Számítógépes Grafika

Hajder Levente és Baráth Dániel
hajder@inf.elte.hu

Eötvös Loránd Tudományegyetem
Informatikai Kar

2019/2020. I. félév

Tartalom

Tartalom

Grafikus szerelőszalag

- A színterünkről készített kép elkészítésének műveletsorozatát nevezik grafikus szerelőszalagnak (angolul *graphics pipeline*)
- A valósidejű alkalmazásoknak lényegében a színterünk leírását kell csak átadnia, a képszintézis lépéseit a grafikus szerelőszalag végzi
- A szerelőszalagban több koordináta-rendszer váltás is történik - mindent feladatot a hozzá legjobban illeszkedő rendszerben próbálunk elvégezni

Grafikus szerelőszalag

- Lépései főbb műveletek szerint:
  - Modellezési transzformáció
  - Nézeti transzformáció
  - Perspektív transzformáció
  - Vágás
  - Homogén osztás
  - Raszterizáció
  - Megjelenítés
- A grafikus szerelőszalag eredménye egy kép (egy kétdimenziós pixeltömb, aminek minden elemében egy színérték található)

Transzformációk

- A szerelőszalag transzformációinak feladata: a modelltérben adott objektumot "eljuttatni" a képernyő-térbe
- Lépései:

modell k.r.
→ világ k.r.
→ kamera k.r.
→ normalizált eszköz k.r.
→ képernyő k.r.

Szerelőszalag bemeneti adatai

- Ábrázolandó tárgyak *geometriai modellje*
- *Virtuális kamera* adatai (nézőpont és látógúla)
- A képkeret megadása (az a pixeltömb, amire a színterünk síkvetületét leképezzük)
- A színtérben található fényforrásokhoz és anyagokhoz tartozó *megvilágítási adatok*

Pipeline

- Minden egyes primitív végigmegy az összes lépésen
- Az egyes lépések az eredményüket a következőnek továbbítják
- Többféleképpen megadható és csoportosítható - mi csak egy példát írtunk fel (pl. hol "színezünk"?)

Koordináta-rendszerek

- Modell KR:
  Az objektumok saját koordináta-rendszerükben adottak.
- Világ KR:
  Az objektumok egymáshoz viszonyított helyzete itt adott, ahogy a kamera/szem pozicó és az absztrakt fényforrások pozicója is.
- Kamera KR:
  A koordináták a kamera poziciója és orientációjához relatíven adottak.

Koordináta-rendszerek

- Normalizált eszköz KR:
  A hardverre jellemző, $[-1,1] \times [-1,1] \times [0,1]$ kiterjedésű KR.
- Képernyő KR:
  A megjelenítendő képnek (képernyő/ablak) megfelelő KR (balkezes, bal-felső "sarok" az origó).

Tartalom

Modellezési transzformácó

- A saját (modell) koordináta-rendszerben adott modelleket a világ-koordináta rendszerben helyezi el
- Tipikusan minden modellre különböző (lehet a színterünk két eleme csak a világtrafóban különbözik!)
- Jellemzően affin transzformációk
- Gyakorlatban: ez a *Model* (vagy *World*) mátrix a kódjainkban

Tartalom

Nézeti transzformáció

- A világ koordináta-rendszert a kamerához rögzített koordináta-rendszerbe viszi át.
- A transzformáció a kamera tulajdonságaiból adódik.
- Gyakorlatban: ez a *View* mátrix.

Kamera transzformácó

- Tulajdonságok, mint sugárkövetés esetén: **eye**, **center**, **up**
- Ebből kapjuk a nézeti koordináta-rendszer tengelyeit:

$$\mathbf{w} = \frac{\mathbf{eye} - \mathbf{center}}{|\mathbf{eye} - \mathbf{center}|}$$

$$\mathbf{u} = \frac{\mathbf{up} \times \mathbf{w}}{|\mathbf{up} \times \mathbf{w}|}$$

$$\mathbf{v} = \mathbf{w} \times \mathbf{u}$$

Kamera transzformácós mátrix

- Mátrixot kapjuk: áttérés az $-\mathbf{eye}$ origójú, $\mathbf{u}, \mathbf{v}, \mathbf{w}$ koordinátarendszerbe:

$$T_{View} = \begin{bmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ w_x & w_y & w_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & -eye_x \\ 0 & 1 & 0 & -eye_y \\ 0 & 0 & 1 & -eye_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Tartalom

