

Virtualna okruženja

Igor S. Pandžić, Tomislav Pejša

Tehnike ubrzavanja iscrtavanja

Kako iscrtavati brže?

- Mjera brzine iscrtavanja – broj slika u sekundi (engl. frames per second, FPS)
- Za interaktivnu 3D grafiku – min. 20–30 FPS
 - (Za brze akcijske igre poželjno i do 60 FPS)
- Tehnologija grafičkog sklopolvlja napreduje strahovito brzo – no, sklopolje nikad neće biti „dovoljno brzo“
- Npr. model Boeing 777 – 500.000.000 trokuta
 - GeForce GTX580 (2010), 2e9 tri/s: procjena 4fps
 - GeForce GTX1080 (2016), 11e9 tri/s: procjena 22fps

Napredak tehnologije

- Napredak tehnologije nije rješenje – potrebne bolje metode ubrzavanja iscrtavanja
- Obradit ćemo metode:
 - Optimalan zapis poligona
 - Selektivno odbacivanje poligona
 - Tehnike razina detalja
 - Optimizacija protočnog sustava

Optimalan zapis poligona

- Naivan pristup – u protočni sustav šaljemo svaki trokut zasebno (3 vrha po trokutu)
- Mnogi vrhovi su dijeljeni među susjednim trokutima te se obrađuju višekratno – neučinkovito!
- Možemo smanjiti broj vrhova organizacijom trokuta u spojenu strukturu:
 - Trake trokuta
 - Lepeze trokuta
 - Mreže trokuta

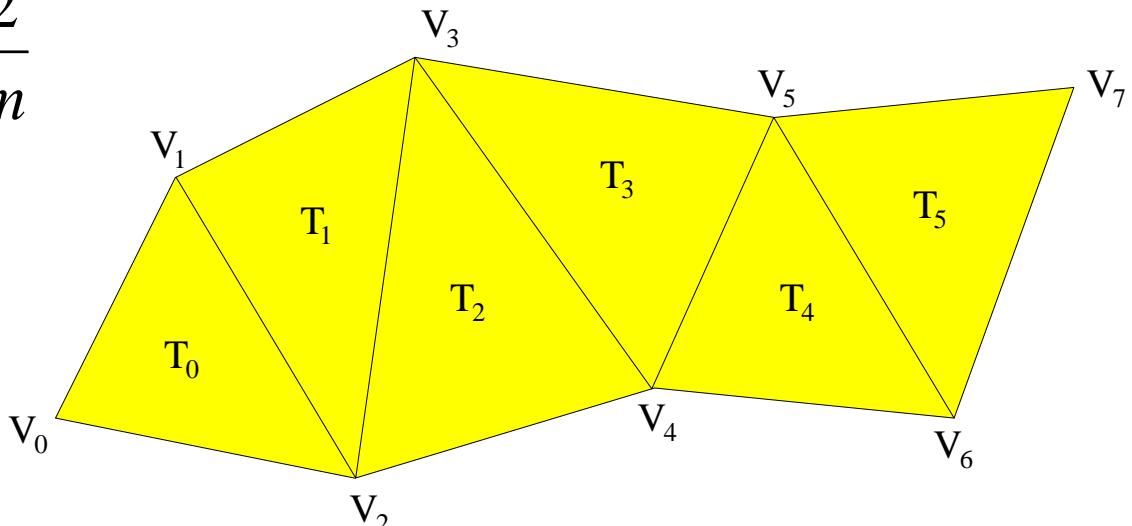
Trake trokuta (triangle strip) (1/2)

- 1. trokut zadan s 3 vrha
- Idući nasljeđuje 2 vrha prethodnog
- $V_0, V_1, V_2, V_3\dots$
- Prosječan broj vrhova po trokutu ($m - \# \text{ trokuta}$):

$$\bar{v} = \frac{3 + (m-1)}{m} = 1 + \frac{2}{m}$$

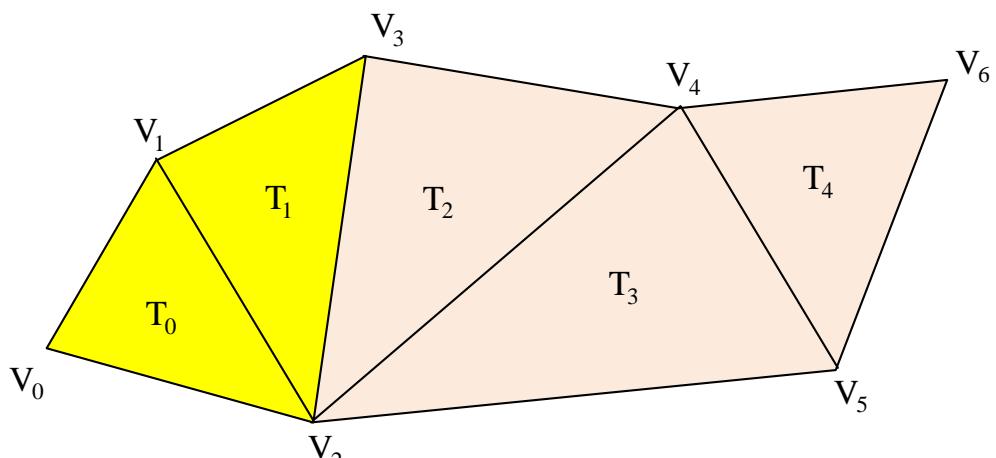
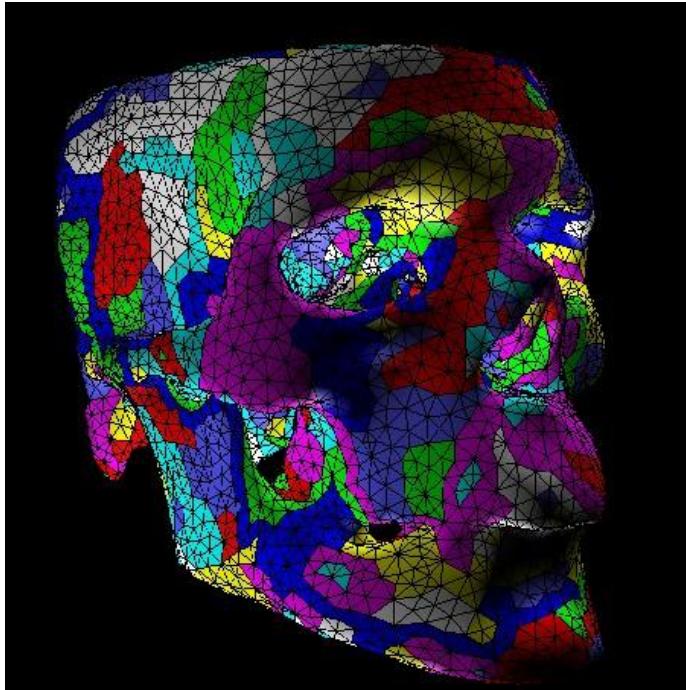
$$m \rightarrow \infty \Rightarrow \bar{v} \rightarrow 1$$

