3D grafika

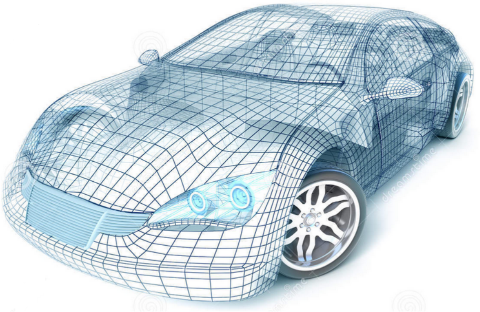
Marko Stanojević

Matematička gimnazija
Nedelja$^{4}_{\text{informatike}}$

30. mart 2018.

Namena 3D grafike

- Prikazivanje 3D objekata na 2D ekranu računara



Paradigme

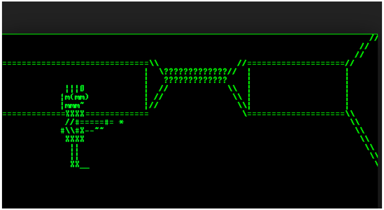
- Mnoštvo različitih spregnutih kriterijuma
  - prostor – vreme
  - lepota/realnost – efikasnost
- Mora se napraviti kompromis
- Različite složenosti 3D modela za isti objekat/pojavu
  - Stvar dogovora koliko ćemo ići u detalje
  - Mora se misliti i na cenu i snagu hardvera

Paradigme

- Ko je krajnji korisnik i kakva mu 3D grafika treba
  - gaming industrija
  - filmska industrija, reklame, spotovi
  - CAD/CAM sistemi
  - simulacije fizičkih pojava
  - machine learning

Paradigme

- 3D grafika se postepeno razvijala



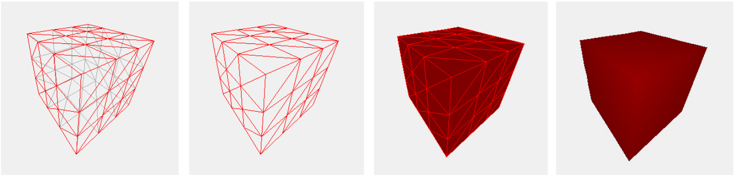
- Što više ljudi želi nešto, to je veći pritisak da se to ostvari
  - jeftiniji GPU $\rightarrow$ pristupačniji većem broju ljudi $\rightarrow$ veća potražnja
- Grafičke biblioteke
  - OpenGL, DirectX, Vulkan

Koraci u rendering-u

- Potrebno je nekako prikazati 3D objekte na 2D ekranu
- To se realizuje kroz sledeće korake:
  - matematičko modelovanje objekata
  - prelazak u svetski koordinatni sistem
  - prelazak u koordinatni sistem posmatrača
  - rasterizacija

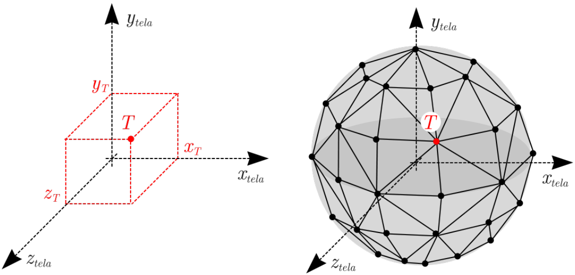
Matematičko modelovanje objekata

- Više opcija
  - point cloud
  - trouglovi
  - Bezier curves



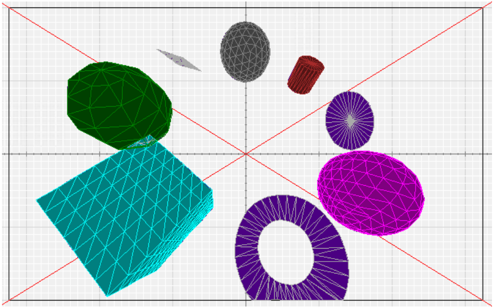
Teselacija

- Teselacija – pretvaranje spoljašnje površine objekta u skup trouglova
- Razlog? GPU radi najbolje nad trouglovima jer ima instrukcije koje su optimizovane za tu primenu



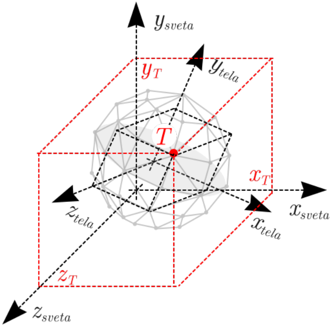
Teselacija

- Kriterijumi za izbor veličine i oblika trouglova
  - da li će biti mnogo skaliranja
  - da li su predmeti mnogo daleko
  - da li je kretanje po sceni mnogo brzo
  - da li su fizički modeli u pitanju



