

Számítógépes Grafika

6. előadás: sugárkövetés

Bán Róbert

`rob.ban@inf.elte.hu`

Eötvös Loránd Tudományegyetem
Informatikai Kar

2025-2026. tavaszi félév

Tartalom

Rekurzív sugárkövetés

- Bevezető

- Illuminációs egyenlet

- Megjegyzések

Sugárkövetés gyorsítása

- Motiváció

- Befoglaló keretek

Térfelosztó eljárások

- Szabályos rács

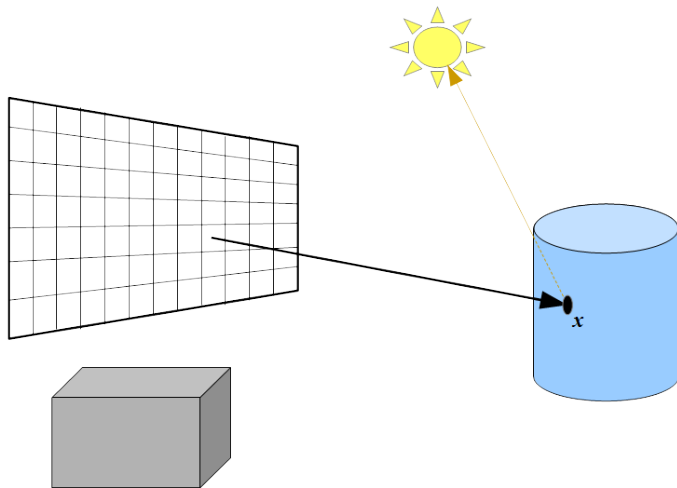
- Octree

- Quadtree

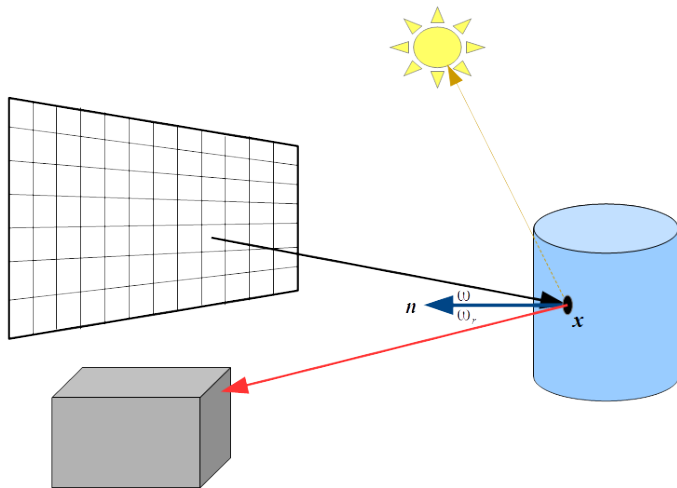
- kd-fa

Sugárkövetés hardverben

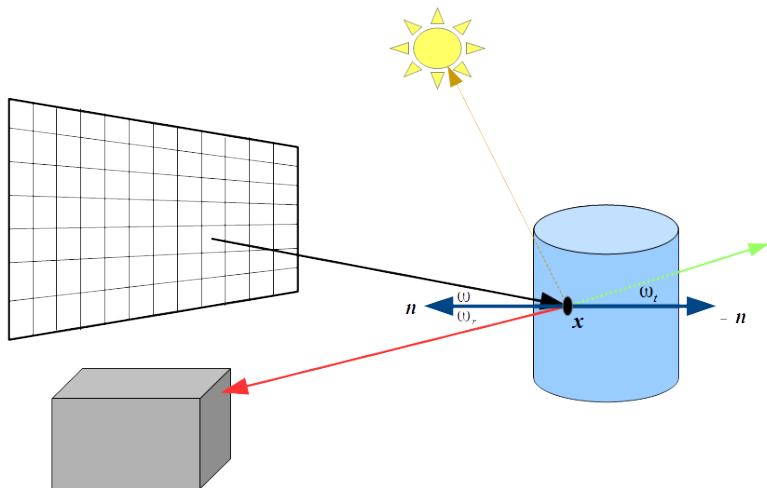
Rekurzív sugárkövetés – árnyék sugar



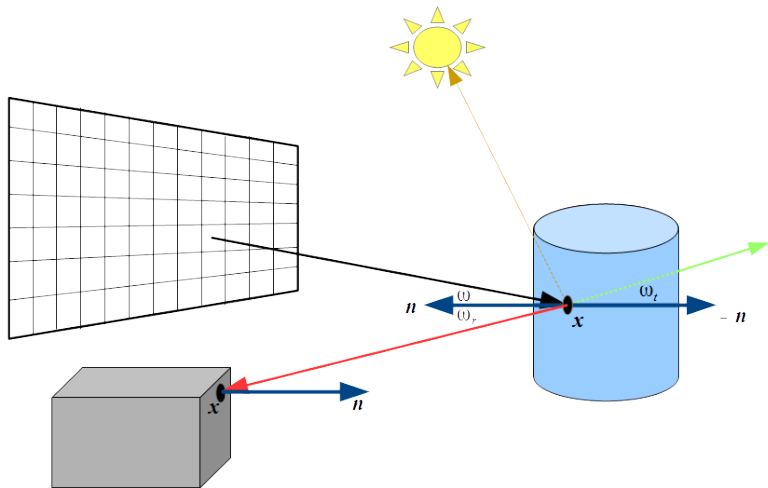
Rekurzív sugárkövetés – ideális visszaverődés



Rekurzív sugárkövetés – ideális törés

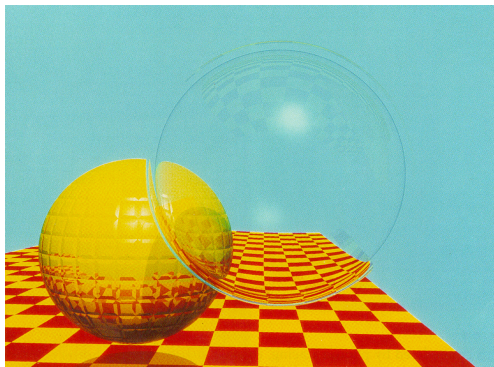


Rekurzív sugárkövetés – rekurzió



Rekurzív sugárkövetés

Minden pixelre egymástól függetlenül határozzuk meg azok színét
– oldjuk meg az *árnyalási* és *takarási* feladatokat.

