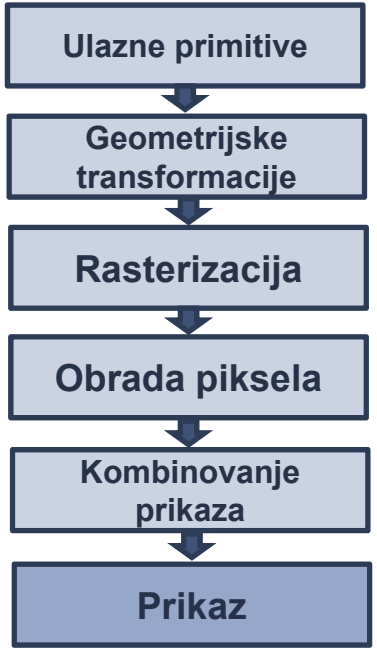
Računarska grafika
(2OER7O02)

3D grafički protočni sistem

Predavanja

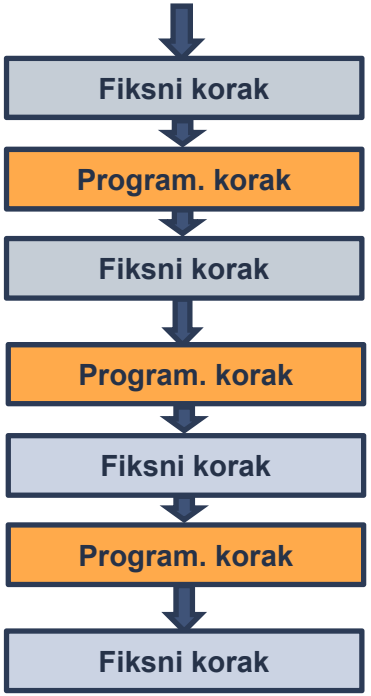
Otvaranje protočnog sistema

- Nekada **funkcionalnost** grafičkih podsistema bila **fiksna**
- Mogli smo upravljati iscrtavanjem menanjem parametara, ali nismo mogli menjati algoritme koji su ugrađeni u sam proces
- Proces obrade podataka, od ulaznih primitiva i njihovih atributa, do konačnog prikaza slike na ekranu, organizovan je u niz koraka koje nazivamo **grafički protočni sistem**
- Vremenom su „otvarani“ pojedini koraci tog protočnog sistema, omogućujući programerima da pišu programe koji implementiraju specijalizovane algoritme

