Virtualna okruženja

Igor S. Pandžić, Tomislav Pejša, Mirko Sužnjević

Grafički procesor (GPU)

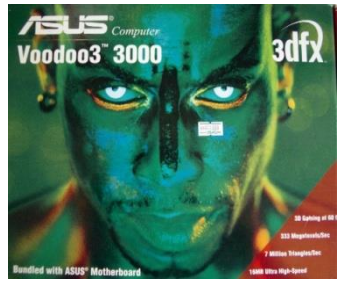
Pregled predavanja

- Uvod
- Grafički protočni sustav
  - Procesor za sjenčanje (engl. shader)
  - Tok podataka kroz protočni sustav
  - Evolucija programabilnog grafičkog sklopovlja
- Programski jezici za sjenčanje
  - Povijest
  - Efekti
  - Primjer programa za sjenčanje
  - Kombiniranje više svjetala i materijala
  - GPGPU

Uvod – Povijesni razvoj grafičkih kartica

- Rani dani grafičkih kartica (1976 – 1995)
  - 1976 – RCA “Pixie” čip
    - Izlazni video signal rezolucije 62x128 piksela
    - Korišten u konzoli RCA Studi II s rezolucijom 64x32
  - 1977 – Television Interface Adapter (TIA) 1A čip – korišten u konzoli Atari 2600
  - 1979 – Intel iSBX 275 Video Graphics Controller Module mogao prikazati rezoluciju 256x256 u boji ili 512x512 monokromatsku (1000 USD)
  - 1981 – Evans & Sutherland - grafički sustav za vojni CT5 simulator leta od 20 milijuna USD
  - 1986 – prva ATI kartica OEM Color Emulation Card
  - 1989 – formiran S3 koji će dugo dominirati tržištem te proizvodi prvi čip S3 911 s 1MB RAM te podrškom za 16 bita boje.
  - 1992 – OpenGL 1.0

Uvod – Povijesni razvoj grafičkih kartica



- 1995 – 1999 Dominacija 3Dfx Voodooo kartica
  - 1996 – 3Dfx izdaje Voodoo grafičku karticu fokusiranu na 3D
    - Kompletna revolucija na tržištu, nestala dominacija S3-a koji je kontrolirao 50% tržišta tada te na vrhuncu popularnosti 3Dfx drži 80 – 85% 3D tržišta
    - Veliki broj kartica postao zastario preko noći
  - 1996 – VideoLogic razvija tehnologiju u kojoj se miče potreba za velikim Z-spremnikom kroz odbacivanje sve geometrije koja se ne vidi prije izračunavanja tekstura, sjena i osvjetljenja
  - 1997 – ATI izdaje Rage II karticu koja se može mjeriti sa Voodoo karticama, a Nividia RIVA 128 (Real-time Interactive Video and Animation accelerator)
  - 1998 – 3Dfx izdaje Vooodoo 2 karticu koja ponovno ima najbolje 3D performanse, ali tržišna prednost se počinje topiti i ATI dostiže 27% tržišta, a Nvidia izdaje visoko popularnu karticu TNT s 32 -bitnom bojom u odnosu na 16-bitnu od Voodoo 2 kartice, a S3 se vraća sa Savage3D karticom.
  - 1999 – izlaze Savage4 od S3, Riva TNT2 od Nvidie, Voodoo 3 od 3Dfx-a te G400 od Matroxa,
  - 1999 – izlazi i Nvidia GeForce 256
    - Prvi potrošački grafički sklop s geometrijskom fazom
    - Tvrtka ga naziva “grafički procesor” (engl. Graphics Processing Unit, GPU), naziv ostaje