Prelazak u drugi koordinatni sistem

- Mali podsetnik iz linearne – drugi koordinatni sistem treba da se izrazi pomoću prvog
- Matrice prelaska – jer predstavljamo temena trouglova preko vektora
- Vrlo lepo i intuitivno za obradu



Prelazak u drugi koordinatni sistem

- u cilju jednostavnije obrade, vektorima tela dodamo još jednu koordinatu
- time ih prebacimo u 4D homogeni koordinatni sistem

$$\vec{u} = \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

Prelazak u drugi koordinatni sistem

- 4D transformaciona matrica rotacije oko x-ose

$$\mathbf{R}_x(t) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos t & -\sin t & 0 \\ 0 & \sin t & \cos t & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Prelazak u drugi koordinatni sistem

- 4D transformaciona matrica rotacije oko y-ose

$$\mathbf{R}_y(t) = \begin{bmatrix} \cos t & 0 & \sin t & 0 \\ 0 & 1 & 0 & 0 \\ -\sin t & 0 & \cos t & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Prelazak u drugi koordinatni sistem

- 4D transformaciona matrica rotacije oko z-ose

$$\mathbf{R}_z(t) = \begin{bmatrix} \cos t & -\sin t & 0 & 0 \\ \sin t & \cos t & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Prelazak u drugi koordinatni sistem

- 4D transformaciona matrica skaliranja

$$\mathbf{S}(k_x, k_y, k_z) = \begin{bmatrix} k_x & 0 & 0 & 0 \\ 0 & k_y & 0 & 0 \\ 0 & 0 & k_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Prelazak u drugi koordinatni sistem

- 4D transformaciona matrica translacije

$$\mathbf{T}(l_x, l_y, l_z) = \begin{bmatrix} 1 & 0 & 0 & l_x \\ 0 & 1 & 0 & l_y \\ 0 & 0 & 1 & l_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Prelazak u drugi koordinatni sistem

- 4D transformaciona matrica za projekciju na XY-ravan

$$\mathbf{P}_{XY}(d) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix}$$

Prelazak u drugi koordinatni sistem

- 4D matricom mogu da se izvrši istovremeno:
  - kretanje u prostoru
    - translacija
    - rotacija po više osa
  - skaliranje
  - smicanje

$$\mathbf{R}(t_x, t_y, t_z) = \mathbf{R}_z(t_z) \cdot \mathbf{R}_y(t_y) \cdot \mathbf{R}_x(t_x)$$

Primer istovremenih 4D transformacija (1/3)

Пример 1. *Имамо вектор*

$$\vec{u} = \begin{bmatrix}2 & 3 & 4\end{bmatrix}^T$$

Рецимо да желимо да прво ротирамо вектор око x-осе за угао $\alpha = \pi/2$ (математички смер), да би га потом скалирали $5\times$ у односу на x-осу, и транслирали за 5 јединица у негативном смеру паралелно x-оси. Како то извести?

→ *Прво трансформишемо вектор у хомогене координате:*

$$\vec{u_0} = \begin{bmatrix}2 & 3 & 4 & 1\end{bmatrix}^T \tag{20}$$

→ *Затим га помножимо са леве стране са матрицом ротације:*

$$\vec{u_1} = \begin{bmatrix}1 & 0 & 0 & 0\\ 0 & \cos(\pi/2) & -\sin(\pi/2) & 0\\ 0 & \sin(\pi/2) & \cos(\pi/2) & 0\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}2\\3\\4\\1\end{bmatrix} \tag{21}$$

Primer istovremenih 4D transformacija (2/3)

→ *Потом га скалирамо:*

$$\vec{u_2} = \begin{bmatrix} 5 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\pi/2) & -\sin(\pi/2) & 0 \\ 0 & \sin(\pi/2) & \cos(\pi/2) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 3 \\ 4 \\ 1 \end{bmatrix} \tag{22}$$

→ *Након тога га транслирамо:*

$$\vec{u_3} = \begin{bmatrix} 1 & 0 & 0 & -5 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 5 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\pi/2) & -\sin(\pi/2) & 0 \\ 0 & \sin(\pi/2) & \cos(\pi/2) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 3 \\ 4 \\ 1 \end{bmatrix} \tag{23}$$

Primer istovremenih 4D transformacija (3/3)

→ *Сада објединимо све матрице у једну помоћу матричног множења:*

$$\vec{u_3} = \begin{bmatrix} 5 & 0 & 0 & -5 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 3 \\ 4 \\ 1 \end{bmatrix} \tag{24}$$

