6. Objasni obujmice i za š to se koriste te nabroji par pr imjera (3 boda)

Obujmica je jednostavan geometrijski oblik koji obuhvaća složeniji geometrijski skup. Obujmice služe za aproksimaciju složenog predmeta jednostavnijim predmetom radi lakše manipulacije (obično ispitivanja presjeka).

Primjeri obujmica su: kugla, elipsoid, kvadar paralelan s osima, općeniti kvadar, konveksna ljuska

9. Objasni princip Z-spremnika (3 boda)

Princip Z-spremnika je tehnika u računalnoj grafici koja se koristi za rješavanje problema preklapanja i određivanja vidljivosti objekata na zaslonu. Z-spremnik je memorijski spremnik koji pohranjuje vrijednosti dubine (Z-koordinate) fragmenata tijekom procesa rasterizacije. Pri svakom novom fragmentu, njegova Z-koordinata se uspoređuje s vrijednošću u Z -spremniku. Ako je nova Z-vrijednost manja, fragment se prihvaća i ažurira se vrijednost u Z-spremniku. Ova tehnika osigurava ispravno prikazivanje vidljivih objekata na zaslonu i sprječava preklapanje objekata.

4. Objasnite postupak rekurzivne podjele volumena prilikom korištenja metode oktalnog stabla.

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 zada nog praga.

10. Što je to obrezivanje? Opišite Sutherland -Hodgeman algoritam.

Obrezivanje (eng. clipping) je tehnika u računalnoj grafici koja se koristi za uklanjanje dijelova objekata koji su izvan vidljivog volumena ili prostora kako bi se optimiziralo iscrtavanje i povećala učinkovitost. Sutherland-Hodgeman algoritam je tehnika obrezivanja poligona koja se koristi u računalnoj grafici. Algoritam se sastoji od nekoliko koraka: podjela poligona na bridove, provjera vidljivosti svakog segmenta u odnosu na obrezne linije (clipping planes) i generiranje novih segmenata koji predstavljaju presjek poligona i obreznih linija. Ovaj postupak se ponavlja za svaku obreznu liniju kako bi se dobio konačni obrezani poligon. Sutherland-Hodgeman algoritam koristi ideju praćenja bridova i izračunavanja presjeka kako bi se smanjio broj točaka koje se moraju dalje obraditi i iscrtati. Ova tehnika je korisna za izbacivanje dijelova poligona koji su izvan vidljivog područja ili granica, č ime se poboljšava učinkovitost iscrtavanja.

1. Što je grafički protočni sustav (GPS)? Koje su njegove glavne faze? (2 boda)

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. Faze grafičkog protočnog podsustava (GPS) uključuju aplikacijsku fazu, geometrijsku fazu i fazu rasteriziranja.

2. Za koji postupak se u GPS-u koriste prednji i stražnji spremnik? U kojoj fazi GPS-a se koristi ovaj postupak? Objasni. (3 boda)

Pri korištenju naivnog pristupa iscrtavanju, boje se crtaju u stražnji spremnik koji se ne vidi, dok video upravljačka jedinica crta na prednji spremnik na zaslonu. Kako su ova dva procesa

asinkrona, mogu se dogoditi nedovršene slike na zaslonu. K ako bi se izbjegao ovaj efekt, koriste se dva spremnika - prednji i stražnji. Rasterizator crta u nevidljivi stražnji spremnik, dok se prednji spremnik prikazuje na zaslonu. Kada je slika spremna, spremnici se zamjenjuju, osiguravajući da video upravljačka jedinica uvijek koristi potpunu sliku za iscrtavanje.

Ovaj postupak pripada fazi rasteriziranja.

4. Objasniti opceniti postupak za ispitivanje presjeka zrake s opcenitim poligonom. Kako možemo ispitati nalazi li se dvodimenzionalna točka P unutar dvodimenzionlnog poligona? (3 boda)

Izracuna se presjek zrake s ravninom u kojoj lezi poligon. Zatim se dobivena tocka i poligon preslikaju u ravninu (xy, xz, yz) u kojoj projekcija ima najvecu povrsinu. Zatim povucemo zraku iz tocke i ukoliko presijeca poligon neparan broj puta, tocka je u poligonu, inace nije.

5. Što su obujmice i čemu služe? Navedi primjere. Objasni pojednostavljeni test kojim se računa presjek obujmice i projekcijskog volumena. U kojim slučajevim a je bolje raditi pojednostavljeni test nego puni? (4 boda)

Obujmica je jednostavan geometrijski oblik koji obuhvaca slozeniji geometrijski skup. To su npr. kugla, elipsoid, konveksna ljuska.

Test: Klasificiraju se dva moguca stanja obujmice. “Vani” ili “vjerojatno unutra” s obzirom na projekcijski volumen. U drugom slucaju postoji vjerojatnost da je citava obujmica, ili neki njezin dio, unutar projekcijskog volumena. U tom slucaju imamo 2 mogucnosti: ili crtamo sve unutar obujmice (ukoliko nije cijela unutar volumena nacrtat cemo nepotrebne poligone) ili ispitujemo sljedecu razinu obujmica (ukoliko ispitujemo sljedecu razinu, a prethodna obujmica je u potpunosti unutar projekcijskog volumena, radimo nepotrebne testove).

Pojednostavljeni testovi se koriste u velikim scenama gdje je velika vjerojatnost da ce i pojednostavljeni test odbaciti puno obujmica koje su izvan projekcijskog volumena.

1. Objasnite algoritam traženja presjeka zraka-kugla.

Presjek zraka-kugla moguć e je pronać i izravnom uporabomm jednadžbi. Uvrš tavanjem izraza koji opisuje zraku u izraz koji opisuje kuglu dobiva se kvadratna jednadžba. Determinanta određuje da li zraka siječ e, dira ili ne dotič e kuglu. Postupak se optimizira koriš tenjem jednostavnih testova za rano odbacivanje. Testovi provjeravaju neke parametre kao š to su udaljenost kugle od zrake ili smjer zrake u odnosu na kuglu te na se osnovu rezultata nastavlja s algoritmom ili dolazi do prekida jer presjek ne postoji.

1. Objasnite algoritam traženja presjeka zraka-kugla.

Presjek zraka-kugla moguć e je pronać i izravnom uporabomm jednadžbi. Uvrš tavanjem izraza koji opisuje zraku u izraz koji opisuje kuglu dobiva se kvadratna jednadžba. Determinanta određuje da li zraka siječ e, dira ili ne dotič e kuglu. Postupak se optimizira koriš tenjem jednostavnih testova za rano odbacivanje. Testovi provjeravaju neke parametre kao š to su udaljenost kugle od zrake ili smjer zrake u odnosu na kuglu te na se osnovu rezultata nastavlja s algoritmom ili dolazi do prekida jer presjek ne postoji

2. Opišite detekciju sudara kod hijerarhije obujmica.

Hijerarhija obujmica zapisuje se u obliku stabla. Pritom račve stabla sadrže obujmice, a listovi poligone. Hijerarhija obujmica može se graditi automatski i to pristupom s vrha ili s dna. U pristupu s vrha prvo se nalazi obujmica za sve poligone u predmetu i to je vrh stabla. Zatim se poligoni dijele u dvije grupe, za svaku se grupu nalazi nova obujmica i te obujmice se dodaju kao račve u stablo. Postupak se rekurzivno ponavlja.ova metoda se puno č ešće koristi. Pri pristupu s dna, poligoni se ubacuju u stablo jedan po jedan, na taj način da minimalno povećaju ukupnu obujmicu.