Uvod – Povijesni razvoj grafičkih kartica

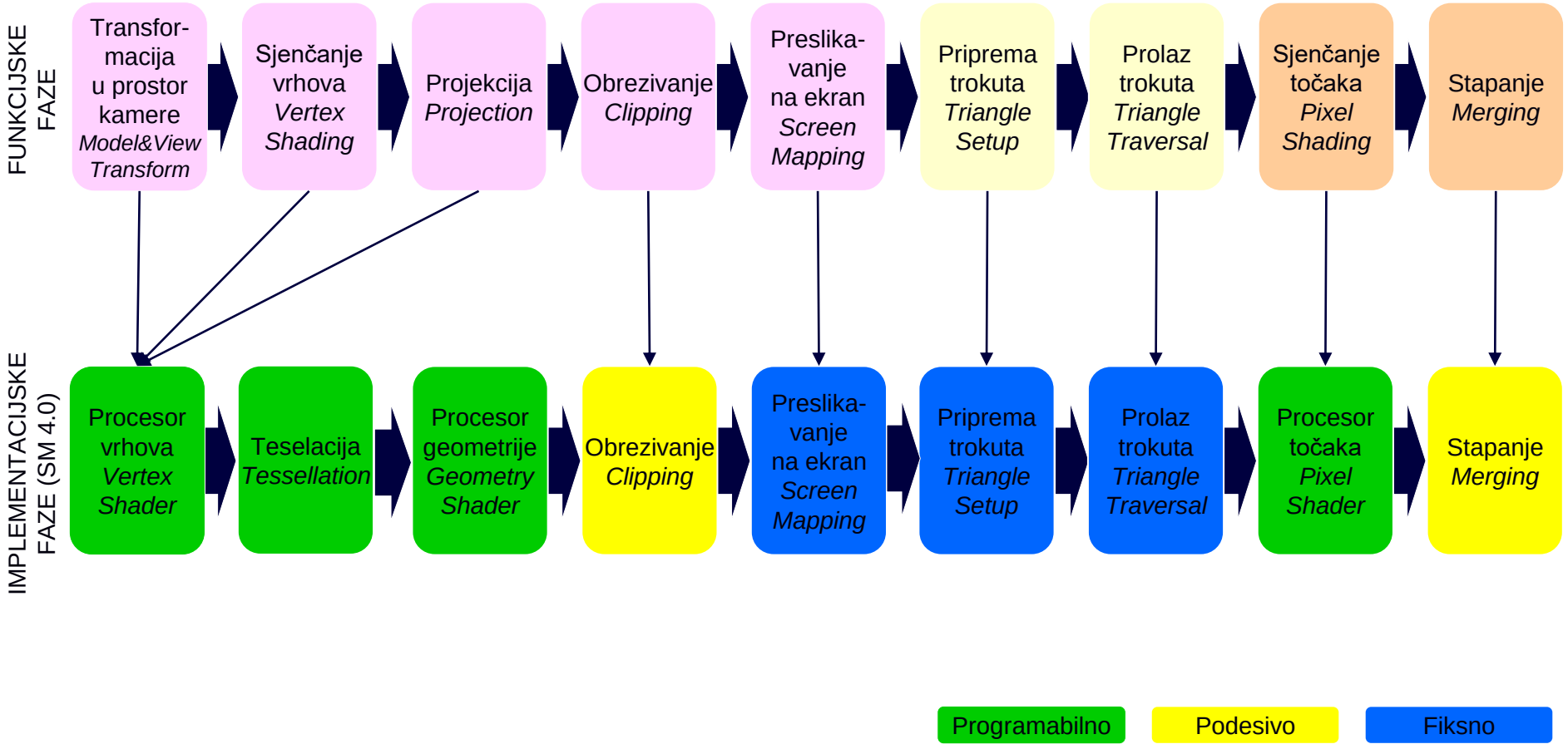
- 2000 – Kreiranje duopola na tržištu ATI i Nvidia
  - ATI kupuje ArtX Inc, za oko 400 milijuna USD
  - VIA kupuje S3 za oko 165 milijuna USD
  - Nvidia kupuje 3Dfx za oko 70 milijuna USD
  - ATI objavljuje svoju prvu Radeon karticu
- 2001 – udio drugih proizvođača (Matrox i S3/VIA) na tržištu grafičkih ka[illegible]
- 2006 – “pobjeda” Nvidie
  - AMD kupuje ATI za 5.4 milijarde USD
  - Nvidia uvodi generičku arhitekturu procesora za sjenčanje (shader model)
- 2007 – počinje era primjene GPU-ova za “generalne” poslove (GPGPU)
  - Izlazi CUDA SDK od Nvidie koji omogućuje iskorištavanje arhitekture grafičkog procesora za generalne visoko paralelizirane poslove
  - Nvidia izdaje Tesla liniju GPGPU

Uvod – Povijesni razvoj grafičkih kartica

- NVIDIA GeForce256, 1999. g.
  - Prvi potrošački grafički sklop s geometrijskom fazom
  - Tvrtka ga naziva “grafički procesor” (engl. Graphics Processing Unit, GPU), naziv ostaje
- Nakon toga, idući veliki trend: programabilnost
  - Dovodi do današnjih općenitih, potpuno programabilnih procesorskih elemenata (engl. shader)

Geometrijska faza | Rasterizacija | Procesiranje piksela

Protočni sustav na grafičkom procesoru



Jedinstvena procesorska jezgra (*Common Shader Core*)

- Procesori (shaderi) se programiraju u jezicima za sjenčanje sličnima C-u
  - HLSL, Cg, GLSL
- Programi se prevode u asemblerski jezik neovisan o hardveru, što tvori *virtualni stroj*
- Jedinstven model za sve procesore (shadere)
  - Od inačice Shader Model 4.0
  - Ranije su procesori vrhova i točaka imali različite programske modele, a procesor geometrije nije ni postojao