$$m = 10 \Rightarrow \bar{v} = 1.2$$



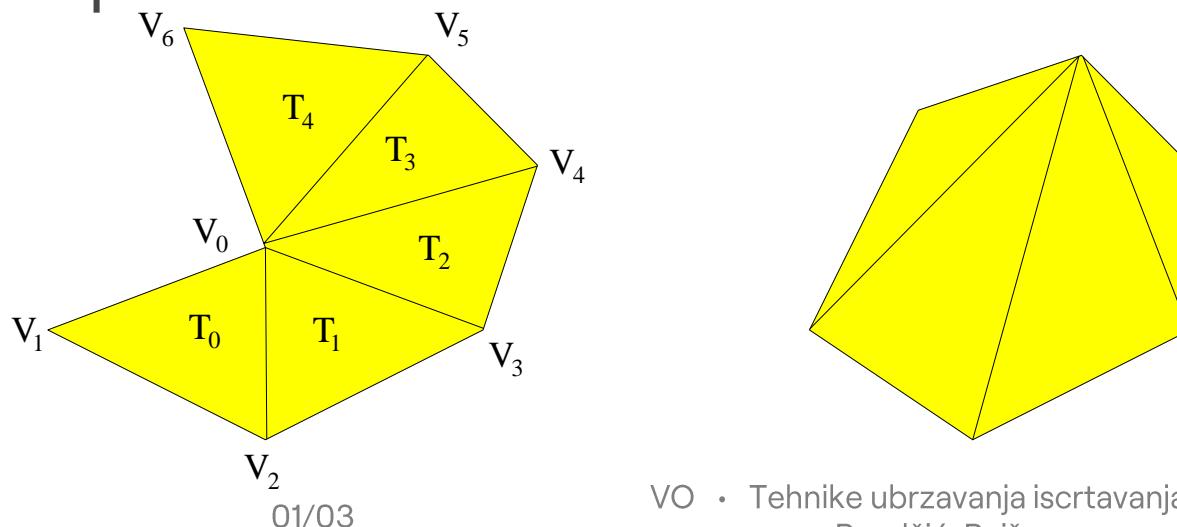
Trake trokuta (2/2)

- Nije učinkovito u g.p.s. slati svaku traku vrhova zasebno (pozivi iscrtavanja su skupi)
- Možemo spojiti više traka u jednu, obrtanjem redoslijeda 2 zadnja vrha
- Primjer:
 - $V_0, V_1, V_2, V_3, V_2, V_4, V_5, V_6$



Lepeze trokuta (triangle fan)

- 1 središnji vrh (V_0) + m vrhova (V_1-V_m) oko njega koji tvore lepezu
- Prosj. broj vrhova po trokutu kao i za traku trokuta, no rjeđe se pojavljuju
- Korisno za pretvorbu n -terokuta u trokute



Mreže trokuta

- Najvažnija struktura za prikaz 3D geometrije
- GPU prilagođen radu s mrežama trokuta
- Skup vrhova + slijed indeksa
- Vrh – koordinate, normala, teksturne koord. itd.
- Slijed indeksa definira trokute
- Pri iscrtavanju predajemo vrhove i indekse u g.p.s.

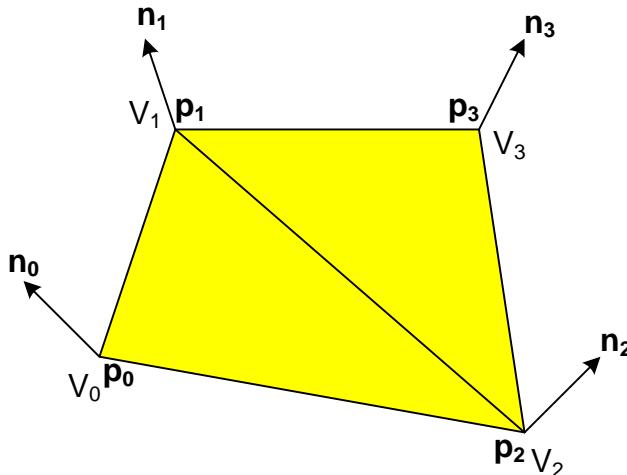
Iscrtavanje mreže trokuta (1/3)

- 2 načina predaje vrhova i indeksa:
 - Spremnik vrhova (vertex buffer)
 - Struje vrhova (vertex streams)
- Spremnik vrhova (+ opc. spremnik indeksa):
 - Slijed podataka na susjednim mem. lokacijama
 - Podaci za 1 vrh pohranjeni zajedno (1 blok)
 - Format vrha definira podatke vrha
- Struje vrhova (+ opc. struja indeksa):
 - Polja podataka – svako polje sadrži podatke određene vrste
 - Npr. struja koordinata, struja normala, struja teksturnih koord. itd.

Iscrtavanje mreže trokuta (2/3)

- Pri iscrtavanju specificiramo g.p.s. koji tip primitiva se iscrtava:
 - Lista točaka
 - Lista linija
 - Lista polilinija
 - Lista trokuta
 - Traka trokuta
 - Lepeza trokuta
- ◆ O tome ovisi kako će g.p.s. interpretirati sadržaj spremnika/struja vrhova
- ◆ Najčešće se koristi lista trokuta u spremniku vrhova i indeksa

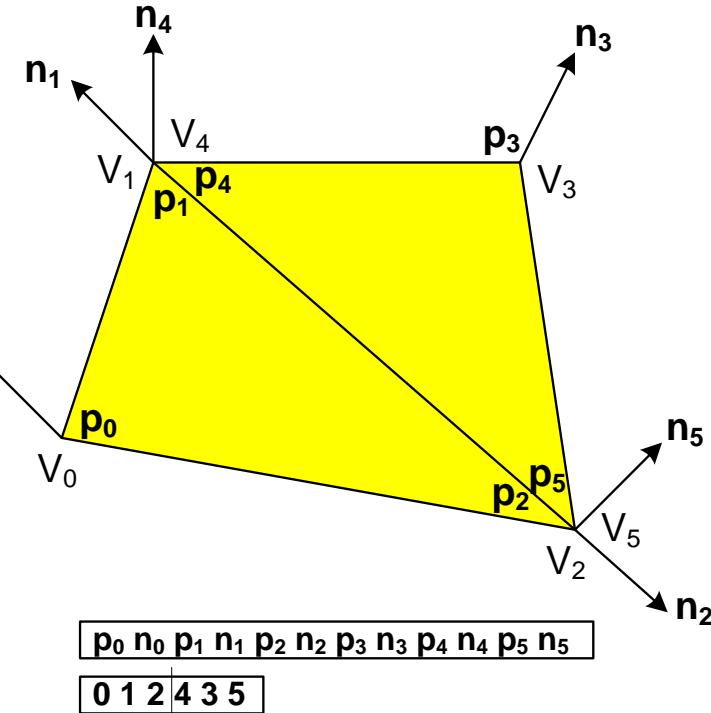
Iscrtavanje mreže trokuta (3/3)



