→ redoslijed obavljanja transformacija: 1. translacija u (0, 0), 2. skaliranje, 3. rotacija, 4. translacija u točku (x, y)

→ kada su ortografska i perspektivna projekcija iste? → kada se izvor svjetlosti d nalazi u beskonačnosti (zrake svjetlosti koje baca su paralelne)!

Elementi simulacije: vizualna simulacija, zvučna simulacija, hipatička simulacija, fizikalna simulacija

Elementi virtualne scene: virtualni predmet, virtualni materijal, virtualna svjetlost, virtualna kamera

Metode/tehnike modeliranja: poligoni, konstruktivna geometrija čvrstih tijela, parametarske krivulje i plohe, razdjelne plohe, brišuće plohe, volumenske reprezentacije, fraktali, sustavi čestica, modeliranje zasnovano na slikama

Modeli: **model kamere** (ortogonalna i perspektivna projekcija), **model osvjetljenja** (globalno i lokalno osvjetljenje, refleksija, refrakcija,…), **modeli izvora svjetlosti** (ambijentno, usmjereno, točkasto, reflektor), **model odbijanja svjetlosti** (Phongov model), **model materijala** (koeficijenti ambijentne, difuzne i spekularne komponente)

Transformacije: translacija, skaliranje, rotacija, smik, kombinacija

Faze iscrtavanja (GPS): aplikacijska faza, geometrijska faza, faza rasteriziranja

Modeli iscrtavanja: Poligoni (prikaz tijela kao skup poligona - trokuti), Konstruktivna geometrija čvrstih tijela (slaganje osnovnih elemenata operacijama), Parametarske krivulje i plohe (Bezier, NURBS, B-spline), Razdjelne plohe (zaglađivanje objekta), Brišuće plohe (povlačenje krivulje – ekstruzija i rotacijsko povlačenje), Volumenski prikaz (isto što i poligoni samo je ovo u 3D – prikaz 3D ćelijama =voxelima), Fraktali (Mandelbrot - planine, biljke, stabla, teren), Sustavi čestica (dim, vatra, voda, magla, kiša), Modeliranje zasnovano na slikama (kompliciranе stvari)

- (Anti)aliasing u kontekstu vektorske grafike objasni te kako se ispravlja.

Aliasing je neželjeni efekt koji se javlja zbog uzorkovanja nekog signala nedovoljnom frekvencijom. Očituje se stepeničastim, nazubljenim rubovima objekta na mjestima gdje bi trebala biti glatki, a ispravlja se tehnikama antialiasinga. Antialiasing radi po principu višestrukog uzorkovanja. Umjesto uzimanja određenog piksela, antialiasingom uzimamo i nekoliko okolnih slikovnih elemenata koji se onda zajedno interpoliraju. Rezultat je „ispravljeni“ slikovni element koji ne rezultira aliasingom.

- Akumulacijski spremnik. Zašto taj spremnik ima dvostruko više bpp od spremnika boje?

Tehnika akumulacijskog spremnika: Tehnika kojom postižemo zamućenje. Akumuliraju se vremenski okviri 1-8 kao kod ''naivnog postupka''. Kada dođe do okvira broj 9, on se doda, a okvir broj 1 se oduzme. U svakom trenutku imamo prosjek 8 okvira, ali samo dva iscrtavanja po okviru. Iscrtava sliku nekoliko puta uzastopno uz male pomake koordinata uzorkovanja i miješa takve uzastopne slike u zajedničkom spremniku.

Akumulacijski spremnik ima duplo više bppa jer mora stopiti više slika u jednu pa mu treba više memorije za to.

- Ambijentna komponenta Phongovog modela. Koji parametri ju karakteriziraju? Ovisi li o upadnom kutu svjetlosti? Od kojih elemenata se sastoji?

Ambijentna komponenta – grubo aproksimira dio efekata globalnog osvjetljenja koja daje minimalno osvjetljenje kojim se izbjegava pojava da predmeti na koje ne pada svjetlost budu potpuno crni. Karakteriziraju je ambijentalni intenzitet svjetlosti Ia (konstanta za cijelu scenu) te ambijentni koeficijent materijala ka (reakcija materijala na ambijentnu svjetlost) – vektori boje sastavljene od R, G i B komponente.

Formula za ambijentalnu komponentu: I = Ia*ka. Ia je ambijentalni intenzitet svjetlosti (konstantan za čitavu scenu), a ka je ambijentalni koeficijent materijala (kako materijal reagira na svjetlost). Iz formule se vidi da ovisi samo o dvije unaprijed zadane konstante, ne o kutu upada svjetlosti.

- Definiraj smik, translaciju, skaliranje i rotaciju, te ispiši matrične 2D oblike (Px, Py). (40,41. str.)

Translacija = pravocrtno pomicanje točke ili objekta na ravnini ili u prostoru.

Translacija točke P za vektor T = [Tx Ty]: P = [Px Py] → **P' = [Px+Tx Py+Ty]**

Rotacija = okretanje točke ili objekta oko ishodišta za kut alfa.

Rotacija točke P oko (0,0) za kut a: P = [Px Py] → **P' = [Px*cos(a)+Py*sin(a) -Px*sin(a)+Py*cos(a)]**

Skaliranje = množenje točke s faktorom skaliranja S.

Skaliranje objekta faktorom S = [Sx Sy]: P = [Px Py] → **P' = [Px*Sx Py*Sy]**

Smik = deformacija objekta uzduž koordinatnih osi (pravokutnik postaje paralelogram, a krug postaje elipsa).

Smik objekta po x-osi u ovisnosti o faktoru kx: P = [Px Py] → **P' = [Px+kx*Py Py]**

Smik objekta po y-osi u ovisnosti o faktoru ky: P = [Px Py] → **P' = [Px Px*ky+Py]**

***2D TRANSFORMACIJE: → redoslijed obavljanja transformacija: 1. translacija u (0, 0), 2. skaliranje, 3. rotacija, 4. translacija u točku (x, y)**

- TRANSLACIJA:

$$\begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ T_x & T_y & 1 \end{matrix}$$

translacija točke P za vektor T = [Tx Ty]: P' = P*T

- ROTACIJA:

$$\begin{matrix} \cos(a) & -\sin(a) & 0 \\ \sin(a) & \cos(a) & 0 \\ 0 & 0 & 1 \end{matrix}$$

rotacija točke P oko (0,0) za kut a: P' = P*R

-PROMJENA VELIČINE:

$$\begin{matrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{matrix}$$

SKALIRANJE objekta faktorom S = [Sx Sy]: P' = P*S

- 2D SMIK:

$$\begin{matrix} 1 & k & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix}$$

smik po x-osi za faktor k: P' = P*S_h

- Dobre i loše strane prikaza geometrije poligonima. (str. 20.)
DOBRE STRANE: vrlo općenit pristup (sve se može pretvoriti u poligone), prikaz osnovnim elementima tj. poligonima (obično trokuti)
LOŠE STRANE: nije intuitivno za ručno modeliranje, često se drugi oblici prikaza pretvaraju u poligone u zadnji čas prije prikaza, za dobru aproksimaciju treba velik broj poligona sa što manjom površinom

- Fraktali? Navedi jedan primjer.
Fraktali su fragmentirani, nepravilni geometrijski objekti koji pokazuju svojstvo samosličnosti. Stvaraju se rekurzivnim ponavljanjem funkcije (proizvoljna razina detalja).
Primjeri fraktala: Mandelbrotov skup, Julijev skup,…

- GPS - faze grafičkog protočnog podsustava, te ukratko objasni svaku od njih.