Párhuzamos vetítés

- A mátrix ami megadja egyszerű, például az XY síkra való vetítés

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

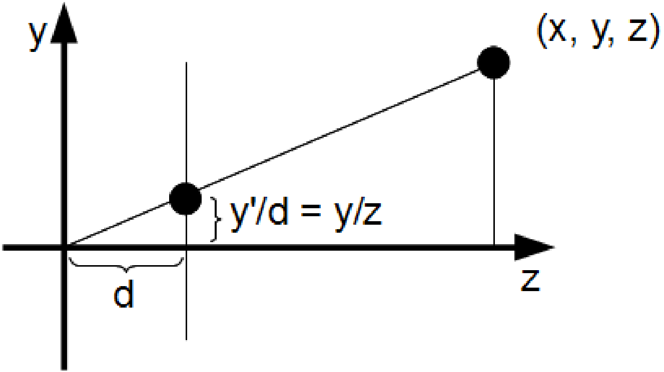
Perspektív transzformácó

- Emlékeztető: 3. EA
- A nézeti csonkagúla által határolt térrészt normalizált eszköz KR-be viszi át
- Ami benne volt a csonkagúlában, az lesz benne a $[-1,1]\times[-1,1]\times[0,1]$ (vagy $[-1,1]\times[-1,1]\times[-1,1]$) tartományban
- A transzformáció a kamerán átmenő *vetítő sugarakból* párhuzamosokat csinál
- A transzformáció a kamerapozíciót a végtelenbe tolja
- Gyakorlatban: ez a *Projection* mátrix

Perspektív transzformácó

- Emlékeztető: tulajdonságok
  - függőleges és vízszintes nyílásszög *(fovx, fovy)* vagy az alap oldalainak az aránya és a függőleges nyílásszög *(fovy, aspect)*,
  - a közeli vágósík távolsága *(near)*,
  - a távoli vágósík távolsága *(far)*

Középpontos vetítés



Középpontos vetítés

- Vagyis:

$$x' = \frac{x}{z} d$$
$$y' = \frac{y}{z} d$$
$$z' = \frac{z}{z} d = d$$

Középpontos vetítés

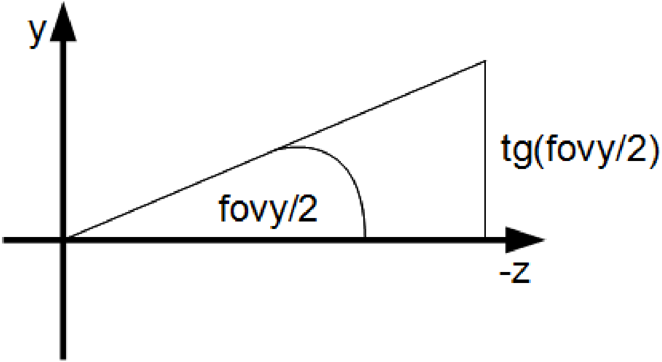
- Az origó, mint vetítési középpont és egy, attól a Z tengely mentén d egységre található, XY síkkal párhuzamos vetítősíkra való vetítés mátrixa:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{d} & 0 \end{bmatrix}$$

- Homogén osztás után ($\frac{z}{d}$-vel) a fentit kapjuk

Normalizált látógúla

- Figyeljünk: a szerelőszalagunk ezen pontján a kamera $-Z$ felé néz és az origóba van
- A fenti térből térjünk át egy "normalizáltabb" gúlába - aminek nyílásszöge x és y mentén is 90 fokos!



Normalizált látógúla

- Mátrix alakban:

$$\begin{bmatrix} 1/\tan\frac{fovx}{2} & 0 & 0 & 0 \\ 0 & 1/\tan\frac{fovy}{2} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Normalizált látógúla

- Ezután már csak a közeli és a távoli vágosík z koordinátáit kell a normalizálásnak megfelelően átképezni ($-1, 1$ vagy $0, 1$-re):

$$T_{Projection} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{far}{far - near} & \frac{near * far}{near - far} \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Tartalom

Vágás

- Cél: ne dolgozzunk feleslegesen azokkal az elemekkel, amikből nem lesz pixel (nem jelennek majd meg).
- Megoldás (kísérlet):
  1. Végezzük el a homogén osztást!
  2. Vágjunk le, ami kilóg a $[-1, 1] \times [-1, 1] \times [0, 1]$-ből!
- Probléma: vegyünk egy olyan szakaszt, ami átlóg a kamera mögé!
- Ez a szakasz átmegy egy ideális ponton $\Rightarrow$ a szakasz képe nem egyezik meg a transzformált pontokat összekötő szakasszal!
- Ez az *átfordulási probléma*.