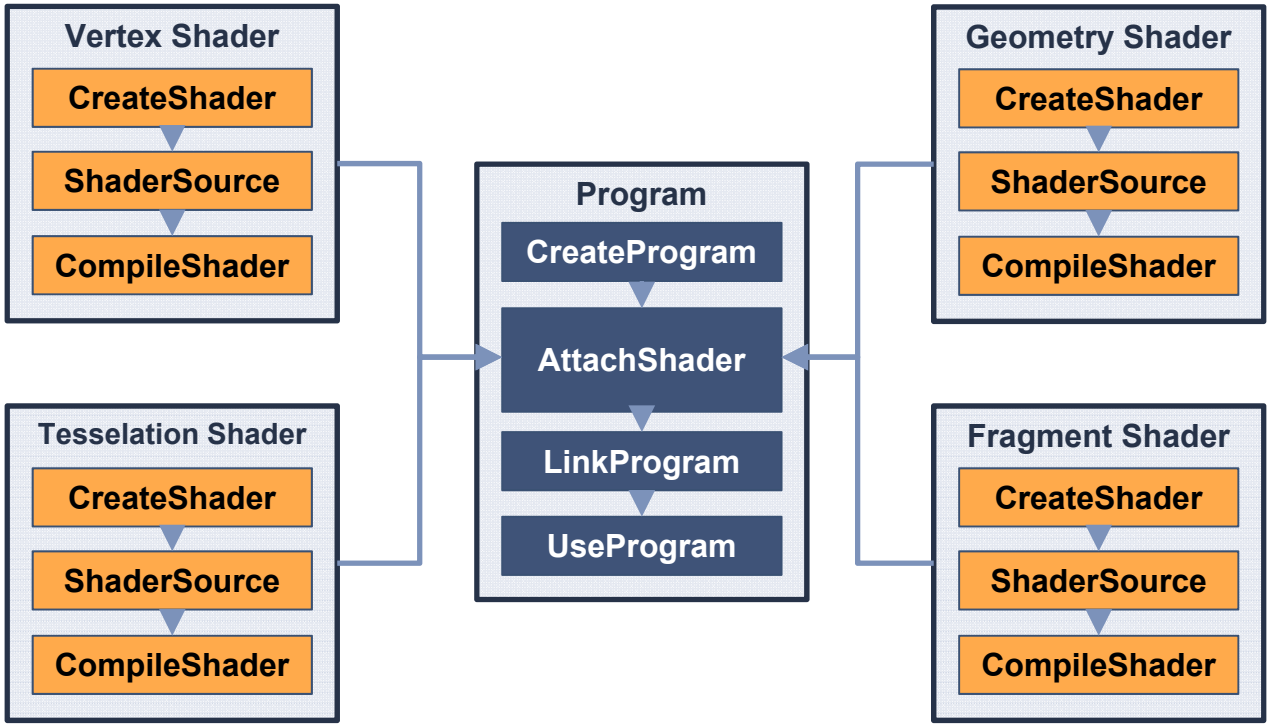


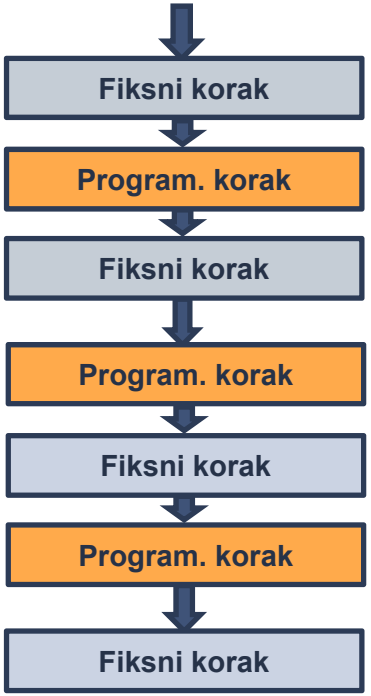
Šejderi

- Program koji se izvršava u nekom od programabilnih koraka grafičkog protočnog sistema naziva se **šejder** (*shader*)
- Svi šejderi koji se koriste u jednom prolazu kroz protočni sistem se linkuju u celinu koja se naziva **program**
- Jezik koji omogućuje pisanje šejdera naziva se *Shading Language* i zapravo je skup srodnih jezika za programiranje odgovarajućih procesora (*vertex, tessellation control, tesselation evaluation, geometry, fragment, comute,* itd)



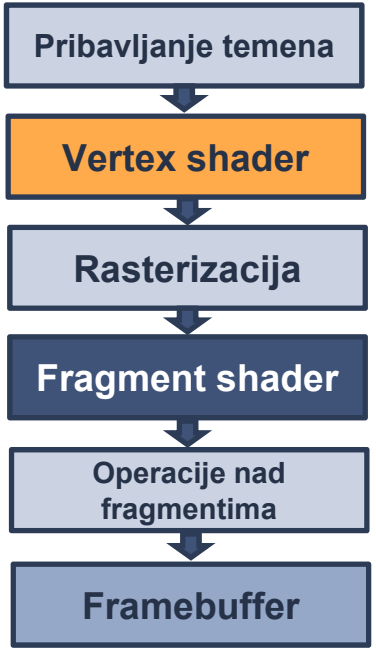
Šejderi


Vertex Shader
CreateShader
ShaderSource
CompileShader
Tesselation Shader
CreateShader
ShaderSource
CompileShader
Program
CreateProgram
AttachShader
LinkProgram
UseProgram
Geometry Shader
CreateShader
ShaderSource
CompileShader
Fragment Shader
CreateShader
ShaderSource
CompileShader


Fiksni korak
Program. korak
Fiksni korak
Program. korak
Fiksni korak
Program. korak
Fiksni korak

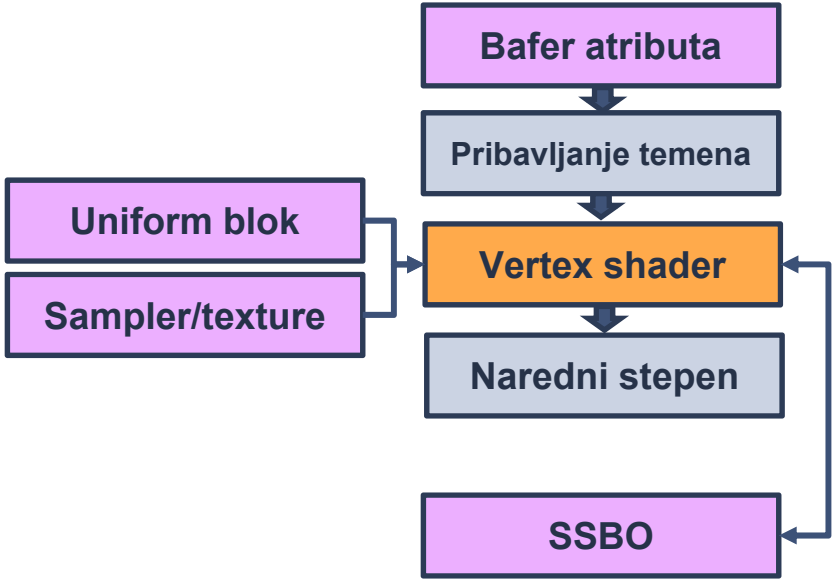
Vertex shader (VS)

- **Vertex shader** vrši transformaciju temena (*vertex*-a)
- Osnovni zadatak je da na osnovu lokalnih koordinata temena, izračuna njegove clip-koordinate (tj. poziciju)
  - $V_c = \mathbf{P} \cdot \mathbf{V} \cdot \mathbf{M} \cdot V_L$
- Obrađuje **samo jedno teme** i nema nikakve informacije o ostalim temenima tekuće primitive
- Svako teme ima jedinstveni identifikator (*VertexID*) koji ga jednoznačno određuje u okviru *draw* poziva
- Sem pozicije, VS definiše i veličinu tačke (u pikselima), kao i rastojanja od ravni odsecanja (*ClipDistance*) i odbacivanja (*CullDIstance*), za upravljanje korisnički-definisanim odsecanjem/odbacivanjem.