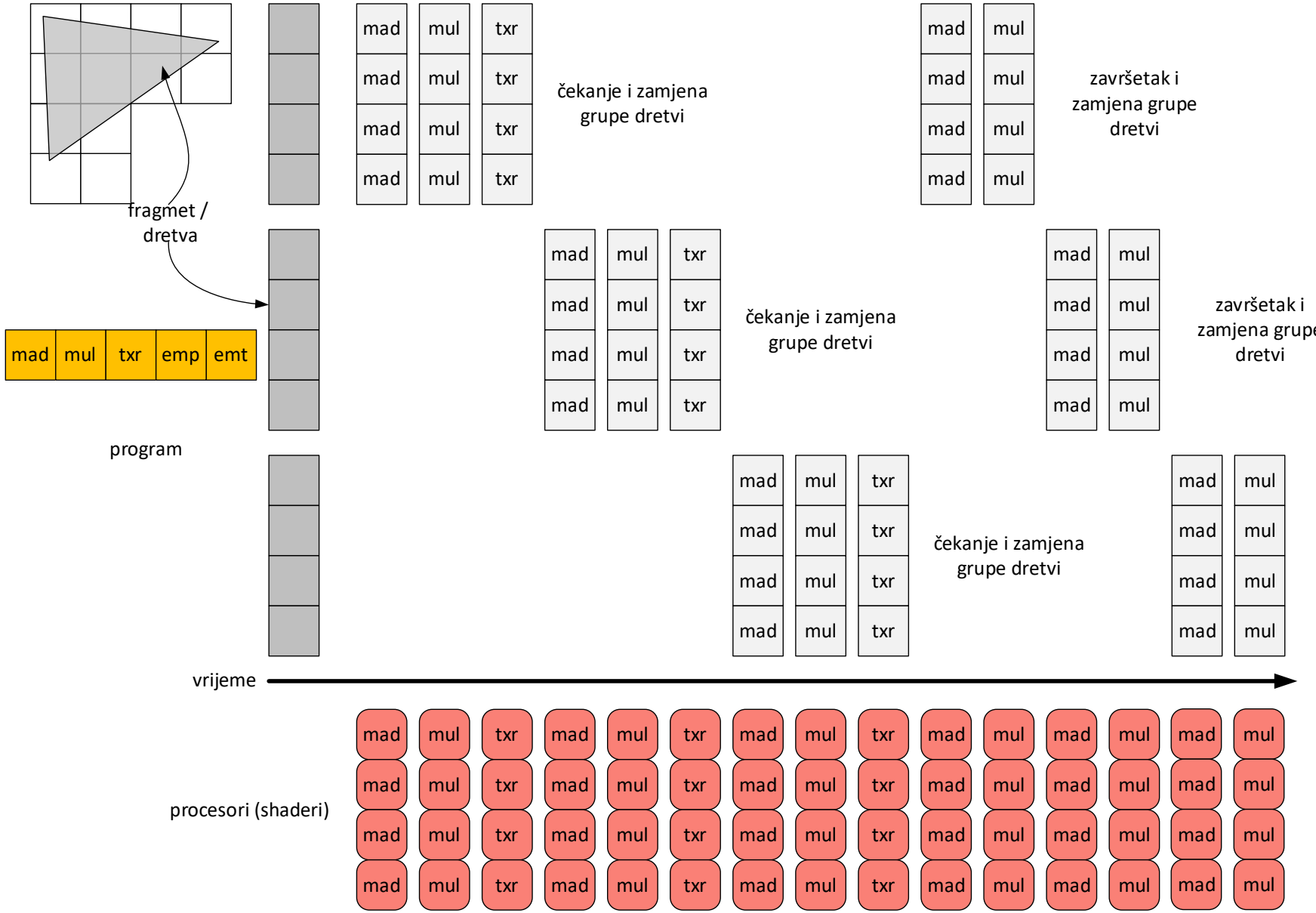
Paralelno procesiranje

- GPU razdvaja logiku izvođenja instrukcija i podatke
- Single Instruction Multiple Data (SIMD) arhitektura
- Primjer
  - Imamo 2000 piksela za koje treba procesirati fragmente te jednu procesorsku jezgru
  - Obrađujemo prvi kroz procesor (shader)
  - Obrada je brza kroz nekoliko aritmetičkih operacija na vrijednostima u lokalnim registrima
  - Nakon toga potrebno je dohvatiti teksturu zapisanu u vanjskoj memoriji što je jako zahtjevno vremenski (stotine ciklusa procesora)
  - Kako bi to bilo brže svakom fragmentu dodjeljuje se malo prostora u lokalnim registrima te umjesto da čekamo na teksturu procesoru se daje na obradu drugi fragment (od dvije tisuće)
  - Prebacivanje između obrade fragmenata je brzo jer ih ne povezuju podaci (ništa iz prvog ne utječe na drugi)
  - Prebacuje se od fragmenta do fragmenta sve dok svi nisu obrađeni, a do tada je i tekstura dostavljena iz memorije
  - Na ovaj način obrada jednog fragmenta je dulja, ali ukupna obrada svih fragmenata je puno brža – kašnjenje sakriva se prebacivanjem na drugi fragment