Vágás homogén koordinátákban

- Megoldás (valóban): Vágás homogén koordinátákban
- Legyen: $[x_h, y_h, z_h, h] = \mathbf{M}_{\text{Proj}}[x_c, y_c, z_c, 1]$
- Cél: $(x, y, z) := (x_h/h, y_h/h, z_h/h) \in [-1, 1] \times [-1, 1] \times [0, 1]$, azaz

Legyen $h > 0$, és

$$-1 < x < 1$$
$$-1 < y < 1$$
$$0 < z < 1$$

Ebből kapjuk:

$$-h < x_h < h$$
$$-h < y_h < h$$
$$0 < z_h < h$$

Tartalom

1. Tartalom
   - Motiváció
2. Grafikus szerelőszalag
   - Áttekintés
   - Modellezési transzformácó
   - Nézeti transzformácó
   - Perspektív transzformácó
   - Vágás
   - Raszterizáció
   - Megjelenítés
     - Triviális hátlapeldobás
     - Festő algoritmus
     - Z-buffer
3. Lokális illumináció

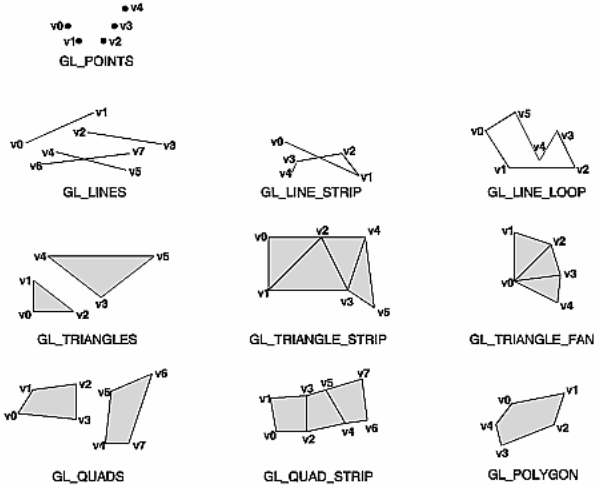
Raszterizáció

- Ne feledjük: eddig minden primitív, amiről beszéltünk folytonos objektum volt
- Azonban nekünk egy diszkrét térben, a képernyő képpontjain kell dolgoznunk
- A primitívek folytonos teréből át kell térni ebbe a diszkrét térbe, ezt hívják *raszterizációnak*

Raszterizáció

- Olyan geometriai primitíveket kell választanunk, amelyeket gyorsan tudunk raszterizálni
- Mi lehet ilyen? Jó lenne pl. ha egyik pixelhez tartozó felületi pontjának koordinátái alapján könnyen számítható lenne a szomszédos pixelekhez tartozó pontok koordinátái, illetve ha síkbeli is lenne...
- A háromszög ilyen!
- Minden egyéb felületet ilyen primitívekkel (lényegében: síklapokkal) közelítünk ← *tesszeláció*

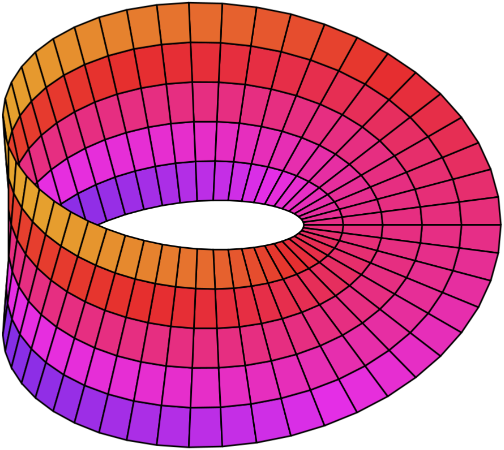
Raszterizáció



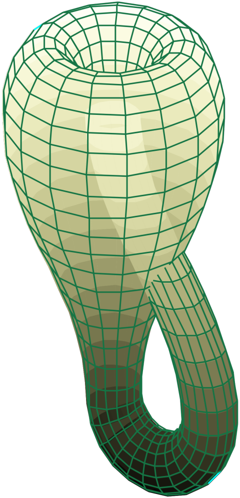
Tesszeláció

- Vigyázzunk: "szép" (teljes oldalakban illeszkedő), 6-reguláris háromszög vagy "szép", 4-reguláris négyszöghálóval nem lehet bármit lefedni degenerált esetek nélkül!
- A fenti reguláris topológiákkal a végtelen síklap, vagy a végtelen hengerpalást, vagy pedig a tórusz topológiájának megfelelő felületek írhatóak le.