	Lista trokuta	Traka trokuta
Struje vrhova	$p_0 \ p_1 \ p_2 p_1 \ p_3 \ p_2$ $n_0 \ n_1 \ n_2 n_1 \ n_3 \ n_2$	$p_0 \ p_1 \ p_2 \ p_3$ $n_0 \ n_1 \ n_2 \ n_3$
Spremnik vrhova	$p_0 \ n_0 \ p_1 \ n_1 \ p_2 \ n_2 p_1 \ n_1 \ p_3 \ n_3 \ p_2 \ n_2$	$p_0 \ n_0 \ p_1 \ n_1 \ p_2 \ n_2 \ p_3 \ n_3$
Struje vrhova + struja indeksa	$p_0 \ p_1 \ p_2 \ p_3$ $n_0 \ n_1 \ n_2 \ n_3$ $0 \ 1 \ 2 \ 1 \ 3 \ 2$	$p_0 \ p_1 \ p_2 \ p_3$ $n_0 \ n_1 \ n_2 \ n_3$ $0 \ 1 \ 2 \ 3$
Spremnik vrhova + spremnik indeksa	$p_0 \ n_0 \ p_1 \ n_1 \ p_2 \ n_2 \ p_3 \ n_3$ $0 \ 1 \ 2 \ 1 \ 3 \ 2$	$p_0 \ n_0 \ p_1 \ n_1 \ p_2 \ n_2 \ p_3 \ n_3$ $0 \ 1 \ 2 \ 3$

Učinkovitost indeksirane geometrije

- Spremnići vrhova i indeksa orijentirani na učinkovitost iscrtavanja, a ne pohrane
- Npr. ako želimo da isti dijeljeni vrh ima više normala (npr. oštiri bridovi kocke) – potrebno višestruko definirati vrh (po 1 za svaku normalu)

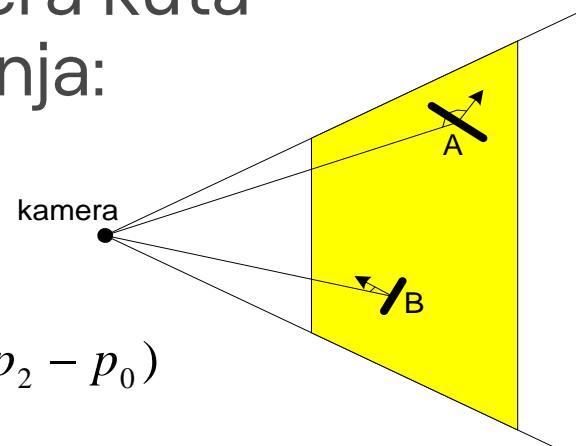


Selektivno odbacivanje poligona (culling)

- Osnovna ideja: poligone koji nisu u nekom trenutku vidljivi na slici ne trebamo iscrtavati
- Najvažnije metode:
 - Odbacivanje stražnjih poligona (backface culling)
 - Odbacivanje po projekcionom volumenu (view-frustum culling)
 - Portalno odbacivanje (portal culling)
 - Odbacivanje prekrivenih poligona (occlusion culling)

Odbacivanje stražnjih poligona

- Standardni dio geometrijske faze g.p.s.
- Poligoni okrenuti od kamere nisu vidljivi
- Okrenutost poligona određena redoslijedom njihovih vrhova (2 konvencije – u smjeru kazaljke na satu ili obrnuto)
- Moguća implementacija – provjera kuta normale poligona i smjera gledanja:



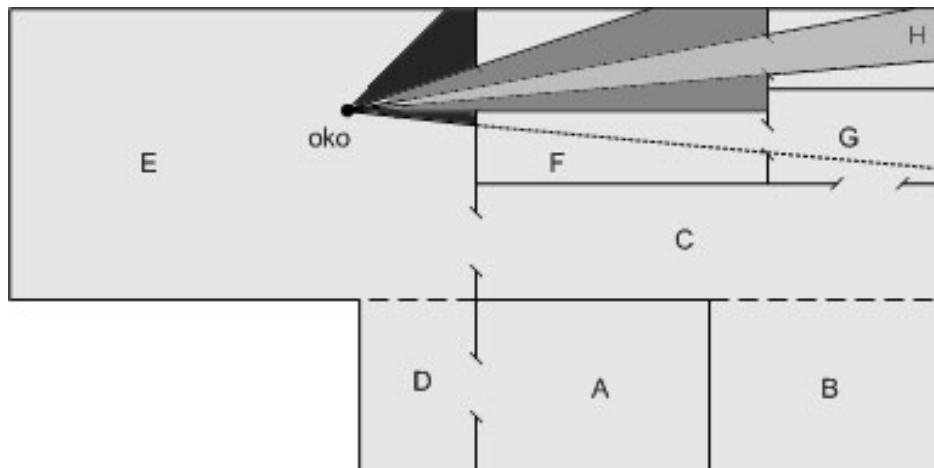
$$n = (p_1 - p_0) \times (p_2 - p_0)$$

Odbacivanje po projekcionom volumenu

- Sve što je izvan projekcionog volumena, nevidljivo je
- Za provjeru vidljivosti koriste se hijerarhije obujmice, BSP i oktalna stabla...
- Npr. hijerarhija obujmica:
 1. Ako je obujmica potpuno izvan projekcionog volumena, odbacuje se sve u njoj
 2. Ako je potpuno unutra, sve se crta
 3. Ako projekcioni volumen siječe obujmicu, provjerava se sljedeća niža hijerarhijska razina obujmica
- Izvodi se u aplikacijskoj fazi

Portalno odbacivanje (1/2)

- Koristi se za scene arhitekture sa sobama
- Scena se dijeli na ćelije (sobe)
- Ćelije imaju portale (vrata) prema drugim ćelijama
- Za svaku ćeliju gradimo graf susjednosti (podaci o portalima i susjednim ćelijama)