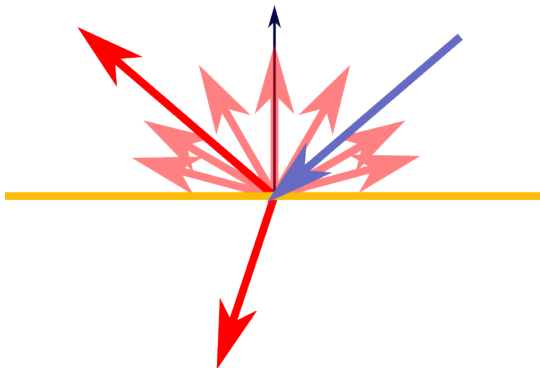


Turner Whitted, 1980

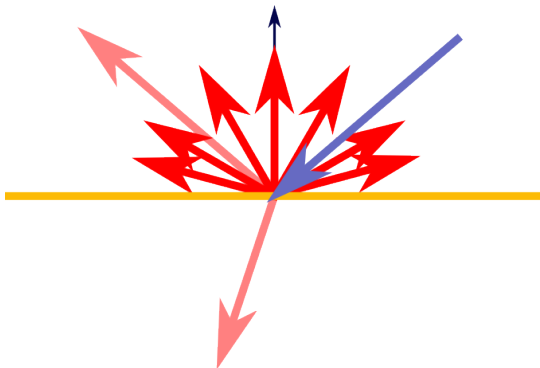
Fénykomponensek

- ▶ A fény útját kétféle komponensre bontjuk:
 - ▶ *Koherens* komponens:
 - ▶ Az optikának megfelelő ideális visszaverődés („tükröződés”) és törés
 - ▶ Rekurzívan továbbkövetjük a fény útját
 - ▶ *Inkoherens* komponens:
 - ▶ Minden egyéb
 - ▶ Ezek közül mi csak az absztrakt fényforrás direkt megvilágítását vesszük figyelembe

Koherens komponens



Inkoherens komponens



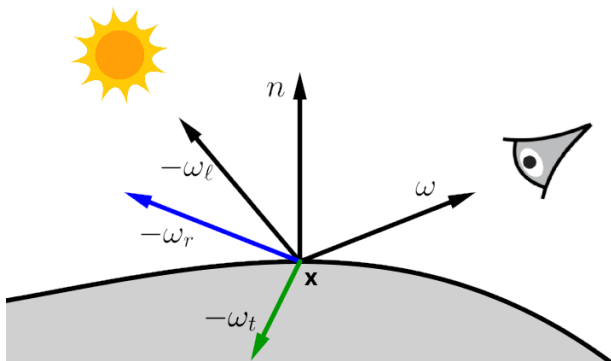
Jelölés

- ▶ Két vektor skalár szorzatát az egyszerűség kedvéért most $\mathbf{a} \cdot \mathbf{b}$ -vel fogjuk jelölni
- ▶ Az *irányok* jelölésére ω, ω' stb. betűket használjuk, de ezek továbbra is egység hosszú vektorok, azaz $\omega \in \mathbb{R}^3 : |\omega| = 1$

Egyszerűsített illuminációs egyenlet

A következő, egyszerűsített megvilágítási egyenletet oldjuk meg:

$$\begin{aligned} L(\mathbf{x} \rightarrow \omega) = & \\ & L_e(\mathbf{x} \rightarrow \omega) + k_a \cdot L_a + \sum_{\ell \in \text{Lights}} f_r(\omega_\ell \rightarrow \mathbf{x} \rightarrow \omega) L_i(\omega_\ell \rightarrow \mathbf{x})(-\omega_\ell \cdot \mathbf{n}) \\ & + k_r \cdot L(\mathbf{x} \leftarrow \omega_r) + k_t \cdot L(\mathbf{x} \leftarrow \omega_t) \end{aligned}$$



Sugárkövetés

$$L(\mathbf{x}, \omega) = L_e(\mathbf{x}, \omega) + k_a \cdot L_a + \sum_{\ell \in \text{Lights}} f_r(\mathbf{x}, \omega_\ell, \omega) L_i(\mathbf{x}, \omega_\ell) (-\omega_\ell \cdot \mathbf{n}) \\ + k_r \cdot L(\mathbf{x}_r, \omega_r) + k_t \cdot L(\mathbf{x}_t, \omega_t)$$

Az $\mathbf{x}$ felületi pontból az ω irányban kibocsátott radiancia

- ▶ A szempozícióból sugarakat indítunk minden pixelen keresztül (például a pixel középpontján át)
- ▶ A sugarak irányát $-\omega$ -val jelöljük (**mínusz ómega!**)
- ▶ A sugár és a színtér objektumainak szemhez legközelebbi metszéspontja adja meg $\mathbf{x}$ -et

Emisszió

$$L(\mathbf{x}, \omega) = L_e(\mathbf{x}, \omega) + k_a \cdot L_a + \sum_{\ell \in \text{Lights}} f_r(\mathbf{x}, \omega_\ell, \omega) L_i(\mathbf{x}, \omega_\ell) (-\omega_\ell \cdot \mathbf{n}) \\ + k_r \cdot L(\mathbf{x}_r, \omega_r) + k_t \cdot L(\mathbf{x}_t, \omega_t)$$

