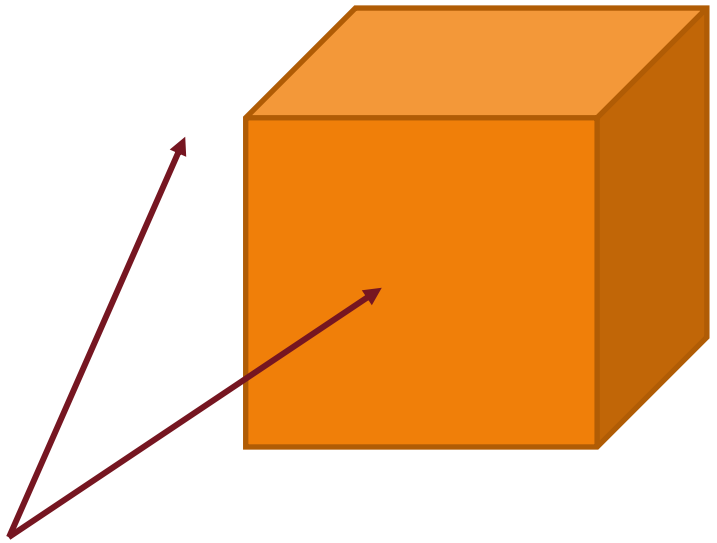
# Presek s trikotnikom

- V **težiščnem** k.s. predstavimo točko $p$ kot
  - $p = a + \beta(b - a) + \gamma(c - a)$
  - $p = \alpha a + \beta b + \gamma c, \quad \alpha + \beta + \gamma = 1$
- Vstavimo enačbo žarka
  - $e + t\vec{d} = a + \beta(b - a) + \gamma(c - a)$
- Dobimo sistem treh enačb $(x, y, z)$ in treh neznank $(t, \beta, \gamma)$, ki ga rešimo
  - obstajajo učinkoviti pristopi, npr. algoritem *Möller–Trumbore*
- Če velja
  - $t > 0$ in $0 < \gamma < 1$ in $0 < \beta < 1 - \gamma$
  - je točka znotraj trikotnika



# Ostali preseki

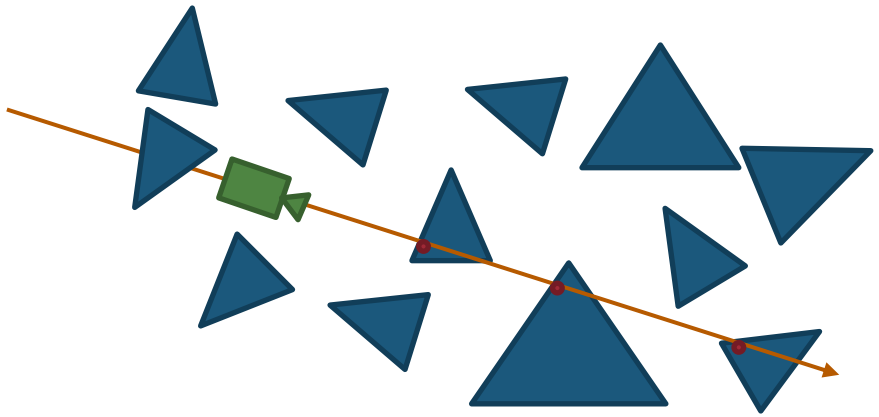
- Algoritmov za določanje presekov je cela vrsta, ker je to časovno najbolj kritičen del algoritma metanja žarka (in vseh ostalih metod, ki sledijo žarkom) žarka
  - stožci, valji, elipsoidi,
  - kocke (veliko se uporabljajo za omejevanje – bounding box)
  - …



# Kako najti prvi presek v sceni

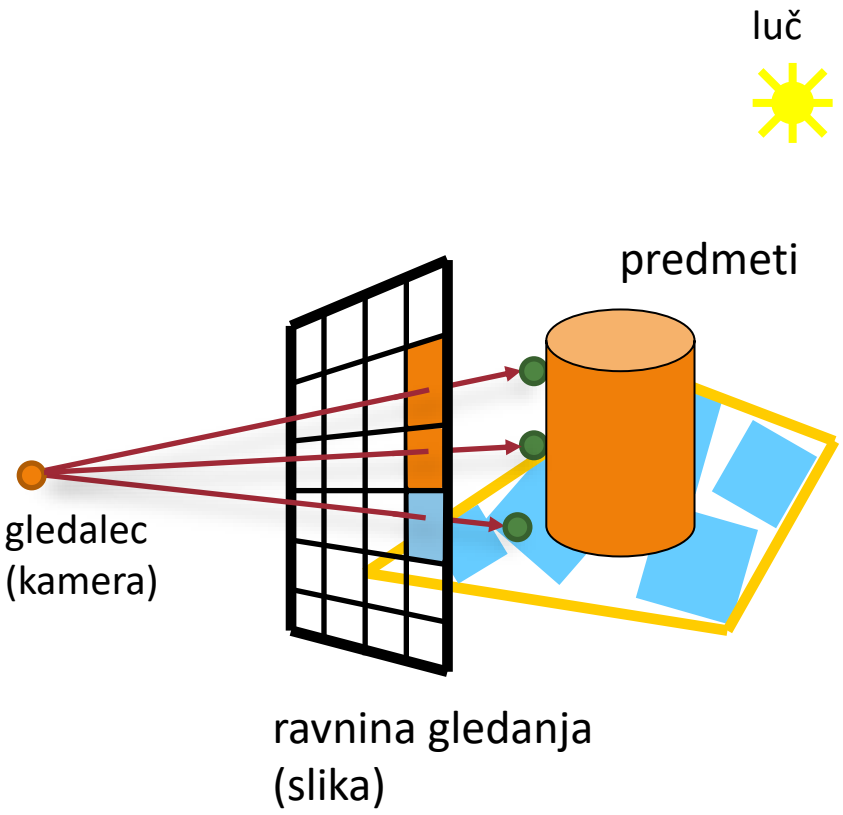
- Računati moramo presek z vsemi predmeti v sceni
- Vrnemo presek z najmanjšim $t$, ki je večji od 0
- Algoritem:

```
Intersect (ray, scene) {
  min_t=Infinity;
  min_p=null;
  foreach (primitive in scene) {
      t=Intersect(ray, primitive);
      if (t>0 && t<min_t) {
        min_p=primitive;
        min_t=t;
      }
  }
  return new Intersection(min_t, min_p);
}
```



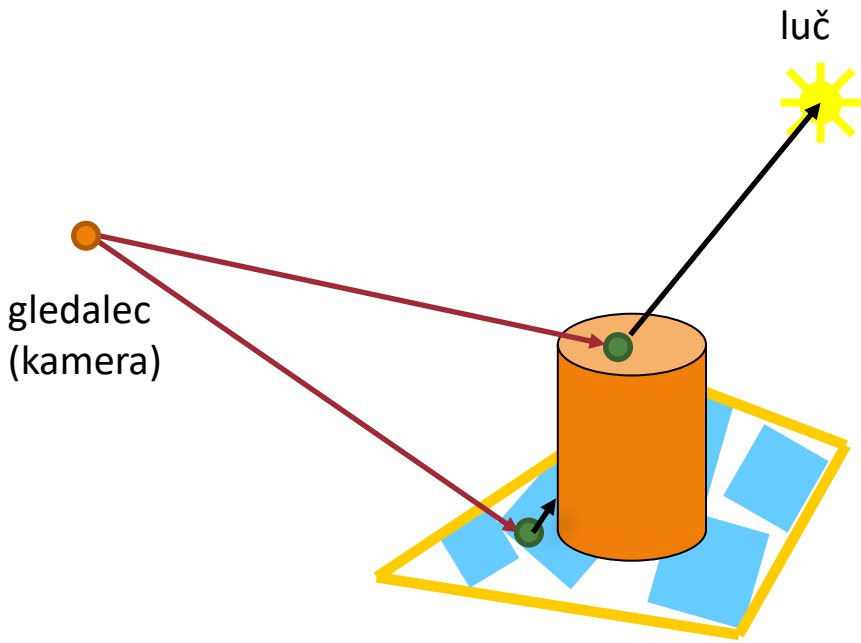
# Metanje žarka - osvetlitev

- Ko najdemo presek žarka s prvim telesom, izračunamo še **osvetlitev**
  - osvetlitev se torej računa v vsaki točki (pikslu slike)
- Osvetlitveni model izberemo, lahko je Phongov
  - $I = k_a L_a + L_i \left( k_d (\vec{L} \cdot \vec{N}) + k_s (\vec{V} \cdot \vec{R})^p \right)$
    - $k_a$ - ambientna svetloba
    - $k_d, k_s$ - difuzna in zrcalna komponenta
    - $p$ – zrcalni koeficient
    - jakost luči $L_a, L_i$