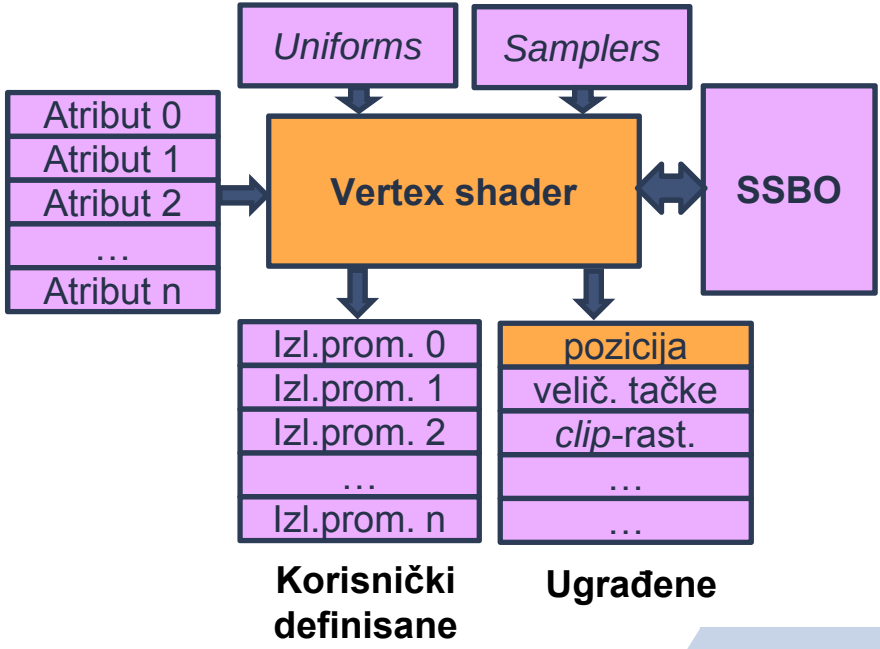
Vertex shader (VS)

- Podatke preuzima iz:
  - **Bafera atributa** (*per-vertex* podaci; npr. normale, tex. koord., boje)
  - **Konstanti** (*uniform* promenljive; konstantne za čitav *draw* poziv)
  - **Tekstura** (pristupa se preko *sampler* promenljivih)
  - **SSBO** (*shader storage buffer object*)
- Rezultate smešta u:
  - **Izlazne promenljive** (moraju se poklapati sa ulaznim prom. narednog programabilnog stepena)
  - **SSBO** (*shader storage buffer object*)



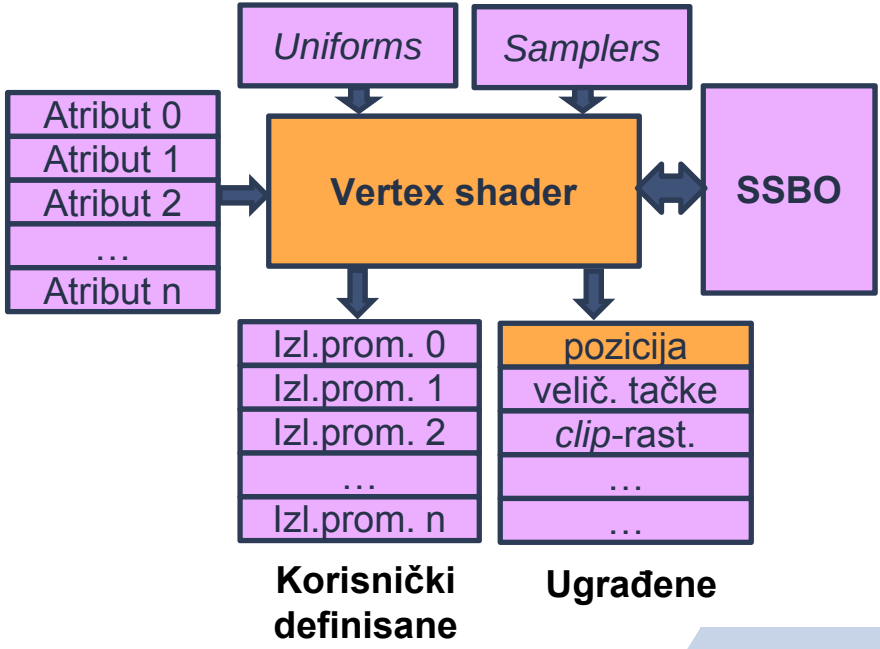
Vertex shader (VS)

- Blok za pribavljanje temen (*vertex puller*) izvlači **atribute** za svako od temena i smešta ih u odgovarajuće slotove (VS ih vidi kao celobrojne vrednosti)
- Skalari i vektori (vec2, vec3 i vec4) zauzimaju po jedan slot, dok matrice (mat3x3, mat4x4) zauzimaju više slotova
- Mapiranje slotova se može prepustiti linkeru (tada se mora očitati) ili definisati u aplikaciji ili shader-u
- ***Uniform*** promenljive su konstantne (npr. MV matrica, pozicija izvora svetlosti, itd.)