Ez a tag a felület saját sugárzását – *emisszióját* – írja le az $\mathbf{x}$ felületi pontból az ω irányba

Ambiens fény

$$L(\mathbf{x}, \omega) = L_e(\mathbf{x}, \omega) + k_a \cdot L_a + \sum_{\ell \in \text{Lights}} f_r(\mathbf{x}, \omega_\ell, \omega) L_i(\mathbf{x}, \omega_\ell) (-\omega_\ell \cdot \mathbf{n}) \\ + k_r \cdot L(\mathbf{x}_r, \omega_r) + k_t \cdot L(\mathbf{x}_t, \omega_t)$$

$k_a \in [0, 1]$ a felület, $L_a \in \mathbb{R}_0^+$ a környezet *ambiens* együtthatója. Az egyenlet *ambiens* tagja közelíti azt a fény mennyiséget, ami általánosan jelen van, minden felületet ér, azok helyzetétől és az absztrakt fényforrásoktól függetlenül. Célja a közelítések miatt elhagyott fény mennyiség pótlása.

Fényforrások

$$L(\mathbf{x}, \omega) = L_e(\mathbf{x}, \omega) + k_a \cdot L_a + \sum_{\ell \in \text{Lights}} f_r(\mathbf{x}, \omega_\ell, \omega) L_i(\mathbf{x}, \omega_\ell)(-\omega_\ell \cdot \mathbf{n}) \\ + k_r \cdot L(\mathbf{x}_r, \omega_r) + k_t \cdot L(\mathbf{x}_t, \omega_t)$$

- ▶ A figyelembe vett inkoherens visszaverődéseket foglalja össze a szummás tag
- ▶ Csak a fényforrások direkt hatását vesszük figyelembe
- ▶ És csak akkor, ha az az $\mathbf{x}$ felületi pontból látszik

Fényforrások

$$L(\mathbf{x}, \omega) = L_e(\mathbf{x}, \omega) + k_a \cdot L_a + \sum_{\ell \in \text{Lights}} f_r(\mathbf{x}, \omega_\ell, \omega) L_i(\mathbf{x}, \omega_\ell) (-\omega_\ell \cdot \mathbf{n}) \\ + k_r \cdot L(\mathbf{x}_r, \omega_r) + k_t \cdot L(\mathbf{x}_t, \omega_t)$$

- ▶ ω_ℓ a fényforrásból a felületi pontba mutató egységvektor.
- ▶ $f_r(\mathbf{x}, \omega_\ell, \omega)$ most csak a diffúz és spekuláris visszaverődést jellemző BRDF. (BRDF-ekről később lesz szó részletesebben.)
- ▶ $-\omega_\ell \cdot \mathbf{n}$ a felületi normális és a fényforrás felé mutató vektor által bezárt szög koszinusza $\approx$ egységnyi idő alatt az ω_ℓ -ből mennyi foton esik rá a felületre

Fényforrások

$$L(\mathbf{x}, \omega) = L_e(\mathbf{x}, \omega) + k_a \cdot L_a + \sum_{\ell \in \text{Lights}} f_r(\mathbf{x}, \omega_\ell, \omega) L_i(\mathbf{x}, \omega_\ell) (-\omega_\ell \cdot \mathbf{n}) \\ + k_r \cdot L(\mathbf{x}_r, \omega_r) + k_t \cdot L(\mathbf{x}_t, \omega_t)$$

- ▶ Ha az ℓ fényforrás teljesítménye felénk Φ_ℓ és pozíciója $\mathbf{x}_\ell$ akkor

$$L_i(\mathbf{x}, \omega_\ell) = v(\mathbf{x}, \mathbf{x}_\ell) \cdot \frac{\Phi_\ell}{\|\mathbf{x} - \mathbf{x}_\ell\|^2}.$$

- ▶ $v(\mathbf{x}, \mathbf{x}_\ell) \in [0, 1]$ láthatósági függvény: *Mi van/van-e valami $\mathbf{x}$ és a fényforrás között?*

Fényforrások – négyzetes elhalás

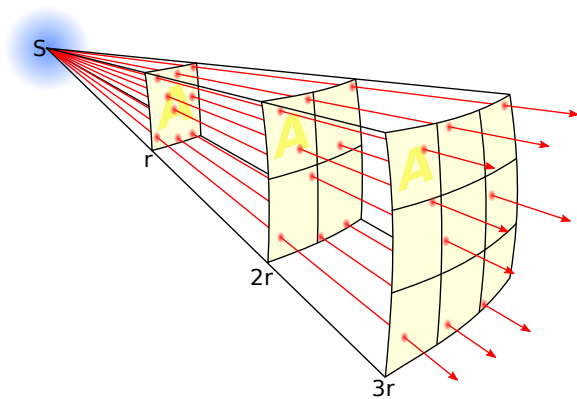
- ▶ Miért a metszéspont és a fényforrás távolságának négyzetével osztunk?
- ▶ Tekintsünk egy pontszerű fényforrást, ami minden irányban egyenletesen sugároz L radianciát
- ▶ Egyre távolodva tőle, a pontszerű fényforrás köré írt gömbön négyzetcentiméterenként mennyi fényt mérnénk? radiancia osztva a távolsághoz tartozó gömb felületével, azaz

$$L_r = \frac{L}{4\pi r^2}$$

- ▶ Tehát két különböző távolságban mért fénymennyiség aránya

$$\frac{L_{r_1}}{L_{r_2}} = \frac{\frac{L}{4\pi r_1^2}}{\frac{L}{4\pi r_2^2}} = \frac{r_2^2}{r_1^2}$$