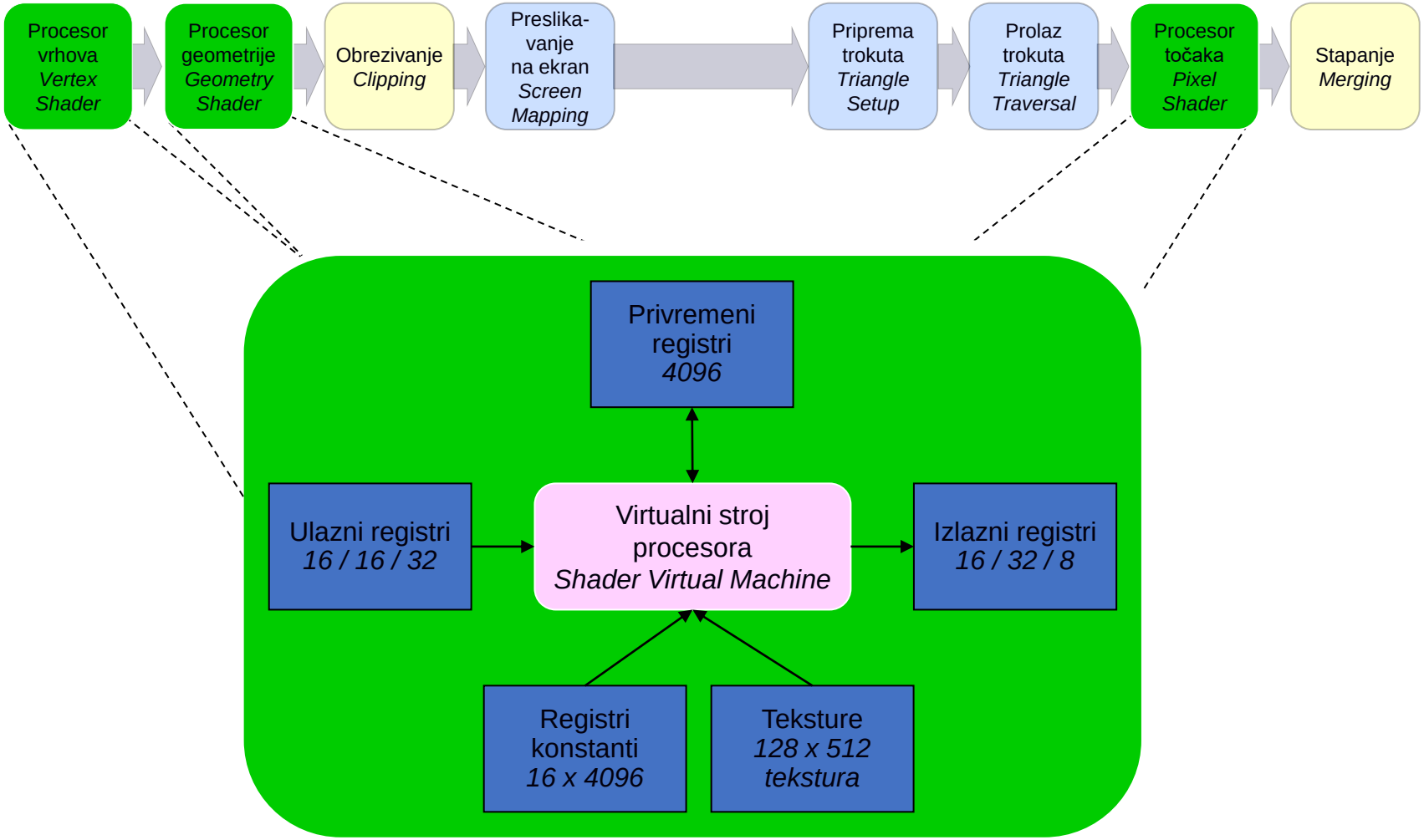
Paralelno procesiranje

- Obrada jednog fragmenta se naziva dretva (thread) i sastoji se od dijela memorije za ulazne podatke za procesor, te memorijskog prostora koji je potreban da procesor završi izračun
- Grupe dretvi su grupirane u valove (engl. wavefronts) u AMD terminologiji odnosno grupe (engl. Warps) u NVIDIA terminologiji
- Koristeći SIMD procesiranje svaka grupa dretvi je raspoređena na određeni broj jezgri procesora (od 8 do 64)
- Svaka dretva je mapirana na SIMD traku
- Primjer
  - Na NVIDIA GPU grupa dretvi može imati 32 dretve
  - Na našem primjeru od 2000 fragmenata imamo 2000/32 =62,5 63 grupe dretvi
  - Zatim se počinje izvoditi prva grupa dretvi na svih 32 procesora
  - Kada dođe do zahtjeva za teksturom isti se događa na svih 32 procesora u isto vrije
  - Cijela grupa dretvi se mijenja drugom dok se ne dostavi tekstura za prvu grupu dretvi