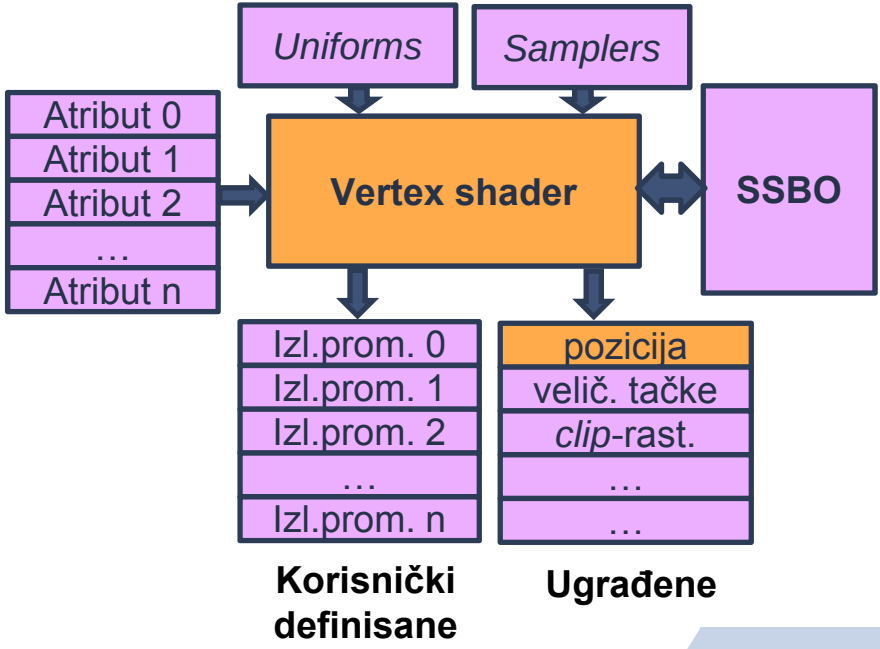
Pristup teksturama

- ***Sampler*** je *handle* za pristup teksturi (poseban tip *uniform* promenljive koji identifikuje objekat teksture – prosleđuje se kao parametar funkcijama za pristup)
- *Sampler* je celobrojna vrednost [0..maxSup) koja selektuje teksturnu jedinicu (ima ih više da omoguće višestruko teksturisanje), a time i teksturu koja je povezana (*bind*-ovana) na nju
- Npr. *sampler2D* selektuje TEXTURE_2D tip teksture na tekturnoj jedinici (koja odgovara vrednosti upisanoj u datu promenljivu) – teksturna jedinica je izabrana u aplikaciji pomoću *ActiveTexture()* a sama 2D tekstura je vezana za nju pozivom *BindTexture()*



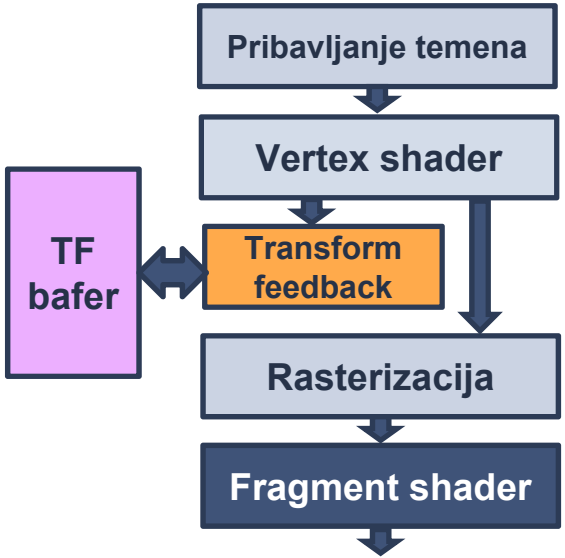
Izlazne promenljive

- VS mora upisati vrednost u ugrađenu izlaznu promenljivu ***position***
- Sve što utiče na fiksni deo protočnog sistema predstavljeno je **ugrađenim promenljivama**
- **Korisnički definisane promenljive** su interfejs ka narednom programabilnom stepenu i moraju da imaju isti tip i naziv kao u tom stepenu
- **SSBO** je bafer-objekat preko koga *shader*-i mogu deliti velike količine podataka (mogu i upisivati i čitati iz njega)



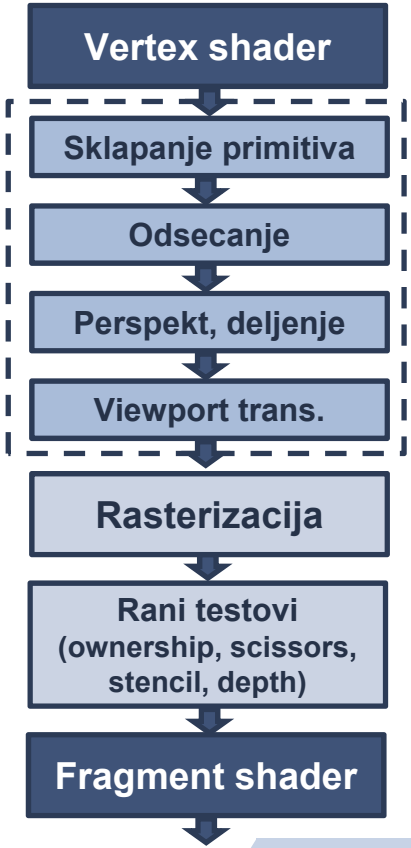
Povratne informacije o transformacijama

- **Transform feedback** (TF) je mehanizam koji omogućuje da se pribave atributi temena, nakon što su izvršene transformacije (nakon poslednjeg geometrijskog stepena – *vertex*, *tessellation* ili *geometry shader*-a)
- Izvorno se koristio za GPGPU (razna negrafička izračunavanja na grafičkom procesoru) i debagiranje
- Prvenstvena namena je da se preuzmu vrednosti generisane na grafičkom procesoru i na osnovu njih izvrši nešto na CPU
- Zadaje se odredišni bafer, u koji se upisuju željeni podaci, a rasterizaciju bi trebalo isključiti