Tesszeláció



Tesszeláció



Tartalom

Takarási feladat

- Feladat: eldönteni, hogy a kép egyes részein milyen felületdarab látszik.
- Objektum tér algoritmusok:
    - Logikai egységenként dolgozunk, nem függ a képernyő felbontásától.
    - Rossz hír: nem fog menni.
- Képtér algoritmusok:
    - Pixelenként döntjük el, hogy mi látszik.
    - Ilyen a sugárkövetés is.

Triviális hátlapeldobás, *Back-face culling*

- Feltételezés: Az objektumaink "zártak", azaz ha nem vagyunk benne az objektumban, akkor sehonnan sem láthatjuk a felületét belülről.
- Körüljárási irány: rögzítsük, hogy a poligonok csúcsait milyen sorrendben *kell* megadni:
  - óramutató járásával megegyező (*clockwise, CW*)
  - óramutató járásával ellentétes (*counter clockwise, CCW*)
- Ha a transzformációk után a csúcsok sorrendje nem egyezik meg a megadással, akkor a lapot *hátulról* látjuk $\Rightarrow$ nem kell kirajzolni, *eldobható*.

Festő algoritmus

- Rajzoljuk ki hátulról előre haladva a poligonokat!
- Ami közelebb van, azt később a rajzoljuk $\Rightarrow$ ami takarva van, takarva lesz.
- Probléma: hogyan rakjuk sorrendbe a poligonokat?
- Már háromszögeknél is van olyan eset, amikor nem lehet sorrendet megadni.

Festő algoritmus

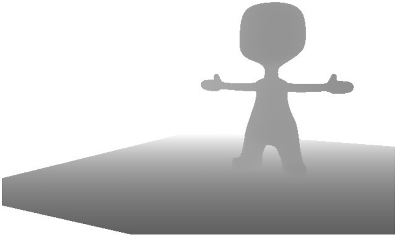
Z-buffer algoritmus

- Képtérbeli algoritmus
- Minden pixelre nyílvántartjuk, hogy ahhoz milyen mélységérték tartozott.
- Ha megint újra erre a pixelre rajzolnánk (*Z-test*):
    - Ha az új Z érték mélyebben van, akkor ez e pont takarva van $\Rightarrow$ nem rajzolunk
    - Ha a régi Z érték mélyebben van, akkor az új pont kitakarja azt $\Rightarrow$ rajzolunk, eltároljuk az új Z értéket.

Z-buffer

- Z-buffer vagy *depth buffer*: külön memóriaterület.
- Képernyő/ablak méretével megegyező méretű tömb.
- Pontosság: a közeli és távoli vágósík közti távolságtól függ.
- Minden pixelhez tartozik egy érték a bufferből.
- Ezzel kell összehasonlítani, és ide kell írni, ha a pixel *átment a Z-teszten*.
- Gyakorlatban:
  - 16-32 bites elemek
  - Hardveres gyorsítás
  - Pl: közeli vágósík: t, távoli: $1000t$, akkor a Z-buffer 98%-a a tartomány első 2%-át írja le.

Z-buffer



by macouno, macouno.com

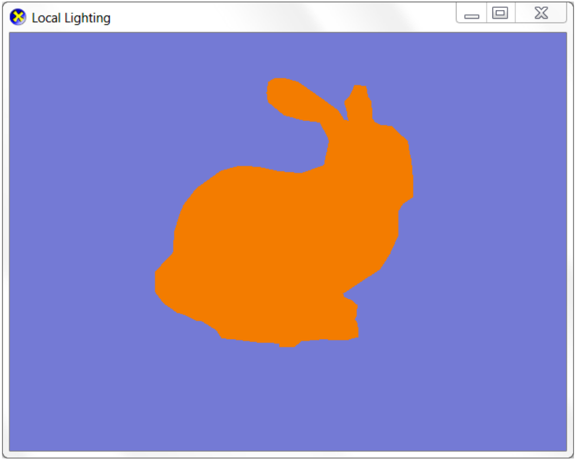
Lokális illumináció

- Ha már megvannak a primitívjeink pixelekre való leképezései, valahogyan számítsunk színeket

Saját színnel árnyalás

- Minden objektumhoz/primitívhez egy színt rendelenünk, és kirajzoláskor el lesz a pixelek értéke.
- Leggyorsabb: az iluminácó gyakorlatilag egyetlen értékadás.
- Borzasztó: se nem valósághű, se nem szép.