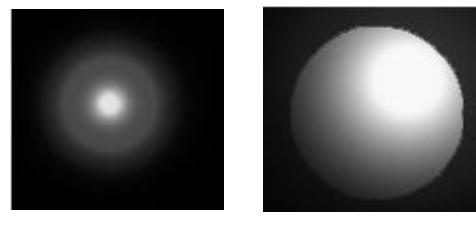
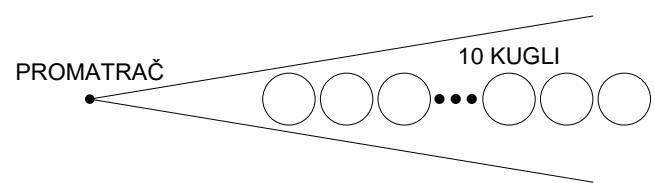


Portalno odbacivanje (2/2)

- IsCRTavanje je rekURzivno:
 1. IsCRTaj trenutnu ćeliju (uz korištenje trenutnog projekcionog volumena)
 2. Na temelju vidljivih portala odredi nove, sužene projekcione volumene
 3. IsCRTaj vidljive susjedne ćelije
- Izvodi se u aplikacijskoj fazi

Odbacivanje putem prekrivenosti

- Predmeti često prekriveni drugim predmetima
- Neće se vidjeti zbog Z-spremnika, no Z-spremnik se primjenjuje tek u fazi rasterizacije
- Dubinska složenost:



DUBINSKA
SLOŽENOST

KONAČNA SЛИKA - VIDI
SE SAMO PRVA
KUGLA

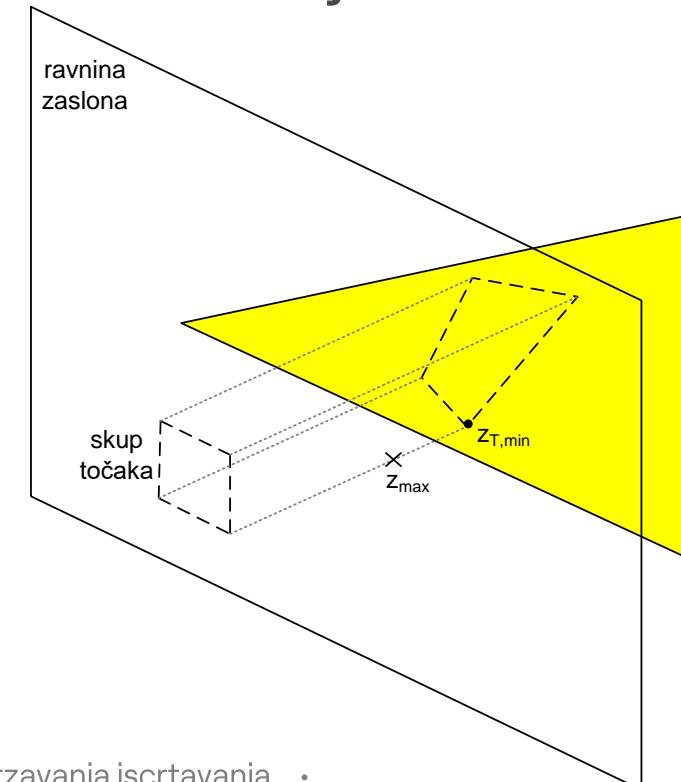
- Ideja: prekrivenu geometriju odbaciti što ranije
- Dosta složene tehnike, no neke su podržane sklopovalni
- Učinak ovisi o redoslijedu iscrtavanja – najbolje od naprijed prema natrag (front-to-back)

Sklopovska provjera prekrivenosti (hardware occlusion queries)

- GPU podržava način iscrtavanja u kojem se provjerava vidljivost skupa poligona (najčešće obujmica) s obzirom na Z-spremnik
- Algoritam:
 1. Iscrtati obujmicu u načinu provjere prekrivenosti (uključen Z-test, isključen Z-write)
 2. Dohvatiti broj vidljivih piksela – ako je veći od nekog praga, iscrtati predmet (uz uključen Z-write)
- Najbolje u redoslijedu front-to-back, uz korištenje hijerarhije obujmica
- Izvodi se u aplikacijskoj fazi (uz djelomičnu sklopovsku podršku)

Z-odbacivanje (Z-cull)

- Ugrađeno u fazi prolaza trokuta
- Rasterizacija se radi u skupovima do 8×8 točaka
- Ispituje je li prekriven isječak trokuta (koji odgovara trenutnom skupu)
- Postupak Z_{MAX} :
 1. Odredi $\sim z_{T,MIN}$
 2. Dohvati z_{MAX} iz Z-spremnika
 3. Ako $z_{T,MIN} > z_{MAX}$, isječak je prekriven



Rani-Z (early-Z)

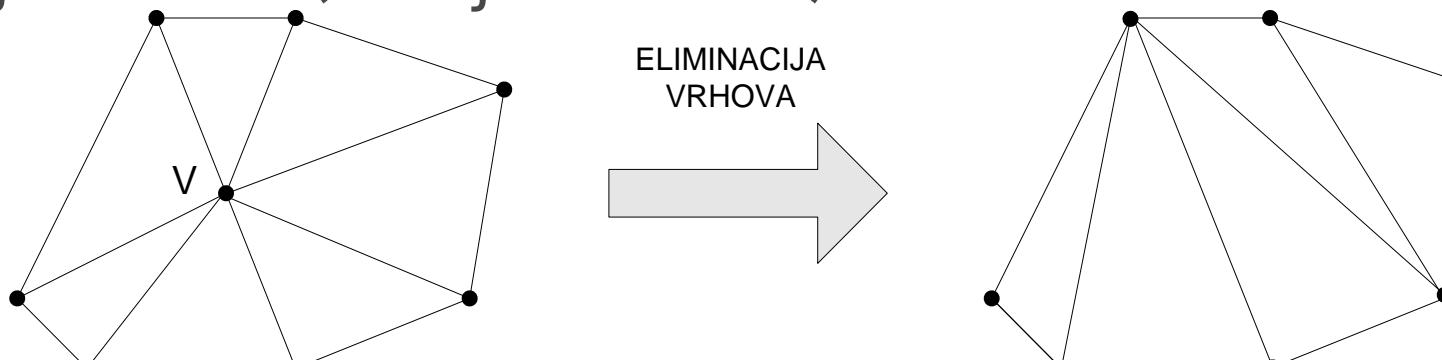
- Nakon generiranja fragmenata, a prije sjenčanja
- Uspoređuje dubinu fragmenta z_F s dubinom u Z-spremniku – ako je veća, odbacuje fragment
- Metode Z-cull i early-Z:
 - Ugrađene u graf. sklopolje i uključene „po defaultu“
 - Smanjuju opterećenje procesora točaka
 - Posebno učinkovite kod višeprolaznog iscrtavanja (jer se Z-spremnik postavlja u 1. prolazu)
 - Automatski se isključuju ako pixel shader mijenja dubinu fragmenta z_F