Fényforrások – négyzetes elhalás



Fényforrások

$$L(\mathbf{x}, \omega) = L_e(\mathbf{x}, \omega) + k_a \cdot L_a + \sum_{\ell \in \text{Lights}} f_r(\mathbf{x}, \omega_\ell, \omega) L_i(\mathbf{x}, \omega_\ell) (-\omega_\ell \cdot \mathbf{n}) \\ + k_r \cdot L(\mathbf{x}_r, \omega_r) + k_t \cdot L(\mathbf{x}_t, \omega_t)$$

$v(\mathbf{x}, \mathbf{x}_\ell) \in [0, 1]$ függvény

- ▶ $= 0$, ha a fényforrás nem látható $\mathbf{x}$ -ből,
- ▶ $= 1$, ha igen,
- ▶ $\in (0, 1)$, ha áttetsző objektumok vannak a kettő között.
- ▶ v kiszámításához úgynevezett *árnyéksugarat* indítunk $\mathbf{x}$ -ből $\mathbf{x}_\ell$ -felé, és az objektumokkal való metszését nézzük.

Tükröződés

$$L(\mathbf{x}, \omega) = L_e(\mathbf{x}, \omega) + k_a \cdot L_a + \sum_{\ell \in \text{Lights}} f_r(\mathbf{x}, \omega_\ell, \omega) L_i(\mathbf{x}, \omega_\ell) (-\omega_\ell \cdot \mathbf{n}) \\ + k_r \cdot L(\mathbf{x}_r, \omega_r) + k_t \cdot L(\mathbf{x}_t, \omega_t)$$

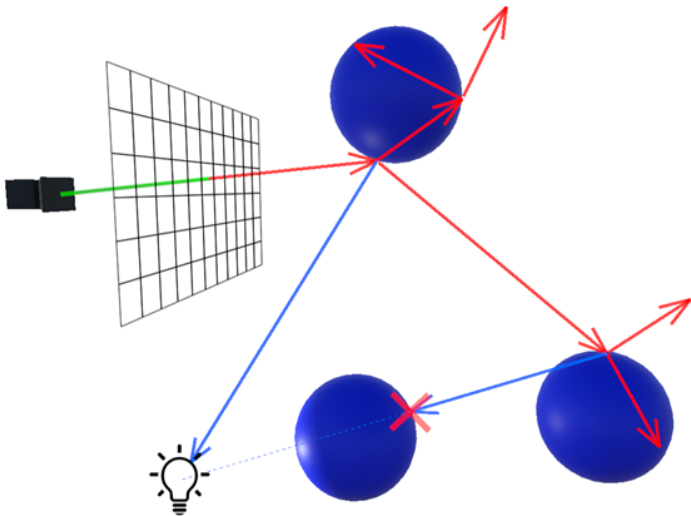
- ▶ A tükrőirányból érkező fényt k_r arányban vesszük figyelembe (k_r a tükrözési Fresnel-együttható).
- ▶ ω_r az ideális tükrőiránynak megfelelő **beeső** vektor.
- ▶ Az $\mathbf{x}_r$ az $\mathbf{x}$ -ből induló, $-\omega_r$ irányvektorú sugár legközelebbi metszéspontja színtérbeli elemmel.
- ▶ $L(\mathbf{x}_r, \omega_r)$ kiszámítása azonos $L(\mathbf{x}, \omega)$ kiszámításával (rekurzió!).
- ▶ Új sugár: szempozíció helyett $\mathbf{x}$ -ből indul, iránya $-\omega_r$.

Fénytörés

$$L(\mathbf{x}, \omega) = L_e(\mathbf{x}, \omega) + k_a \cdot L_a + \sum_{\ell \in \text{Lights}} f_r(\mathbf{x}, \omega_\ell, \omega) L_i(\mathbf{x}, \omega_\ell) (-\omega_\ell \cdot \mathbf{n}) \\ + k_r \cdot L(\mathbf{x}_r, \omega_r) + k_t \cdot L(\mathbf{x}_t, \omega_t)$$

- ▶ A törési-irányból érkező fényt k_t arányban vesszük figyelembe (k_t a törési Fresnel-együttható).
- ▶ ω_t a törésiránynak megfelelő **beeső** vektor.
- ▶ Az $\mathbf{x}_t$ az $\mathbf{x}$ -ből induló, $-\omega_t$ irányvektorú sugár legközelebbi metszéspontja színtérbeli elemmel.
- ▶ $L(\mathbf{x}_t, \omega_t)$ kiszámítása megint azonos $L(\mathbf{x}, \omega)$ kiszámításával (rekurzió!).
- ▶ Új sugár: szempozíció helyett $\mathbf{x}$ -ből indul, az iránya pedig $-\omega_t$.

Rekurzív sugárkövetés – koherens komponensek



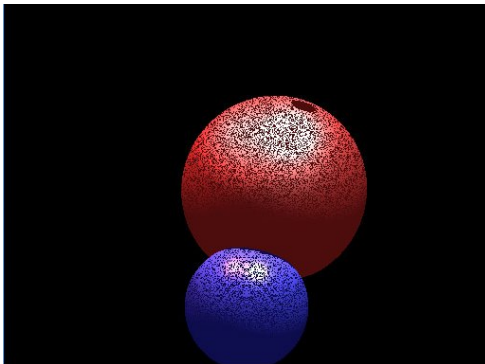
Megjegyzés – rekurzió

- ▶ Az első metszéspont megtalálása után a feladatunk, hogy meghatározzuk az innen a kamera felé induló fénymennyiséget – ezt írja le az illuminációs egyenlet
- ▶ Az egyenlet jobb oldalán a koherens komponensek az L függvényre hivatkoznak, amit éppen ki akarunk számítani, csak éppen más paraméterekkel – itt történik rekurzió újabb sugarak indításával
- ▶ A rekurzió leállítást általában egy maximális „mélységgel” szokás korlátozni
- ▶ A rekurzió leállítható korábban is, ha a számítandó sugár hozzájárulása a végső fénymennyiséghez elenyésző – a különböző k_r , k_t együtthatók szorzódnak össze a rekurzió során
- ▶ A fényforrások hozzájárulásának számításakor a v kiszámításához szintén sugarakat (árnyéksugarat) indítunk, ez azonban nem rekurzív!