1. Što je graf scene, č emu služi i od kojih se dijelova sastoji i nacrtaj njegove osnovne dijelove (3 boda)

Graf scene je podatkovna struktura u koju se virtualna scena sprema na organiziran i strukturiran način. Omogućuje logičnu organizaciju scene, lakšu manipulaciju i učinkovitije iscrtavanje. Prolaz kroz graf scene standardni je rekurzivni postupak za obavljanje neke operacije nad svim čvorovima u sceni.

Kreće se od korijena sc ene te se rekurzivno spušta po hijerarhiji, pritom obrađujući svu djecu korijena scene, zatim njihovu djecu itd.

2. Što su obujmice i navedi 2 oblika. (2boda)

Obujmica je jednostavan geometrijski oblik koji obuhvaća složeniji geometrijski skup. Obujmice služe za aproksimaciju složenog predmeta jednostavnijim predmetom radi lakše manipulacije (obično ispitivanja presjeka). Primjeri obujmica su: kugla, elipsoid, kvadar paralelan s osima, općeniti kvadar, konveksna ljuska.

3. Nabroji 3 osnovne vrste presjeka ili 3 pravila računanja detekcije presjeka (3 boda)

Presjeci sa zrakom, presjeci obujmica, presjeci obujmica s projekcionim volumenom

4. Objasni presjek između 2 trokuta (3 boda)

Presjek trokuta ispitujemo sljedecim postupkom: (Trokut T1 u ravnini R1 i T2 u ravnini R2) Izracunamo udaljenost svih vrhova T1 do R1, ako su udaljenosti istog predznaka nema presjeka. Isto ponovimo za T2. Pravac L je presjek R1 i R2. Nalazimo presjek T1 i T2 sa L i dobijemo intervale I1 i I2, ako se intervali preklapaju imamo presjek

1.Što je BSP stablo?

BSP stablo je struktura za organizaciju scene kod koje se prostor scene rekurzivno djeli na dva dijela nekom ravninom. Njegova glavna prednost u odnosu na hijerarhiju obujmica je u tomeš to prolaz BSP stabla omogućuje učinkovito sortiranje geometrije od naprijed prema natrag u odnosu na kameruš to ima ulogu kod nekih medota ubrzavanja iscrtavanja.

2. Zašto se koriste u interaktivnoj 3D grafici testovi presjeka koji se razlikuju od analitičkih rješenja.(analitička rješenja -kad koristimo jednadžbu zrake ,jednadžbu kugle.....)

Zato jer nam nekad nisu potrebni svi elementi,nego se testira samo jedan element(moguće izračunati u pripremnoj fazi).Npr. kod auta i podloge provjeravamo samo da li su gume direktno iznad površine,ostalo nas ne zanima.Drugi razlog je zbog optimalnosti,jer analitička rješenja

nekad mogu biti dosta kompleksna,pa se onda koriste testovi odbacivanja koji su podosta jednostavniji.Ukoliko se koristi više testova/faza,prvo jednostavnije.

4. Ukratko opisati programska sučelja niske razine. Koje je sučelje najraširenije?

Sto je nize razine, to znaci da je sucelje po strukturi i nacinu rada blize protocnom sustavu. Izravnija je veza s protocnim sustavom sto znaci i vise fleksibilnosti u koristenju svih funkcija protocnog sustava. To su npr. DirectX i OpenGL, ne znam koje je najrasirenije.

5. Opisati i navesti neke karakteristike formata VRML. Da li je to standardni, otvoreni ili vlasnički format?

Zasnovan je na grafu scene i ima sve normirane čvorove. Standardni format. Karakteristike: moze se kreirati i urediti u obicnom editoru, ne ovisi o operacijskom sustavu i podrsci. Napredne karakteristike: naprednije funkcije senzori, interpolatori i staze kojim se mogu definirati jednostavne animacije i interakcije sa korisnikom.

3. Opisati i navesti neke karakteristike formata VRML. Da li je to standardni, otvoreni ili vlasnički format?

Zasnovan je na grafu scene i ima sve normirane čvorove. Standardni format. Karakteristike: moze se kreirati i urediti u obicnom editoru, ne ovisi o operacijskom sustavu i podrsci. Napredne karakteristike: naprednije funkcije senzori, interpolatori i staze kojim se mogu definirati jednostavne animacije i interakcije sa korisnikom.

4. Ukratko opisati programska sučelja niske razine. Koje je sučelje najraširenije? (str. 77)

Sto je nize razine, to znaci da je sucelje po strukturi i nacinu rada blize protocnom sustavu. Izravnija je veza s protocnim sustavom sto znaci i vise fleksibilnosti u koristenju svih funkcija protocnog sustava. To su npr. DirectX i OpenGL, ne znam koje je najrasirenije.

7. Objasniti općeniti postupak za ispitivanje presjeka zrake s općenitim poligonom. Kako možemo

ispitati nalazi li se dvodimenzionalna točka P unutar dvodimenzionlnog poligona? Izracuna se presjek zrake s ravninom u kojoj lezi poligon. Zatim se dobivena tocka i poligon preslikaju u ravninu (xy, xz, yz) u kojoj projekcija ima najvecu povrsinu. Zatim povucemo zraku iz tocke i ukoliko presijeca poligon neparan broj puta, tocka je u poligonu, inace nije.

8. Što su obujmice i čemu služe? Navedi primjere. Objasni pojednostavljeni test kojim se računa presjek obujmice i projekcijskog volumena. U kojim slučajevima je bolje raditi pojednostavljeni test nego puni? (str. 115)

Obujmica je jednostavan geometrijski oblik koji obuhvaca slozeniji geometrijski skup. To su npr. kugla, elipsoid, konveksna ljuska.

Test: Klasificiraju se dva moguca stanja obujmice. “Vani” ili “vjerojatno unutra” s obzirom na projekcijski volumen. U drugom slucaju postoji vjerojatnost da je citava obujmica, ili neki njezin dio, unutar projekcijskog volumena. U tom slucaju imamo 2 mogucnosti: ili crtamo sve unutar obujmice (ukoliko nije cijela unutar volumena nacrtat cemo nepotrebne poligone) ili ispitujemo sljedecu razinu obujmica (ukoliko ispitujemo sljedecu razinu, a prethodna obujmica je u potpunosti unutar projekcijskog volumena, radimo nepotrebne testove). Pojednostavljeni testovi se koriste u velikim scenama gdje je velika vjerojatnost da ce i pojednostavljeni test odbaciti puno obujmica koje su izvan projekcijskog volumena.