→ *Потом помножимо вектор са обједињеном матрицом:*

$$\vec{u_3} = \begin{bmatrix} 5 & -4 & 3 & 1 \end{bmatrix}^T \tag{25}$$

→ *И коначно вратимо вектор из хомогеног у Декартов координатни систем:*

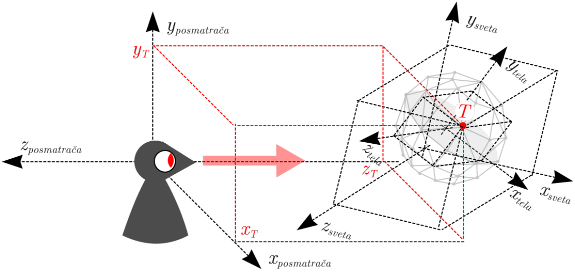
$$\vec{u_4} = \begin{bmatrix} 5 & -4 & 3 \end{bmatrix}^T \tag{26}$$

Prelazak u drugi koordinatni sistem

- Moguće su i druge matrice, ali bi bile drugačije za svaku tačku (npr. matrica prelaska u ravan bi bila različita za svaki trougao i svako teme trougla)
  - ako je matrica ista, onda se samo prosledi GPU da je pomnoži sa svim trouglovima
  - inače GPU prosleđujemo funkciju, a to usporava obradu

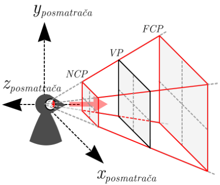
Samo je jedan svet...

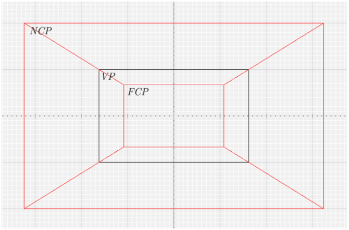
- Sve trouglove objekata treba prebaciti u jedan, svetski koordinatni sistem
- Nakon toga, treba preći u koordinatni sistem izabranog posmatrača



...ali je posmatrača više

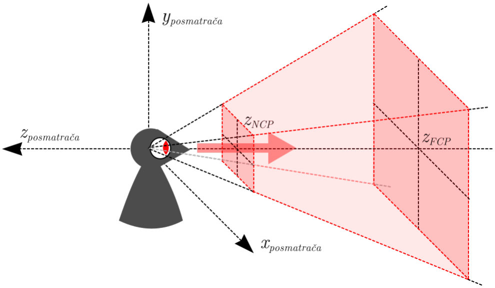
- Posmatrač je tačka ispred monitora, uobičajeno na normali iz centra ekrana, koji gleda ka monitoru sa neke udaljenosti (podešavanjem FOV-a)
- Može istovremeno da bude više posmatrača
  - zgodno ako se nešto modeluje da se posmatra iz više uglova istovremeno





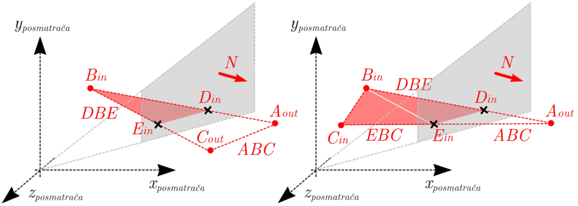
Polje vida

- Zamislimo da je monitor prozor u svet, a da je posmatrač tačka u 3D prostoru
- Posmatrač ne vidi beskonačno daleko, ali ni bliže od udaljenosti oka (ne vidimo unutrašnjost oka)
- Pogled je ograničen i sa strana
  - vidimo zarubljenu piramidu (viewing frustrum)



Polje vida

- Trouglove koji su u celini izvan piramide treba odbaciti
- Trouglove koji su samo delimično unutar piramide treba iseći na delove koji pripadaju piramidi, a ostatak odbaciti
  - trougao može da bude isečen više puta od strane 6 ravni koje formiraju polje vida
  - bitno je samo da je ostatak u unutrašnjosti ili na granici piramide



Polje vida

- Računanje tačke preseka duži i ravni
  - za svaku ravan zasebna obrada

Тачка пресека дужи троугла и равни се рачуна по формули:

$$\vec{v_p} = \vec{v_1} + \frac{(\vec{L} - \vec{v_1}) \cdot \vec{N}}{(\vec{v_2} - \vec{v_1}) \cdot \vec{N}} \cdot (\vec{v_2} - \vec{v_1})$$

где су:

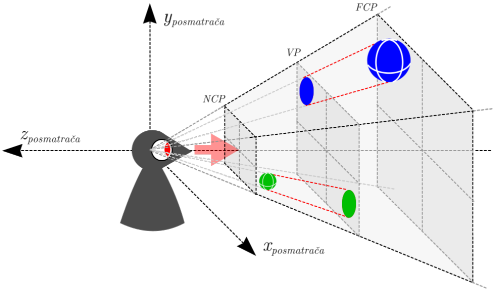
$\vec{v_1}$ и $\vec{v_2}$ – вектори крајњих тачака дужи,

$\vec{L}$ – вектор произвољне тачке у равни,

$\vec{N}$ – вектор нормале на раван.