1. aplikacijska faza – početna faza u GPS i glavna joj je uloga da pripremi elemente za iscrtavanje (najčešće trokuti, crte i točke) i da ih šalje dalje u pipeline. Operacije: logika aplikacije, animacija, simulacija, ulaz/izlaz, detekcija sudara i sl. Ne izvodi se sklopovski već se programira.

2. geometrijska faza – implementirana sklopovski na GPU. U nju ulaze 3D trokuti, svjetla i kamera, a izlaze 2D koordinate koje će se iscrtati na zalonu. Dijeli se u **podfaze**: transformacija u k.s. kamere, sjenčanje vrhova, projekcija, obrezivanje i preslikavanje na zaslon.

3. faza rasteriziranja – imamo sve spremno za iscrtavanje, ali je svemu još potrebno dodati boju. Također, ako postoje dodat će se i tražene teksture. Ovdje se određuje i koji predmeti su vidljivi. Tipa mi još iz geometrijske faze znamo da u sceni recimo imamo dvije kocke, ali zanemarujemo činjenicu da nam je jedna kocka možda u sceni ispred druge i da se druga uopće ne bi trebala vidjeti od prve. U ovoj fazi se određuje koji dijelovi koje kocke će se vidjeti i da li ih uopće treba iscrtavati. Dijeli se u **podfaze**: priprema trokuta, prolaz trokuta, sjenčanje i stapanje.

→ priprema trokuta: priprema potrebnih podataka za prolaz trokuta, diferencijali koordinata duž površine trokuta

→ prolaz trokuta: utvrditi koje točke zaslona trokut prekriva, redak po redak (na x koordinate rubnih točaka dodaju se njihovi diferencijali čime se dobiva lijevi i desni rub trokuta u novom retku zaslona) i prolaz točku po točku unutar retka, interpolacija svih zadanih podataka u vrhovima - nastaje fragment

→ sjenčanje (dodavanje boje): programabilna faza u kojoj se određuje boja pojedine točke trokuta, ulaz su podaci fragmenta dobiveni interpolacijom, izlaz je boja u točki, kombinacija boje teksture i boje sjenčanja

→ stapanje: boja se upisuje u spremnik boje (matrica X×Y gdje su X i Y razlučivosti prozora), vektor [R G B], izračunata boja točke stapa se s postojećom točkom u spremniku boje, **rasterske operacije** (= sklopovski izvedene matematičke i logičke operacije koje se izvode na sadržaju raznih spremnika (Z-spremnik)), **maskiranje** (= tehnika kod koje oblik iscrtan u zasebnom spremniku (spremniku maske) određuje područje zaslona u kojem se točke iscrtavaju, a ostatak je maskiran, određivanje vidljivosti metodom Z-spremnika)

- GPS - grafički protočni sustav? Koje su njegove glavne faze?

Grafički protočni sustav u stvarnom vremenu je niz funkcija koje se izvode jedna za drugom, a koje virtualnu scenu pretvaraju u sliku. Funkcije se mogu izvoditi istovremeno, kao na pokretnoj traci. Budući da se funkcije mogu izvoditi istovremeno, ovdje važi princip najslabije karike, što znači da najsporija operacija određuje brzinu i postaje usko grlo. Grafički protočni sustavi danas su optimizirani za rad s trokutima, te mogu izuzetno velikom brzinom iscrtavati scene sastavljene od trokuta.

Protočni sustav ima 3 glavne faze: aplikacijska faza, geometrijska faza i faza rasteriziranja.

Tehnike koje se koriste prilikom iscrtavanja su: sjenčanje, preslikavanje tekstura (teksturiranje), određivanje vidljivosti (metoda Z-spremnika), prozirnost, antialiasing.

- GPS - faza rasteriziranja. Što joj je ulaz a što izlaz? Navedi neke metode koje se u njoj koristi. Kako se implementira?

Faza rasteriziranja je zadnja glavna faza grafičkog protočnog sustava (GPS). Ulaz su joj trokuti s 2D koordinatama spremnim za iscrtavanje na zaslonu (zaslonske koordinate) već izračunate u geometrijskoj fazi, a izlaz su te iste koordinate, ali s dodatkom boje, tekstura, prozirnosti i sličnih efekata. Implementira se sklopovski na GPU. Metode (podzafe) koje se u njoj koriste su redom priprema trokuta, prolaz trokuta, sjenčanje (dodavanje boje), stapanje.

→ priprema trokuta: priprema potrebnih podataka za prolaz trokuta, diferencijali koordinata duž površine trokuta

→ prolaz trokuta: utvrditi koje točke zaslona trokut prekriva, redak po redak (na x koordinate rubnih točaka dodaju se njihovi diferencijali čime se dobiva lijevi i desni rub trokuta u novom retku zaslona) i prolaz točku po točku unutar retka, interpolacija svih zadanih podataka u vrhovima → nastaje fragment

→ sjenčanje: programabilna faza u kojoj se određuje boja pojedine točke trokuta, ulaz su podaci fragmenta dobiveni interpolacijom, izlaz je boja u točki, kombinacija boje teksture i boje sjenčanja

→ stapanje: boja se upisuje u spremnik boje (matrica X×Y gdje su X i Y razlučivosti prozora), vektor [R G B], izračunata boja točke stapa se s postojećom točkom u spremniku boje, **rasterske operacije** (= sklopovski izvedene matematičke i logičke operacije koje se izvode na sadržaju raznih spremnika (Z-spremnik)), **maskiranje** (= tehnika kod koje oblik iscrtan u zasebnom spremniku (spremniku maske) određuje područje zaslona u kojem se točke iscrtavaju, a ostatak je maskiran, određivanje vidljivosti metodom Z-spremnika)

- GPS - prva faza grafičkog protočnog sustava. Je li ona programabilna? Objasni!

Aplikacijska faza je početna faza u GPS i glavna joj je uloga da pripremi elemente za iscrtavanje (najčešće trokuti, crte i točke) i da ih šalje dalje u pipeline. Da, aplikacijska faza je programibilna, ne bi imalo smisla da ne možemo programirati logiku naše aplikacije ili recimo koje sve predmete želimo prikazati u njoj. Sve predmete koje želimo prikazati moramo najprije zadati (programirati) u aplikacijskoj fazi da bi ih dalje predali geometrijskoj fazi.

- GPS - uloga aplikacijske faze u grafičkom protočnom sustavu. Koje operacije vežemo uz tu fazu?

Aplikacijska faza je početna faza u GPS i glavna joj je uloga da pripremi elemente za iscrtavanje (najčešće trokuti, crte i točke) i da ih šalje dalje u pipeline. Operacije: logika aplikacije, animacija, simulacija, ulaz/izlaz, detekcija sudara i sl. (str. 62.)

- Homogene koordinate i kako se koriste za translaciju točke u 3D?

Homogene koordinate Px i Py služe nam za prikaz 2D točke uređenom trojkom [Px, Py, w] i prikaz 3D točke uređenom četvorkom [Px, Py, Pz, w], gdje je w = faktor proporcionalnosti (za naše potrebe w = 1). Omogućuju nam prikaz transformacija pomoću matrica. Te matrice 4x4 su karakteristične za svaku od transformacije koju opisuje.

Koriste se tako da se matrice kombiniraju i množe (translacija, rotacija, promjena veličine, smik i kombinacija). Množenje matrica nije komutativno (T1R1 ≠ R1T1).