13. Objasni puni test za obujmice, te je bila slika s 4 kružnice(kugle) i zraka svijetlosti intenziteta I, pa je pitanje da se skicira i izdračuna intenzitet zrake svijetlosti, jer se zraka odbije od 4. kugle, pa od 2. i na kraju od 1.

Puni test za obujmice je test gdje se programski provjerava za svaku točku obujmice s obzirom na zraku iz izvora da li je vani,unutra ili je presjek.Ako je vani,odbacuje sve po projekcionom volumenu,ako je unutra crta sve,a ako je presjek testira sljedeću razinu i rekurzivno provjerava dalje.Za kugle se to provjerava ako je udaljenost središta kružnica strogo veći od r1+r2,onda se ne siječu,inače se siječu.Bila je slika s 4 kružnice(kugle) i zraka svijetlosti intenziteta I, pa je pitanje da se skicira i izdračuna intenzitet zrake svijetlosti, jer se zraka odbije od 4. kugle, pa od 2. i na kraju od 1. (to je valjda to)

14. Kako se traži presjek zrake sa scenom(ne računski) neko općenito. Ako u scenu dodamo Kuglu i valjak da li i kako moramo promijeniti raytracing metodu.

Presjek zrake sa scenom gdje je d udaljenost ishodišta zrake do njenog najbližeg presjeka sa scenom:

d=0 -> predmet dotiče scenu

d>0 -> predmet je iznad scene

d<0 -> predmet ulazi u scenu

Da,jer jednadžbe kugle i valjka su drugačije kao i optimalna rješenja za njih.Dodati generičku metodu koja s obzirom na tip objekta provjerava na optimalan način presjek zrake sa predmetom.

15. Objasnite algoritam traženja presjeka zraka-kugla?

Presjek zraka-kugla moguće je pronaći izravnom uporabomm jednadžbi. Uvrštavanjem izraza koji opisuje zraku u izraz koji opisuje kuglu dobiva se kvadratna jednadžba. Determinanta određuje da li zraka siječe, dira ili ne dotiče kuglu. Postupak se optimizira korištenjem jednostavnih testova za rano odbacivanje. Testovi provjeravaju neke parametre kao što su udaljenost kugle od zrake ili smjer zrake u odnosu na kuglu te na se osnovu rezultata nastavlja s algoritmom ili dolazi do prekida jer presjek ne postoji.

26.Graf scene?Nešto objasni način rekurzivnog prolaska grafom scene.

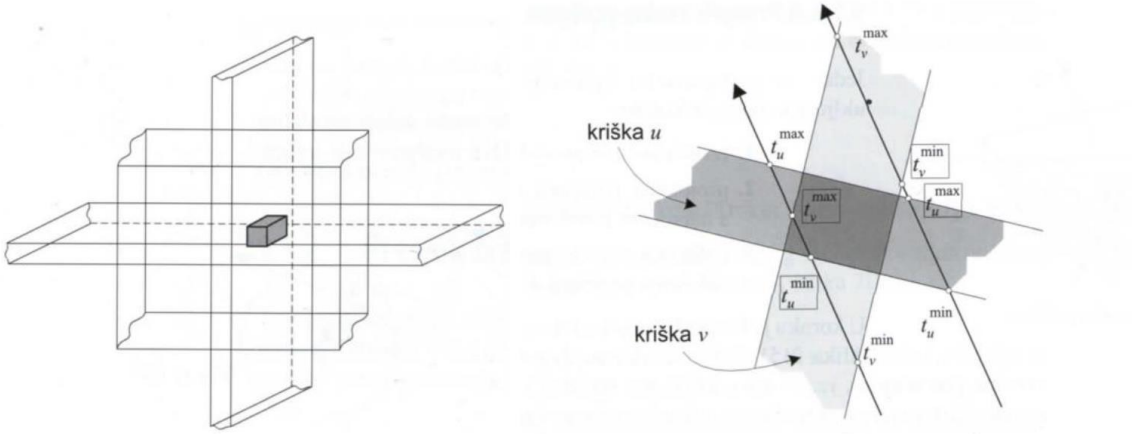
Prolaz kroz graf scene je normirani rekurzivni postupak za obavljanje neke operacije na svim čvorovima u u sceni. Pritom se kreće od korijena scene, te se rekurzivno spušta po hijerarhiji scene, dakle obrađuju se sva djeca korijena scene, zatim dijeca svakog djeteta itd.

27.Zašto se koriste u interaktivnoj 3D grafici testovi presjeka koji se razlikuju od analitičkih rješenja.(analitička rješenja -kad koristimo jednadžbu zrake ,jednadžbu kugle.....)

Zato jer nam nekad nisu potrebni svi elementi,nego se testira samo jedan element(moguće izračunati u pripremnoj fazi).Npr. kod auta i podloge provjeravamo samo da li su gume direktno iznad površine,ostalo nas ne zanima.Drugi razlog je zbog optimalnosti,jer analitička rješenja nekad mogu biti dosta kompleksna,pa se onda koriste testovi odbacivanja koji su podosta jednostavniji.Ukoliko se koristi više testova/faza,prvo jednostavnije.

28.Postupak računanja presjeka zrake i kvadra (2D). Na svom primjeru objasnit kako proširit na 3D.

Za presjek zrake i kvadra može se korstiti tzv. metoda krišaka. Za 2D slučaj koristimo pravokutnik. Pravokutnik možemo zamisliti kao presjek dvaju krišaka, pri čemu je svaka kriška omeđena dvjema paralelnim pravcima u kojima leže nasuprotne stranice pravokutnika. Zraka siječe svaku krišku u dvije točke, ulaznoj(timin) i izlaznoj(timax), gdje je i index kriške. Od dvije ulazne točake za dvije kriške biramo onu točku koja je najudaljenija od ishodišta zrake. Isto tako tmax je izlazna točka najbliža ishodištu. Ukoliko je tmin <= tmax zraka siječe pravokutnik. Za proširivanje u 3D slučaj koristimo kvadar. Razlika je što onda imamo tri zrake i biramo tmax i tmin od tri točke.



30. Što je BSP stablo?

BSP stablo je struktura za organizaciju scene kod koje se prostor scene rekurzivno djeli na dva dijela nekom ravninom. Njegova glavna prednost u odnosu na hijerarhiju obujmica je u tome što prolaz BSP stabla omogućuje učinkovito sortiranje geometrije od naprijed prema natrag u odnosu na kameru što ima ulogu kod nekih medota ubrzavanja iscrtavanja.

34. Presjek zrake i trokuta

Jedan od načina računanja presjeka zrake i trokuta je korištenje baricentričnih koordinata. Baricentrične koordinate opisuju bilo koju točku u ravnini trokuta linearnom kombinacijom triju vrhova trokuta po izrazu T(u,v)=(1-u-v)*V0+uV1+vV2 za toče unutar trokuta u>=0, v>=0, u+v<=1. Uvrštavanjem izraza zrake(P(t)= O + tD, t>=0) u izraz za točku trokuta po baricentričnim koordinatama dobivamo sustav jednadžbi s tri nepoznanice u,v,t. Ove vrijednosti definiraju točku presjeka, ako su u i v u intervalui [0,1] zraka siječe trokut.