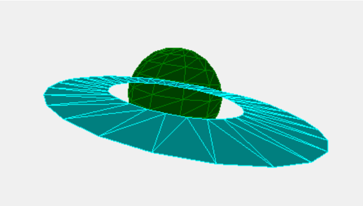
The great flattening

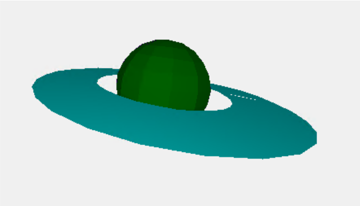
- Trouglove iz vidnog polja posmatrača treba projektovati u ravan ekrana, da bi ih prikazali pikselima monitora
- Matematički, da smo spljoštili sve trouglove u ravan ekrana, ne bi znali koji trougao da prvi iscrtamo
- Nije svejedno koji se trouglovi prvi iscrtavaju



Rasterizacija

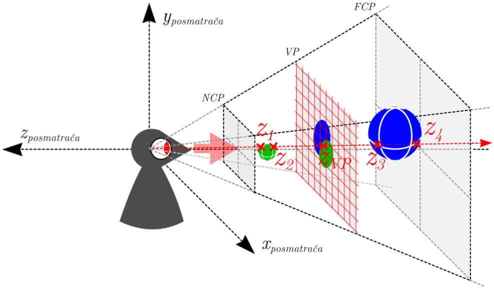
- Painter's algorithm
  - nije baš najbolji osim za jednostavne modele
  - radi dobro ako je jasno koje telo je bliže posmatraču
  - ograničene primene





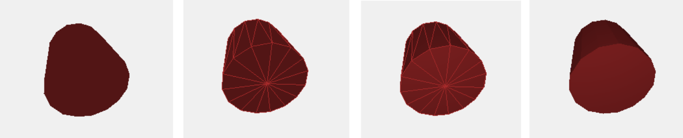
Rasterizacija

- Z-buffer, log-Z buffer
  - ispaljujemo zrake iz oka posmatrača u svaki piksel ekrana
  - pamtimo boju najbližeg tela kojeg pogodimo zrakom
  - kroz svaki piksel po jedan ili više zraka
  - ovde treba shading
  - ne rešavaju sve probleme - transparency

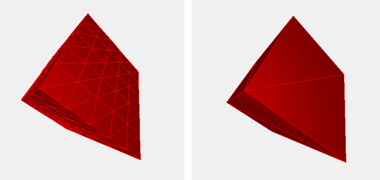


Rasterizacija

- Bez sencenja nam svet izgleda nerealno
- Fixed shading



- Flat shading



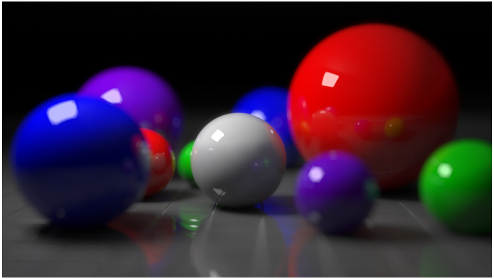
- Phong shading

Mapiranje

- bump mapping
- texture mapping

Svetlost

- Rendering equation
- Realno osvetljenje teško postići perfektno bez smanjenja performansi
  - vokselizacija
- Zavisno od primene postoji više rešenja rendering jednačine



Svetlost

- Ray tracing
  - ispaljujemo zrake iz oka posmatrača u svet
  - dozvoljena su odbijanja zraka o predmete
  - ray tracing može da reši:
    - senčenje, osvetljenje od netačkastog izvora
    - refleksiju, transparentnost, mat i upijajuće površine
    - uske proreze, caustics

Metode ubrzanja iscrtavanja

- Hardversko ubrzanje
  - 3D grafička obrada je najčešće ograničena na mali broj instrukcija
  - podela posla iscrtavanja
    - 3600 grafičkih core-ova u grafičkom procesoru (GPU)
    - svaki core GPU-a proračunava samo deo slike
    - core-ovi koji su završili rad dobiju nove zadatke iz zajedničke memorije
  - farme grafičkih kartica
- Softversko ubrzanje
  - pipeline - istovremeno obavljanje više koraka u različitim stadijumima izvršenja (kao pokretna traka)
  - uklanjanje geometrije koja se ne vidi i kompleksnosti koja ne doprinosi kvalitetu slike

Optimizacije i doterivanje

- Razne približne metode
  - fast inverse square root method
- Double + triple buffering
- Anti-aliasing
  - FXAA
  - DSR scaling
  - Subpixel (font smoothing)