# Sence

- Metanje žarka upošteva tudi **sence**
  - v vsaki točki preseka pošljemo senčni žarek (*shadow ray*) proti vsaki luči
    - za senčni žarek ponovno računamo preseke z vsemi predmeti v sceni
  - če na poti senčnega žarka najdemo kak presek s predmetom, je točka v senci ($V_i = 0$), sicer ni ($V_i = 1$)
  - $I = k_a L_a + V_i L_i \left( k_d (\vec{L} \cdot \vec{N}) + k_s (\vec{V} \cdot \vec{R})^p \right)$



# Rasterizacija vs. metanje žarka

Rasterizacija:

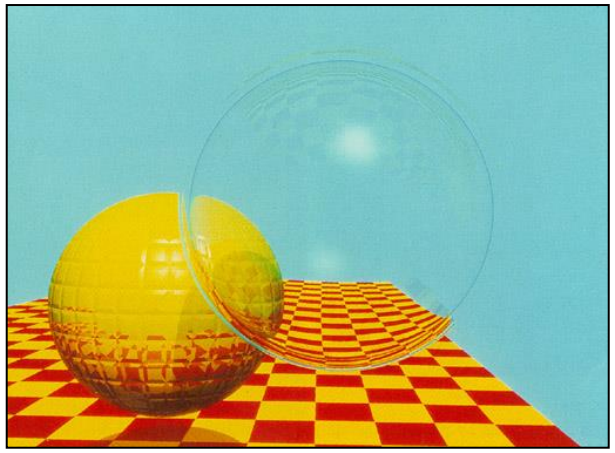
```
OutputImage img
foreach Polygon p:
  foreach Pixel pp in p:
     c=CalculateColor(pp)
     if Visible(pp)
       img[pp]=c
```

Metanje žarka:

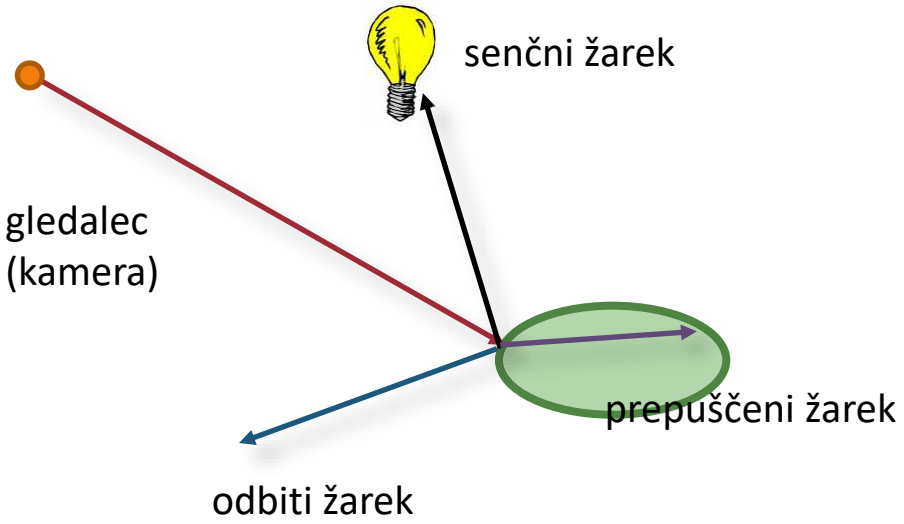
```
OutputImage img
foreach Pixel ip in img:
   ray=GetRay(ip)
   foreach Polygon p in Scene:
      t=FindIntersection(ray,p)
      mint=FindIfClosest(t)
   img[ip]=CalculateColor(mint)
```

# Sledenje žarku

- Metoda **globalne** osvetlitve
- Osnova je metanje žarka
- Žarku sledimo tudi **po prvem dotiku** s predmetom v dve smeri:
  - **popolni** odboj za zrcalne odboje
  - **prepuščeni** žarek za prosojne materiale
- Pri obeh novih žarkih rekurzivno ponovimo celoten postopek sledenja
  - **seštevamo** svetlobne prispevke žarkov

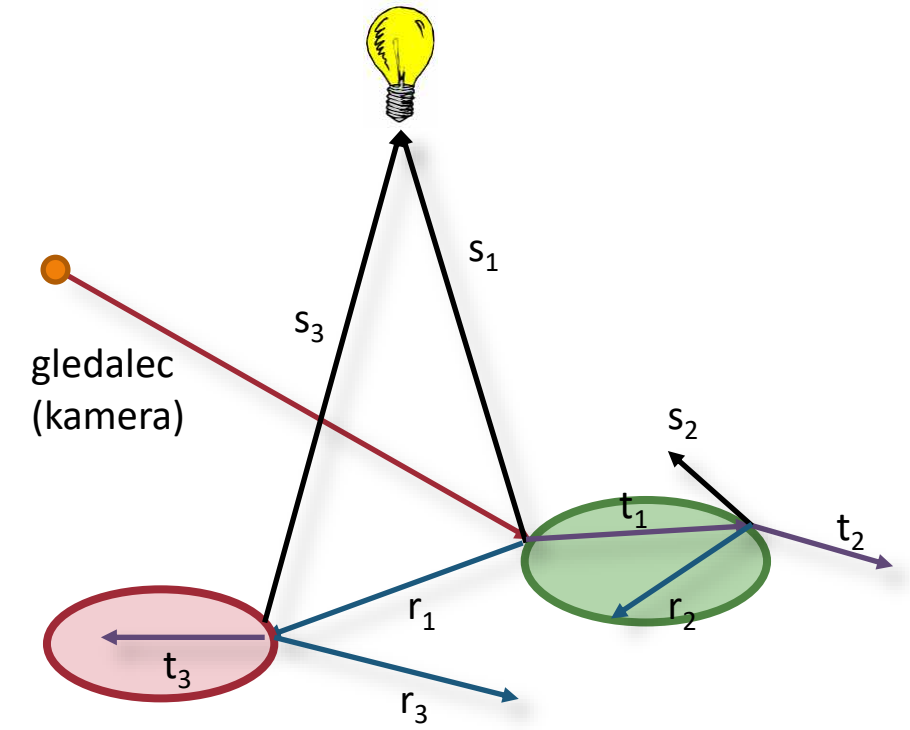


Turner Whitted 1980



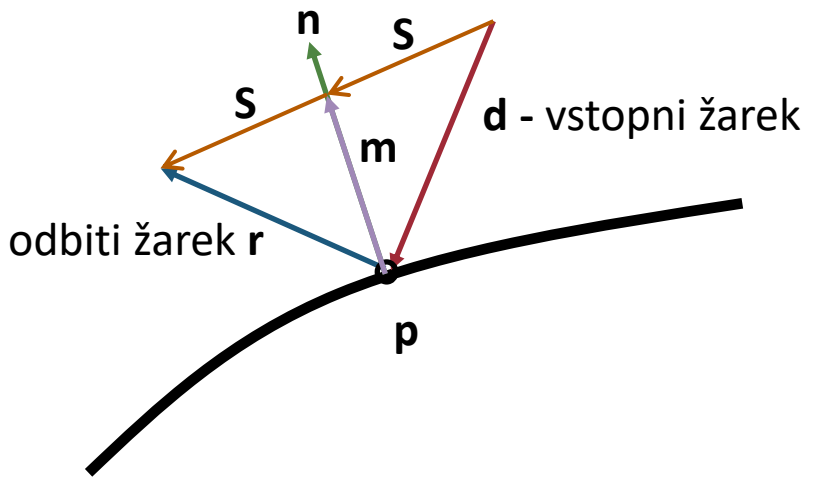
# Rekurzija

- Kdaj rekurzijo pri nekem žarku **ustavimo**?
  - ko žarek zadene luč (dobi barvo luči)
  - ko žarek ne zadene ničesar (tema)
  - ko žarek pride do predmeta, ki nima zrcalnih odbojev in ni prosojen
- Ti pogoji niso dovolj, omejiti moramo **globino** rekurzije
  - koliko nivojev rekurzije rabimo? Odvisno od kompleksnosti scene, npr. 4
  - z večjo globino raste tudi kompleksnost, saj moramo slediti vse več žarkom in računati vse več presekov