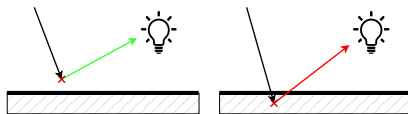
Megjegyzés – színek

- ▶ A képletek csak fénymennyiséggel számolnak, így valójában egyetlen hullámhosszon számítottuk ki a kamerába érkező fényt
- ▶ A hullámhossztól függ majdnem minden:
 - ▶ A felület és a fényforrások által kibocsátott fény: L_e, L_i
 - ▶ A felület fényvisszaverő tulajdonsága (BRDF, közelítő és koherens együtthatók): f_r, k_a, k_r, k_t
 - ▶ Az ideális törésirány: ω_t
- ▶ Tehát el kell végeznünk a számítást a látható fénytartomány minden hullámhosszára
- ▶ Ehelyett rendszerint csak néhány hullámhosszon számolunk
- ▶ Gyakorlatban sokszor csak az RGB színhármasnak megfelelően (ez egy durva közelítés!)

Megjegyzés – ???



Megjegyzés – önárnyékolás

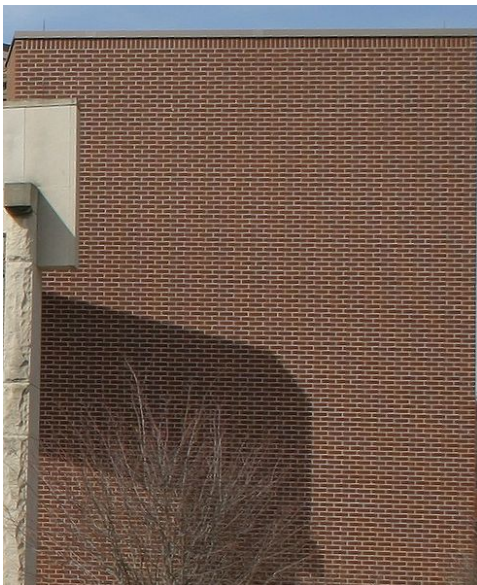


- ▶ A numerikus pontatlanságok miatt előfordulhat, hogy a metszéspont valójában kissé a testen belül van
- ▶ Ilyenkor a fényforrások felé lőtt sugarak beleütközhetnek a kiindulási felületbe!
- ▶ Megoldás: toljuk el a sugár kezdőpontját normális irányában
- ▶ Figyeljünk arra is, hogy mi is kell pontosan: a metszéspont helye(i) vagy elég-e maga a metszés ténye?

Megjegyzés – aliasing

- ▶ Egyenközű pontonkénti mintavételezést csinálunk lényegében
→ a mintavételezési frekvenciánál gyorsabb változások
alias-olnak, azaz nem létező, alacsonyabb frekvenciás
jelkomponensként regisztráljuk őket

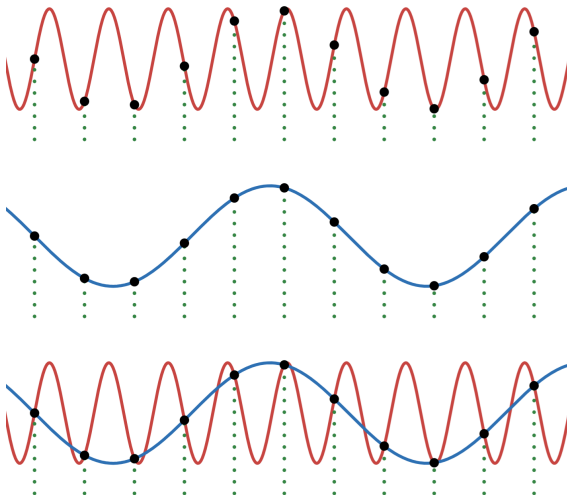
Megjegyzés – aliasing



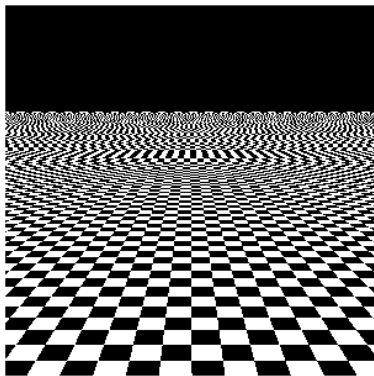
Megjegyzés – aliasing



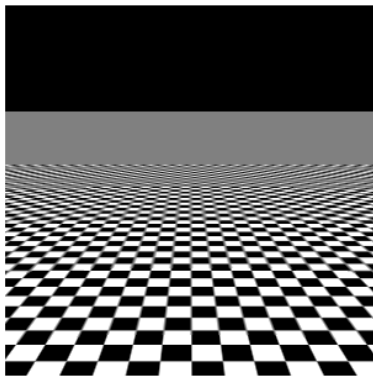
Megjegyzés – aliasing



Mintavételezés grafikában – alias



<https://textureingraphics.wordpress.com/what-is-texture-mapping/anti-aliasing-problem-and-mipmapping/>



- ▶ Bal: alias
- ▶ Jobb: egy lehetséges megoldás Mipmap segítségével (később)