Tehnike razina detalja

- Engl. Level Of Detail – LOD
- Ideja: smanjiti razinu detalja (broj poligona) kada je predmet udaljen od kamere
 - Razlika se neće primijetiti
- Ako brzina padne, smanji razinu detalja
- Podtehnike:
 - Generiranje – stvaranje jednostavnijih inačica osnovnog modela predmeta
 - Odabir – odluka koja verzija će se iscrtavati
 - Zamjena – zamjena trenutne inačice drugom

Pojednostavljivanje mreže trokuta

- Smanjiti broj trokuta u modelu, nastojeći pritom što manje promijeniti izgled modela
- Osim za LOD, koristi se i za pojednostavljivanje vrlo složenih modela, npr. dobivenih skeniranjem
- Eliminacija vrhova (starija metoda)



Eliminacija bridova (edge collapse)

- Novija i jednostavnija metoda
- Općeniti postupak:
 1. Izračunaj funkciju troška za sve moguće eliminacije bridova; poredaj ih po trošku
 2. Izvedi operaciju s najmanjim troškom
 3. Ponovo izračunaj trošak gdje se mijenja; ponovi 2

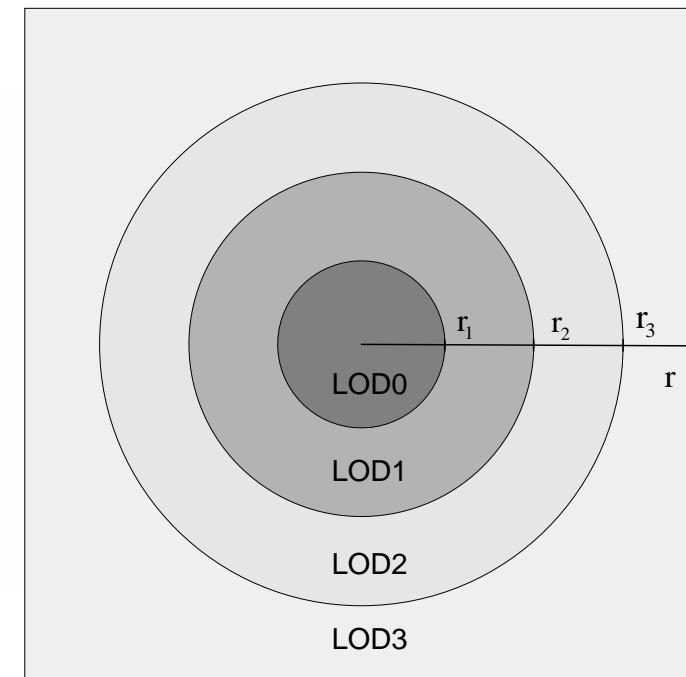


Odabir razina detalja

- Kako odabrati LOD razinu koja će se u nekom trenutku iscrtavati?
- U upotrebi 2 metrike:
 - Udaljenost predmeta od kamere
 - Površina projekcije obujmice

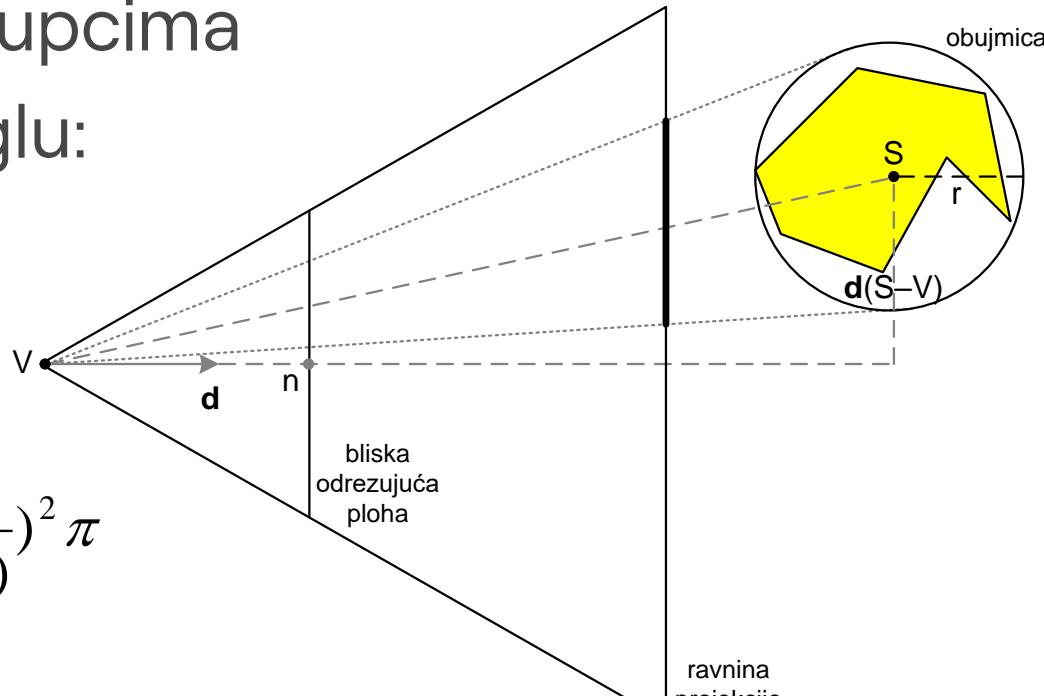
Udaljenost predmeta od kamere

- Najjednostavnija i najčešće korištena metoda
- Svaka razina LOD_i ima raspon udaljenosti od kamere r_i do r_{i+1} na kojoj je aktivna



Površina projekcije obujmice

- Kao i za udaljenost od kamere, definiraju se rasponi površina P_i do P_{i+1}
- Površina projekcije se računa približnim, ali brzim postupcima
- Npr. za kuglu:



$$P = \left(\frac{nr}{d(S-V)}\right)^2 \pi$$

Zamjena razina detalja

- Mora biti što neprimjetnije
- Nagla zamjena modela rezultira efektom skokova (popping)
- Glavne metode:
 - Diskretne razine detalja
 - Miješanje razine detalja
 - α razine detalja
 - Geomorfne razine detalja