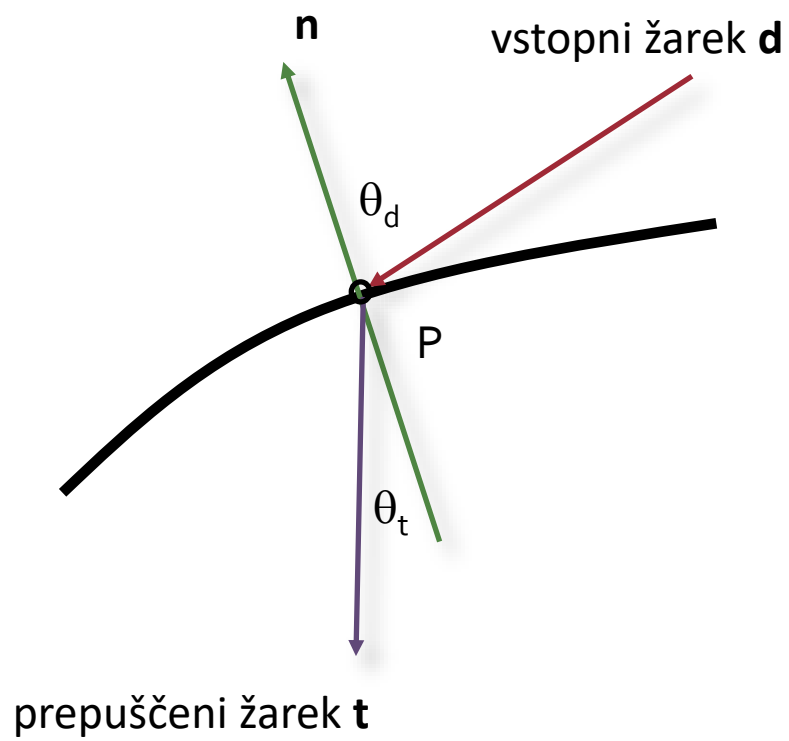
# Odboj

- Odbiti žarek računamo kot popolni odboj
  - $\vec{m} = -\vec{n}\left(\vec{n} \cdot \vec{d}\right)$
  - $\vec{s} = \vec{d} + \vec{m}$
  - $\vec{r} \quad = \vec{m} + \vec{s} = \vec{d} + 2\vec{m}$
    $= \vec{d} - 2\vec{n}\left(\vec{n} \cdot \vec{d}\right)$
- Nov odbiti žarek je parametrično zapisan kot:
  - $r(t) = p + t\vec{r}$



# Lom svetlobe

- **Prosojni** materiali (steklo, voda itn.) prepuščajo svetlobo
  - in jo tudi lomijo
  - svetloba se **upočasni**, ko preide v bolj gost medij
- Velja **Snellov zakon**:
  - $\eta_d \sin\theta_d = \eta_t \sin\theta_t$
    - $\eta_d$ - lomni količnik v snovi vstopnega žarka
    - $\eta_t$ - lomni količnik v snovi prepuščenega žarka
- $\eta = \frac{hitrost\ svetlobe\ v\ vakuumu}{hitrost\ svetlobe\ v\ mediju}$
  - $\eta_{zrak} \sim 1, \eta_{voda} = 1{,}33, \eta_{steklo} = 1{,}5$

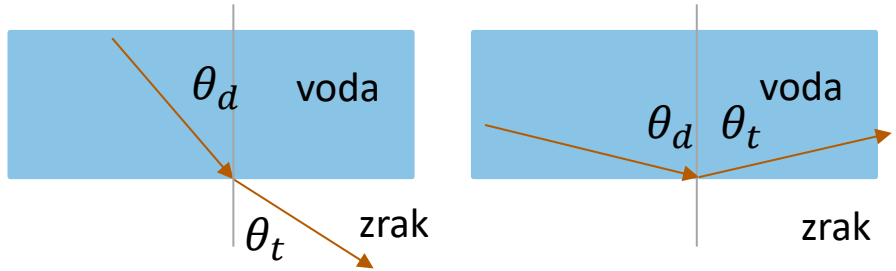


# Lom svetlobe

- Primer
  - voda $\eta = 1.333$
    - hitrost svetlobe v vodi ~ 225056 km/s
  - Zrak-voda
    - Vstopni kot: $\theta_i$: 60°
    - $\eta_d \sin\theta_d = \eta_t \sin\theta_t$
    - Prepuščeni kot $\theta_t$: 40.53°
      - v vodi je svetloba počasnejša, se lomi navznoter
  - Voda-zrak (nazaj)
    - Vstopni kot $\theta_d$: 40.53 °
    - Prepuščeni kot $\theta_t$: 60°
      - na zraku je svetloba hitrejša, se lomi navzven
- Pri vstopu v hitrejši medij lahko pride do **popolnega** notranjega odboja
  - ko je kot vstopne svetlobe prevelik
  - Primer

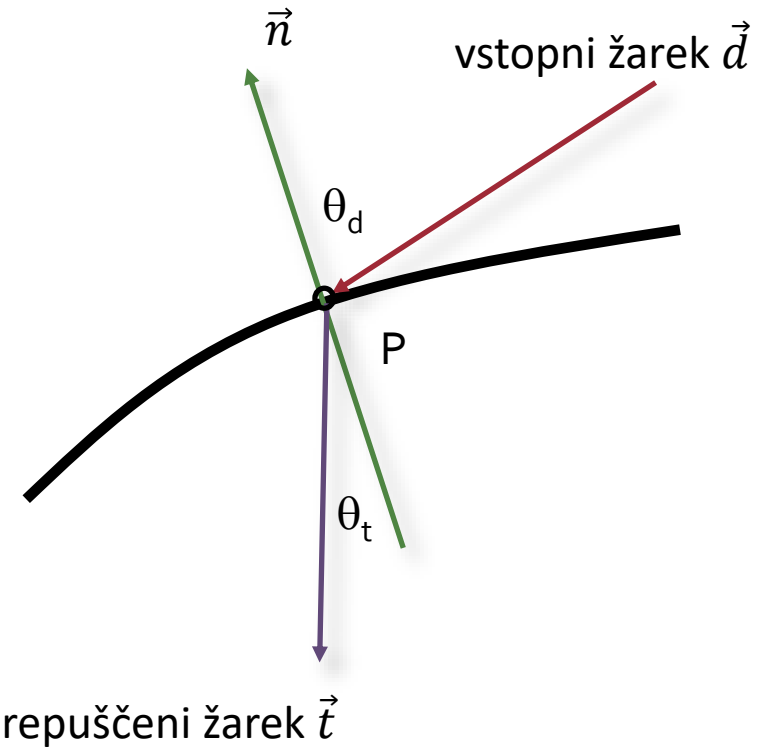
| Material | *n* |
|---|---|
| Vacuum | 1 |
| Gases at 0 °C and 1 atm | |
| Air | 1.000 293 |
| Helium | 1.000 036 |
| Hydrogen | 1.000 132 |
| Carbon dioxide | 1.000 45 |
| Liquids at 20 °C | |
| Water | 1.333 |
| Ethanol | 1.36 |
| Olive oil | 1.47 |
| Solids | |
| Ice | 1.31 |
| PMMA (Plexiglas) | 1.49 |
| Window glass | 1.52[11] |
| Polycarbonate (Lexan™) | 1.58[12] |
| Flint glass (typical) | 1.62 |
| Sapphire | 1.77[13] |
| Cubic zirconia | 2.15 |
| Diamond | 2.42 |
| Moissanite | 2.65 |

