1. Objasni procesor vrhova i procesor točaka (vertex i pixel shader)
    a. Nabrojati bar 2 shader jezika
    b. Koji je C-like shader jezik
    c. Što je Ubershader
2. Sve 4 metode opiši:
    a. Ubershader
    b. Dinamičko grananje
    c. Višeprolazno osvjetljenje
    d. Odgođeno sjenčanje
3. Objasni potrebu za filtriranjem tekstura. Objasni metodu najbližeg susjeda i bilenearne interpolacije
4. Objasni mipmapping and trilinearnu interpolaciju
5. Što je preslikavanje okoline (environment mapping)? Objasni dva algoritma.
6. Što je preslikavanje neravnina (bump mapping?) Koja je razlika između preslikavanja normala i preslikavanja pomaka (normal and displacement mapping)?
7. Što je efekt, od čega se sastoji? Što je common shader core?
8. Što je GPGPU, nabroji namjene
9. Objasni odbacivanje stražnih poligona(backface culling), odbacivanje po projekcijskom volumenu(view-frustum culling), odbacivanje prekrivenih poligona(occlusion culling) i portalno odbacivanje(portal culling).
10. Objasni Z-culling and backface culling. Koja je ideja iza nivoa detalja (LOD - level of details)? Objasni unpleasant "popping" effect.
11. Koja je ideja iza UVO (umreženog virtualnog okruženja) i navedi neke primjere. Što je upravljanje prema području interesa (area of interest management)?
12. Objasni trake trokuta(triangle strips), lepeze trokuta(fans) i mreže trokuta(meshes).
13. Objasniti shadow mapping.

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.

Kombiniranje vise svjetala i materijala
1.dinamicko
2.ubserhader
...

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.

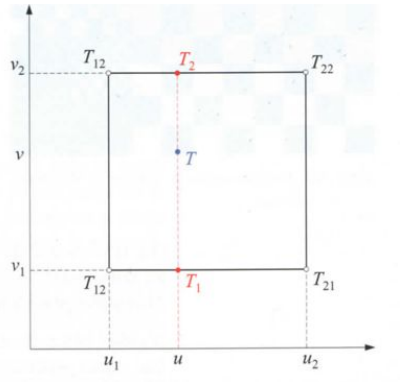
d. Odgođeno sjenčanje

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).

3. Objasni potrebu za filtriranjem tekstura. Objasni metodu najbližeg susjeda i bilenearne interpolacije

Filtriranje teksture - kada se pomoću u, v koordinata iz slike dohvaća vrijednost teksela, javljaju se problemi poduzorkovanja ili naduzorkovanja (ovisno o udaljenosti od kamere). Konačni rezultat preslikavanja teksture je pojava slike na zaslonu, gdje veličina ovisi o udaljenosti predmeta od kamere. Idealno, korespondencija piksel-teksel je 1:1, no ako je slika umanjena (predmet udaljen od kamere) postoji mogucnost da nastane alijasing

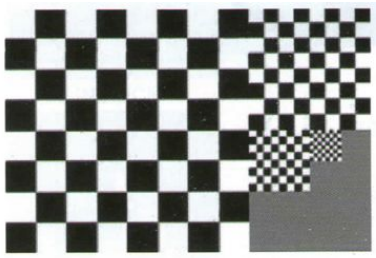
Metoda najbližeg susjeda - Jednostavna metoda u kojoj se dohvaća se teksel najbliži u,v koordinatama, no javljau se problemi efekta blokova. (slika pikselizirana)
Bilinearna interpolacija - vrši se interpolacija unutar kvadra 2x2 susjena teksela odnosno 4 točke najbliže centru točke koje iscrtavamo. Svodi se na dvije linearne interpolacije, prvo po osi u, pa po v. Slika je zamučena ali je ublažen efekt blokova.



Bikubna interpolacija - interpolira se unutar mreže 4x4 ili 5x5 susjednih teksela. Postupak je analogan bilinearnoj interpolaciji samo se u ovom slučaju vrše dvije uzastopne bilinearne interpolacije. Rezultat je mnogo bolji i gotovo je izgubljen efekt blokova navodno nije bitno

4. **Objasni mipmapping i trilinearnu interpolaciju**

MipMapping - kada 1 piksel prekriva više teksela. Mipmap je metoda koja se najčešće koristi za rješavanje problema umanjene slike na način da se skup manjenih i filtriranih slika teksture pripremi unaprijed i pohrani zajedno s originalnom slikom. Svaka slika 2x manja od prethodne, a datoteka je veća samo za ⅓. Prilikom preslikavanja teksture koriste se tekseli iz slika čija veličina trenutno odgovara veličini piksela, a određena je parametrom d (d-razina detalja, LOD).