35. Oktalno stablo

Oktalno stablo je struktura vrlo slična BSP stablu poravnotom s osima. Gradi se na sljedeći način:

• Čitava scena se obujmljuje kvadrom poravnatim s osima – to je korijenski čvor stabal

• Kvadar se rekurzivno dijeli na sve manje potkvadre; u svakom koraku podjela se vrši duž sve tri osi istodobno, a točka podjele je središte kvadra trenutnog čvora X- trenutni kvadar se tako zapravo dijeli na 8 jednakih potkvadra, koji se dodaju kao djeca čvora X.

Oktalno stablo može biti efikasnije zbog svoje jednostavnosti - ne moraju se pohranjivati ravnine podjele u pojedinim čvorovima, jer je točka podjele uvijek u središtu kvadra.

36. Opišite detekciju sudara kod hijerarhije obujmica.

Hijerarhija obujmica zapisuje se u obliku stabla. Pritom račve stabla sadrže obujmice, a listovi poligone. Hijerarhija obujmica može se graditi automatski i to pristupom s vrha ili s dna. U pristupu s vrha prvo se nalazi obujmica za sve poligone u predmetu i to je vrh stabla. Zatim se poligoni dijele u dvije grupe, za svaku se grupu nalazi nova obujmica i te obujmice se dodaju kao račve u stablo. Postupak se rekurzivno ponavlja.ova metoda se puno češće koristi. Pri pristupu s dna, poligoni se ubacuju u stablo jedan po jedan, na taj način da minimalno povećaju ukupnu obujmicu.

2. Objasnite osnovnu ideju selektivnog odbacivanja poligona i ukratko opišite barem dvije metode.

Osnovna ideja selektivnog odbacivanja poligona (culling) je da ne šaljemo kroz cijeli sustav i ne obrađujemo (sjenčamo) SVE poligone koji se nalaze u sceni, već samo one koji su vidljivi iz trenutnog pogleda.

Metode koje se koriste:

- Backface culling
    - U geometrijskoj fazi
    - Odbacuju se poligoni koji su okrenuti od kamere. Pretpostavka je da se oni nalaze s „druge“ strane predmeta pa ih ne treba crtati. Orijentaciju trokutaodređuje redoslijed vrhova
- View-frustum culling
    - Odbacuju se svi poligoni koji su izvan projekcijskog volumena
- Portal culling
    - Scena se dijeli na ćelije koje imaju portale/vrata.
    - Rekurzivno se odbacuju dijelovi scene i iscrtavaju se samo vidljivi dijelovi ćelija
- Occlusion culling
    - Koristi se Z-buffer (dubina) kako bi se odbacili prekriveni predmeti (early-Z)

3. Na kojoj se ideji temelje tehnike razine detalja (LOD)? Objasnite kako nastaje problem skokova (popping) kod diskretnih razina detalja. Kako se on rješava?

Ideja je da se predmeti koji su udaljeniji od kamere mogu zamijeniti jednostavnijim modelom (manje poligona) bez da korisnik uoči razliku u detaljima radi udaljenosti samog predmeta, a pritom se povećavaju performanse.

Kod diskretnih razina detalja imamo za jedan predmet npr. 3 modela različite detaljnosti. Na različitim granicama udaljenosti predmeta od kamere se učitavaju različiti modeli (udaljenije – manje detaljni modeli). Prilikom prelaska te granice simo-tamo (recimo igrač trza naprijed-natrag) uočljivo je da se događaju promjene modela. To se rješava histerezom – granica za učitavanje manje detaljnog modela nije ista kao i granica na kojoj se taj manje detaljni model natrag zamjenjuje više detaljnim modelom. Problem se dodatno rješava miješanjem razina detalja. Na kratko vrijeme su učitana oba modela predmeta te se između njih izvodi miješanje

po prozirnosti (alpha-blending). Moguće je još i napraviti morphing iz jednog modela u drugi ili vršiti eliminaciju bridova kako je predmet sve udaljeniji.

1. Objasni vertex i pixel shader

Vertex Shader (procesor vrhova) - je prva programabilna faza koja implementira funkcije transformacije u prostor kamere, sjenčanje vrhova i projekcije. Često se koristi za animaciju a danas rijeđe za sjenčanje. Procesor vrhova pokreće se za svaki vrh koji je dan GPU. Vrh je predstavljen položajem, normalom, teksturom i bojom (samo položaj je nužan). Procesor vrhova u jednom trenu radi na jednom vrhu i nema pristup drugim vrhovima (ne može dodati ni brisati vrhove). Minimalni rezultat je položaj vrha nakon projekcije.

Pixel Shader (procesor točaka) - je programabilna faza koja implementira funkcijsku fazu sjenčanja točaka. Ulazni podatak u procesor točaka je fragment, a zadatak procesora je sjenčanjem odrediti boju fragmenta te ju poslijediti na izlaz (u fazi stapanja). U jednoj točki može biti više fragmenata. Nije moguće dobiti podatke o drugim točama. Primarno se koristi za preslikavanje ravnina, sjena i spekularno sjenčanje.

Shader jezici : HLSL, GLSL, Cg

C-like Shader jezici : HLSL, GLSL, Cg

Ubershader: Složeni program za sjenčanje koji se više puta prevodi kako bi se dobile specijalizirane inačice za svaku kombinaciju materijala i svjetla. U idealnom slučaju sve se inačice mogu generirati unaprijed i učitavati prema potrebi, no kod složenijih programa to nije izvedivo, već je potrebno odrediti koje se kombinacije mogu pojaviti u aplikaciji.

2. Sve 4 metode opiši:

a. Ubershader -> odgovoreno iznad

b. Dinamičko grananje

Pretpostavimo da neka aplikacija podržava M svjetla i N vrsti materijala, te da na neki predmet može u bilo kojem trenutku djelovati do L svjetala. Možemo sjenčanje za te vrste svjetla i materijala implementirati unutar jednog programa pseudokoda:

```
float4 myShader (...){

float4 color;

for(i = 0 do L){

color+=light_compute(L[i],materijal);

}

return color;

}

float4 light_compute(light, materijal){

switch(light){

Case0:
```

switch(materijal):

Case 0:

Return ... formula;

}

}

Ovakvo rješenje je jednostavno no daje loše performanse na starijim grafičkama zbog dinamičkog grananja.

c. Višeprolazno osvjetljenje

Kod višeprolaznog osvjetljenja u svakom se prolazu računa osvjetljenje za jednu vrstu svjetlosi, a rezultat se aditivnim miješanjem u fazi stapanja zapisuje u spremnik boja. Prednost je pojednostavljenje programa za sjenčanje (jer se računa za svaki tip svjetlosti), a mana je visok stupanj korištenja sabirnice te ponavljanje dijela proračuna u programima za sjenčanje u svakom prolazu.

d. Odgođeno sjenčanje

Tehnika sjenčanja kod koje se sjenčanje radi nakon određivanja vidljivosti. Postupak se provodi u dva ili više prolaza iscrtavanja. U prvom prolazu, čitava se geometrija iscrtava bez sjenčanja, ali uz uključen Z-spremnik. U geometrijski spremnik se zapisuju podaci o vidljivim točkama površine (dubina, normala, teksturne koordinate, parametri materijala). U drugom prolazu ne iscrtava se cijela geometrija već samo jedan četerokut koji prekriva cijeli zaslon, a na koji se efektivno lijepi slika screne. Spremljeni podaci iz prethodnog koraka se koriste za izracun osvjetljenja u pojedinim tockama.