Diskrete razine detalja (1/2)

- Koristi se više verzija istog modela, sa sve manjim i manjim brojem poligona
- Kad se ispune uvjeti (npr. udaljenost od kamere), jedna inačica se zamjenjuje drugom



Diskrete razine detalja (2/2)

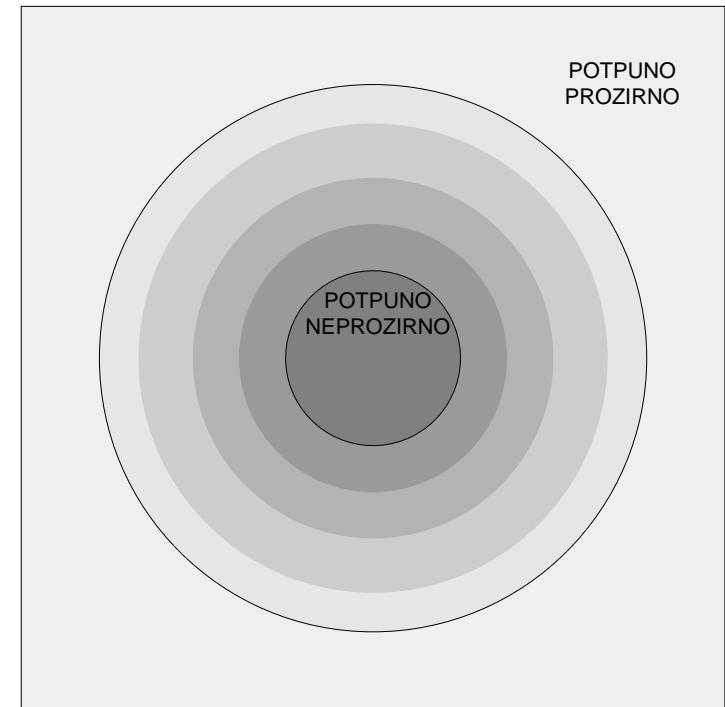
- ◆ Jednostavna tehnika
- ◆ Vrlo dobra sklopovska podrška – različite LOD inačice modela mogu se pohraniti zajedno u video mem.
- ◆ Izražen efekt skokova, pogotovo ako r „oscilira“ oko granice raspona
- ◆ Potonje moguće ublažiti histerezom – granice promjene u jednom smjeru drukčija nego u drugom

Miješanje razina detalja

- Ideja: kratko vrijeme iscrtavamo obje inačice istodobno, uz uključen alpha blending
- Postupak:
 1. Iscrtaj LOD1 bez prozirnost u spremnik boje, uz uključen Z-spremnik
 2. Iscrtaj LOD2 u spremnik boje uz uključenu „over“ funkciju prozirnosti
 - Pritom linearno povećavati α od 0 do 1
 3. Kad je LOD2 iscrtan uz $\alpha = 1$, počni iscrtavati LOD1, uz uključen Z-test, ali isključen Z-write
 - Pritom linearni smanjivati α od 1 do 0
- Nedostatak – kratkotrajan gubitak performansi zbog istodobnog iscrtavanja 2 inačica predmeta

α -razine detalja

- Predmet postaje prozirniji do nestajanja
- Nije potrebno raditi dodatne LOD inačice
- Jednostavna metoda
- Eliminira efekt skokova
- Ubrzanje se postiže tek po nestajanju predmeta



Geomorfne razine detalja

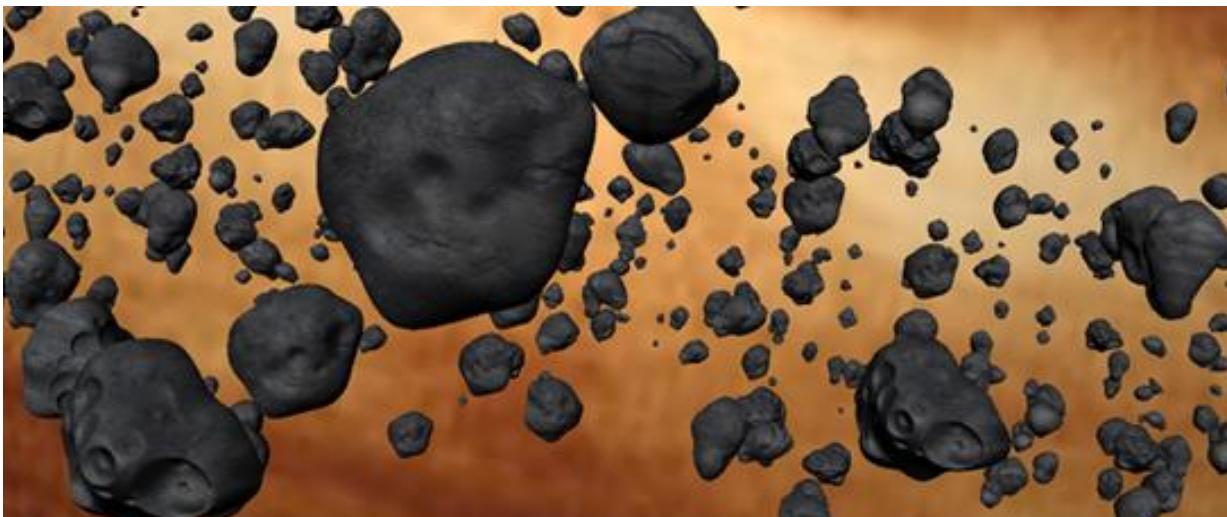
- Koristi se niz modela proizведен eliminacijom bridova
- Podjela vrhova – inverz eliminacije bridova
- Možemo zamisliti da je složenija LOD inačica nastala nizom podjela vrhova u jednostavnijoj inačici
- Ideja: za svaki vrh upamtimo njegov „vrh-roditelj”, te interpoliramo vrhove između susjednih LOD inačica
- Gladak prijelaz između dviju inačica modela
- Predmet neprestano mijenja oblik, što može zasmetati

Srodne tehnike

- Ideja LOD – zamijeniti resurs (model) jednostavnijom inačicom / aproksimacijom ovisno o relativnom položaju kamere
- To se može poopćiti na druge tehnike iscrtavanja
- Npr. zamjena materijala/tekstura niže kvalitetnima:
 - Isključivanje tekture detalja
 - Isključivanje tekture okoline
 - Zamjena shadera jednostavnijom inačicom
 - ...