Primjer


fragmet / dretva
mad mul txr emp emt
program
čekanje i zamjena grupe dretvi
završetak i zamjena grupe dretvi
čekanje i zamjena grupe dretvi
završetak i zamjena grupe dretvi
čekanje i zamjena grupe dretvi
vrijeme
procesori (shaderi)

Jedinstvena procesorska jezgra SM4.0



Tipovi podataka i operacije

- 4x SIMD arhitektura
- Osnovni tip podataka – 4x32-bit cjelobrojni ili *floating point* vektori (novije kartice podržavaju i 64-bita)
  - 2D, 3D i 4D vektori su vrlo česti u grafici – koordinate, normale, boje, kvaternioni, reci transf. matrica
  - Cijeli brojevi – indeksi, brojači, bitovne maske
- Operacije
  - Množenje i zbrajanje skalara i vektora – 1 takt
  - sqrt, pow, log, sin, cos... – do 4 takta

Memorija (1/2)

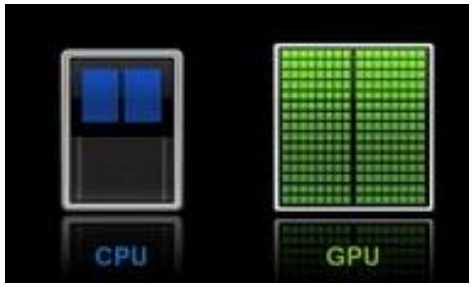
- Ulazni i izlazni registri
  - Varijabilni parametri – specifični za pojedini primitiv (vrh, trokut ili fragment)
  - Ulaz – podaci za obradu (npr. koordinate, normala, teksturne koordinate)
  - Izlaz – rezultati obrade (npr. koordinate, teksturne koordinate, boja)
- Registri konstanti
  - Konstante ili uniformni parametri – podaci konstantni u jednom pozivu iscrtavanja
  - Npr. transformacijske matrice, svjetla

Memorija (2/2)

- Privremeni registri
  - = registri opće namjene
  - Međurezultati operacija
- Teksture
  - Memorijski spremnici
    - Nema pristupanja proizvoljnim memorijskim lokacijama
  - Najsporiji oblik
    - Registri su fizički dio GPU
    - Teksture se u memoriji na graf. kartici ili memoriji sustava
  - Dohvat podataka (*uzorkovanje*)
    - Kontinuirane teksturne koordinate (0-1)
    - Odgovaraju jednoj ili više memorijskih lokacije
    - Ako ih je više, sadržaj se interpolira (filtriranje tekstura)

Kontrola toka

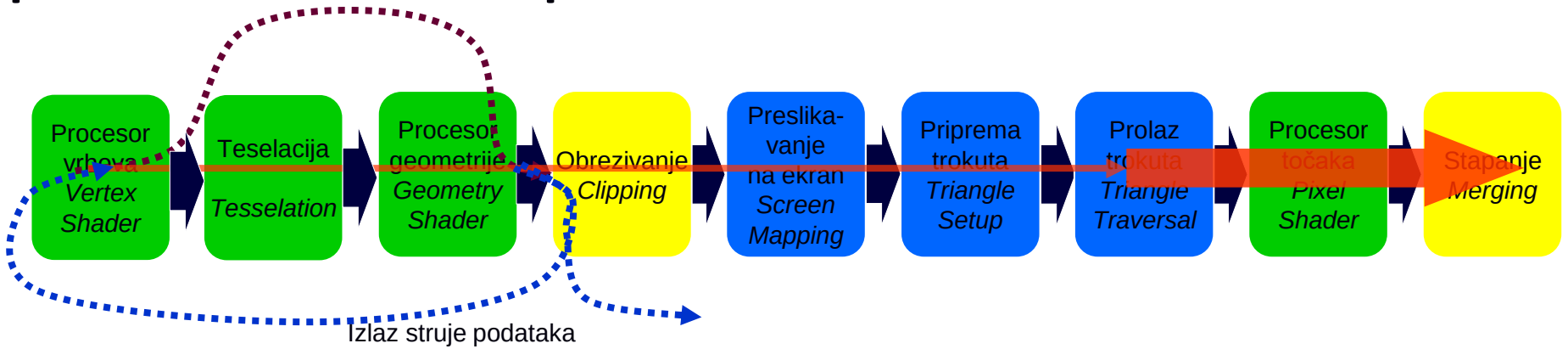
- Kao i na CPU – if-else, for, while
- Zapravo je stvar složenija
- Statička kontrola toka
  - Samo *konstante* u uvjetu – tijek izvođenja isti u jednom prolazu (dobre performanse)
- Dinamička kontrola toka
  - Varirajući podaci u uvjetu – tijek izvođenja može biti različit za različite primitive
  - Slabije performanse zbog *granularnosti*
  - GPU obrađuje po nekoliko desetaka ili stotina primitiva u paraleli – aktivne grane izvođenja moraju se izvesti za *sve* primitive