Transformacija predmeta zahtijeva transformaciju svakog vrha P= [Px Py Pz] → P' = [P'x, P'y, P'z]

2D: P =[Phx/w Phy/w] → Ph = [Phx Phy w]

W = 1 : P = [Px Py] → Ph = [Phx Phy 1]

3D: P =[Phx/w Phy/w Phz/w] → Ph = [Phx Phy Phz w]

W = 1 : P = [Px Py Pz] → Ph = [Phx Phy Phz 1]

Translacija točke P za vektor T = [Tx Ty Tz]: **P' = P*T = [Px+Tx Py+Ty Pz+Tz 1]**

Rotacija točke P oko x-osi za kut a: P' = [Px Py*cos(a) – Pz*sin(a)]

Promjena veličine objekta (skaliranje) faktorom S = [Sx Sy Sz]: P = [Px Py Pz 1] → **P' = P*S = [Px*Sx Py*Sy Pz*Sz 1]**

Smik objekta po x-osi u ovisnosti o faktoru kx: P = [Px Py Pz 1] → **P' = [Px+kx*Py Py]**

Smik objekta po y-osi u ovisnosti o faktoru ky: P = [Px Py Pz 1] → **P' = [Px Px*kx+Py]**

P_{2D} = [Px Py] → P_{3D} = [Px Py Pz 1]

***3D TRANSFORMACIJE: → redoslijed obavljanja transformacija: 1. translacija u (0, 0), 2. skaliranje, 3. rotacija, 4. translacija u točku (x, y)**

- TRANSLACIJA:

1	0	0	0
0	1	0	0
0	0	1	0
T_x	T_y	T_z	1

-PROMJENA VELIČINE 3D OBJEKTA:

Sx	0	0	0
0	Sy	0	0
0	0	Sz	0
0	0	0	1

- ROTACIJA:

1	0	0	0
0	cos(a)	sin(a)	0
0	-sin(a)	cos(a)	0
0	0	0	1

oko osi X (Rx)

---

cos(a)	0	-sin(a)	0
0	1	0	0
sin(a)	0	cos(a)	0
0	0	0	1

oko osi Y (Ry)

---

cos(a)	sin(a)	0	0
-sin(a)	cos(a)	0	0

0	0	1	0	**oko osi Z (Rz)**
0	0	0	1	

- 3D SMIK:

1	kyx	kzx	0	
kxy	1	kzy	0	**komb. smikova po svim ravninama (kx je kut deformacije po x-osi, ky po y-osi i kz po z-osi)**
kxz	kyz	1	0	
0	0	0	1	

- LOKALNI KOORDINATNI SUSTAV:

Xx	Xy	Xz	0
Yx	Yy	Yz	0
Zx	Zy	Zz	0
Tx	Ty	Tz	1

- Izračun osvjetljenja?

Model osvjetljenja služi za računanje osvjetljenja tj. boje u promatranoj točki predmeta u sceni. Kada predmet iz virtualne scene projiciramo na projekcijsku plohu, te tu sliku preslikamo na zaslon, postavlja se pitanje koje će boje biti slika predmeta na zaslonu, tj. svaka točka u toj slici. Model osvjetljenja tj. boja u kojoj vidimo točku predmeta ovisi o: materijalu predmeta, svjetlima i o relativnim položajima kamere, svjetala i predmeta. Svjetlost se odbija od predmet i neizravno dolazi do drugih predmeta pri čemu dobivamo efekte kao što su mekane sjene, razlijevanje boje, odrazi itd. (str. 31.)

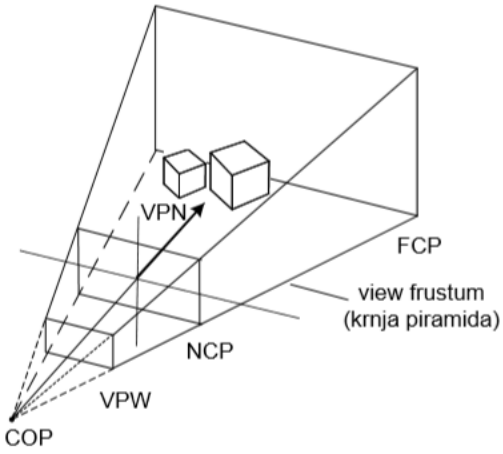
- Kako se može ubrzati postupak transformacije neke točke P u P' množenjem matrica?

Tako da se prvo pomnože sve matrice transformacija, i onda se sve točke množe s tom jednom matricom, umjesto da za svaku točku radimo pojedinačno množenje.

- Kod iscrtavanja, kako radi traženje presjeka sa scenom (u funkciji trace-ray), te da li se treba nešto mijenjati u funkciji ako se dodaju novi objekti (nije trebalo nikakve formule pisat).

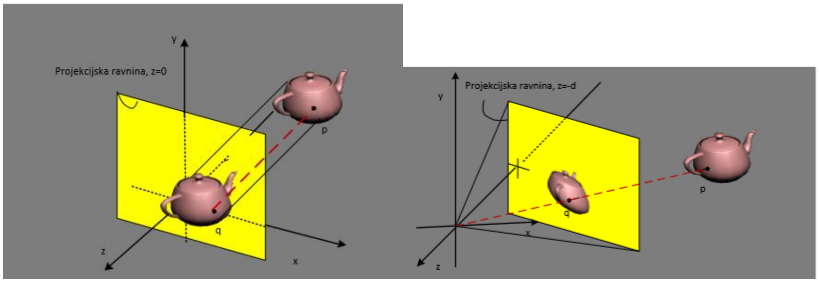
Ray-trace algoritam dijelimo u 7 operacija: računanje zrake kroz točku u prozoru, ispitivanje dubine rekurzije, nalaženje najbližeg presjeka zrake sa scenom, računanje lokalnog osvjetljenja u točki presjeka, ispitivanje utjecaja sjene, računanje odbijene i lomljene zrake, kombiniranje lokalnog osvjetljenja i doprinosa odbijene i lomljene zrake. Tu nas pita za presjek zrake sa scenom. Nakon što nađemo zraku kroz VPW, ta zraka će se negdje sudarit s nekim predmetom (ili nekad i neće i u tom slučaju ne moramo ništ računat). Naš cilj ovdje je odredit točku presjeka ta dva predmeta. Zraku prikazujemo kao pravac. U najjednostavnijem slučaju objekt s kojim tražimo presjek bit će kugla pa onda samo uvrstimo jednu jednadžbu u drugu i nađemo presjek. No često nije tako pa za različite „vrste“ objekata bit će potrebni različiti postupci nalaženja presjeka. Zato, ako bi se u scenu dodali novi objekti, morali bi dodati i metode koje će određivati kako se s njima i zrakom nalazi presjek. Za svaku vrstu objekta po jedna metoda za nalaženje presjeka. (str.58.)

- Model kamere. Čime je određen dio scene koji će se prikazati na zaslonu? Koje dvije vrste projekcije razlikujemo? (str. 30.)
Model kamere određuje pogled u virtualnu scenu koji će se iscrtati. Dio scene koji će se prikazati ograničen je s VPW, FCP i NCP. Razlikujemo ortogonalnu (z = 0) i perspektivnu (z = -d) projekciju.