Glavna prednost su performanse jer se bitno smanjuje količina proračuna sjenčanja. Druga prednost je to što smo proračun vezan za materijal (prvi prolazak) odvojili od proračuna vezanih za svjetla (drugi prolazak).

7. Što je efekt, od čega se sastoji? Što je common shader core?

Efekt - skup programa za sjenčanje i svih potrebnih postavki protočnog sustava (npr. Uključen/isključen Z-spremnik). Zapisuje se u jedinstvenoj datoteci korištenjem jezika za efekte. Korisnik upravlja svojstvima efekta kroz globalne parametre efekta koje je moguće postavljati kroz sučelje alata za razvoj efekata (npr. Phongovo sjenčanje -parametri materijala, svijetla) Jedinstvena procesorska jezgra (common shader core) - pojam se odnosi na činjenicu da se za sve vrste procesora koristi jedinstven programski model. Procesori (shaderi) se programiraju u jezicima za sjenčanje sličnima C-u (HLSL, Cg, GLSL), a programi se prevode u asemblerski jezik neovisan o hardveru, što tvori virtualni stroj.

8. Što je GPGPU, nabroji primjene

GPGPU (General purpose GPU) - GPU sve više nadilaze svoju primarnu zadaću stoga su sve pogodni i za općenite proračune. Današnji GPU ima SIMD arhitekturu za strujnu obradu podataka (stream processing) te mogu ostvariti i do 2 reda velicine veće brzine nego CPU. NVIDIA - CUDA, Microsoft - DirectX, OpenGL - open source Primjene : fizikalne simulacije, obrada slike, obrada videa, kriptografija, kriptoanaliza, baze podatka, neuronske reže, složene operacije linearne algebre, sortiranja i pretraživanja

9. Objasni odbacivanje stražnih poligona(backface culling), odbacivanje po projekcijskom volumenu(view-frustum culling), odbacivanje prekrivenih poligona(occlusion culling) i portalno odbacivanje(portal culling).

Odbacivanje stražnjih poligona (backface culling) - Tehnika odbacivanja poligona koji su okrenuti od kamere koja se uobičajeno odvija u geometrijskoj fazi protočnog sustava. Ako normala poligona (određena redoslijedom vrhova) i kut gledanja zatvaraju tupi kut, poligon se odbacuje.

Odacivanje po projekcijskom volumenu (view-frustum culling) - oslanja se na ideju da ono što kamera trenutno ne vidi nije potrebno iscrtavati tj. svi elementi izvan projekcijskog volumena su odbačeni. Za provjeru vidljivosti koriste se hijerarhije obujmice, BSP i oktalna stabla. Portalno odbacivanje (portalno odbacivanje) - često je potrebno simulirati arhitekture koje sadrže velik broj soba, gdje se dio soba ne vidi iz trenutne pozicije kamere te ih nije potrebno iscrtavati. Scena se dijeli na čelije (sobe) i te sobe imaju portale (izlaze/vrata) prema drugim čelijama. Za svaku čeliju gradimo graf susjedstva i scena se iscrtava rekurzivno. Izvodi se u aplikacijskoj fazi.

Odbacivanje prekrivenih poligona (occlusion culling) - predmeti u sceni su česo prekriveni drugim predmetima i u završnoj sceni neće biti iscrtani jer će korištenjem Z-spremnika biti odbačeni. Metoda Z-spremnika izvodi se tek u konačnoj fazi i odrađuju se nepotrebne prethodne obrade. Ideja je prekrivenu geometriju odbaciti čim ranije. Neke metode koje se koriste su sklopovska provjera prekrivenosti (hardware occlusion queries), Z-odbacivanje (Z-cull) i rani-Z.

10. Objasni Z-culling and backface culling. Koja je ideja iza nivoa detalja (LOD – level of details)? Objasni zamjenu razine detalja i efekt skokova (unpleasant "popping") effect.

Backface culling - objašnjeno iznad Z-culling - u fazi prolaza trokuta ispituje je li neki dio trokuta prekriven ranije iscrtanom geometrijom, ako je, odbacuje ga prije ulaska u fazu sjenčanja točaka Rani Z (nije bilo ali je usko povezano) - nakon generiranja elemenata, a neposredno prije izvođenja programa za sjenčanje, na pojedinom elementu GPU još jednom provodi test prekrivenosti. Izravno uspoređuje izračunatu dubinu fragmenta s dubinom iz Z-spremnika. Ako je veća za trenutnu točku, odbacuje element.

Ideja iza LOD - predmet je moguće zamijeniti jednostavnijom inačicom (s manjim brojem poligona) kada je udaljen od kamere, a da promatrač ne vidi razliku u kvaliteti Zamjena razine detalja - nakon što se ispune kriteriji za zamjenu trenutne razine detalja nekog predmeta, potrebno je provesti način zamjene modela što neprimjetnije, u protivnom se javlja ružan efekt skokova (popping).

12. Objasni trake trokuta(triangle strips), lepeze trokuta(fans) i mreže trokuta(meshes).

Trake trokuta - prvi trokut je definiran sa 3 vrhova. Svaki sljedeći je definiran sa zadnja dva dodana vrha prvog trokuta kojemu se doda treći vrh i tako čini sljedeći trokut. S toga dolazimo da je prosječan broj vrhova po trokutu 1 + 2/m (m - broj trokuta). Za velik broj trokuta ta formula teži 1.

Lepeze trokuta - sastoje se od jednog središjeg vrha V0 i n vrhova (V1,..,Vn) koji oko njega tvore lepezu. Prosječan broj vrhova isti je kao i za trake poligona : 1 + 2/m, no u stvarnom svijetu lepeze se pojavljuju rijeđe nego trake. Lepeze su dobre za pretvorbu kompleksnih poligona s više vrhova u trokute.

Mreže trokuta - najvažnija sktruktura za prikaz 3D geometrije i moderno grafičko sklopovlje prilagođeno je njihovom radu. Mreža trokuta predstavljena je skupom vrhova i slijedom indeksa. Vrh definiraju koordinate, normala, teksturne koordinate,... Slijed indeksa definira kako su vrhovi međusobno povezani u trokute. Iscrtavanje mreže trokuta radi se tako da pošaljemo vrhove i indekse protočnom sustavu pozivom odgovarajućih funkcija programskog sučelja.

Dva načina predaje vrhova :

- Spremnik vrhoa (vertex buffer)
- Struje vrhova (vertex streams)

1. Što je grafički protočni sustav (GPS)? Koje su njegove glavne faze? (2 b)

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. Bududi 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 sastoji se od triju glavnih faza:

- Aplikacijska faza,
- Geometrijska faza,
- Faza rasteriziranja.