Mintavételezés grafikában – alias

- ▶ Eddig csak pixelenként („ablakonként”) egyetlen sugarat indítottunk
- ▶ Ha többet indítunk, akkor a Nyquist-frekvencia nő
- ▶ Pixelenként egy szín kell csak $\rightarrow$ a sugarak által behozott különböző színeket összegezni kell valahogy (pl. átlagolni)
- ▶ De a több sugár eredményét is többféleképpen összegezzhetjük (vagyis: szűrhetjük)

Mintavételezés grafikában – alias

- ▶ Egyenletes mintavételezéssel az alias nem szüntethető meg (csak ha a bejövő jel garantáltan nem tartalmaz túl magas frekvenciás komponenseket)
- ▶ Azonban ha nem egyenletes a mintavételezés, hanem megfelelő eloszlás szerint történik, akkor az alias helyett véletlenszerű, nagyfrekvenciás zaj lesz a képen
- ▶ Ehhez a szemünk már alkalmazkodott!

Megjegyzés

- ▶ Vegyük észre: a fényt fent végig részecskeként kezeltük
- ▶ Emiatt a fény hullám természetéből adódó jelenségeket (interferencia, diffrakció stb.) nem tudjuk visszaadni
- ▶ Mikor számítanak ezek?
 - ▶ Az interferencia miatt látjuk olyan színben a páva tollait vagy a szappanbuborék felszínét amilyenben látjuk
 - ▶ A diffrakció pedig a finom árnyékjelenségek egy részében játszik szerepet

Metszészvizsgálat gyorsítása – motiváció

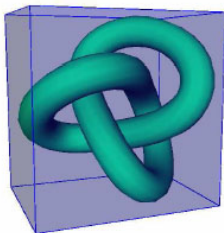
- ▶ Az algoritmus sebessége leginkább a *metszészvizsgálat* sebességétől függ.
- ▶ Hogyan gyorsíthatnánk ezt?
- ▶ Ne vizsgáljunk metszést olyan objektumokra, amiket biztosan nem metsz a sugár!
- ▶ Ne vizsgáljunk metszést olyan objektumokra, amik biztosan távolabbi metszéspontot adnak, mint a már megtalált!

Befoglaló keretek

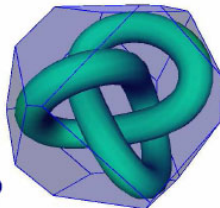
- ▶ Minden objektumot vegyünk körbe valamilyen *kerettel* amivel gyorsan lehet metszést számolni.
- ▶ Ha egy sugár metszi az objektumot, akkor biztosan metszi a keretet is → tehát ha a befoglaló keretet nem metszi a sugár, akkor nem kell vizsgálni az objektummal a metszést
- ▶ Akkor lesz hatékony, ha fordítva is minél nagyobb a valószínűsége. (Minél jobban közelítse a befoglaló test az igazit)
- ▶ Befoglaló gömb: másodfokú egyenlet megoldás.
- ▶ Befoglaló doboz: élei a tengelyekkel párhuzamosak, gyorsan számítható, lásd előző előadás!

További befoglalók – konvex poliéderek

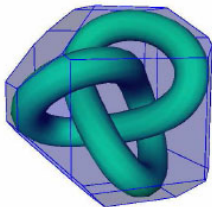
6-DOP
(AABB)



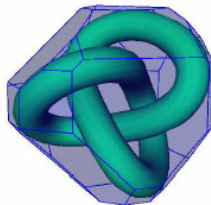
14-DOP



18-DOP



26-DOP



DOP: discrete oriented polytope

További befoglalók – konvex poliéderek

- ▶ Egy n -oldalú konvex poliéderek felírható n féltér metszeteként (féltér: $ax + by + cz + d \leq 0$)
- ▶ Azaz az oldallapok síkjait kifelé mutató normálisokkal felírhatjuk

$$a_i x + b_i y + c_i z + d_i = 0, \quad i = 1.., n$$

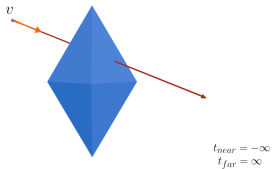
alakban

- ▶ A korábban látott sugár-AAB metszés könnyen általánosítható erre az esetre!
- ▶ Legyen a sugár $\mathbf{p}(t) = \mathbf{p}_0 + t\mathbf{v}$

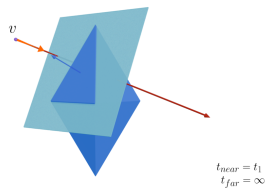
Sugár metszése konvex poliéderrel

1. Legyen $t_{near} := -\infty$, $t_{far} := \infty$
2. Minden $i \in \{1, 2, \dots, n\}$ oldallapra:
legyen a lap $a_i x + b_i y + c_i z + d_i = 0$, egyenlete olyan, hogy az $\mathbf{n}_i = [a_i, b_i, c_i]^T$ lapnormális a poliéderből kifelé mutat
 - 2.1 Számítsuk ki a sugár és az oldallap síkjának t_i metszéspontját
 - 2.2 **Ha** $\mathbf{v} \cdot \mathbf{n}_i < 0$, **akkor** $t_{near} := \max\{t_{near}, t_i\}$
 - 2.3 **Különben** $t_{far} := \min\{t_{far}, t_i\}$
3. **Ha** $t_{near} > t_{far}$, **akkor** nincs metszéspont
4. **Különben** a sugár egyenese t_{near} -nél lép be és t_{far} -nál hagyja el a konvex poliédert (azaz metszi a sugár a konvex poliédert, ha $[t_{near}, t_{far}] \cap \mathbb{R}^+ \neq \emptyset$)