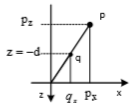
COP - centar projekcije (engl. center of projection)
NCP/FCP - bliska i daleka odrezujuća ploha (engl. near/far clipping plane)
VPW - projekcioni prozor (engl. view-plane window)
VPN - normala na projekcionu plohu (engl. view-plane normal)

- Ortografska projekcija na ravninu z=0
  - Postavlja z koordinate na 0



- Perspektivna projekcija na ravninu z=-d

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



$$\frac{q_x}{p_x} = \frac{-d}{p_z} \Rightarrow q_x = -d\frac{p_x}{p_z}$$

$$q = \left[-d\frac{p_x}{p_z} \quad -d\frac{p_y}{p_z} \quad -d \quad 1 \right]$$

- Modeliranje zasnovano na slikama. Navedi neke izazove na koje možemo naići koristeći ovu metodu. *NADOPUNITI*
Koristi se pri rekonstrukciji scena koje je prekomplicirano modelirati iz nule. Tipa hoće se napravit model cijelog grada. Uđe se u helikopter i uzme neki posebni laser/skener sa sobom i s tim se snima teren. Ono što je bitno je da se time dobiva jako jako puno točaka od koje se sastoji taj model pa je zbog toga nezgodno. i skupa je oprema. Ali nekad je ta metoda potrebna jer se recimo želi biti jako precizan u modeliranju. Uglavnom, ako želimo točno rekonstruirati neki kompliciran objekt iz stvarnosti, šanse su da ćemo koristiti ovu metodu. (str. 26.-30.)

- Na proizvoljnom primjeru virtualne scene objasni transformaciju u koordinatni sustav kamere. Koje dvije transformacije se koriste? U kojoj je to fazi GPS-a?

Nalazimo se u geometrijskoj fazi GPS. Uzmimo scenu s jednom kockom. Transformacija iz lokalnog k.s. kocke u k.s. kamere obavlja se u dva koraka. Najprije se lokalni k.s. kocke svojom matricom transformacije preslika u globalni k.s. (zajednički svim predmetima u sceni), a zatim se kocka u globalnom k.s. još jednom matricom transformacije preslika u k.s. kamere. Sve to isto se ponavlja za eventualne dodatne predmete u sceni.

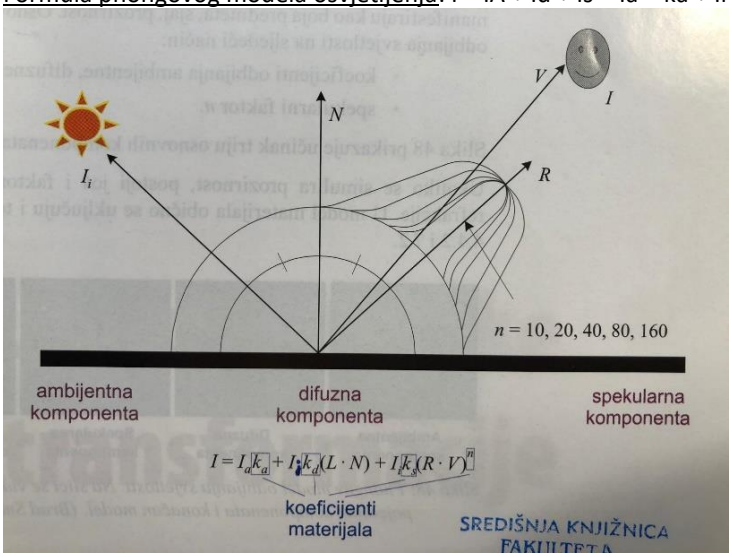
- Oktalno stablo (octree).
Oktalno stablo metoda je automatskog stvaranja hijerarhijske strukture od nestrukturirane scene (tzv. juhe poligona). Oktalno stablo „razrezuje“ scenu po sredini koristeći 3 ravnine i time stvara 8 volumno jednakih dijelova scene. Zatim svaki dio rekurzivno dijeli na opisani način. Rekurzija staje kad je promatrani djelić potpuno prazan, sadrži dovoljno malo poligona (često 1) da ga je nepotrebno dalje dijeliti, ili kad je veličina promatranog dijela manja od unaprijed zadanog praga.

- Ortografska i perspektivna projekcija, napiši njihove matrice te objasni kad su njihove matrice jednake.
Projekcije su posebne vrste transformacije.
Ortografska projekcija: možemo ju zamisliti kao sjenu koju baca objekt osvjetljen točkastim izvor svjetlosti smješten u beskonačnosti. Dakle, zrake svjetlosti koje baca su paralelne. Slika se projicira na unaprijed određenu ravninu u k.s.
Perspektivna projekcija: za razliku od ortografske projekcije uzima u obzir činjenicu da su predmeti koji su udaljeni manji od onih koji su bliže kameri. Ovdje zrake izvora nisu paralelne kao kod ortografske projekcije, osim kada bi se taj izvor d nalazio u beskonačnosti. U tom slučaju ortografska i perspektivna projekcija daju jednak rezultat. (str.50.-52.)

- Phongov model osvjetljenja - od čega se sastoji difuzna komponenta, te što se događa ako se kut upada svijetlosti mijenja od 45° do upada paralelno sa normalom?
IZ UDŽBENIKA: Difuzno odbijanje ide u svim smjerovima jednakim intenzitetom, a taj intenzitet ovisi o upadnom kutu te je najveći kada zraka upada okomito na površinu. Ova zakonitost je opisana skalarnim produktom vektora smjera upadne zrake L i normale na površinu N. Difuzna komponenta je usto proporcionalna intenzitetu izvora svjetlosti Ii i koeficijentu materijala kd.
FORMULA ZA DIFUZNU KOMPONENTU: Ii*kd(L*N)
ODGOVOR: Dakle, Ii i kd su konstante, N je vektor koji se ne mijenja. Mijenja se vektor L, a samim time i skalarni umnožak L*N. Skalarni umnožak je najveći kada su vektori paralelni, a nula kada su okomiti. Dakle, odgovor je da će se intenzitet svjetlosti koje objekt reflektira povećati i doseći svoj MAX kada vektor upada bude paralelan s normalom.

- Phongovog model osvjetljenja (formula i skice!).
Phongov model osvjetljenja je najčešće korišteni model za grafiku u stvarnom vremenu. Modelira difuzno i spekularno odbijanje te globalno osvjetljenje pomoću aproksimacije ambijentnim svjetlom. Jednostavan je za računanje i ima dobru aproksimaciju. Sastoji se od zbroja tri komponente: difuzne, ambijentalne i spekularne.
Difuzna → opisuje Lambertov zakon koji opisuje difuzno odbijanje svjetla na predmetu; difuzna komponenta proporcionalna je intenzitetu izvora i difuznom koeficijentu materijala kd, a opisuje ju skalarni produkt vektora smjera upadne zrake i normale na površinu.
Ambijentna → grubo aproksimira dio efekata globalnog osvjetljenja koja daje minimalno osvjetljenje kojim se izbjegava pojava da predmeti na koje ne pada svjetlost budu potpuno crni. Karakteriziraju je intenzitet Ia (konstanta za cijelu scenu) te ambijentni koeficijent materijala ka (reakcija materijala na ambijentnu svjetlost) – vektori boje sastavljene od R, G i B komponente.
Spekularna → aproksimira spekularni odsjaj na predmetu, a karakterizira ju krivulja sa oštrim maksimumom koji je izraženiji za sjajnije materijale (idealan materijal – ogledalo); proporcionalna je intenzitetu izvora i spekularnom koeficijentu materijala k_a.
Formula za difuznu: Id = Ii*kd(L*N). Ii je intenzitet izvora, kd je koeficijent materijala, L je vektor upadne zrake, a N normala površine na koju zraka pada.
Formula za ambijentalnu: IA = Ia * ka. Ia je intenzitet ambijentalne svjetlosti i jednak je za cijelu scenu, ka je ambijentalni koeficijent materijala.
Formula za spekularnu: Is = Ii*ks*(R*V)^n. ks je spekularni koeficijent materijala, V (view) je vektor smjera od točke gledanja do površine, R(reflect) je vektor smjera zrcaljenja od površine. Što je veći n zadan, objekt će biti sjajniji.