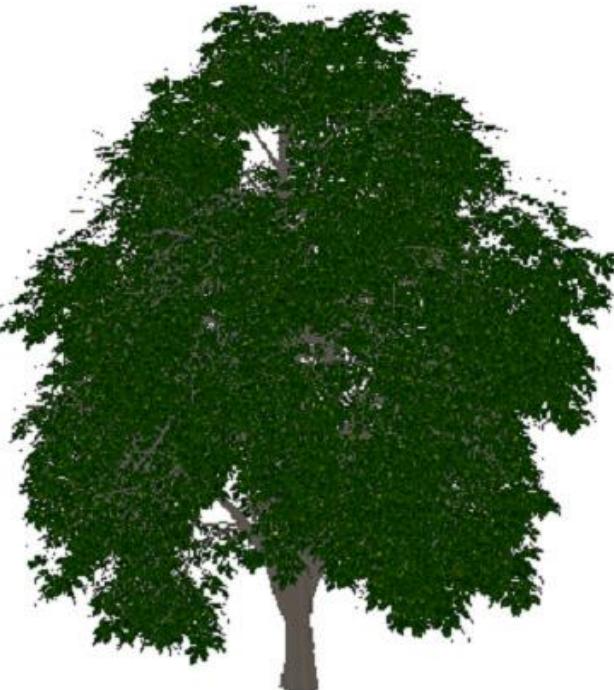
Varalice (impostors)

- Složeni predmet iscrtamo u teksturu
- Predmet zamijenimo panoom (billboard), i na njega nalijepimo teksturu
- Kod većih promjena relativnog položaja kamere potrebno ponoviti postupak

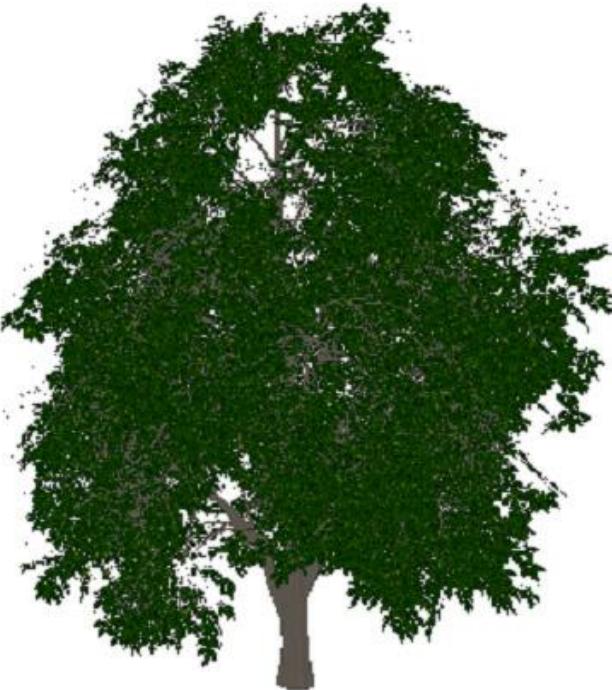


Oblaci panoa (billboard clouds)

- Složen predmet zamjenjujemo skupom panoa
- Svaki pano sadrži sliku nekog dijela predmeta
- Nije potrebno višekratno iscrtavati predmet



01/03



Optimizacija protočnog sustava

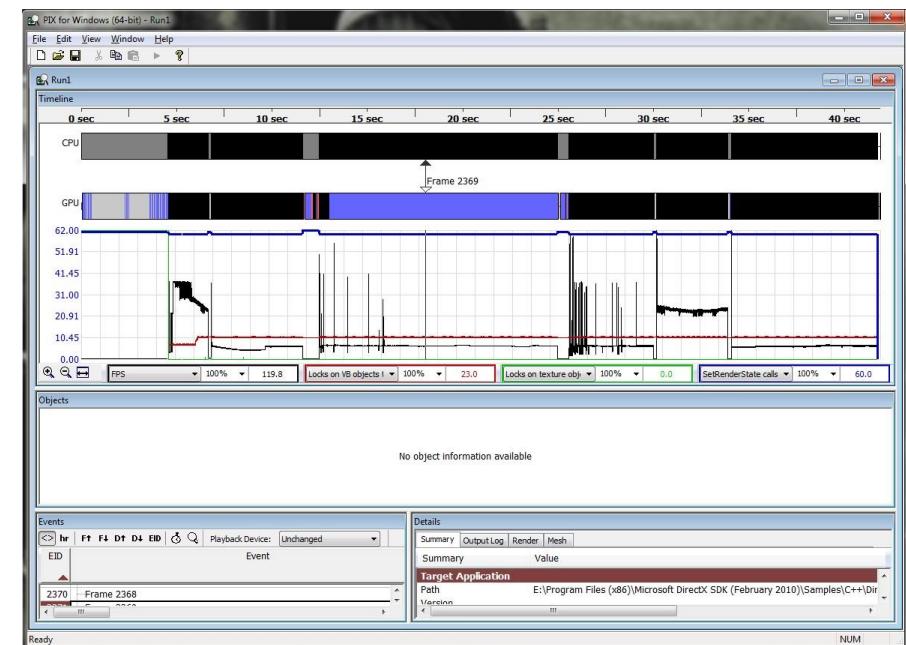
- Najsporija faza stvara usko grlo (kao kod pokretnih traka)
- Postupak
 1. Pronaći usko grlo
 2. Ubrzati tu fazu
 3. Ponoviti postupak
- Ako ne možemo ukloniti usko grlo, u ostalim fazama se može obaviti više posla i povećati kvaliteta

Mjerenje performansi (1/2)

- Brzina iscrtavanja – broj slika u sekundi (frames per second, FPS)
- Brzina geom. faze – broj vrhova u sekundi
- Brzina rast. faze – broj točaka u sekundi
- Te mjere nisu same dovoljne:
 - Usko grlo se često seli već unutar jedne slike
- Složeni testovi (benchmark):
 - Prethodno isključiti dvostruko spremanje

Mjerenje performansi (2/2)

- Alati za profiliranje korištenja CPU/GPU:
 - NVIDIA PerfKit
 - AMD GPU PerfStudio
 - PIX for Windows
 - gDEBugger
- ◆ Omogućuju vremensko praćenje raznih statistika:
 - Broj poziva iscrtavanja, čitanja tekstura, izvođenja shadera
 - Potrošnja memorije
 - Iskorištenost CPU-a
 - ...



Traženje uskog grla (1/3)

- Testirati svaku fazu posebno
- Aplikacijska faza:
 - 100% na CPU
 - Koristiti programe za prikaz tereta procesora:
 - Ako je procesor skoro 100% opterećen, tu smo!
 - ... osim ako glavna petlja nije radno čekanje
 - Koristiti alate za profiliranje kôda:
 - AMD CodeAnalyst, VS Team System Profiler...
 - Eliminirati ostale faze slanjem „praznih“ naredbi
 - Npr. za iscrtavanje koristiti null driver
 - Ako nema ubrzanja, našli smo krivca!