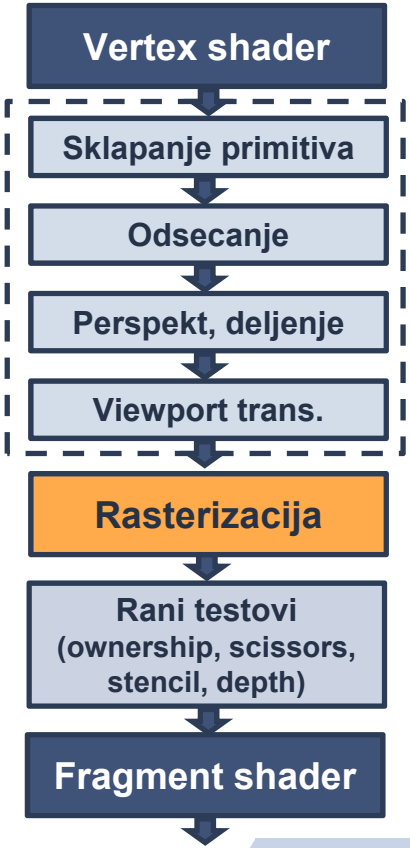
Sklapanje primitiva

- Primitiva je geometrijski objekat koji se može iscrtati nekom od *draw* komandi (mogu biti tačke/sprajtovi, linije i trouglovi)
- **Sklapanje primitiva** (*primitive assembly*) – na osnovu koordinata temena (x,y,z,w), tipa primitive i indeksa definišu se pojedinačne primitive
- **Odsecanje** (u odnosu na prostor pogleda, ali i dodatnih ravni;
  $-w_c \leq x_c \leq w_c$, $-w_c \leq y_c \leq w_c$, $-w_c \leq z_c \leq w_c$
  na mestu preseka dodaju se nova temena)
- **Perspektivno deljenje** (deljenje *clip*-koordinata (x_c,y_c,z_c,w_c) sa w_c dobijaju se normalizovane koord. uređaja)
- **Viewport transformacija** (konverzija normalizovanih koordinata uređaja u koordinate prozora)



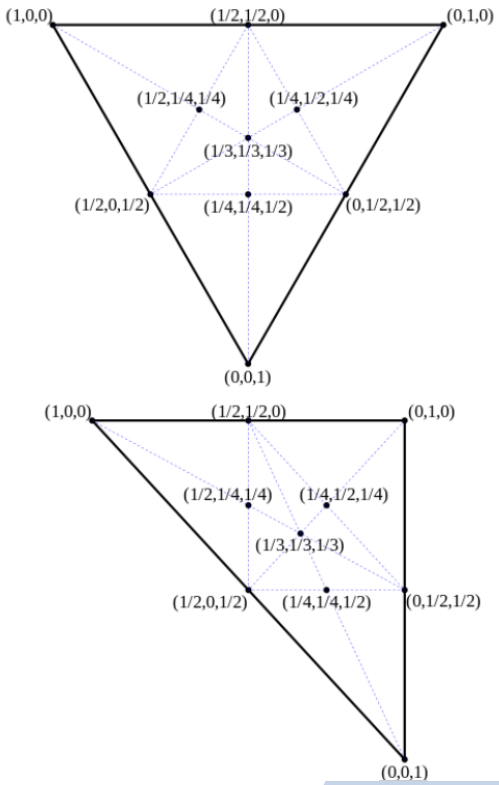
Rasterizacija

- Odbacivanje primitiva pre rasterizacije (ako je uključeno RASTERIZER_DISCARD; koristi se kod *transform feedback*-a)
- Određivanje strane (*front/back-facing*) i odbacivanje (*culling*)
- Rasterizacija je proces konverzije primitive (tačke, linije ili trogla) u 2D sliku, pri čemu svaka tačka te slike ima bar informaciju o boji i dubini
- Interpolacija atributa
  - ***flat*** – nema interpolacije, vrednost dolazi od jednog (*provoking*) temena
  - ***noperspective*** – linearna interpolacija u *screen* prostoru
  - ***smooth*** – perspektivno-korektna interpolacija