Formula phongovog modela osvjetljenja: I = IA + Id + Is = Ia * ka + Ii*kd(L*N) + Ii*ks*(R*V)^n



- Postupak ispitivanja utjecaja sjene kod principa praćenja zrake. Koji je lokalni doprinos kad je objekt koji baca sjenu proziran, a koji kada je neproziran?

Odmah idući korak nakon računanja osvjetljenja. Od točke presjeka zrake i predmeta šalje se zraka prema izvoru svjetlosti. Ako se na tom putu nađe neproziran objekt, onda je doprinos sjene maksimalan, odnosno intenzitet osvjetljenja u toj točki jednak nula. Ako se pak na tom putu nalazi proziran objekt (ili se ne nalazi nikakav objekt) onda je doprinos sjene nula. Ako se na tom putu nalazi neko tijelo faktora prozirnosti k, onda se utjecaj sjene modulira s obzirom na taj faktor. Za određivanje puta od objekta do svjetlosti (i da li se na tom putu nalaze neki objekti) mogu se koristiti iste metode iz Ray-trace algoritma. (str. 59.)

- Primjene sustava čestica u 3D grafici.

Sustavi čestica pogodni su za simulaciju prirodnih pojava (vodopad, vatra, dim, kiša,…).

- Princip praćenja zrake (ray-tracing). (str. 56.)

Princip praćenja zrake najčešće se koristi u offline grafici jer je računski zahtjevno i odvijalo bi se presporo u stvarnom vremenu. Izvrsno prikazuje refleksije, oštre sjene i prozirnost. Cilj je simulirati zrake svjetlosti kako bi se ponašale i u stvarnosti. Budući da umjesto okom, scenu vidimo na zaslonu, naše oko (kameru tj. zaslon) trebamo najprije podijeliti u piksele i kroz svaki od tih zraka prema nama bi u stvarnosti dolazila jedna zraka svjetlosti. Ovdje radimo suprotno (jer je računski lakše) - od kamere kroz svaki piksel šaljemo jednu zraku prema sceni. To se radi u 1. koraku. S jednom takvom zrakom koja prolazi kroz neki piksel ulazimo u **2. korak:** određivanje dubine rekurzije. Naša zraka iz prvog koraka će naići na neki premet u sceni. Dio te zrake će predmet upiti, a dio će se reflektirati. Za dio koji se odbio trebamo rekurzivno dalje računati kamo će se kretati. Ta zraka koja se odbila će vjerojatno naići na još neki predmet u sceni i tu će se opet dio upiti, a dio reflektirati. Postupak ponavljamo sve dok se više nema što reflektirati, tj. dok intenzitet zrake padne ispod neke granice za koju smo odredili da je nevidljiva (0 ili blizu 0). I to je u biti to što se misli pod ograničavanjem rekurzije. Još jedan način kako to možemo postići je tako da odredimo neki fiksni broj N i kažemo da nećemo gledati reflektirane zrake nakon one N-te koja se odbila. Tu podvlačimo crtu i kažemo da je kraj. Prva metoda daje bolje rezultate jer je bliža onome kako stvari funkcioniraju u stvarnom svijetu. **Sljedeći koraci su:** pronalaženje presjeka naše zrake sa predmetom u koji udara → određivanje lokalnog osvjetljenja u točki presjeka → ispitivanje utjecaja sjene → određivanje reflektirane i lomljene zrake. Treba samo odrediti normalu površine na koju pada zraka i odbiti je pod istim kutom pod kojim je i upala. Lomljena zraka računa se ako imamo neki proziran objekt tipa staklo. Tada se dio upadne zrake lomi na tom staklu po Snellovom zakonu i prolazi dalje. Obje zrake dalje tretiramo zasebno i gledamo kamo se kroz scenu kreću sve dok ne udarimo u limit rekurzije koji smo zadali prije nekoliko koraka. Konačno, potrebno je kombinirati sve doprinose osvjetljenja. To su doprinosi od osvjetljenja u sceni (koje smo računali u koraku lokalnog osvjetljena) i doprinosi od zraka koje su lomile ili odbile od nekih predmeta i udarile u točku (piksel) koju promatramo. Točna formula je na strani 60. (str. 56.-61.)

Ray trace algoritam dijelimo u **7 operacija**: računanje zrake kroz točku u prozoru, ispitivanje dubine rekurzije, nalaženje najbližeg presjeka zrake sa scenom, računanje lokalnog osvjetljenja u točki presjeka, ispitivanje utjecaja sjene, računanje odbijene i lomljene zrake, kombiniranje lokalnog osvjetljenja i doprinosa odbijene i lomljene zrake.

→ **Kratki postupak praćenja zrake:** za svaku točku ekrana se prati zraka koja kroz tu točku ulazi u scenu. Za zraku se traži presjek sa predmetima u sceni. Ako se nađe presjek, računa se osvjetljenje u točki presjeka, te se računaju zrcaljena zraka i refraktirana zraka. Za ove dvije zrake postupak se rekurzivno ponavlja, te se zbrajaju doprinosi osvjetljenja svih nađenih točaka presjeka.

→ PSEUDO-KOD ZA POSTUPAK PRAĆENJA ZRAKE (RAY-TRACING):

```
za svaki x,y u prozoru iscrtavanja
        izračunaj zraku R kroz x,y
                zraku promatramo kao pravac kroz dvije točke u 3D xyz-sustavu. Ekran za iscrtavanje postavljamo u xy-
                        ravninu tako da točka (0,0,0) bude u središtu ekrana. Početna točka koja definira zraku je oko
                                promatrača koje se nalazi na unaprijed određenoj i fiksiranoj poziciji (0,0,z). Druga točka koja
definira zraku je piksel na ekranu                za iscrtavanje. Položaj oka se uzima za hvatište zrake, a smjer zrake će biti
normalizirani vektor od oka promatrača do piksela na                ekranu i računa se kao smjer pravca kroz dvije točke
(oko i piksel)
        boja C = TraceRay(R,0)
        obojaj točku x,y bojom C
kraj
funkcija TraceRay (R,dubina)
        ako je dubina > max dubine, prekini funkciju i vrati crnu boju
        nađi najbliži presjek zrake R sa scenom
        ako nema presjeka, prekini funkciju i vrati kao rezultat boju pozadine
        izračunaj boju lokalnog osvjetljenja Clocal u točki presjeka
        izračunaj odbijenu zraku Rrefl
        boja Crefl = TraceRay(Rrefl, dubina+1)
        izračunaj refraktiranu zraku Rrefr
        boja Crefr = TraceRay (Rrefr, dubina+1)
        boja C = kombinacija (Clocal, Crefl,Crefr)
kraj: vrati boju C
```

- Programska sučelja niske razine. Koje je sučelje najraširenije? (78.-79. str.)

Što je niže razine, to znači da je sučelje po strukturi i načinu rada bliže protočnom sustavu. Izravnija je veza s protočnim sustavom što znači i više fleksibilnosti u korištenju svih funkcija protočnog sustava. To su npr. DirectX i OpenGL. Najraširenija je OpenGL.

- Quaternion i slerp - napisati jednadžbe. (str 86./87.)