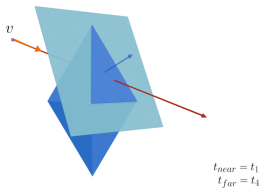
Sugár metszése konvex poliéderrel



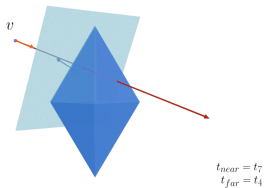
$i = 1$



$i = 4$



$i = 7$



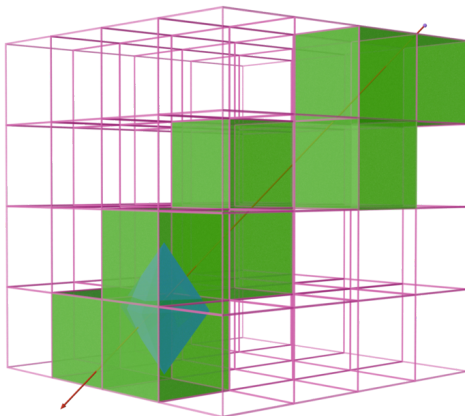
Hierarchikus befoglaló keretek

- ▶ A kisebb kereteket nagyobb keretekbe fogjuk össze.
- ▶ Fa struktúrát kapunk.
- ▶ Egy részfat csak akkor kell kiértékelni, ha a gyökérrel van metszés.
- ▶ Angolul *Bounding Volume Hierarchy* – BVH

Szabályos felosztás

- ▶ Egy szabályos 3D ráccsal lefedjük az egész színteret.
- ▶ Előfeldolgozás: minden cellához feljegyezzük a beletartozó objektumokat.
- ▶ Használat: csak azokkal az objektumokkal végzünk sugár metszést, amiknek a celláján a sugár áthalad
- ▶ Előnye: a vizsgálandó cellák gyorsan számíthatók szakaszrajzoló algoritmussal! (Lásd később a raszterizációnál)
- ▶ Hátránya: feleslegesen sok cella – nagy részük üres teret fed le.

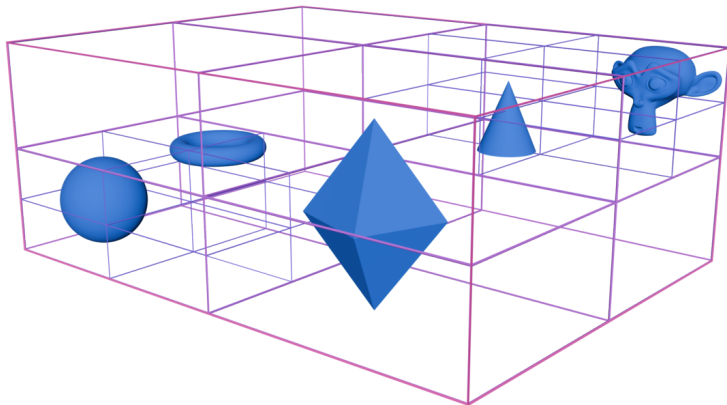
Szabályos felosztás



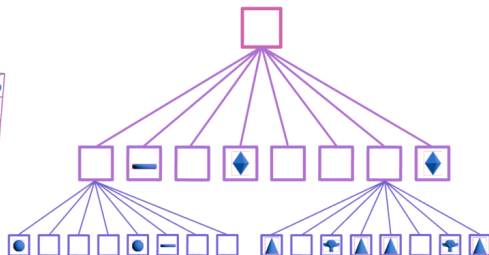
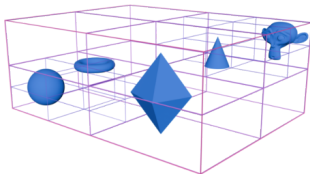
Oktális fa

- ▶ Fa gyökere: a teljes színteret magában foglaló tengelyekkel párhuzamos élű befoglaló doboz (*AABB*)
- ▶ Vágjuk ezt nyolc egyenlő részre!
- ▶ Minden új dobozra: ha *elég sok* objektum van benne, akkor tovább osztjuk, különben megállunk.
- ▶ Előny: az üres részeket nem osztjuk tovább feleslegesen.
- ▶ Hátrány: bonyolultabb bejárás.
- ▶ Hátrány: a fa mélysége elszállhat → a gyakorlatban egy előreírt mélység is adott, amit elérve már nem osztjuk tovább a cellákat (még ha több objektum is jutna oda, mint a maximum)

Octree – Oktális fa



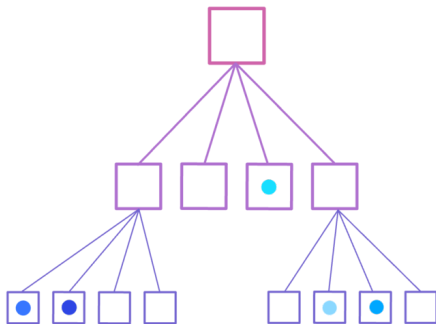
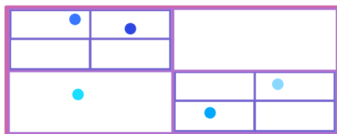
Octree – Oktális fa



Quadtree

- ▶ Az oktális fa síkbeli változata
- ▶ Az aktuális cellát mindig négy egyenlő részre osztjuk, a tengelyekkel párhuzamosan
- ▶ Használható 3D-ben is, pl. a függőleges irány figyelmen kívül hagyásával