Traženje uskog grla (2/3)

- Geometrijska faza:
 - Najčešće 100% na GPU
 - Glavne operacije – dohvati i sjenčanje vrhova
 - Testiranje dohvata vrhova:
 - Povećati format vrha (npr. dodati „prazne“ teksturne koord.)
 - Ako brzina padne, to je to!
 - Testiranje sjenčanja vrhova:
 - Dodati naredbe u vertex shader
 - Ako brzina padne, to je to!

Traženje uskog grla (3/3)

- Faza rasterizacije:
 - 100% na GPU
 - Glavne operacije – sjenčanje točaka i ROP
 - Testiranje sjenčanja točaka:
 - Dodati naredbe u pixel shader ili smanjiti razlučivost
 - Promjena brzine => tu je usko grlo
 - Testiranje ROP:
 - Smanjiti dubinu boje u spremniku boja (npr. 32-bit => 16-bit)
 - Ako brzina poraste, tu je problem!

Optimizacija aplikacijske faze

- Opća pravila optimizacije koda i pristupa memoriji
 - Npr. izbjegavati dijeljenje, spremati podatke u memoriju redoslijedom korištenja...
 - Koristiti optimizacijske opcije compilera
 - ...
- 2 važne strategije:
 - Koristiti paralelizam
 - Optimizirati promjene stanja

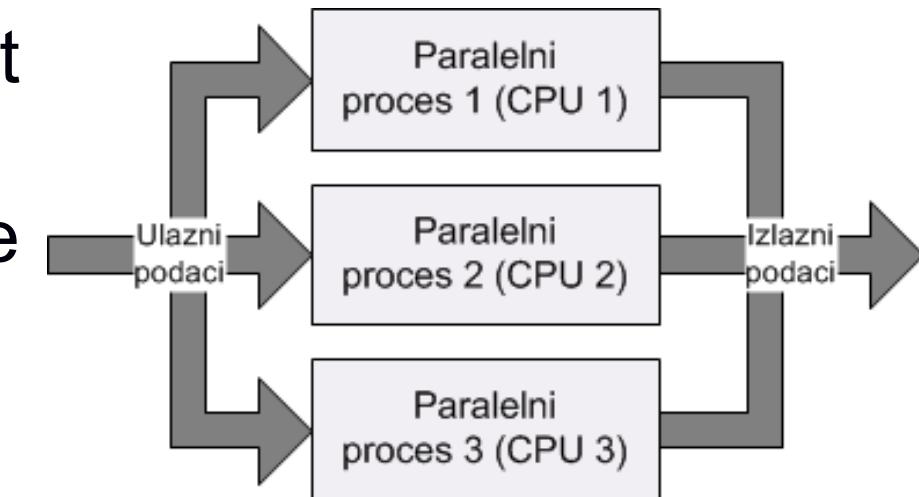
Paralelizam u aplikacijskoj fazi (1/2)

- Grafički algoritmi se često daju paralelizirati
- Višeprocesorska protočnost (vremenski paralelizam):
 - Aplikacijsku fazu podijeliti u protočne podfaze
 - Svaka podfaza se izvodi na zasebnoj CPU jezgri
 - Npr. u sustavu OpenGL Performer:
 - APP – logika aplikacije
 - CULL – odbacivanje poligona
 - DRAW – iscrtavanje
 - Nedostatak – akumulacija kašnjenja zbog protočnosti



Paralelizam u aplikacijskoj fazi (2/2)

- Paralelno izvođenje (prostorni paralelizam):
 - Pojedine algoritme zamijeniti višedretvenim inačicama
 - Nema akumulacije kašnjenja
 - Neke algoritme nije moguće učinkovito paralelizirati
- ◆ Napomena: mogućnost paralelnog pristupa grafičkom API-ju iz više dretvi uvedena tek u DirectX 11!



Optimizacija promjene stanja (1/3)

- Operacije promjene stanja:
 - Zadavanje spremnika vrhova i spremnika indeksa
 - Zadavanje tekstura
 - Učitavanje i konfiguracija shadera
 - Poziv iscrtavanja
- Te operacije su skupe:
 - Izvode se na CPU
 - Zahtijevaju pražnjenje g.p.s. (npr. priručnih memorija) → prazan hod!
- Učestale promjene stanja su danas glavni uzrok loših performansi!

Optimizacija promjene stanja (2/3)

- Rješenje – bolje grupirati predmete u sceni!
- Spajanje predmeta (batching):
 - Više manjih predmeta spojiti u jedan
 - Što ako imaju različite materijale i teksture?
 - Dodati ID predmeta u format vrha, te u shader grananje na temelju ID
 - Time se opterećenje seli u druge faze g.p.s.

Optimizacija promjene stanja (3/3)

- Instanciranje:
 - Ako se isti predmet pojavljuje mnogo puta (npr. drveće)
 - Predmet možemo iscrtati N puta jednim pozivom iscrtavanja
 - Podaci specifični za instancu (npr. transformacije) u posebnom spremniku
- Organizacija scene s obzirom na stanje:
 - Neki predmeti dijele materijale, shadere i/ili teksture
 - Grupirati ih i iscrtavati slijedno!
 - Manje učinkovito (svaki predmet i dalje ima svoj spremnik vrhova)
 - Najpraktičnije realizirati u sklopu grafa scene

Optimizacija geometrijske faze

- Transformacije, osvjetljenje, obrezivanje, projekcija i preslikavanje na ekran
- U načelu ne optimiziramo izravno
- Smanjiti količinu geometrije u g.p.s.
(odbacivanje, LOD)
- Koristiti mreže trokuta s indeksima, trake
trocuta i sl.

Optimizacija faze rasterizacije

- Koristiti odbacivanje stražnjih poligona
- Isključiti Z-spremnik ako nije potreban:
 - Npr. crtanje pozadine
- Koristiti tehnike smanjenja dubinske složenosti:
 - Z-odbacivanje i rani-Z ne isključivati bez razloga
 - Uvesti preliminarni prolaz radi inicijalizacije Z-spremnika
- Miješanje boja koristiti samo kad je potrebno
- Koristiti kompresiju tekstura
- Smanjiti broj svjetala ili pojednostaviti sjenčanje
- Imati više varijanti pixel shadera
- Koristiti jednostavniji anti-aliasing
- Smanjiti razlučivost iscrtavanja