SLERP = sferna linearna interpolacija (spherical linear interpolation)

QUATERNION = proširenje kompleksnih brojeva trima imaginarnim komponentama i, j, k

QUATERNION: $q = (q_v, q_w) = (q_x, q_y, q_z, q_w) = iq_x + jq_y + kq_z + q_w$

$i^2 = j^2 = k^2 = ijk = -1$

$jk = -kj = i$

$ki = -ik = j$

$ij = -ji = k$

točka P = (px, py, pz) izražena kao quaternion: $p = (p_x, p_y, p_z, 1) = ip_x+jp_x+kp_z+1$

rotacija izražena kao quaternion: $q = (q_v, q_w) = (u_q \sin\phi, \cos\phi)$

→ uzastopna rotacija dobiva se množenjem quaterniona!

$$slerp(\hat{r}, \hat{q}, t) = \frac{\sin(\phi(1-t))}{\sin\phi}\hat{q} + \frac{\sin(\phi t)}{\sin\phi}\hat{r}$$

SLERP: $\cos\phi = q_x r_x + q_y r_y + q_z r_z + q_w r_w$

- Razlika između offline i real-time grafike. (str. 5.)

Offline grafika: puno veće vrijeme iscrtavanja slike, proizvodnja pojedinačnih slika, proizvodnja video sekvenci, izuzetno visoka kvaliteta slike (animacije i specijalni efekti)

Real-time grafika: iscrtavanje brzinom od barem 30 slika u sekundi, manja kvaliteta slike, mogućnost interakcije

→ real-time grafika = VO + zvuk + dodir + fizikalna simulacija (virt. stvarnost, proš. stvarnost, web3D, NVE, CAD)

- Razlika između proširene i virtualne stvarnosti.

PROŠIRENA STVARNOST = dodaje elemente virtualnog okruženja u stvarni svijet te na taj način izgledaju kao dio stvarnog svijeta. Sastoji se od kombinacije stvarnog svijeta i virtualnih elemenata, interakcije u stvarnom vremenu, te poravnavanja virtualnog sa stvarnim. Slika virtualnog svijeta miješa se sa slikom stvarnog svijeta (u stvarnom vremenu)

VIRTUALNA STVARNOST = računalne simulacije kojima je cilj stvoriti osjećaj prisutnosti u virtualnom okruženju. Skup tehnologija koje korisnika uranjaju u virtualno okruženje, korisnik osjeća samo virtualne podražaje

→ Računalne simulacije virtualnih stvarnosti mogu biti simulacije stvarnih lokacija ili potpuno novi imaginarni svjetovi, dok proširena stvarnost dodaje elemente virtualnog okruženja u stvarni svijet a ne stvara potpuno novi svijet.

- Rotiracija oko osi r za kut FI.

Uzmimo kocku sa svojim lokalnim koordinatnim sustavom u globalnom koordinatnom sustavu. Najprije treba rotacijom i translacijom transformirati lokalni koordinatni sustav kocke tako da se os r poklopi sa osi x globalnog sustava. Zatim, rotirati za kut FI i konačno primijeniti inverznu transformaciju prve transformacije tj. vratiti kocku u početnu točku.

Ukupna transformacija: $R_{r,FI} = MRx(FI)M^{-1} = MRx(FI)M^{T}$

- Slika i pogled. Opiši postupak praćenja zrake iz pogleda. Na koji način određujemo kraj rekurzije? Kako se računa intenzitet?
Princip praćenja zrake najčešće se koristi u offline renderiranju grafike jer je računski zahtjevno i odvijalo bi se presporo u stvarnom vremenu (za zadovoljavajuće rezultate treba nam minimalno 30ak frame-ova u sekundi). Svaki frame može se iscrtavati minutama ako želimo preciznu sliku. Cilj je simulirati zrake svjetlosti kako bi se ponašale i u stvarnosti. Budući da umjesto okom, scenu vidimo na zaslonu, naše oko (kameru tj zaslon) trebamo najprije podijeliti u piksele i kroz svaki od tih zraka prema nama bi u stvarnosti dolazila jedna zraka svjetlosti. Ovdje radimo suprotno (jer je računski lakše al dođe na isto), od kamere kroz svaki piksel šaljemo jednu zraku. To se radi u prvom koraku. S jednom takvom zrakom koja prolazi kroz neki piksel ulazimo u drugi korak – određivanje dubine rekurzije. Naša zraka iz prvog koraka će naići na neki premet u sceni. Dio te zrake će predmet upiti, a dio će se reflektirati. Za dio koji se odbio trebamo rekurzivno dalje računati kamo će se kretati. Ta zraka koja se odbila će vjerojatno naići na još neki predmet u sceni i tu će se opet dio upiti, a dio reflektirati. Postupak ponavljamo sve dok se više nema što za reflektirati, tj. dok intenzitet zrake padne ispod neke granice za koju smo odredili da je nevidljiva (0 ili blizu 0). I to je u biti to što se misli pod ograničavanjem rekurzije. Još jedan način kako to možemo postići je tako da odredimo neki broj tipa 20 i kažemo da nećemo gledati reflektirane zrake nakon one dvadesete koja se odbila. Tu podvlačimo crtu i kažemo to je to. Prva metoda daje bolje rezultate jer je bliža onome kako stvari funkcioniraju u stvarnom svijetu. Sljedeći korak je pronalaženje presjeka naše zrake sa predmetom u koji udara i taj korak je opisan u 5. pitanju gore. Sljedeći korak je određivanje lokalnog osvjetljenja u točki presjeka i on je opisan u 6. pitanju. Idući korak je ispitivanje utjecaja sjene i on je objašnjen u pitanju 7. Idući korak je određivanje reflektirane i lomljene zrake. Ništa pametno, treba samo odredit normalu površine na koju pada zraka i odbit je pod istim kutom pod kojim je i upala. Lomljena zraka računa se ako imamo neki proziran objekt tipa staklo. Tada se dio upadne zrake lomi na tom staklu (fiz2) po snellovom zakonu i prolazi dalje. Ništa više od toga mislim da ne treba znat. Obje zrake dalje tretiramo zasebno i gledamo kamo se kroz scenu kreću sve dok ne udarimo u limit rekurzije koji smo zadali prije nekoliko koraka. Konačno, potrebno je kombinirati sve doprinose osvjetljenja. To su doprinosi od osvjetljenja u sceni (koje smo računali u koraku o lokalnom osvjetljenu) i doprinosi od zraka koje su lomile ili odbile od nekih predmeta i udarile u točku (piksel) koju promatramo.

- Slika kamere u sceni koja snima nekoliko objekata (čajnik, stožac, kuglu...). Kamera je bila definirana sa 3 ravnine i centrom (COP). Od čega se sastoji kamera? (COP, NCP, FCP, VPW). Koji će se objekti iscrtati?
Model kamere se sastoji od COP, VPN, VPW, NCP, FCP.
Iscrtat će se svi objekti koji se nalaze u prostoru omeđenom s (NCP), (FCP) i (VPW).

- Slika sa 4 kugle (nešto poput one na str. 35 - slika 48). Opiši postupak praćenja zrake i opiši kako se računa osvjetljenje u zadanoj točki (bila je zadana 1. točka u koju zraka udara). *DOPUNITI*
Pitaju nas za računanje lokalnog osvjetljenja u točki presjeka (4. korak ray trace algoritma). Za piksel kojem smo pronašli presjek s objektom u prošlom koraku potrebno je pripisati određeni intenzitet svjetlosti. To se može računati Phongovim modelom osvjetljenja pri kojem se konačan intenzitet sastoji od zbroja ambijentalne, difuzne i spekularne komponente osvjetljenja. (str. 59.)

- Slika scene (ormarić na kojem su dvije vaze i zdjelica). Opisati tehnike modeliranja za koje misliš da su najpogodnije za modeliranje te scene i zašto.
Poligoni, CSG, razdjelne plohe i brišuće plohe.

- Sustavi čestica. (str. 25.)

SUSTAVI ČESTICA : fizikalna simulacija velikog broja jednostavnih čestica. Svaka čestica se prikazuje točkicom, crticom ili slično. Najzanimljivije svojstvo je svojstvo dinamičnosti. Parametri su položaj, boja i oblik. Korisno je za simulaciju prirodnih pojava (vodopad, vatra, dim).