Model izvođenja

- U pripremnom koraku:
  - Učitavanje programa za sjenčanje (preko grafičkog API-ja)
  - Postavljanje vrijednosti konstanti, alokacija memorijskih resursa (tekstura, spremnika vrhova...)
- U petlji iscrtavanja:
  - Poziv funkcije za iscrtavanje (engl. draw call)
  - 1 poziv iscrtavanja = 1 prolaz geometrije kroz protočni sustav
  - Pritom će procesori za sjenčanje izvesti učitane programe na svim primitivima
  - Rezultat se zapisuje u 1 ili više memorijskih spremnika (npr. spremnik boje, teksturu...)

Tok podataka kroz protočni sustav



- Umnožavanje podataka interpolacijom pri prolazu trokuta
- Procesor geometrije te teselacija se ne moraju koristiti
- Izlaz struje podataka (engl. Stream Output)
  - Povrat u sustav za iterativnu obradu
  - Izlaz iz sustava, praktično za opće primjene grafičkog procesora

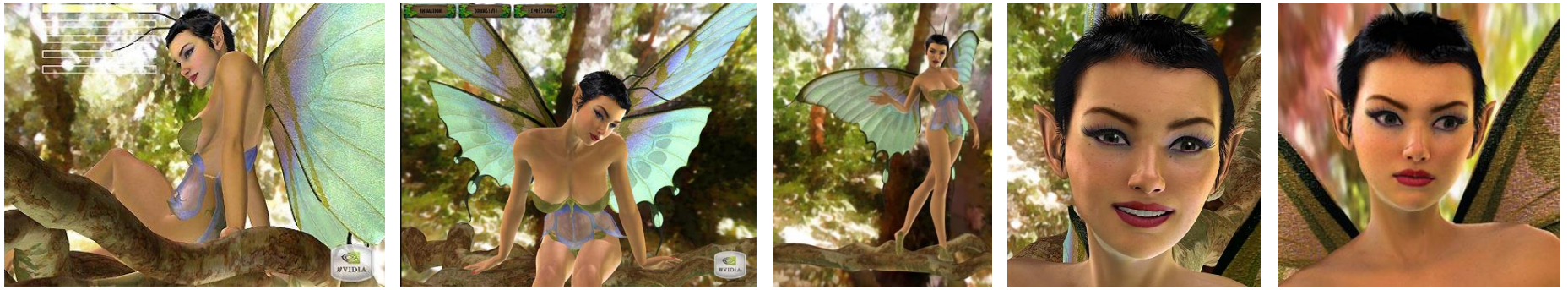
Programabilno | Podesivo | Fiksno

Procesor vrhova (1/2)

- Ulaz: vrh sa pripadajućim podacima
  - Minimalno: koordinate
  - Dodatno: normala, teksturne koordinate, boja...
  - Rad samo na pojedinom vrhu (nema pristupa drugim vrhovima)
- Operacije na ulaznim podacima
  - Minimalno: transf. u prostor kamere, projekcija
  - Sve ostalo ovisno o željenom efektu (npr. proračun osvjetljenja za Gouraudovo sjenčanje)
  - Ne može brisati niti dodavati vrhove
- Izlaz
  - Minimalno: položaj nakon projekcije
  - Izlazni registri se interpoliraju u prolazu trokuta, interpolirane vrijednosti za svaku točku ulaze u procesor točaka

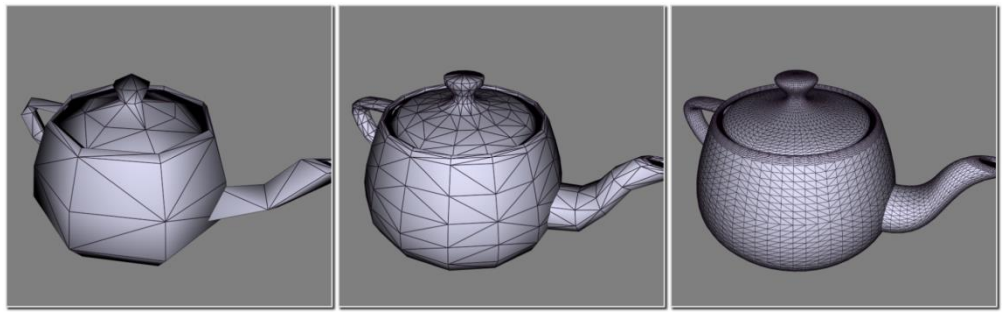
Procesor vrhova (2/2)

- Ukupni efekt je rezultat suradnje procesora vrhova i točaka (i geometrije)
- Neki primjeri efekata
  - Animacija kože pomoću kostura (engl. skinning)
  - Interpolacija oblika (engl. morphing)
  - Efekti objektiva kao npr. riblje oko
  - Proceduralne animacije (npr. zastave, valovi...)