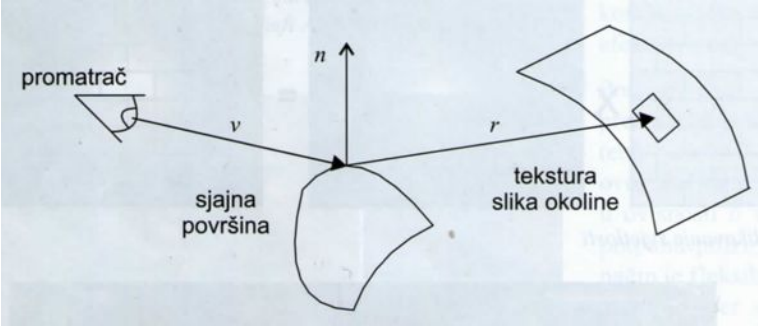


Slika 145: Primjer mipmap teksture s 4 razine detalja

Trilinearna interpolacija - funkcija korespodencije, ulazi su parametri : u,v,d. Teksel se dobiva tako da se korištenjem parametra d odrede dvije susjedne mipmap razine teksture, te se potom iz njih dohvate tekseli korištenjem bilinearne interpolacije. Dobiveni tekseli se linearno interpoliraju po parametru d i dobiva se konačni uzorak.

5. Što je preslikavanje okoline (environment mapping)? Objasni dva algoritma.

Preslikavanje okoline - je metoda simuliranja zrcaljenja na oblim površinama. Ideja je da se slika okoline pohrani na teksturu predmeta, koja se zatim primjenjuje na predmet. Koordinate teksture se računaju prilikom iscrtavanja (ovisi o kutu gledanja promatrača)



Kuglasto preslikavanje - Najjednostavnija metoda, kuglasta tekstura okoline sadrži sliku okoline dobivenu ortografskom projekcijom na površinu savršene kugle koja se nalazi oko predmeta. Teksel se odreduje reflektiranom zrakom R (gdje sjece kuglu).

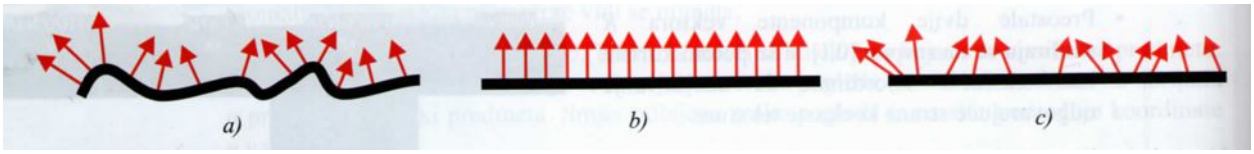
Kockasto preslikavanje - najšire korištena metoda, kockasta slika generira se projekcijom okoline na 6 strana kocke postavljene oko predmeta, a rezultat se pohranjuje u jednu teksturu od 6 djelova kockaste teksture. Pri sjenčanju predmeta vektor reflektirane trake R koristi se za određivanje teksturnih koordinata za uzorkovanje kockaste teksture okoline.
Postupak :

- Vektor R se normalizira
- Određuje se najveća komponenta od R(x,y ili z) i predznak i određuje se strana kocke za uzorkovanje, ostale dvije komponente od R skaliraju se na raspon [0-1]
- Uzorkuj odgovarajuću stranu (u,v koordinate 2 manje komponente r)

6. **Što je preslikavanje neravnina (bump mapping?) Koja je razlika između preslikavanja normala i preslikavanja pomaka (normal and displacement mapping)?**

Preslikavanje neravnina (bump mapping) - skup tehnika za simuliranje neravnih površina. Ideja je da se poremećajem normale u svakoj točki predmeta stvori dojam neravnina bez dodatne geometrije. Phongov model osvjetljenja - difuzna i spekularna komponenta ovise o normali što daje iluziju hrapave površine.
Postoje dva načina:

- Koristi se mapa normala
- Koristi se visinska mapa koja govori koji dio površine je viši od drugih, tada shader izrančuna normalu površine prema znanoj visini i izračuna interakciju površine sa svjetlima - procesorski jako zahtjevno



Preslikavanje normala - u teksturu neravnina izravno se pohranjuju normale u pojedinoj točki površine (tekstura normala). U programu za sjenčanje točaka uzorkuje se tekstura normala korištenjem interpoliranih u,v koord u toj točki. Dobivena normala se koristi za proračun osvjetljenja. Normale se zadaju u kooordinatom sustavu tangente, definiran trima vektora: vektor tangente, bitangente i geometrijske normale.