- Tehnike modeliranja.

1. Poligoni: „zajednički nazivnik“ svim metodama. Koristi se u više manje svim prikazima. Ideja je svako tijelo prikazati kao skup poligona (najčešće trokuti, rjeđe četverokuti). Tipa kuglu nećemo prikazati kao glatku kuglu (niti ne možemo u računalu) nego kao tijelo popločeno poligonima.
2. CSG (Konstruktivna geometrija čvrstih tijela): Imamo osnovne elemente: kvadar, kuglu, stožac, torus i valjak. Njih proizvoljno kombiniramo u složenije objekte operacijama unije, presjeka i oduzimanja. Udžbenik str.22.
3. Parametarske krivulje i plohe: Iz naziva je očito da se radi o krivuljama i plohama tak da to sigurno nećete koristit ako je na slici uz zadatak slika svemirskog broda. Ideja je prikazati neku krivulju (ili plohu) parametarski. Najpoznatije takve krivulje su Bezierova krivulja, B-spline i NURBS. Svaka od njih ima svoju formulu u koju se ubaci neki parametar/ri i krivulja poprimi oblik s obzirom na njih. Dakle, ako je na slici uz zadatak neka glatko zakrivljena ploha ili krivulja, šanse su da se koristila ova metoda. Udžb.str.23.
4. Razdjelne plohe: ovo je više metoda poboljšavanja prikaza. Najbolje se skuži iz slike u udžbeniku str 24. Imamo neki „loš“ model auta i želimo mu zagladit rubove da izgleda ljepše. Tad koristimo ovu metodu i njene algoritme. Rezultat je ljepši model sa zaglađenim rubovima po našem izboru.
5. Brišuće plohe: Imamo neki 2D lik ili krivulju. Metoda brišućih ploha je u biti uzimanje te početne krivulje/lika i njeno „izvlačenje“ (extrude) ili rotacija oko neke osi. Time se dobivaju složeniji objekti ili plohe. Odma se skuži iz slike na strani 24.
6. Volumenski prikaz: Minecraft. Kao metoda poligoni, samo u 3D. Za prikaz objekata su korištene male 3D ćelije (voxeli) za razliku od poligona gdje su korišteni 2D trokuti ili četverokuti. (str.25.)
7. Fraktali: Udžb.str.25 i 26. Ponekad se koriste za modeliranje prirodnih pojava (planine, biljke, stabla, teren) i to je sve. Osim ako bude baš slika fraktala... onda je očito odgovor da je metoda fraktala najpogodnija za modeliranje fraktala.
8. Sustavi čestica: Udžb.str.26. Ako na slici bude neki dim, vatra, voda, magla, kiša i sl. Sve to se prikazuje sustavima čestica.
9. Modeliranje zasnovano na slikama: Koristi se pri rekonstrukciji scena koje je prekomplicirano modelirati iz nule. Tipa hoće se napravit model cijelog grada. Uzme se neki laser/skener sa sobom i s tim se snima teren. Ono što je bitno je da se time dobiva jako jako puno točaka od koje se sastoji taj model pa je zbog toga nezgodno. Ali nekad je ta metoda potrebna jer se recimo želi biti jako precizan u modeliranju. Ako želimo točno rekonstruirati neki objekt iz stvarnosti, vjv ćemo koristiti ovu metodu.

- Teksturiranje - u,v koordinate. (str. 70.)

Teksturiranje je u osnovi lijepljenje 2D slike na 3D geometriju. Za svaki vrh u modelu zadan je skup teksturiranih koordinata koje se nazivaju u,v koordinate. Svakom skupu teksturiranih koordinata odgovara neka točka u slici teksture. Teksturne koordinate su normallizirane u odnosu na veličinu slike (točka (0,0) odgovara donjem lijevom kutu slike, a točka (1,1) odgovara gornjem desnom kutu). Pomoću ovih koordinata slika se preslikava (lijepi) na model i dobivamo model s teksturom. Teksturne koordinate zadane su za vrhove svakog trokuta, te svakom trokutu modela odgovara neki trokut u slici teksture. Preslikavanje teksture se događa u fazi rasterizacije i radi se za svaku pojedinu točku. U,v koordinate se interpoliraju u fazi prolaza trokuta čime se dobivaju interpolirane teksturne koordinate u pojedinoj točki površine. Zatim se u fazi sjenčanja pomoću interpoliranih u,v koordinata vrši uzorkovanje teksture (dohvat boje teksture na tim koordinatama). Točka teksture na određenim u,v koordinatama se naziva teksel. Dohvaćeni teksel potom se koristi za određivanje konačne boje točke npr. težinskim miješanjem s bojom osvjetljenja.
Nedostatak linearne interpolacije u,v koordinata unutar trokuta očituje se kroz nedostatak perspektive jer se interpolacija radi u zaslonskim, a ne u 3D koordinatama (nemamo osjećaj dubine). Rješava se tako da se u obzir uzme i dubina (z koordinata).

- Teksturiranje - u kojoj fazi se provodi i koji je nedostatak linearne interpolacije? Kako to riješiti?

Teksturiranje se provodi u fazi rasterizacije. U fazi rasterizacije obavlja se preslikavanje teksture i radi se za svaku pojedinu točku. U fazi prolaza trokuta teksturne se koordinate interpoliraju. U fazi sjenčanja pomoću interpoliranih koordinata vrsi se uzorkovanje teksture (dohvat boje teksture na tim koordinatama). Nedostatak linearne interpolacije u,v koordinata unutar trokuta očituje se kroz nedostatak perspektive jer se interpolacija radi u zaslonskim, a ne u 3D koordinatama (nemamo osjećaj dubine). Rješava se tako da se u obzir uzme i dubina (z koordinata).

- Točka u sceni, matrica pomaka i matrica rotacije, izračunaj točku P'. (samo pomnožiti te matrice). Kako se to može ubrzati?

Točku P treba pomaknuti i rotirati. P će biti zadana kao matrica 1x3 (2D) ili kao 1x4 (3D), a matrice rotacije i pomaka 3x3(2D) ili 4x4(3D). Sve što treba napraviti je pomnožiti P s tim matricama. Međutim, nije svejedno kojim redoslijedom se to radi. Ako u zadatku piše da točku najprije treba translatirati, onda je najprije treba pomnožiti s matricom translacije, a tek onda matricom rotacije vice versa. To je zato jer se rotacija vrši uvijek iz ishodišta koordinatnog sustava. Recimo da je naša točka u ishodištu. Ako je najprije rotiramo, dobit ćemo opet tu istu točku na istoj poziciji. Međutim, kad bismo je prvo translatirali pa rotirali dobili bi skroz neku drugu točku jer bi se rotacija vršila nad točkom koja je već translatirana iz ishodišta. Ako je matrica rotacije R i matrica translacije T onda su koraci: P' = P*T, P'' = P'*R. Točku uvijek držite s lijeve strane množenja. Ako je s desne, formule iz knjige ne važe. Ovaj postupak se može ubrzati tako da najprije pomnožimo sve matrice međusobno (T*R) i onda rezultat pomnožimo s početnom točkom (P*TR) umjesto da točku postepeno množimo sa svakom od matrica. Ovo se često koristi ako imamo hrpu transformacijskim matrica i hrpu točaka na koje ih želimo primijeniti.

- Virtualna stvarnost i zatvorena petlja korisnika?