Procesor teselacije

- Teselacija nam omogućava iscrtavanje zakrivljenih površina
- Latinski *tessella* – mali kamen ili pločica u mozaiku
- Predstavlja uzorak koji se može proširiti u svakom smjeru
- Prednosti
  - Sama teselacija štedi na resursima pošto su parametri teselacije manji nego dostava svih trokuta vezanih za zakrivljene površine
  - Dodatno može sprječavati zagušenje sabirnice između CPU-a i GPU-a za likove čiji se oblik mijenja u svakom okviru
- Uvijek se sastoji od tri koraka
  - DirectX terminologija: hull shader, tessellator i domain shader
  - OpenGL terminologija: tesselation control shader, tesselator i tesselation evolution shader

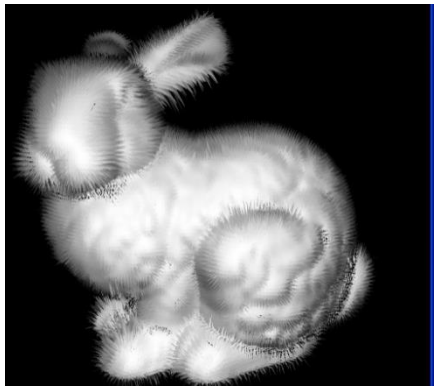


Procesor teselacije

- Hull shader
  - Ulaz je poseban set kontrolnih točaka koji definiraju zakrivljenu površinu (primjerice Bezier površina)
  - Može modificirati ulazne kontrolne točke, dodavati ih ili uklanjati
  - Teselatoru šalje tip površine (trokut, četverokut ili isocrta – poseban oblik traka koji se koriste za renderiranje kose) te faktore teselacije
  - Unutarnji faktori teselaciji određuju koliko teselacije se dešava unutar jedne jedinice (trokuta)
  - Vanjski faktori teselacije govore na koliko se dijeli svaki vanjski brid
  - Domain shaderu se šalju transformirani set kontrolnih točaka te kontrolne podatke
- Teselator generira mesh s te ga šalje domain shaderu
- Domain shader svaki vrh iz teselatora prilagođavaja na temelju kontrolnih točaka te generira izlazne vrijednosti vrhova

Procesor geometrije

- Uporaba je izborna
- Ulaz
  - Objekt (trokut, crta, točka) s pripadajućim vrhovima
    (+ dodatno susjedni vrhovi sa shader modelom 5 čak do 32 dodatne kontrolne točke)
- Izlaz
  - Nula ili više objekata: procesor može brisati, dodavati i mijenjati primitive (skupa operacija, mogućnost učinkovite teselacije dodana u Shader Modelu 5.0)
- Primjeri korištenja
  - Generiranje čestica raznih veličina i oblika u sustavima čestica
  - Iscrtavanje siluete u efektu krzna
  - Generiranje proceduralnih objekata (npr. *metaballs*)
  - Generiranje fraktalne geometrije



Izlaz struje podataka

- Engl. stream output
- Izlaz procesora vrhova/geometrije može se zapisati izravno u memorijski spremnik
- Već u idućem prolazu se može koristiti kao spremnik vrhova – iterativna obrada podataka!
- Faza rasterizacije se može potpuno isključiti (bolje performanse)
- Korisno za GPGPU

Procesor točaka (fragmenata) (1/2)

- Ulaz: fragment
  - Fragment: točka trokuta sa podacima za sjenčanje (**koordinate, dubina**, boja, teksturne koord., normala...)
  - Podaci fragmenta – određeni interpolacijom u prolazu trokuta
  - Procesor točaka sjenčanjem računa boju fragmenta
  - Ako fragment preživi određivanje vidljivosti (Z-buffer), boja se zapisuje u spremnik boje (postaje bojom piksela)
- Operacije
  - Izračun modela osvjetljenja, teksturiranje, efekt magle, razni specijalni efekti...