2. Za koji postupak se u GPS-u koriste prednji i stražnji spremnik? U kojoj fazi GPS-a se koristi ovaj postupak? Objasni. (3b)

Kod naivnog pristupa iscrtavanju rasterizator crta boje u spremnik, koji se svaki put prethodno izbriše. Istovremeno, video upravljačka jedinica iz spremnika crta na zaslon. Ova dva procesa su asinkrona, dakle, u trenutku kada video upravljačka jedinica iz spremnika crta na zaslon, vrlo je vjerovatno da rasterizator nije dovršio čitavu sliku, nego je negdje u procesu crtanja. Dakle, pri svakom osvježavanju zaslona dobit demo sliku u drugačijoj fazi dovršenosti. Da se izbjegne ovaj efekt koriste se dva spremnika prednji i stražnji. Rasterizator crta u stražnji spremnik koji se ne vidi, jer video upravljačka jedinica crta na zaslon uvijek prednji spremnik. Kada je slika spremna, izvršava se zamjena uloga spremnika tj. stražnji postaje prednji. U sljededem iscrtavanju na zaslon video upravljačka jedinica poslužuje novu sliku. Ovime se osigurava da video upravljačka jedinica uvijek na raspolaganju ima dovršenu sliku.

5. Što su obujmice i čemu služe? Navedi primjere. Objasni pojednostavljeni test kojim se računa presjek obujmice i projekcijskog volumena. U kojim slučajevima je bolje raditi pojednostavljeni test nego puni? (4b)

Obujmica je jednostavan geometrijski oblik koji obuhvada složeniji geometrijski skup. Obujmice služe za aproksimciju složenog predmeta jednostavnijim predmetom radi lakše manipulacije (obično ispitivanja presjeka). Primjeri obujmica su: kugla, elipsoid, kvadar paralelan s osima, opdeniti kvadar, konveksna ljuska. Pri pojednostavljenom testu ne ustanovljava se točno siječe li obujmica volumen, nego se ona samo klasificira u dva moguda stanja: ''vani'' ili ''vjerojatno

unutra''. U ovom drugom slučaju velika je vjerojatnost da je čitava obujmica, ili barem njezin dio, unutar projekcijskog volumena. U ovom slučaju postoje dvije strategije: ili se crta sva geometrija unutar obujmice, ili se ispisuje sljededa razina obujmice. Pošto ne znamo jeli obujmica unutar volumena, niti jedna od ovih strategija nije zajamčeno optimalna. Pojednostavljeni test je bolje raditi u slučajevima gdje su velike scene, pa je poprilična vjerojatnost da de i jednostavni test odbaciti puno obujmica koje su daleko od projekcijskog volumena.

1. Objasnite algoritam traženja presjeka zraka-kugla.

Presjek zraka-kugla moguće je pronaći izravnom uporabomm jednadžbi. Uvrštavanjem izraza koji opisuje zraku u izraz koji opisuje kuglu dobiva se kvadratna jednadžba. Determinanta određuje da li zraka siječe, dira ili ne dotiče kuglu. Postupak se optimizira korištenjem jednostavnih testova za rano odbacivanje. Testovi provjeravaju neke parametre kao što su udaljenost kugle od zrake ili smjer zrake u odnosu na kuglu te na se osnovu rezultata nastavlja s algoritmom ili dolazi do prekida jer presjek ne postoji.

2. Opišite detekciju sudara kod hijerarhije obujmica.

Hijerarhija obujmica zapisana je u obliku stabla. Pritiom račve sadrže obujmice, a listovi geometriju. Pri pristupu s vrha, prvo se nalazi obujmica za sve poligone u predmetu, odnosno vrh stabla. Zatim se poligoni dijele u dvije grupe te se za svaku grupu nalazi nova obujmica. Ovaj postupak se rekurzivno ponavlja. Sudar hijerarhija obujmica obavlja se rekurzivno. Ispituje se sudaraju li se dva predmeta opisana hijerarhijskim stablom. Ukoliko se utvrdi da se čvorovi više razine stabla ne sudaraju, postupak se prekida jer nema sudara, a ukoliko se sudaraju, isti postupak se provodi nad djecom.

- BSP stablo? (116. str.)
BSP stablo je struktura za organizaciju scene kod koje se prostor scene rekurzivno dijeli na dva dijela nekom ravninom. Njegova glavna prednost u odnosu na hijerarhiju obujmica je u tome što prolaz BSP stabla omogućuje učinkovito sortiranje geometrije od naprijed prema natrag u odnosu na kameru što ima ulogu kod nekih metoda ubrzavanja iscrtavanja.

- Detekcija sudara.
Detekcija sudara je metoda u kojoj se testovi presjeka koriste za ustanovljavanje sudara između predmeta i scene, te gdje i kako se sudaraju.
Detekcija sudara - metode testiranja da li se dva geometrijska elementa sijeku (i gdje).

- Definiraj graf scene.
Graf scene = podatkovna struktura u koju se virtualna scena sprema na organiziran i strukturiran način. Omogućuje logičnu organizaciju scene, lakšu manipulaciju i učinkovitije iscrtavanje.

- Detekcija sudara kod aproksimacije predmeta zrakama.
Postavimo strateška mjesta na predmetu tj. na mjesta gdje se očekuje sudar s okolinom (dno kotača na automobilu) i tražimo presjek zrake sa scenom: *ukoliko je **d = 0** → predmet dotiče scenu
*ukoliko je **d < 0** → predmet je u sceni
*ukoliko je **d > 0** → predmet je iznad scene/površine tj. nema presjeka doticaja.

- Detekcija sudara kod hijerarhije obujmica.
Svaki predmet predstavimo kao skup poligona. Pretvorimo ga u hijerarhiju (stablo) obujmica. Pritom račve sadrže obujmice, a listovi geometriju.
Pri pristupu s dna (bottom-up approach), poligoni se ubacuju u stablo jedan po jedan, na taj način da minimalno povećaju ukupnu obujmicu.

Pri pristupu s vrha (top-down approach), krećemo od vrha hijerarhija. Prvo se nalazi obujmica za sve poligone u predmetu i to je vrh stabla.
Ako se već najviša razina obujmica ne siječe → postupak se prekida i nema sudara
Ukoliko se već najviša razina obujmica siječe → ispitujemo obujmice A i B koje mogu biti čvorovi ili listovi

Ukoliko su listovi → ispitujemo sve poligone ispod njih

Ukoliko je A list, a B čvor → pozivamo funkciju testsudara rekurzivno za B i ispitujemo sudar sa A

Ako su oboje čvorovi → svaka obujmica ispod A se ispituje na sudar sa B

Ako kroz cijelu hijerarhiju nije pronađen sudar → onda nema sudara.

- Format VRML. Da li je to standardni, otvoreni ili vlasnički format?
VRML je standardni format. Zasnovan je na grafu scene i ima sve normirane čvorove.
Karakteristike: može se kreirati i urediti u običnom editoru, ne ovisi o operacijskom sustavu i podrščci.
Napredne karakteristike: naprednije funkcije senzori, interpolatori i staze kojim se mogu definirati jednostavne animacije i interakcije sa korisnikom.