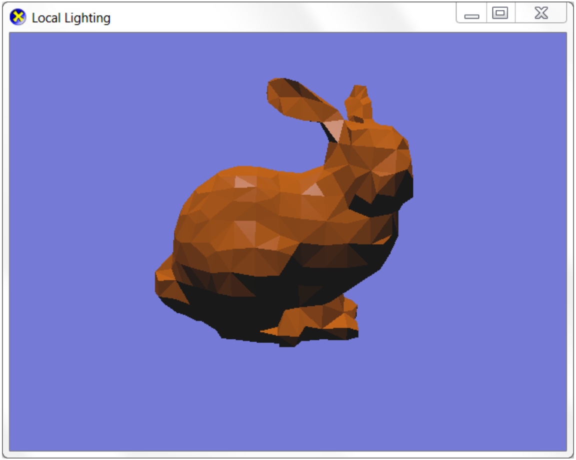
Saját színnel árnyalás



Konstans árnyalás, *Flat shading*

- A megvilágítást poligononként egyszer számítjuk ki, a szín homogén a lapon belül.
- Gyors: a műveletek száma a poligonok számától függ, a pixelek számától független.
- Van hogy használható: íves részeket nem tartalmazó, diffúz, egyszinű objektumokra

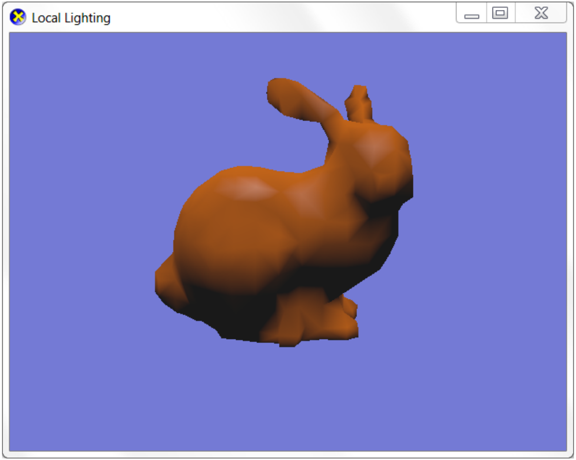
Konstans árnyalás



Gouraud árnyalás

- A megvilágítást csúcspontonként számítjuk ki, a lapon lineáris interpolációval számítjuk a színeket.
- Lassabb: N db megvilágítás számítás + minden pixelre interpoláció.
- Szebb: az árnyalás minősége nagyban függ a poligonok számától. Nagy lapokon nem tud megjelenni a csillanás.

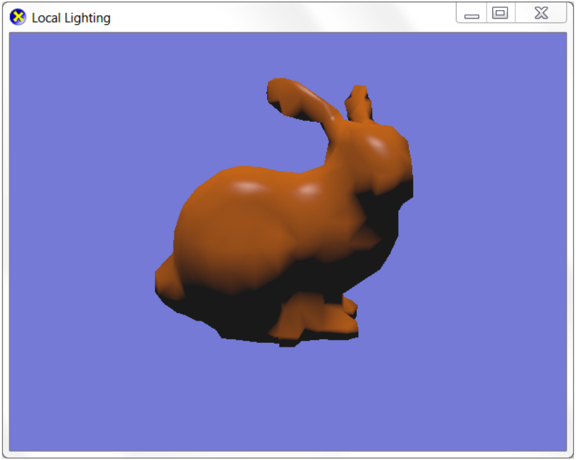
Gouraud árnyalás



Phong árnyalás

- Csak a normálvektorokat interpoláljuk, a megvilágítást minden pixelre kiszámítjuk.
- Leglassabb: *pixelek száma* db megvilágítás számítás.
- Legszebb: az árnyalás minősége nem függ a poligonok számától. Csillanás akár poligon közepén is meg tud jelenni.

Phong árnyalás



Gouraud vs Phong árnyalás