Preslikavanje pomaka - rješava problem kada se površina promatra pri oštrom kutu zbog iluzije promjene geometrije. Koristi podatke iz teksture visina za deformaciju same geometrije predmeta pa generira neravnine bez trikova sjenčanja.
Jednostavna izvedba: svakom vrhu mijenja visinu, geometrija mora biti pripremljena (ne povecava detaljnost geometrije) - moze se primjeniti za npr. more.
“Pravo” preslikavanje pomaka: stvaraju se dodatni trokuti za neravnine, potrebna sklopovska podrška za teselaciju (SM5.0) - detekcija presjeka i sudara postaje problem (vrši se u aplikaciji, prije pomaka).

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 često 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).

11. **Koja je ideja iza UVO (umreženog virtualnog okruženja) i navedi neke primjere. Što je upravljanje prema području interesa (area of interest management)?**

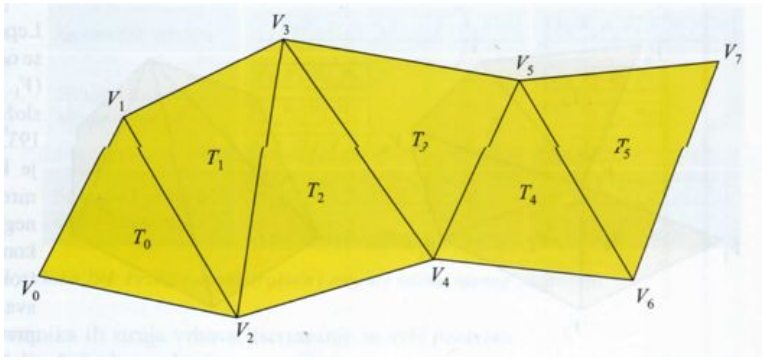
UVO - raspodijeljeni programski sustav koji korisniku omogućuje prisutnost i sudjelovanje u zajedničkom virtualnom okruženju s drugim korisnicima te interakciju sa samim okruženjem i predmetima na njemu. Provodi se sinkronizacija kako bi svaki avatar (grafički prikaz lika - korisnika) imao jednaku okolinu na mreži.
Primjeri : Video igre, simuacije

ovo je gradivo iz osnova virtualnih okruzenja

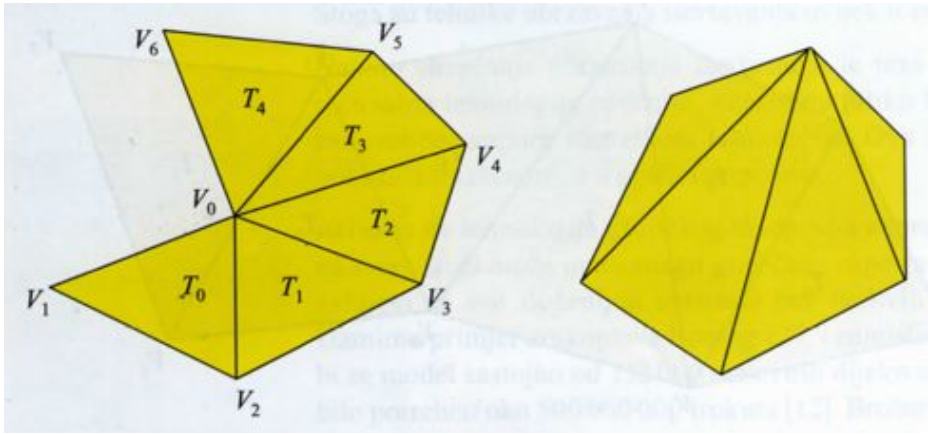
Upravljanje prema području interesa - Porastom broja korisnika u UVO raste i broj poruka koje treba razmjeniti. Postoje određena ograničenja korisnikove percepcije, što se može iskoristiti za filtriranje poruka koje je potrebno poslati. Npr. avataru koji se nalazi u jednoj sobi nije potrebno javljati promjene pozicija avatara u drugoj sobi ako ih ne vidi.

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)

13. Objasniti metodu teksture sjena (shadow mapping).

Metoda teksture sjena je najšire korištena metoda za simulaciju sjena u stvarnom vremenu. Zasniva se na ideji da sve točke scene koje su vidljive iz perspektive izvora svjetla trebaju biti osvijetljene, dok je ostatak scene u sjeni.

Osnovni algoritam: Scena se iscrta iz prespektive izvora svjetlosti u Z-spremnik (iscrtaju se samo dubine). U sljedećem prolazu iscrtavanja scena se crta iz perspektive kamere. Za svaku točku se uzorkuje sadržaj teksture sjena i provjerava se ako je dubina točke veća od dubine u teksturi sjena. Ako je, onda je točka u sjeni i primjenjuje se samo ambijento osvjetljenje. Pri usporedbi dubine obje dubine trebaju biti izražene u istom koordinatnom sustavu, pa se koordinate točke transformiraju u koordinatni sustav svjetla.

Na slici vidimo da je Vb u sjeni, jer je udaljenost Svjetlo-Vb veća od vrijednosti u Z-spremniku.