- Obujmice i čemu služe? Navedi primjere. Objasni pojednostavljeni test kojim se računa presjek obujmice i projekcijskog volumena. U kojim slučajevima je bolje raditi pojednostavljeni test nego puni? (115.str.)
Obujmica je jednostavan geometrijski oblik koji obuhvaća složeniji geometrijski skup. Obujmice služe za aproksimaciju složenog predmeta jednostavnijim predmetom radi lakše manipulacije (obično ispitivanja presjeka).
Primjeri obujmica su: kugla, elipsoid, kvadar paralelan s osima, općeniti kvadar, konveksna ljuska.
Pojednostavljeni test: ne ustanovljava se točno siječe li obujmica volumen, nego se ona samo klasificira u dva moguća stanja: "Vani" ili "vjerojatno unutra" s obzirom na projekcijski volumen. U ovom drugom slučaju postoji velika vjerojatnost da je čitava obujmica, ili neki njezin dio, unutar projekcijskog volumena. U ovom slučaju postoje dvije strategije: ili crtamo sve unutar obujmice (ukoliko nije cijela unutar volumena nacrtat ćemo nepotrebne poligone) ili ispitujemo sljedeću razinu obujmica (ukoliko ispitujemo sljedeću razinu, a prethodna obujmica je u potpunosti unutar projekcijskog volumena, radimo nepotrebne testove).
Pojednostavljeni testovi se koriste u velikim scenama gdje je velika vjerojatnost da će i pojednostavljeni test odbaciti puno obujmica koje su izvan projekcijskog volumena.

- Obujmice i za što se koriste te nabroji par primjera.
Obujmice su jednostavan geometrijski oblik koji obuhvaća složeniji geometrijski skup. Služe za aproksimaciju složenog predmeta jednostavnijim predmetom radi lakše manipulacije. Važna uporaba obujmica je u ubrzavanju iscrtavanja.
Složenija obujmica → bolja aproksimacija predmeta → raste složenost testa presjeka.
Primjeri obujmica: kugla, elipsoid, kvadar paralelan s osima, općeniti kvadar, konveksna ljuska,...

- Odabir predmeta bacanjem zrake i iscrtavanjem? (str. 167./168.)
Odabir 2D pokazivačem bacanjem zrake – iz točke pokazivača baca se zraka u ravninu te se gleda presjek s elementima scene. Uzima se najbliži presjek. Ova je metoda pogodna za korištenje kod programa za crtanje s podrškom za više slojeva platna.

- 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.

DRUGI ODGOVOR

Oktalno stablo je struktura vrlo slična BSP stablu poravnotom s osima. Gradi se na sljedeći način:

- Čitava scena se obujmljuje kvadrom poravnatim s osima – to je korijenski čvor stabal
- Kvadar se rekurzivno dijeli na sve manje potkvadre; u svakom koraku podjela se vrši duž sve tri osi istodobno, a točka podjele je središte kvadra trenutnog čvora X- trenutni kvadar se tako zapravo dijeli na 8 jednakih potkvadra, koji se dodaju kao djeca čvora X.

Oktalno stablo može biti efikasnije zbog svoje jednostavnosti - ne moraju se pohranjivati ravnine podjele u pojedinim čvorovima, jer je točka podjele uvijek u središtu kvadra.

- Općeniti postupak za ispitivanje presjeka zrake s općenitim poligonom. Kako možemo ispitati nalazi li se dvodimenzionalna točka P unutar dvodimenzionlnog poligona?

Općenito presjek zrake i bilo kojeg geometrijskog tijela može se dobiti na način da uvrštavamo izraz zrake u jednadžbu tijela koje zraka siječe. Ispitivanje jeli točka P u poligonu može se vršiti na razne načine. Jedan od poznatijih je provlačenje zrake iz točke u proizvoljnom smjeru (obično smjer jedne od osi u kojoj projekcija ima najveću površinu.). Zraka presijeca poligon n puta. Ako je n neparan broj onda se točka P nalazi u poligonu, a ako je paran ili 0 točka je izvan poligona.

- Presjek ravnine i kvadra.

Ako su bilo koja dva vrha kvadra na suprotnim stranama ravnine, očito je da imamo presjek kvadra i ravnine. Uvrštavanjem bilo koje točke u jednadzbu ravnine mozemo izracunati udaljenost tocke od ravnine. Tu udaljenost izracunamo za svih 8 vrhova kvadra. Ako svih 8 udaljenosti ima isti predznak (odnosno s iste strane ravnine su) Tada nema presjeka.
Ukoliko je jedan predznak razlicit (nalazi se s druge strane ravnine) → imamo presjek.
Postupak možemo pojednostaviti tako da ispitujemo samo dva vrha na dijagonali kvadra najbližoj smjeru normalne na ravninu.

- Presjek trokuta i trokuta.

Presjek trokuta ispitujemo sljedecim postupkom: (Trokut T1 u ravnini R1 i T2 u ravnini R2)

1. Izracunamo udaljenost svih vrhova T1 do R1, ako su udaljenosti istog predznaka nema presjeka
2. Isto ponovimo za T2
3. Pravac L je presjek R1 i R2. Nalazimo presjek T1 i T2 sa L i dobijemo intervale I1 i I2, ako se intervali preklapaju → imamo presjek.

- Presjek zrake i kugle.

Presjek zraka-kugla možemo dobiti uvrštavanjem jednadžbe zrake u jednadžbu kugle i rješavanjem kvadratne jednadžbe dobivamo točke presjeka zrake s kuglom, međutim postoji i jednostavniji način u kojem provjeravamo gleda li zraka prema središtu kugle ili od njega. U slučaju da ne gleda prema središtu kugle ($d<0$), a ishodište zrake se ne nalazi unutar kugle ($l2>r2$) zraka sigurno ne siječe kuglu, ako dosad nismo ustanovili da zraka ne siječe kuglu nastavljamo s ispitivanjem u kojem provjeravamo udaljenost zrake od kugle (m), ako je ta udaljenost veća od polumjera zraka ne siječe kuglu i postupak se prekida, u protivnom znamo da postoji presjek i postupak se nastavlja kako bi našli točke presjeka.

- Presjek zrake i kvadra (2D). Na svom primjeru objasni kako proširiti na 3D.