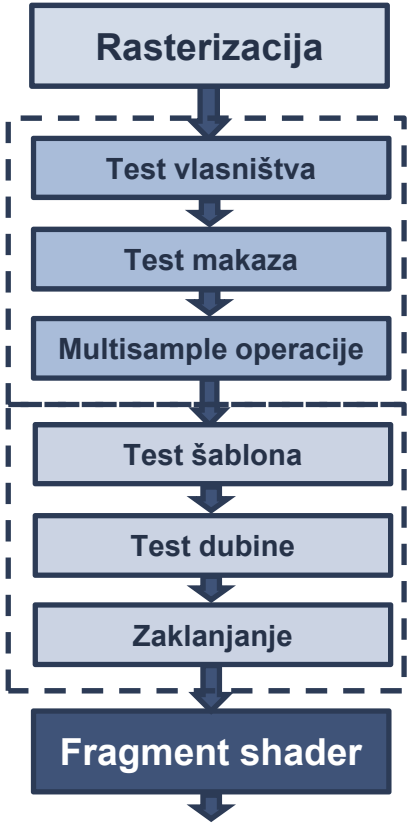
Interpolacija

- Perspektivno-korektna interpolacija se obavlja u **baricentričnim** (*barycentric*) koordinatama
- Za trougao, baricentrične koordinate (*u*,*v*,*w*) definišu relativni pomeraj u odnosu na temena
- Za baricentrične koordinate važi:
  - u + v + w = 1
  - u, v, w ∈ [0..1]



Rani testovi fragmenata

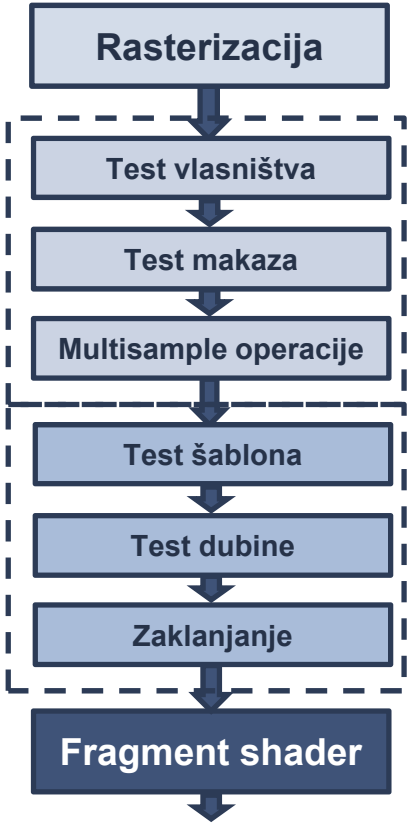
- Da bi se izbegla izračunavanja za fragmente koji nisu vidljivi, svi zaklonjeni se odbacuju na osnovu „ranih“ testova
- Tri operacije su obavezne, i to:
  - **Test vlasništva piksela** (*pixel ownership test*) – da li tekući fragment pripada kontekstu (ako je prozor delimično prekriven drugim prozorom treba sprečiti njegovo iscrtavanje)
  - **Test makaza** (*scissor test*) – ako je fragment van definisane pravougaone (clip) oblasti za svaki viewport odbacuje se
  - **Višestruko uzorkovanje** (*multisample*) – obavezno samo ako je uključeno višestruko uzorkovanje i definiše masku i pokrivanje (koji uzorci se koriste u određivanju boje fragmenta, na osnovu maske i logičke operacije)



Multisampling uzorkuje piksele više puta na različitim subpikselskim lokacijama da bi se generisao efekat umekšavanja (***antialiasing***).

Rani testovi fragmenata

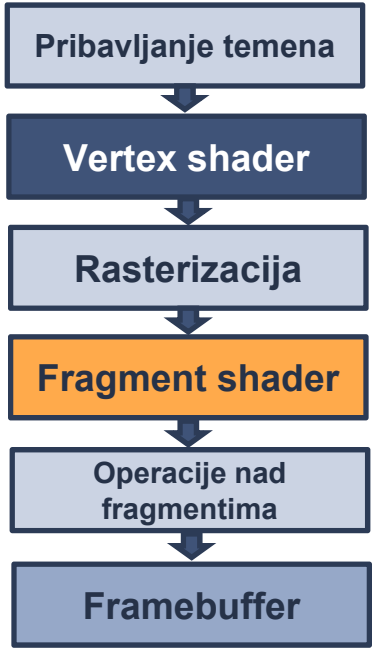
- Tri operacije se dodatno mogu uključiti, i to:
  - **Test šablona** (*stencil test*) – poredi tekući fragment sa vrednostima u odgovarajućem baferu i na osnovu toga ga odbacuje ili obrađuje dalje (omogućuje definisanje složene maske)
  - **Test dubine** (*depth test*) – poredi tekući fragment sa vrednostima u Z-baferu
  - **Zaklanjanje** (*occlusion*) – odbacivanje na osnovu upita da li će tekući fragment biti zaklonjen drugim objektima



Occlusion query je tehnika koja utvrđuje koliko piksela je iscrtano tokom crtanja jednog ili više objekata. Ako nije nijedan, znači da se ne vide i mogu se odbaciti. Za potrebe ispitivanja crtaju se uprošćeni objekti (nakon što je startovan upit), zatim obavlja nešto drugo (da bi se završio upit) i nakon toga, ako je broj „iscrtanih“ piksela veći od 0 (ili nekog praga), crtaju se objekti sa svim detaljima.