*wiki*



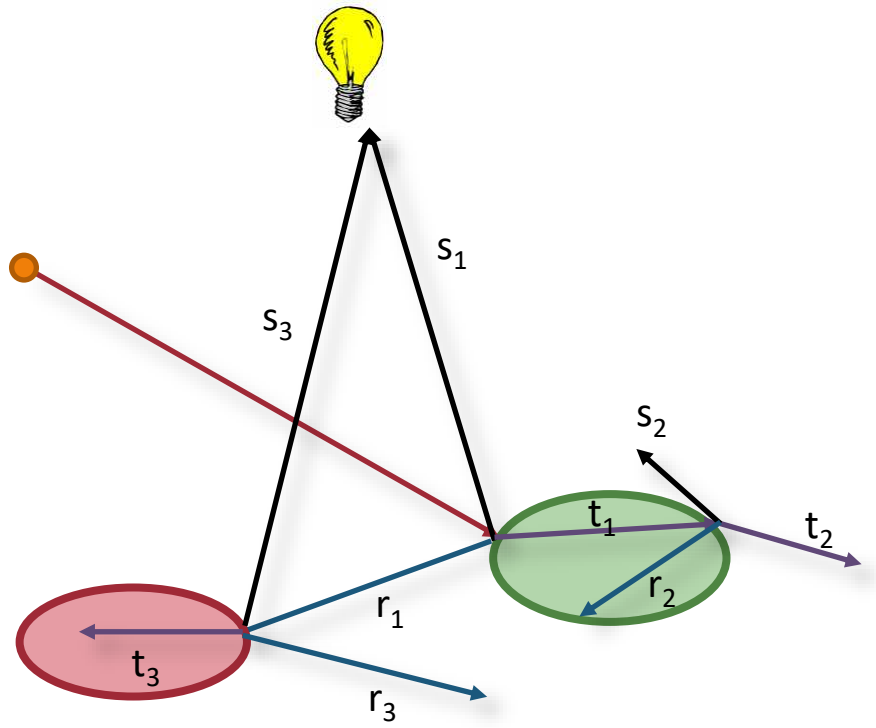
# Lom svetlobe

- Smer prepuščenega žarka izpeljemo iz
  - $\eta_d \sin\theta_d = \eta_t \sin\theta_t$
- Dobimo:
  - $\vec{t} = \frac{\eta_d}{\eta_i}\left(\vec{d} - \vec{n}\left(\vec{n}\cdot\vec{d}\right)\right) - \vec{n}\sqrt{1 - \frac{\eta_d^2}{\eta_i^2}\left(1 - \left(\vec{n}\cdot\vec{d}\right)^2\right)}$



# Osvetlitev

- Osvetlitev v točki računamo kot pri metanju žarka, le da prištejemo še deleže svetlobe $I_r$ in $I_t$, ki pridejo od obeh novih žarkov (po rekurzivnem sledenju)
  - $I = k_a L_a + V_i L_i \left( k_d (\vec{L} \cdot \vec{N}) + k_s (\vec{V} \cdot \vec{R})^p \right) + k_r I_r + k_t I_t$
  - $I_r$ – rekurzivni zrcalni prispevek
  - $k_r$ - zrcalni koeficient (Fresnel)
  - $I_t$ – rekurzivni prepustni prispevek
  - $k_t$ - prepustni koeficient (Fresnel)
- Dejansko osvetlitev torej lahko izračunamo šele po koncu rekurzije