Procesor točaka (fragmenata) (2/2)

- Izlaz: boja
  - Može i izbrisati fragment ili mu promijeniti dubinu (z)
  - Višestruki spremnici (engl. Multiple Render Targets, MRT)
    - Više boja na izlaz, zapisuju se u različite spremnike
    - Iscrtavanje više slika u jednom prolazu, stapanje u jednu završnu sliku
- Nema pristupa susjednim točkama, ali ima diferencijalima podataka (ddx, ddy)
  - Zato što se točke obrađuju u blokovima 2x2 – 8x8
  - Korisno za mipmapping ili anti-aliasing
  - Nedostupni unutar grane dinamičkog grananja

Stapanje

- Zadatak: odrediti konačnu boju točke (piksela) koja će se vidjeti na ekranu
  - Ulaz: boja više fragmenata koji se preslikavaju u točku, Z-spremnik, spremnik maske...
- Određivanje vidljivosti metodom Z-spremnika
- Operacije sa spremnikom maske
- Nije programabilna, ali velika fleksibilnost u postavkama (rasterske operacije, ROP)
  - Množenje, zbrajanje, oduzimanje, min/max, bitovne operacije na vrijednostima boja i prozirnosti
  - Najčešće za miješanje boje
  - Neovisno miješanje višestrukih spremnika (MRT)

Procesor računanja (engl. compute shader)

- Koristi se za općenite izračune – GPU računarstvo ili GPGPU
- Procesor računanja je uveden u DirectX 11
- Platforme za GPGPU
  - CUDA
  - OpelCL
  - …
- Nema predefiniranu poziciju u protočnom sustavu GPU
- Temelji se na istoj standardiziranoj jedinstvenoj procesorskoj jezgri
- Jedna od prednosti procesora računanja je pristup GPU memoriji što je puno brže od komunikacije između GPU i CPU
- Koriste se i za čestične sustave, procesiranje mesheva kao što je animacija lica, sjene, itd…

Evolucija programabilnog grafičkog sklopovlja

- Povijest ideje programibilnosti u procesu iscrtavanja
  - Shade trees 1984.
  - RenderMan Shading Language, kasne 80te, koristi se i danas
- Shader Model 1.1, 2001. g., DX8.0 i OpenGL, NVIDIA GeForce 3
  - Procesor vrhova, vrlo ograničeni procesor točaka
  - Programiranje u asembleru
- SM 2.x, 2002. g., DX9.0 i OpenGL
  - Potpuno programibilni procesor točaka
  - Povećanje svih ograničenja resursa (broj instrukcija, tekstura, registara...)
  - Jezici više razine (HLSL, GLSL, Cg)
  - Statička kontrola toka
- SM 3.0, 2004 g., DX9.0c i OpenGL (2005. g. Xbox 360, 2006. g. Playstation 3)
  - Dinamička kontrola toka
  - Daljnja povećanja resursa
- SM 4.0, 2007 g., DX10.0 i OpenGL
  - Jedinstvena procesorska jezgra
  - Izlaz struje podataka
  - Procesor geometrije
  - Asembler više nije podržan osim za debugging
  - Daljnja povećanja resursa
- SM 5.0, 2009 g., DX11.0 i OpenGL
  - Teselacija
  - Elementi OOP u HLSL-u
- SM 6.0, 2016 g., DX12.0
  - Razdvojeni kompajliranje i optimizacija
  - Proceduralne teksture, mijenjanje veličine piksela
- Trenutno SM 6.6, video o najnovijim mogućnostima kartica koje implementiraju dani model https://www.youtube.com/watch?v=5rYBLjUmGkA

Programski jezici za sjenčanje (1/2)

- Program za sjenčanje (engl. *shader*) – izvodi se na procesoru vrhova/teselacije/geometrije/točaka
- *Shaderi* se pišu u jezicima za sjenčanje (engl. shading languages) i prevode u *assembler*
- Jezici slični C-u, prilagođeni specifičnostima grafičkog sklopovlja
  - *Stream processing*, vektorske operacije, nema slobodnog pristupa memoriji
  - Glavni tip podataka 4x32-bit vektor, no postoje i skalarni tipovi, cjelobrojni tipovi, matrice
  - Operacije množenja, zbrajanja, oduzimanja...
  - Ugrađene funkcije – mat. i log. operacije, uzorkovanje tekstura, izračun gradijenta (samo u procesoru točaka)
  - Definiranje funkcija i struktura (kao u C-u)
  - Preprocesor – #include, #define, #ifdef itd. (kao u C-u)

Programski jezici za sjenčanje (2/2)

- Razvili se iz ekvivalentnih jezika za *offline* iscrtavanje (Pixar PhotoRealistic RenderMan)
- Za prvu generaciju (prije SM2.0) morao se koristiti *assembler*
- Cg (NVIDIA, 2003.)
  - Prvi visokorazinski jezik za sjenčanje
  - kratica – *C for Graphics*
- HLSL (Microsoft, 2003.)
  - *High Level Shading Language*
  - Gotovo identičan Cg-u, no namijenjen korištenju uz Direct3D
- GLSL (OpenGL ARB, 2004.)
  - *OpenGL Shading Language*
- Od SM4.0 *shaderi* se više ne mogu pisati u *assembleru*