VIRTUALNA STVARNOST = računalne simulacije kojima je cilj stvoriti osjećaj prisutnosti u virtualnom okruženju. Skup tehnologija koje korisnika uranjaju u virtualno okruženje, dok korisnik osjeća samo virtualne podražaje
ZATVORENA PETLJA KORISNIKA : korisnik → ulazne jedinice (razni senzori) → računalo (simulacija VO) → izlazne jedinice (vizualni izlazni uređaji, zvučnici, slušalice,…)
Zatvorena petlja korisnika sastoji se od korisnika, ulazne jedinice, računalne simulacije i izlazne jedinice. Ova petlja prikazuje osnovni princip komunikacije korisnika i računala u virtualnoj sceni. Korisnik se nalazi u zatvorenoj petlji. Ulazne jedinice prate pokrete korisnika i šalju ih računalu koje na osnovu tih i ostalih podataka vrši simulaciju virtualnog okruženja. Izlazne jedinice mogu biti slušalice, zaslon koji korisnik nosi na glavi, itd. Računalo pomoću izlaznih jedinica prikazuje virtualno okruženje korisniku. U idealnom slučaju korisnik bi trebao dobivati podražaje samo od računala i time stvarati potpuno odvojeni svijet.

- Virtualno okruženje, virtualni predmet i virtualna scena?

Virtualno okruženje je složeniji skup virtualnih predmeta. Radi sa složenijim virtualnim predmetima, odnosno skupom virtualnih predmeta koji mogu prikazivati čitave zgrade, gradove i dr.
Virtualni predmet je predmet definiran u memoriji računala na takav način da ga računalo može na zaslonu prikazati korisniku uz mogućnost interakcije.
Virtualna scena je prikaz virtualnog okruženja u memoriji računala. Stvarna scena sastoji se od stvarnih predmeta, a virtualna scena od virtualnih predmeta. Slika stvarne i virtualne scene je stvarna. (str. 3.)

- Vrste transformacija. (str. 79.)

Translacija (pravocrtno pomicanje točke/objekta na ravnini/prostoru),
rotacija (rotacijom točke P oko ishodišta za kut α dobivamo točku P'),
promjena veličine (množenje točke s faktorom skaliranja S),
smik (deformacija objekta uzduž koordinatnih osi).

- Web3D? Ukratko opišite i usporedite Java applet i plug-in rješenja, koja omogućuju 3D grafiku u WWW preglednicima.
Web3D su tehnologije za prikaz interaktivne 3D grafike na Web-u. Veći broj tvrtki od kojih su najpoznatije Macromedia, Viewpoint, Cult 3D, Pulse, nudi 3D plugin za WWW preglednike. Poteškoća za tvrtke koje nude plug-in rješenja je u tome što krajnji korisnici moraju instalirati njihov plug-in da bi vidjeli 3D sadržaj. Korisnici iz sigurnosnih razloga, kao iz zbog izbjegavanja čekanja, nisu skloni takvim instalacijama pogotovo ako nisu čuli za ime tvrtke čiji plug-in trebaju instalirati. Time prednosti dobivaju tvrtke čija su imena ved poznata. Kod rješenja temeljenih na Java applet-u, applet čita i iscrtava 3D format te pruža mogućnost interakcije. Ovo je trenutno najpraktičnije rješenje. Problem je veličina appleta (~100K).

- Zadane su matrice translacije i rotacije dimenzija 4x4 te točka koju je potrebno prvo translatirati, a potom rotirati. Pomoću zadanih matrica odredi transformiranu točku P'. Kada bismo ovaj postupak provodili nad velikim brojem točaka, kako bismo ga mogli optimizirati odnosno smanjiti broj računskih operacija? Je li moguće provesti ovaj postupak pomoću matrica dimenzija 3x3? Obrazloži!
Točku P treba pomaknuti i rotirati. P će biti zadana kao matrica 1x3 (2D) ili kao 1x4 (3D), a matrice rotacije i pomaka 3x3(2D) ili 4x4(3D). Sve što treba napraviti je pomnožiti P s tim matricama. Međutim, nije svejedno kojim redoslijedom se to radi. Ako u zadatku piše da točku najprije treba translatirati, onda je najprije treba pomnožiti s matricom translacije, a tek onda matricom rotacije vice versa. To je zato jer se rotacija vrši uvijek iz ishodišta koordinatnog sustava. Recimo da je naša točka u ishodištu. Ako je najprije rotiramo, dobit ćemo opet tu istu točku na istoj poziciji. Međutim, kad bismo je prvo translatirali pa rotirali dobili bi skroz neku drugu točku jer bi se rotacija vršila nad točkom koja je već translatirana iz ishodišta. Ako je matrica rotacije R i matrica translacije T onda su koraci: P' = P*T, P'' = P'*R. Točku uvijek držite s lijeve strane množenja. Ako je s desne, formule iz knjige ne važe.
Ovaj postupak se može ubrzati tako da najprije pomnožimo sve matrice međusobno (T*R) i onda rezultat pomnožimo s početnom točkom (P*TR) umjesto da točku postepeno množimo sa svakom od matrica. Ovo se često koristi ako imamo hrpu transformacijskim matrica i hrpu točaka na koje ih želimo primijeniti.
Pošto radimo sa matricama 4x4, nalazimo se u 3D prostoru. Translaciju (u 3D prostoru) koju nas traži da obavimo ne možemo iskazati matricom 3x3 pa bi odgovor bio ne. Osim, ako se ispostavi da je točka zadana u zadatku takva da leži recimo u xy ravnini (z = 0) i treba je translatirati u neku točku koja leži u istoj toj z = 0 ravnini. Tada problem možemo svesti na 2D prostor i zapisati tražene matrice u 3x3 formatu.

- **Zatvorena petlja kod virtualne stvarnosti.**
ZATVORENA PETLJA KORISNIKA = korisnik → ulazne jedinice (razni senzori) → računalo (simulacija VO) → izlazne jedinice (vizualni izlazni uređaji, zvuk, haptički izlani uređaji)
Zatvorena petlja korisnika sastoji se od korisnika, ulazne jedinice, računalne simulacije i izlazne jedinice. Ova petlja prikazuje osnovni princip komunikacije korisnika i računala u virtualnoj sceni. Korisnik se nalazi u zatvorenoj petlji. Ulazne jedinice prate pokrete korisnika i šalju ih računalu koje na osnovu tih i ostalih podataka vrši simulaciju virtualnog okruženja. Izlazne jedinice mogu biti slušalice, zaslon koji korisnik nosi na glavi, itd. Računalo pomoću izlaznih jedinica prikazuje virtualno okruženje korisniku. U idealnom slučaju korisnik bi trebao dobivati podražaje samo od računala i time stvarati potpuno odvojeni svijet.

- Z-spremnik (Z-buffer)! Navesti još neke metode kojima se može riješiti problem vidljivosti.
Metoda Z-spremnika koristi dodatnu memoriju zvanu Z-spremnik u koju se sprema dubina objekta na svakoj točki zaslona. Z-spremnik ili spremnik dubine koristi se za izračun vidljivosti objekata i poligona u sceni i koristi tipično 24bpp. Ovaj spremnik uz upotrebu istoimene metode, osigurava da se na zaslonu vide samo vidljivi poligoni, a oni skriveni iza njih ostaju nevidljivi. Broj bitova u z-spremniku je važan zbog preciznosti. Ukoliko preciznost nije dovoljna a da poligona su vrlo blizu jedan drugome po dubini, može doći do pogreške, tj. do krivog određivanja vidljivosti.
Neke od ostalih metoda za rješavanje problema vidljivosti su spremnik boje, dvostruko i trostruko spremanje, stereo spremnici, spremnik predloška, slikarski algoritam, BSP stablo i ray casting.