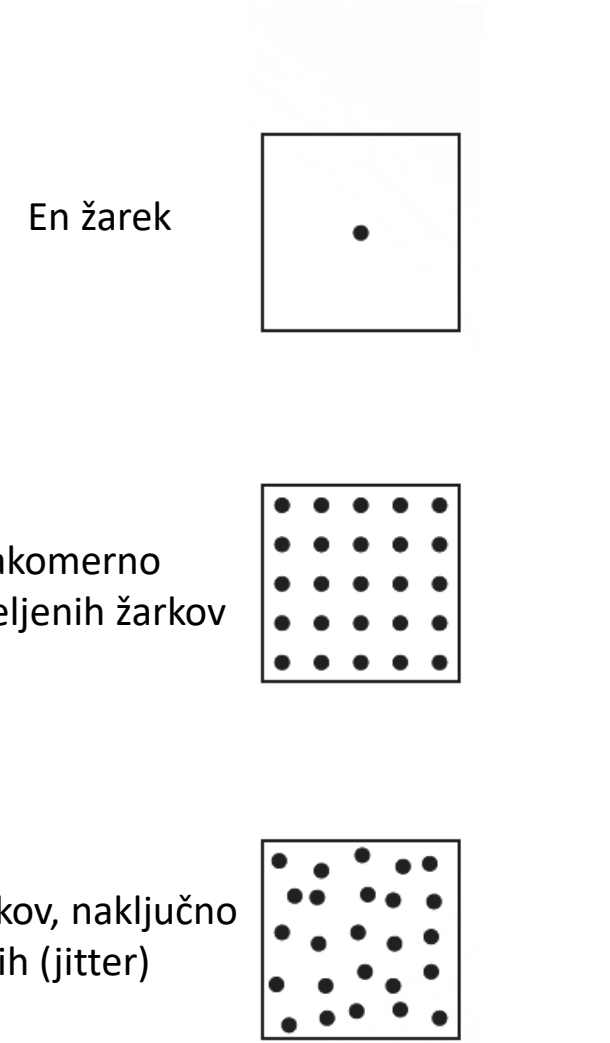


# Sledenje porazdeljenim žarkom

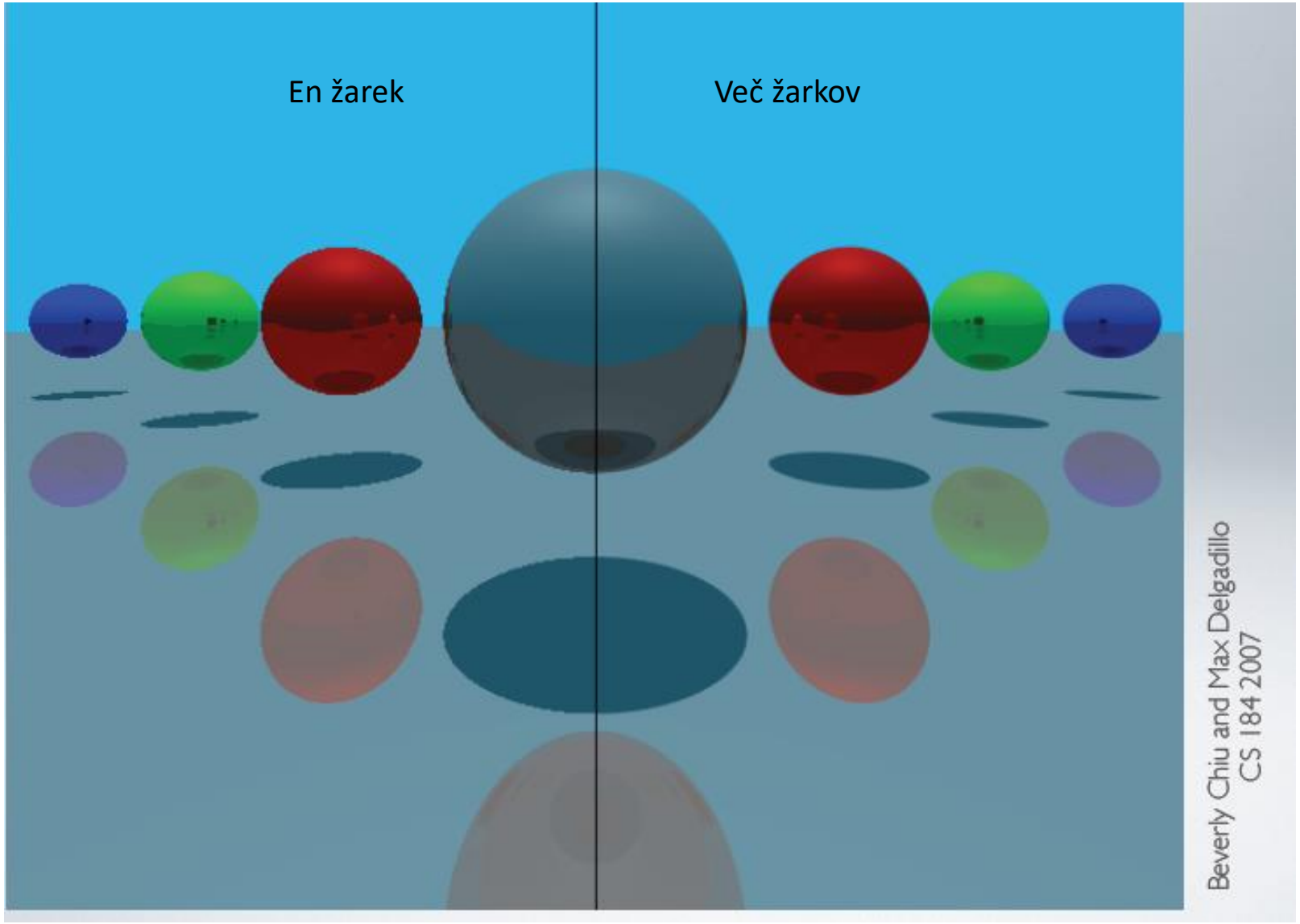
- Osnovni algoritem sledenja žarku ima pomanjkljivosti, izgled slik je precej **umeten**
  - ostri robovi
  - trde sence
  - vse je v fokusu
  - površine “perfektno” sijejo
  - steklo “perfektno” prepušča svetlobo
- Veliko pomanjkljivosti odpravlja t.i. **sledenje porazdeljenim žarkom -** *distribution/distributed ray tracing*
  - *Cook, Porter, Carpenter, 1984*
- Namesto enega žarka delamo v vsaki fazi z več žarki (porazdelitev žarkov)

# Mehčanje robov

- Namesto enega žarka iz očesa pošljemo skozi vsak piksel **več žarkov**
  - supersampling
    - fiksno število žarkov
    - adaptivno (začnemo z majhnim številom, če so rezultati zelo različni, število povečamo)
  - navadno žarke naključno stresemo (jitter)
- Končna barva piksla je **uteženo povprečje** barv vseh žarkov
- Dobimo bolj mehke prehode med ostrimi kontrasti (*antialiasing*)

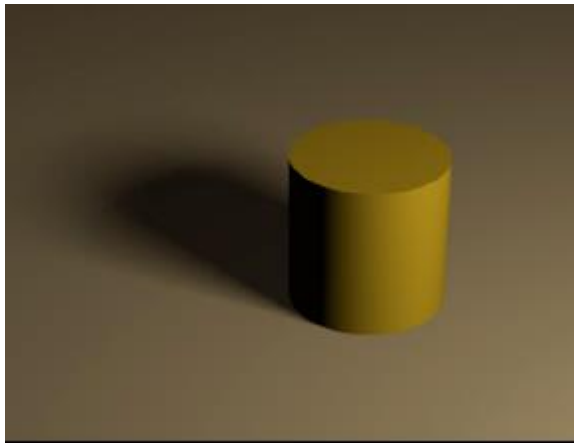


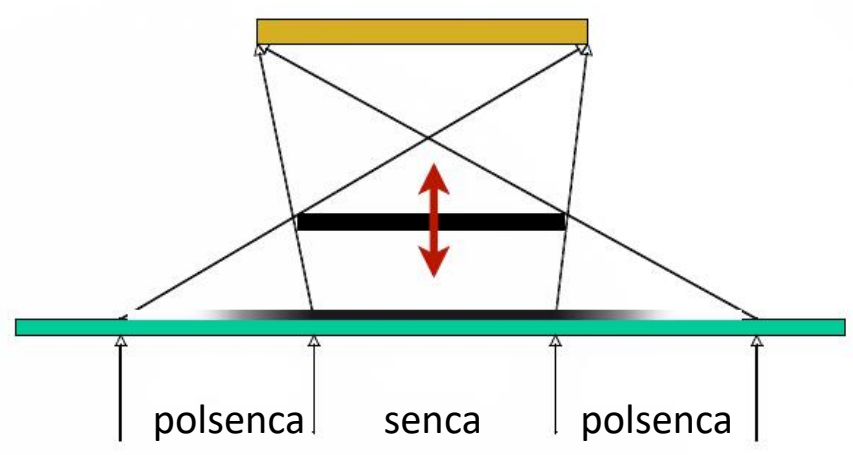
# Mehčanje robov



# Mehke sence

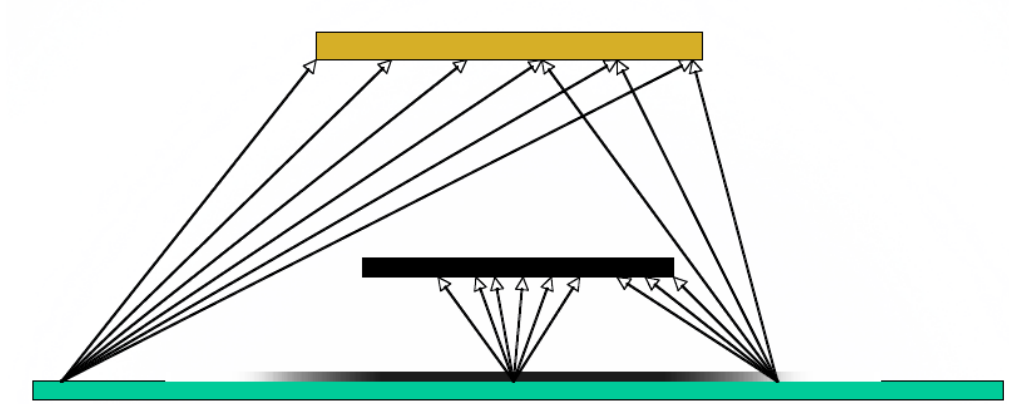
- Točkasta luč **ni realističen** model
  - trde sence
- Luči naj zajemajo večjo površino – površinska luč (area light)