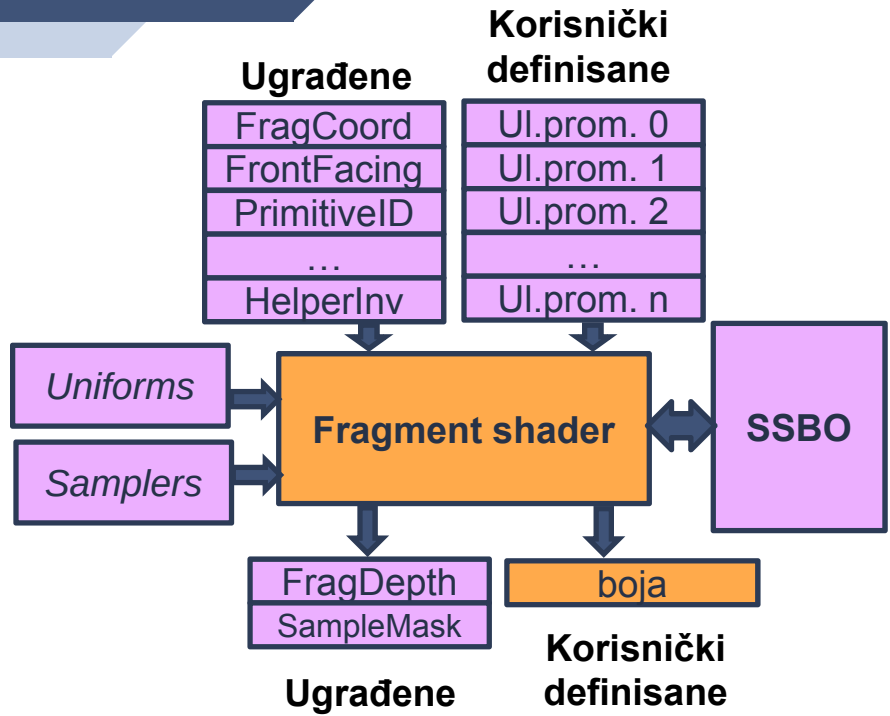
Fragment shader (FS)

- **Fragment shader** vrši transformaciju fragmenta (prototip piksela)
- Obrađuje **samo jedan fragment**, ne može pristupiti susednim fragmentima, niti može promeniti poziciju (x,y) tekućeg fragmenta, a izračunate vrednosti se koriste da ažuriraju *framebuffer* ili teksturu
- Osnovni zadatak je da definiše konačnu boju piksela, ali može i odbaciti fragment (**discard**)
- Iako ne može pristupiti susednim fragmentima, posebnim funkcijama se može dobiti „priraštaj“ (tj. aproksimacija parcijalnog izvoda) neke vrednosti po X i Y pravcu (obzirom da hardver obično „zahvata“ 2x2 fragmenta)



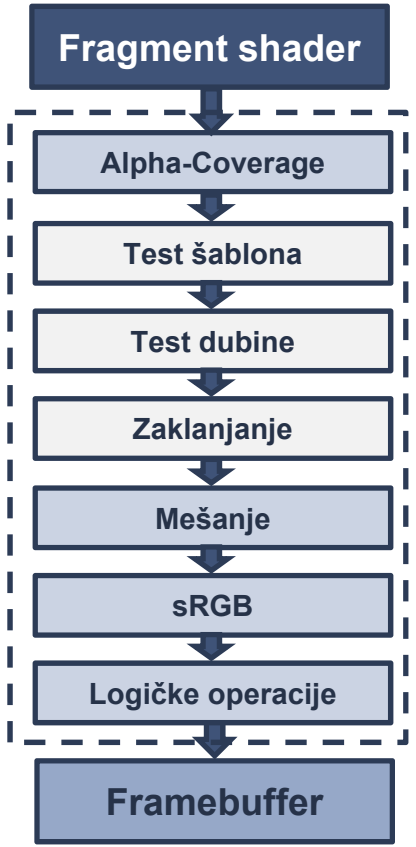
Fragment shader (FS)

- FS preuzima podatke od prethodnih koraka preko ulaznih promenljivih, koje mogu biti
  - **Ugrađene** (*built-in*) – podaci potiču od fiksnog dela protočnog sistema i
  - **Korisnički definisane** (*user-defined*) – potiču od prethodnog programabilnog koraka (*shader*-a)
- Korisnički definisane ulazne promenljive se moraju po nazivu i tipu poklapati sa izlaznim promenljivama prethodnog prog. Stepena
- Jedna korisnički definisana izlazna promenljiva je obavezna i ona definiše konačnu boju fragmenta



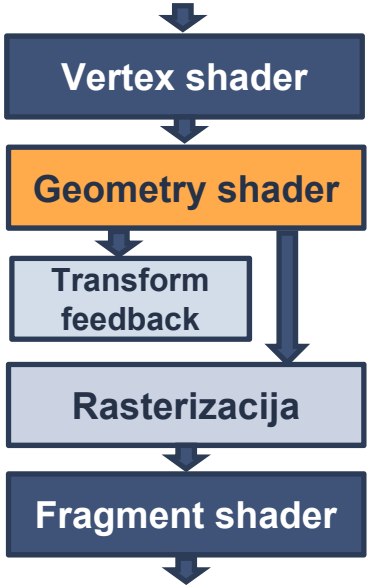
Operacije nad fragmentima

- Nakon fragment shader-a, nad fragmentima se primenjuju sledeće operacije, kao deo fiksne funkcionalnosti:
  - **Alpha-coverage** – ako je omogućen modifikuje *alpha* i *coverage* vrednosti (ako je onemogućeno višestruko uzorkovanje ili je bafer u kome se crta celobrojni, ovaj korak se preskače)
  - **Testovi šablona** (*stencil*), **dubine** (*depth*) i **zaklanjanja** (*occlusion query*) se preskaču ako su obavljeni kao „rani“ testovi
  - **Mešanje** (*blending*) – kombinuje boju tekućeg fragmenta sa bojom koja je već upisana u *framebuffer*
  - Konverzija u **sRGB** – opciona standardizacija boja prikaza (primenljivo samo na neoznačene normalizovane kolor-bafere sa fiksnom tačkom)
  - **Logičke operacije** (slično mešanju) tekućeg fragmenta i onoga što je već smešteno u *framebuffer*