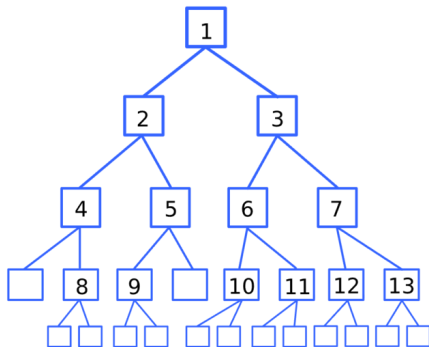
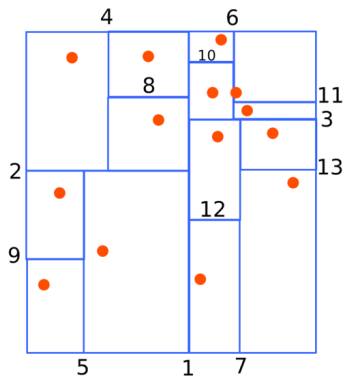
Quadtree



kd-fa

- ▶ Probléma az oktális fával: mindig középen és minden sík mentén vág – nem veszi figyelembe az objektumokat.
- ▶ Oktális fa: keresési idő $\approx$ fa magassága. DE! az oktális fa kiegyensúlyozatlan.
- ▶ kd-fa: minden lépésben egyetlen síkkal vágunk, ami egy tengelyre merőleges.
- ▶ Sorrend: $X, Y, Z, X, Y, Z, \dots$
- ▶ Felező sík elhelyezése (heurisztika):
 - ▶ térbeli középvonal módszer
 - ▶ test középvonal módszer
 - ▶ költség modell alapú módszer

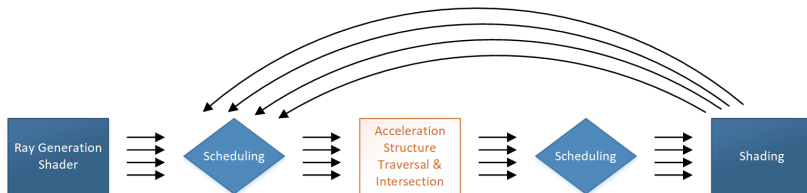
kd-fa



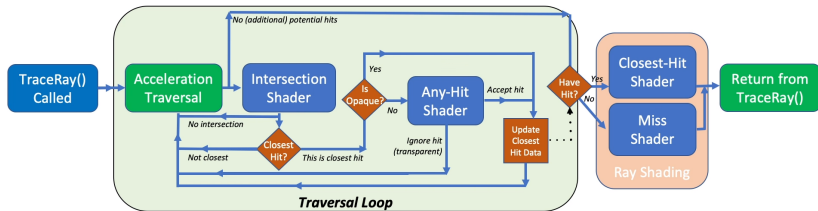
*DXR

- ▶ 2018 óta NVIDIA GPU-kban hardveresen gyorsított raytracing érhető el
- ▶ Ez a DX12 szabvány része lett
- ▶ Compute shaderes fallback-kel kibővített támogatás van Pascal architektúráig lefelé
- ▶ De valódi hardveres raytracing csak Volta, Turing és Ampere és újabb architektúrákon van

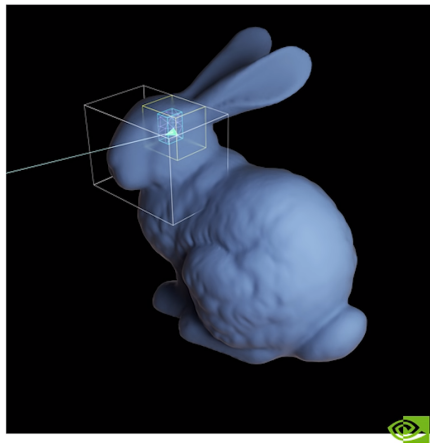
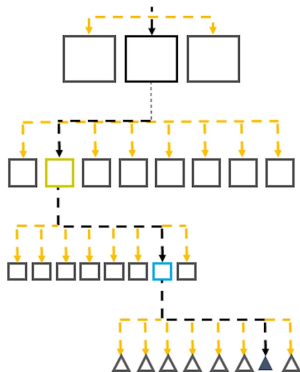
*HW RT



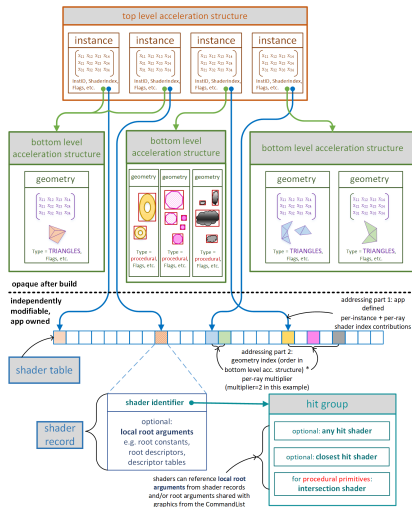
*HW RT



*HW RT – Bounding Volume Hierarchy



*HW RT



*DXR 1.0

5 új shader típus:

- ▶ **Raygeneration shader:** sugarak előállítását végző shader
- ▶ **Intersection shader:** procedurális geometriákra; van beépített hardveres sugár-háromszög metszés, nem kell feltétlen megírni (és nem is biztos, hogy akarnátok - ld. Watertight Ray/Triangle Intersection)
- ▶ **Closest hit shader:** lényegében a fragment shader megfelelője; a legközelebbi metszéspontra egyszer lefutó shader
- ▶ **Anyhit shader:** átlátszósági tesztek eldöntésére meghívódó shader bejárás közben; többször is meghívódhat; a meghívások sorrendjére nincs semmilyen garancia
- ▶ **Miss shader:** ha nem volt metszett geometria, akkor ez hívódik meg (itt számíthattok például égbolt színt vagy amit akartok)

*DXR 1.1

Ami igazán érdekes, hogy innentől bármilyen shader-ben lehet hívni TraceRay parancsot (nem csak grafikus szerelőszalagbeliből, hanem compute shader-ből is!).

Illetve DispatchIndirect is van, ami rengeteg új lehetőséget ad.