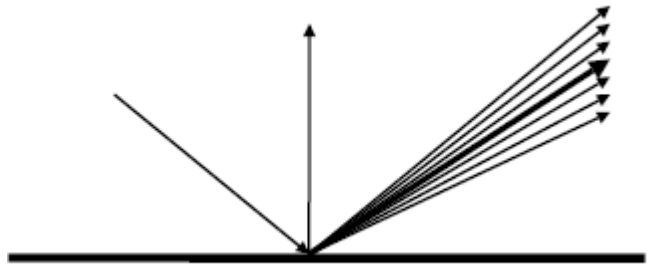
# Mehke sence

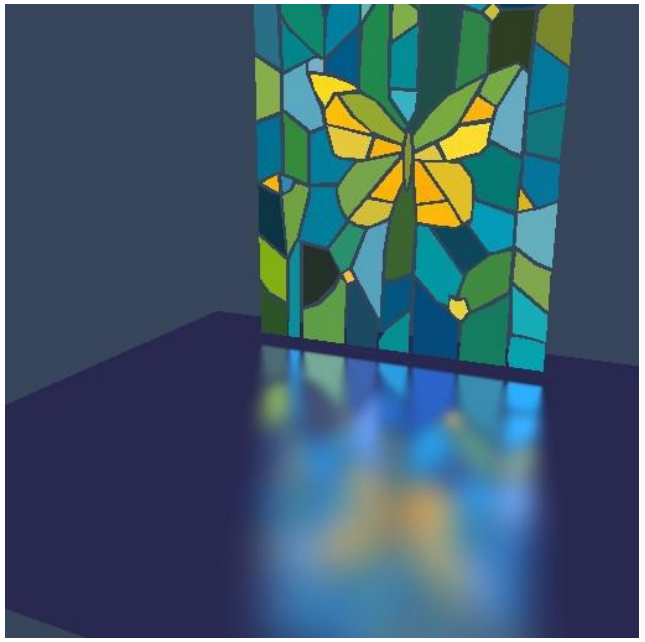
- V luč pošljemo **več** (nekoliko naključno porazdeljenih) **senčnih žarkov**, porazdeljenih po njeni površini, seštejemo doprinose
  - število zadetkov/število žarkov = %osvetlitve



# Mehki odboji

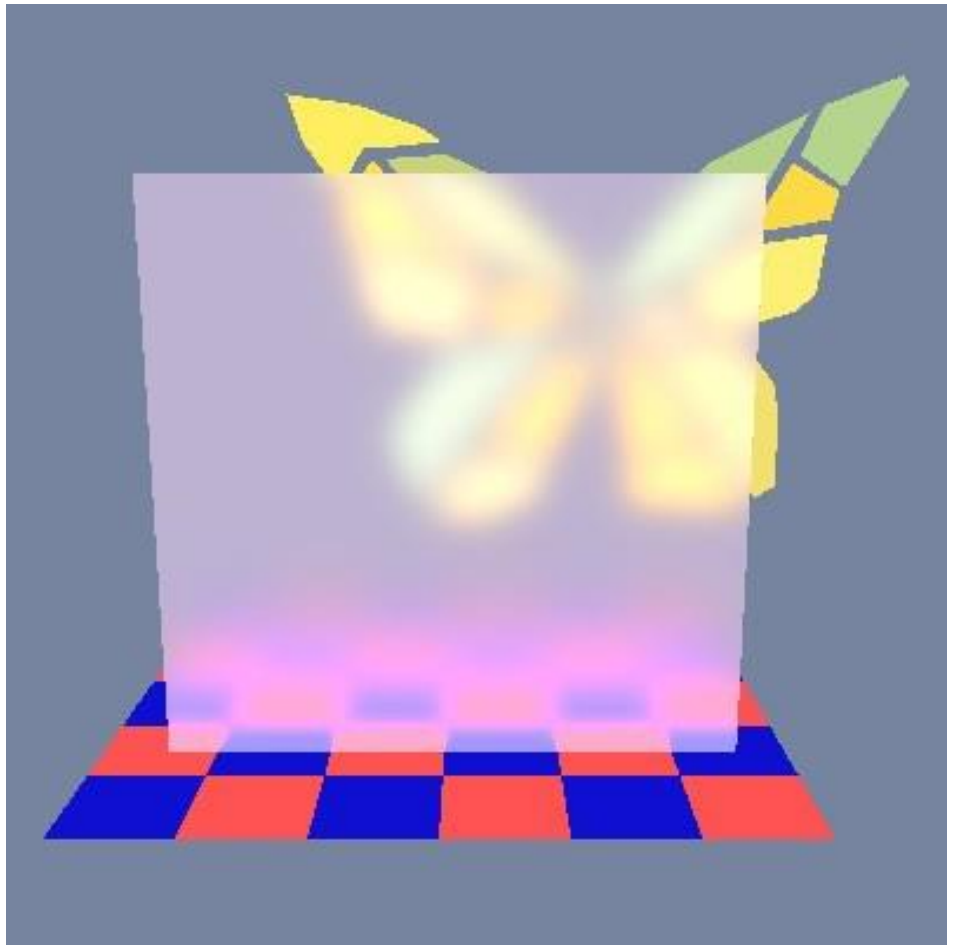
- Pri standardnem sledenju žarkov so odboji **preveč podobni zrcalu** – idealni
- Zaradi grobosti materialov so v realnosti precej bolj zabrisani
- Idejo prenesemo tudi na odbite žarke; namesto enega jih ustvarimo več v (nekoliko naključnih) smereh
  - uporabimo lahko lastnosti materiala (BRDF ali Phong) za določanje števila in smeri





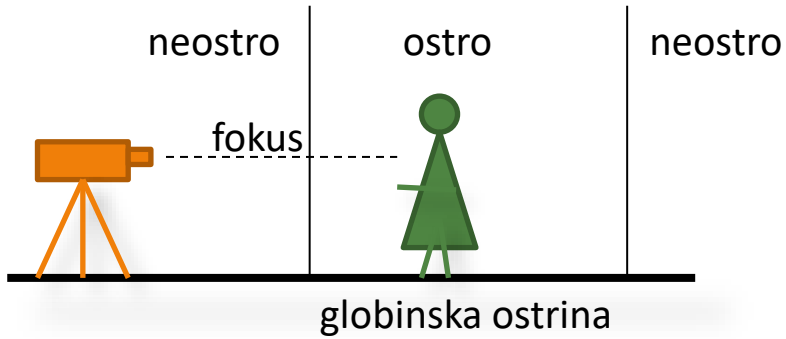
# Prosojnost - translucency

- Difuzna prozornost
- Kot za odbite žarke, lahko tudi pri prepuščenih žarkih ustvarimo **več** “naključno” **porazdeljenih žarkov** okoli idealnega žarka
- Dobimo efekt prosojnosti oz. pol-prozornosti



# Globinska ostrina

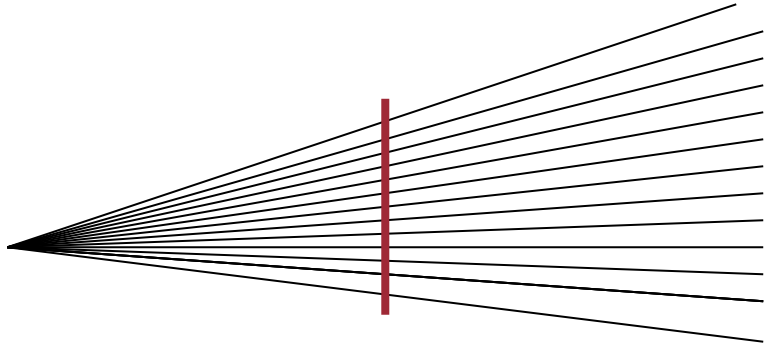
- Globinska ostrina: področje okoli fokusne razdalje, v katerem je slika še vedno sprejemljivo ostra

