grafika

Nikola Bebić

šta će nama grafika?

šta će nama kako grafika radi?

model

- 3d model: gomila *primitiva*
- koji su pak gomila tačaka (vektora) u 3D prostoru, sa bojom ili teksturom

rasterizacija

- 2d tačke (pikseli) na ekranu sa bojom

plan rada

- **model transform** - model u prostoru
- **world transform** - prostor pred kamerom
- **perspective transform** - kamera na ekran
- **rasterization** - sve to u piksele

linearna algebra

- matrice
- množenje matrica

vektori

- u matematici, član A^n
- za nas: niz od 2, 3, ili 4 broja određenog tipa (najčešće `float`)
- GPU ima specijalnu podršku za njih
  - sabiranje, oduzimanje
  - množenje matricama

vetrex

- ima svoje (x, y, z, w) koordinate
- $w = 0$ ili $w = 1$
- drugačije zapisano kao

$$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

model transform

- objekte treba da "postavimo" na svoj položaj u svetu
- pomeramo koordinatne sisteme, što će se odraziti na sve objekte (i podobjekte) nakon toda

linearne transformacije

- svaka transformacija je linearna: možemo je predstaviti kao matricu
- transformacije primenjujemo množenjem sa odgovarajućom matricom

$$\tau(\mathbf{x}) = A \cdot \mathbf{x}$$

skaliranje

skaliranje

- širenje, skupljanje, preslikavannje oko neke ose
- množenje svake ose nekim skalarom

$$(x, y, z) \rightarrow (s_x x, s_y y, s_z z)$$

- `glScalef( sx, sy, sz );`

skaliranje

$$S(s_x, s_y, s_z) = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

translacija

- pomeranje za određen vektor (t_x, t_y, t_z)

$$(x, y, z) \rightarrow (x + t_x, y + t_y, z + t_z)$$

- `glTranslatef( tx, ty, tz );`

translacija

$$T(t_x, t_y, t_z) = \begin{bmatrix} 0 & 0 & 0 & t_x \\ 0 & 0 & 0 & t_y \\ 0 & 0 & 0 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

šta ako nije $w = 1$?

rotacija

rotacija

- rotacija oko vektora (x, y, z) za ugao θ
- komplikovano ...
  - specijalni slučajevi - oko koordinatnih osa
- `glRotatef( theta, x, y, z );`

rotacija

- oko x-ose:

$$R_x(\theta) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

a sad sve zajedno

- $\tau_A \circ \tau_B = \tau_{A \cdot B}$
- iz asocijativnosti množenja matrica i vektora
- možemo transformacije *"akumulirati"* tako što množimo odgovarajuće matrice
- stek matrica $\Rightarrow$ brzo možemo menjati transformacije

world transform

- treba naš svet da pozicioniramo pred kameru
- kamera uvek stoji u $(0, 0, 0)$ i gleda ka $-z$
- `gluLookAt( ex, ey, ez, lx, ly, lz, ux, uy, uz)`
- uvek je *prva* transformacija koju radimo - da bi bila poslednja primenjena

perspective transform

- *perspektiva* - bliže stvari su veće
- blize stvari zaklanjaju dalje
- *frustrum* - deo sveta koji vidimo
- FoV, aspect ratio, ...
- `gluPerspective( fov, aspect, zNear, zFar )`

perspective transform

- od $[-afz, afz] \times [-fz, fz] \times [-z_n, -z_f]$
- do $[-a, a] \times [-1, 1] \times [-1, 1]$, ali pazeći na perspektivu

perspective transform

- delimo x i y sa z
- homogenous divide: $\mathrm{hdiv}(x, y, z, w) = (\frac{x}{w}, \frac{y}{w}, \frac{z}{w}, 1)$
- stavimo $w = -z$

rasterizacija

- od trougla na 2D ekranu treba doći do pojedinačnih tačaka
- problemi:
  - koje tačke "zahvata" svaki trougao?
  - šta je ispred/iza?
  - šta je koje boje / teksture?
  - ...

crtanje trougla

- metoda 1: "edge walking"
  - loša paralelizacija, komplikovan algoritam
- metoda 2: "brute force"
  - može se paralelizovati
  - mnogo optimizocija

crtanje trougla

- odredimo bounding-box (brzo)
- izračunamo jednačine linija za svaku stranicu
- za svaki piksel u BB:
  - proverimo da li pripada

depth-buffer

- šta je ispred, a šta iza?
- poseban "sloj" za čuvanje daljine
- tačka se crta samo ako je $z_t(x, y) > z_{buf}(x, y)$

depth-buffer

- problem: ne zavisi linearno nakon projekcije
- $1/z$ zavisi linearno!
- možemo iskoristiti homogenizaciju od ranije

boja / tekstura

- linearna interpolacija izmedju temena
- nije idealno, ali sta god

shaderi

- uopštenje čitavog ovog procesa
  - vertex shader: 3D do 2D
  - fragment shader: 2D do pixel (fragment)
  - ...

osvetljenje

- teško da se uradi kako treba
- razni "otprilike" modeli:
  - Phongov model

pitanja?