U 2D slučaju imamo zraku i neki pravokutnik. Možemo svaki brid pravokutnika zapisati kao jedan pravac. Sad imamo 4 pravca umjesto bridova i jednu zraku koju usmjeravamo prema njima. Postupak se svodi na traženje zajedničke točke naše zrake i zadanih pravaca. Za dva pravca ćemo naći rješenje: na mjestu gdje zraka ulazi (tmin) u pravokutnik i na mjestu gdje izlazi (tmax) iz njega. Mi ćemo kao točku presjeka uzeti onu koja nam je bliža (ulazna) jer nema smisla da izlazna točka bude točka u kojoj se naša zraka sudarila s pravokutnikom. Rješenja se traže samo u segmentima pravaca ograničenim vrhovima pravokutnika. Pravac se proteže u beskonačnost i može se desiti da dobijemo besmisleno rješenje, tj. točku presjeka koja je dio pravca brida, ali

uopće nije dio pravokutnika. Dakle, od dvije ulazne točke biramo onu točku koja je najudaljenija od ishodišta zrake. Isto tako tmax je izlazna točka najbliža ishodištu. Ukoliko je tmin <= tmax zraka siječe pravokutnik.
Ideja za proširiti ovo u 3D prostor bila bi da radimo sve iste opisane korake kao i u 2D samo što ih ponavljamo u iteracijama i u svakoj iteraciji uvećamo z koordinatu za 1. I dalje ćemo u svakoj iteraciji imati po 4 pravca samo što ćemo u svakoj iteracijom promatrati jedan viši kat našeg kvadra. Z uvećavamo sve dok ne dođemo do z-a koji je jednak zadanoj visini kvadra. Ako pravac neće biti paralelan sa našim „katom“ (kao što je to uvijek bio slučaj u 2D prostoru) pravokutnikom dobit ćemo max jednu točku presjeka u jednoj iteraciji (više ih nema nužno 2 kao u 2D) ili 0 ako ih pravac uopće ne siječe.

DRUGI ODG Za ovaj presjek koristimo metodu krišaka. Kvadar zamislimo kao presjek triju krišaka, Zraka siječe te krište u dvije točke Tmin(ulazna) i Tmax(izlazna). Kao glavnu ulaznu točku biramo onu koja je najudaljenija od ishodišta zrake: Tmin=max(Tmin1, Tmin2, Tmin3), te isto vrijedi i za izlaznu Tmax=max(Tmax1,Tmax2,Tmax3).
Ako je Tmin > Tmax → zraka siječe kvadar.

- Presjek zrake i poligona.
Presjek se računa u 3 koraka:

1. Izračunamo presjek zrake s ravninom u kojoj leži poligon
2. Zatim poligon i točku presjeka projiciramo u koordinatnu ravninu s najvećom površinom projekcije.
3. Ispitujemo da li je preslikana točka presjeka u 2D poligonu dobivena projekcijom na ravninu. Odnosno ispitujemo da li je točka unutar 2D poligona. To možemo lako ispitat tako da povučemo zraku iz te točke u proizvoljnom smjeru.

Ako povučena zraka siječe poligon neparan broj puta → točka je u poligonu.

- Presjek zrake i trokuta.
Presjek nalazimo korištenjem baricentričnih koordinata (opisuju bilo koju točku u ravnini trokuta linearnom kombinacijom triju vrhova po izrazu T(u,v) = (1-u-v)V0 + u*V1 + v*V2 za točke unutar trokuta (u,v>=0, u+v<=1).
Uvrštavanjem jednadžbe zrake u izraz za točku trokuta po baricentričnim koordinatama (V(x,y)) dobivamo sustavj jednadzbi s 3 nepoznanice (U,V,T) koje definiraju točku presjeka.
Ukoliko su U i V unutar [0,1] → zraka siječe trokut.

- Presjek zrake sa scenom (ne računski nego općenito)? Ako u scenu dodamo kuglu i valjak da li i kako moramo promijeniti ray tracing metodu?
Gleda se presjek zrake sa svakim pojedinim objektom scene. Potom se uzima najbliži presjek. Ako u raytracing metodi imamo definirane funkcije za pronalaženje presjeka zrake i kugle te zrake i valjka, tada ne moramo ništa mijenjati. Inače, moramo napisati nove generičke metode kojima definiramo traženje presjeka zrake s kuglom i valjkom.

Presjek zrake sa scenom gdje je d udaljenost ishodišta zrake do njenog najbližeg presjeka sa scenom:

d=0 → predmet dotiče scenu
d>0 → predmet je iznad scene
d<0 → predmet ulazi u scenu

- 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. Najrašireniji je OpenGL.

- Rekurzivni prolazak kroz stablo grafa scene (pseudokod). (119-124. str)
Rekurzivna funkcija prima kao parametre čvor s kojim se trenutno radi i trenutno stanje procesa. U prvom pozivu funkciji prosljeđuje se kao parametar ishodišni čvor (korijenski čvor) grafa scene

i početno inicijalizirano stanje. U funkciji se računa novo stanje s obzirom na čvor koji se trenutno obrađuje. Stanje može sadržavati razne podatke (najvažnija je matrica transf.). U funkciji UpdateNode se globalna matrica transf. množi s lokalnom matricom transf. čvora te se matrice akumuliraju. Na taj način stanje uvijek sadrži globalnu matricu transf. kojom se trenutni čvor može prevesti iz lokalnih u globalne koordinate. U varijablu localState se sprema obrada trenutnog čvora. Na kraju se za eventualne čvorove djecu rekurzivno poziva ista funkcija processNode.

→ ***PSEUDOKOD:***

```
processNode (node,state) {
        localState = update (node,state); // računanje novog stanja
        switch (node.type)  {
                case <XYZ>: // posebna obrada za tip XYZ, npr. crtanje
        }
        for (i=0; i < node.numberOfChildren; i++)  {
                process (node.child[i],localState); // rekurzija
        }
}
updateNode (node,state) {
        StateType localState;
        localState.gt = state.gt • node.t; // globalna transf. ; izračun ostalih parametara stanja, brojčanika i sl.
        return (localState);
}
```

→ **POSTUPAK PROLASKA KROZ GRAF SCENE:**

Graf scene je podatkovna struktura u koju se virtualna scena sprema na organiziran i strukturiran način. Omogućuje logičnu organizaciju scene, lakšu manipulaciju i učinkovitije iscrtavanje. Prolaz kroz graf scene standardni je rekurzivni postupak za obavljanje neke operacije nad svim čvorovima u sceni.

Kreće se od korijena scene te se rekurzivno spušta po hijerarhiji, pritom obrađujući svu djecu korijena scene, zatim njihovu djecu itd. Ovakvim prolaskom izvode se operacije iscrtavanja scene, testiranja presjeka, traženja elemenata i pojednostavljivanja scene.

→ Iscrtavanje scene - Test presjeka/sudara - Traženje pojedinog elementa - Pojednostavljivanje scene - Razne operacije specifične za pojedine primjene

- Slika grafa scene ormarića i ostalih objekata. Nešto slično kao onaj primjer u knjizi sa avionom. Pitanja:

a) nabroji koji su korijenski čvorovi, koji su račve i koji su listovi.

Korijenski čvor R obuhvaća čitavu scenu.

Račve su transformacijski čvor T (mjesta lijepljenja manjih dijelova na veće, na svako mjesto transformacije staviti račvu T)

Listovi su pojedini elementi (dijelovi) scene.

b) nešto tipa da si morao kopirati objekte koji su jednaki.

Imali smo dvije vaze i da bi uštedili na memoriji oni zadnji čvorovi pokazuju na isti list.

c) koje bi čvorove trebao promijeniti ako bi htio pomaknuti i smanjiti ormarić?

Ispod čvora R i iznad prvog čvora T staviti još jedan čvor T koji smanjuje i pomiče čitavu scenu.

- 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.