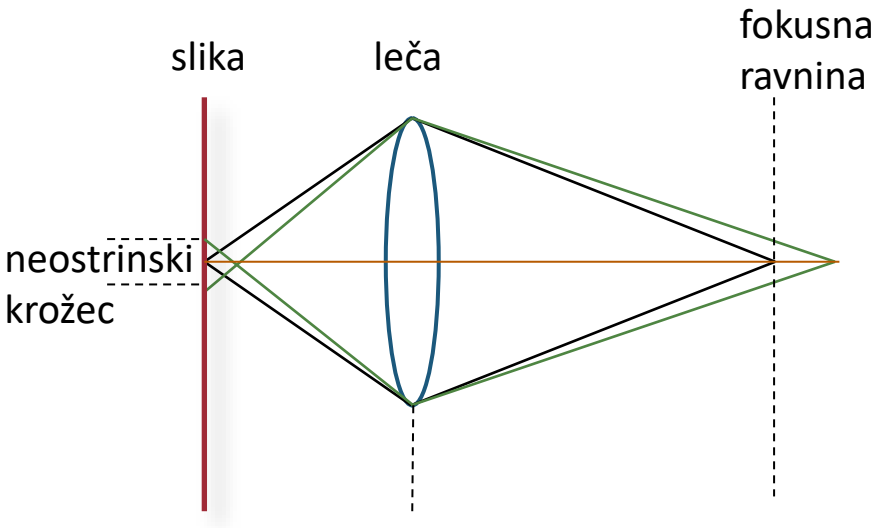



*Lord of the Rings, Newline Cinema*

# Globinska ostrina

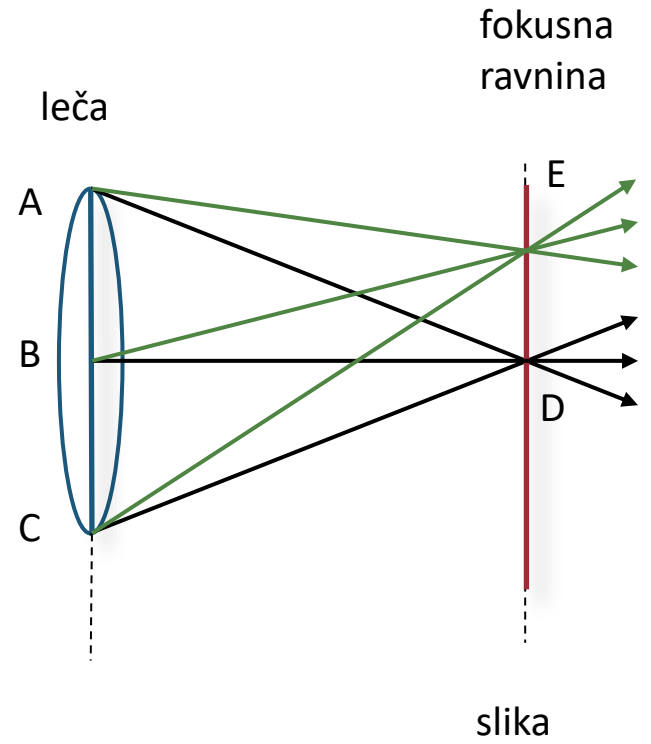
- **RG kamera**:
  - idealna, vsi žarki gredo iz ene točke (leča velikosti 0), vse je ostro
  - ena točka v sceni = ena točka na sliki
- **Leča**: bolj realističen model, žarki razporejeni po leči
  - ena točka na sceni = krožec na sliki, razen za točke v fokusni ravnini





# Globinska ostrina

- Simuliramo jo s **porazdelitvijo žarkov po površini leče**
- Slika je postavljena na fokusno ravnino
- Primer na desni:
  - idealna RG kamera:
    - vsi žarki gredo iz B
    - piksel D dobimo z žarkom BD
    - piskel E dobimo z žarkom BE
  - z globinsko ostrino:
    - žarki so porazdeljeni po ravnini leče
    - slika je na fokusni ravnini
    - piksel D dobimo iz AD, BD, CD …
    - piskel E dobimo iz AE, BE, CE …



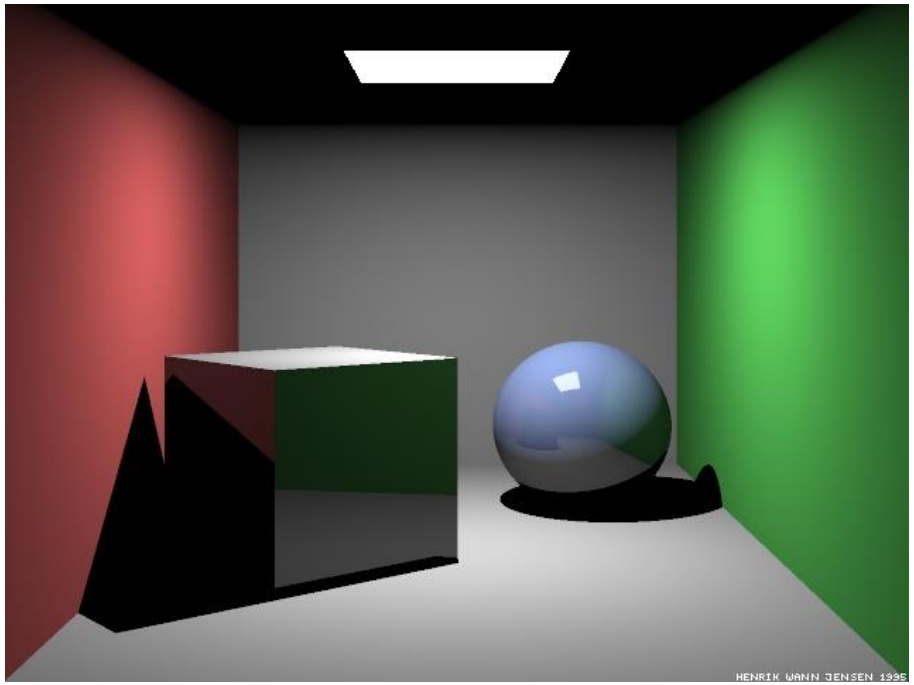
# Zabrisano gibanje

- Zabrisano gibanje - “motion blur” – žarke porazdelimo po času in povprečimo
- Predmeti, ki se premikajo, bodo zamegljeni

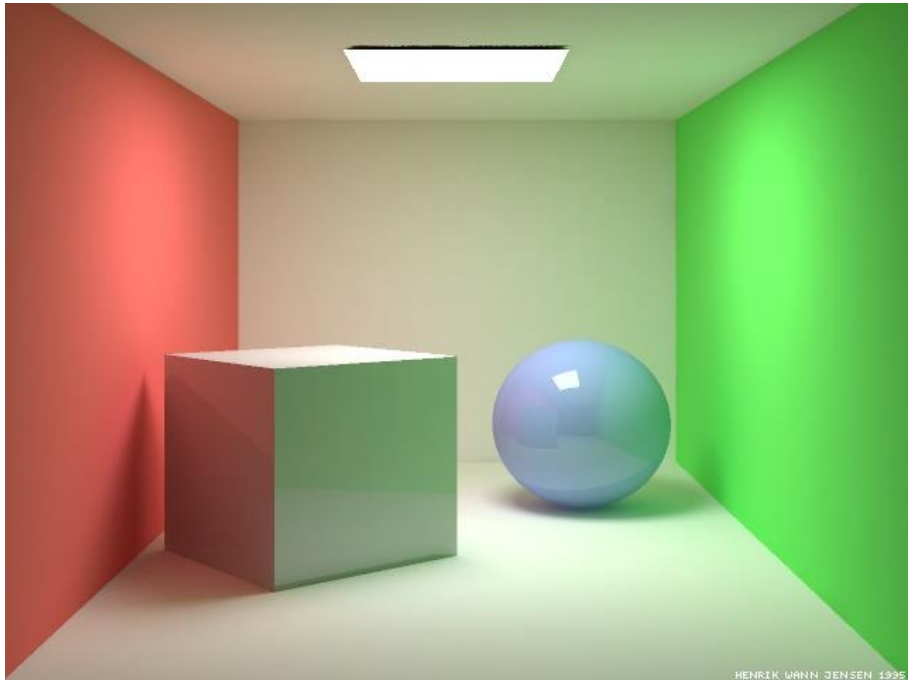
**Pool Balls**
Tom Porter
RenderMan

# Sledenje žarka vs. prava globalna osvetlitev

- Sledenje žarka
  - upoštevamo le zrcalne odboje




- Sledenje poti (path tracing)
  - upoštevamo tudi difuzne odboje



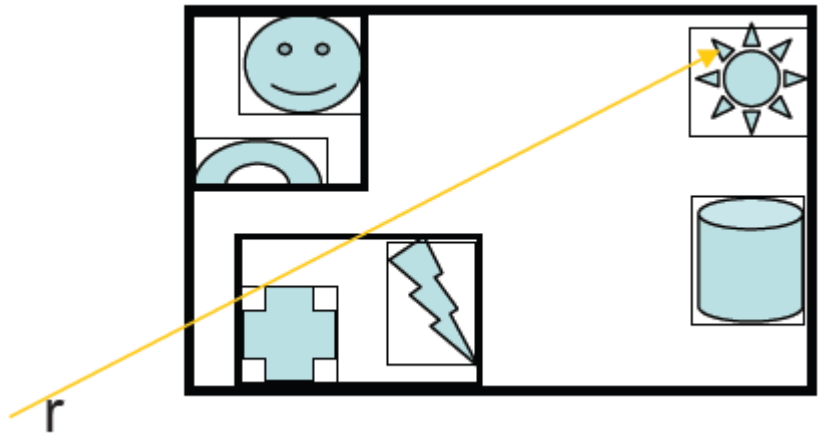
# Sledenje žarkov - pohitritve

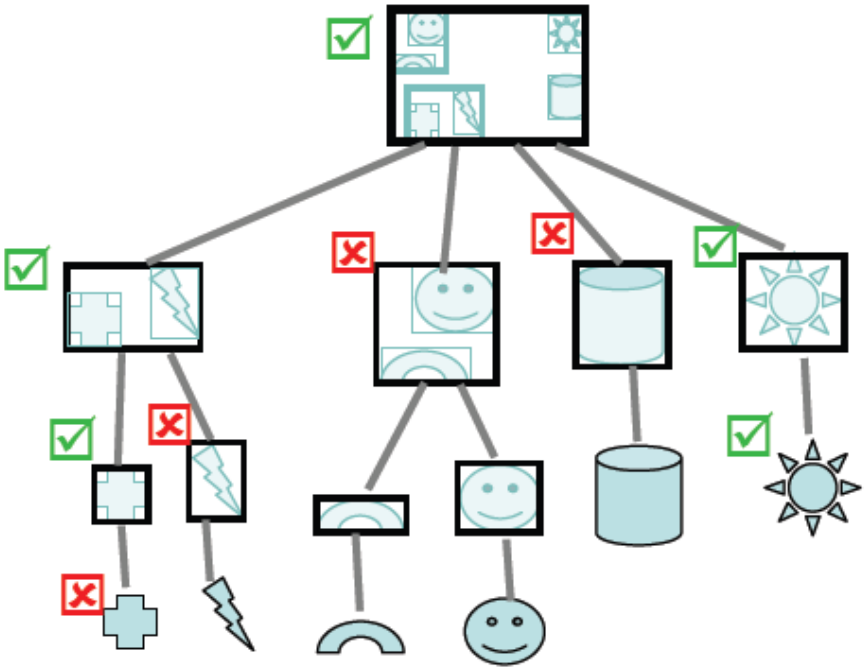
- Sledenje žarkov je **počasna metoda**, predvsem zaradi potrebe po računanju presekov
- Primer
  - slika 1280x1024, 10 žarkov/piksel
  - 1000 predmetov
  - 3 nivoji rekurzije
  - 39.321.600.000 testov presekov
  - 1.000.000 testov/sekundo -> 10.9 ur!

- Moramo uporabiti metode za **pohitritev**!

# Hierarhije očrtanih teles (BVH)

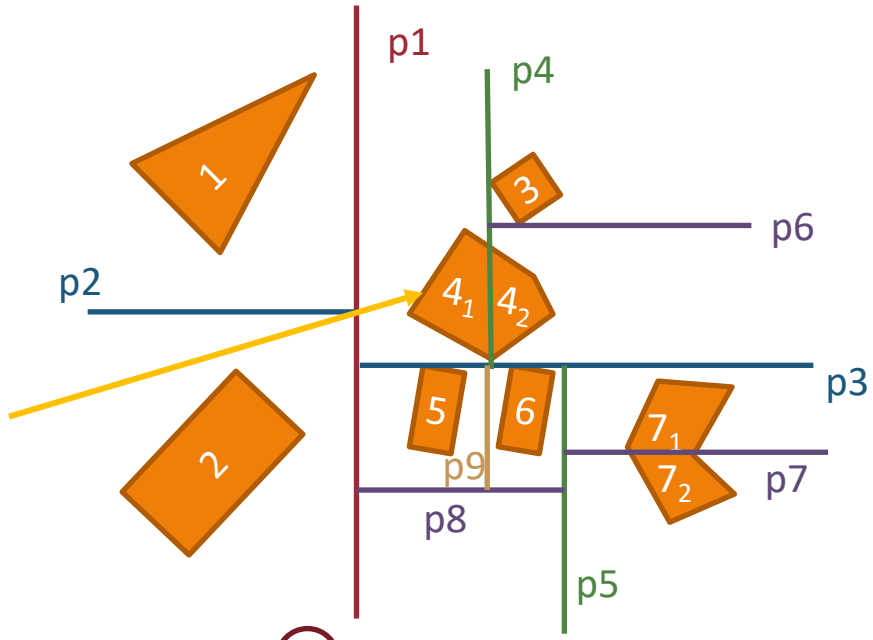
- Predmete očrtamo s hierarhijo teles, s katerimi lahko enostavno izračunamo preseke – **očrtana telesa** (glej poglavje o tehnikah razdelitve prostora)
  - če žarek ne seka očrtanega telesa, ne seka predmeta v njem
- Ko sledimo žarkom, začnemo pri korenu drevesa (celotna scena) in nadaljujemo proti predmetom v listih

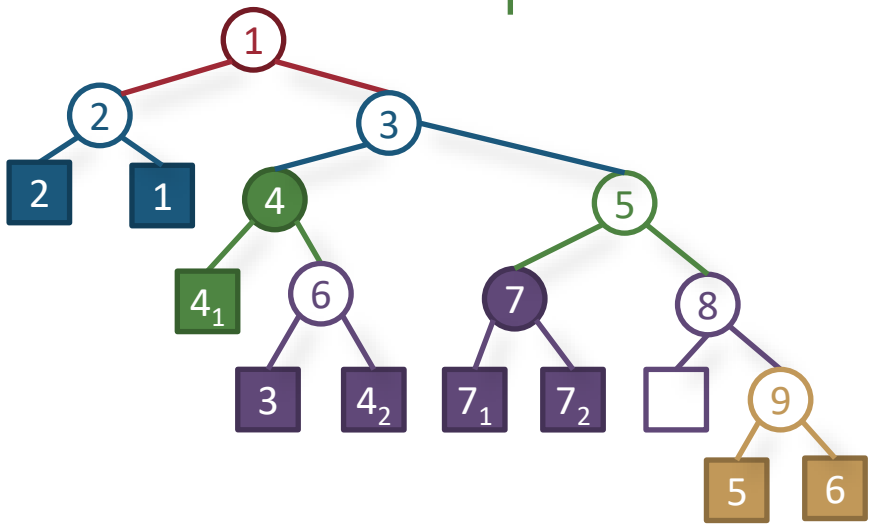


# Razdeljevanje prostora

- Uporabimo lahko tudi druge **tehnike razdelitve prostora**
  - navadna mreža, octree, BSP drevo, kd drevo …
- Hierarhično računamo preseke žarka, dokler ne pridemo do posameznih primitivov (trikotnikov ipd.), s katerimi računamo presek



# Sledenje žarku v realnem času

- S povečevanjem hitrosti GPU prihajamo v čas, ko  je sledenje žarku z ustreznimi kompromisi že mogoče v realnem času
  - Nvidia RTX
    - *demo*
  - Nvidia OptiX API for real-time ray tracing
  - Microsoft DXR API for real-time ray tracing

- https://devblogs.nvidia.com/ray-tracing-games-nvidia-rtx/


*Metro Exodus: GeForce RTX Real-Time Ray Tracing (2018)*

# REFERENCE


- P.H. Christensen et al. Ray Tracing for the Movie ‘Cars’
- N. Guid: Računalniška grafika, FERI Maribor
- J.D. Foley, A. Van Dam et al.: Computer Graphics: Principles and Practice in C, Addison Wesley