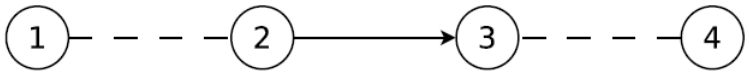


Geometry shader (GS)

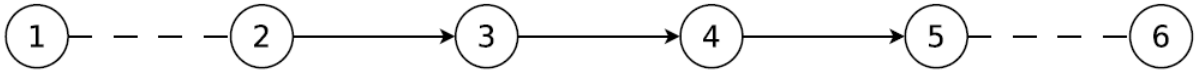
- **Geometry shader** (GS) je opcioni korak protočnom sistemu, i sledi nakon obrade temena u VS i njihovog organizovanja u primitive, a pre TF
- GS obrađuje **jednu primitivu** i može emitovati jednu ili više primitiva istog tipa (ali se tip ne mora slagati sa ulaznim)
- Originalne primitive se odbacuju nakon izvršenja GS
- Ulaz su atributi svih temena koja čine primitivu (umesto skalara, atributi se predstavljaju vektorima sa 1 do 6 komponenti)
- Uvedene su dodatne (ulazne) primitive, tzv. **primitive sa susedstvom**, koje, sem osnovnih, sadrže i susedna temena
- Ulaz mogu biti: tačke, linije, linije sa susedstvom, trouglovi i trouglovi sa susedstvom, a izlaz: tačke, trake linija i trake trouglova



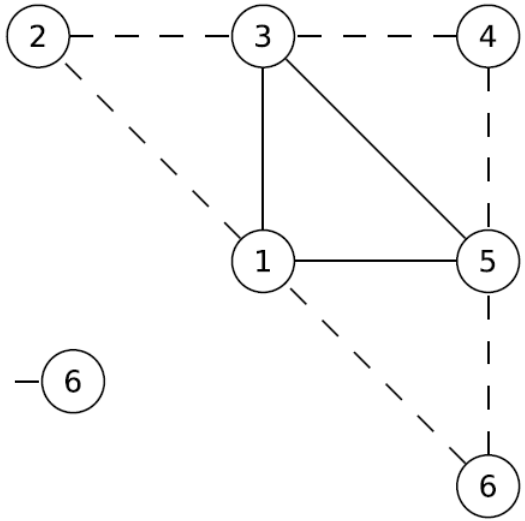
Primitive sa susedstvom



Linija sa susedstvom

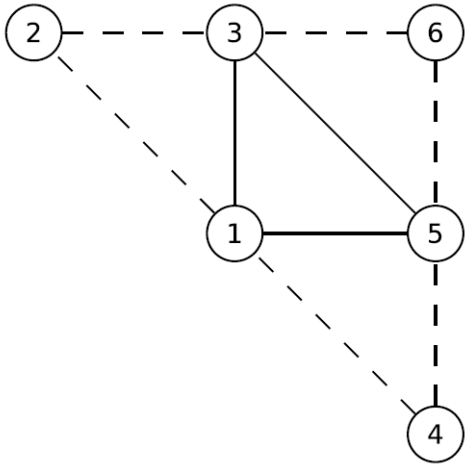


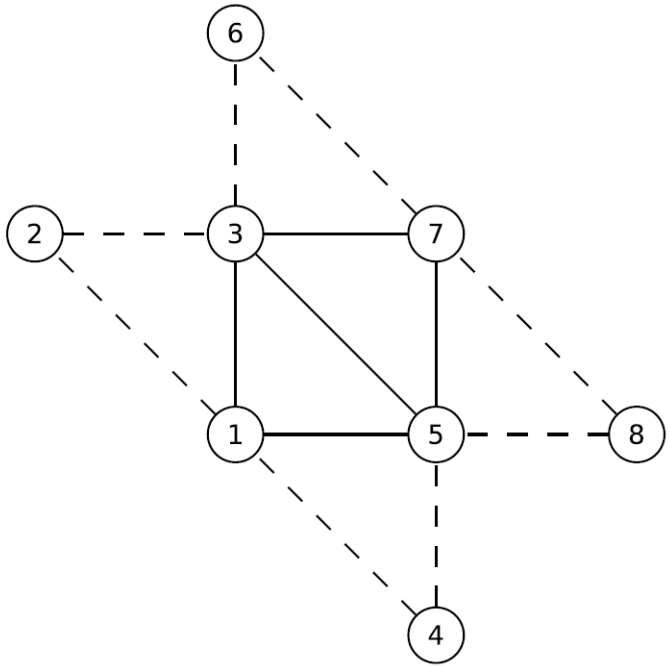
Traka linija sa susedstvom



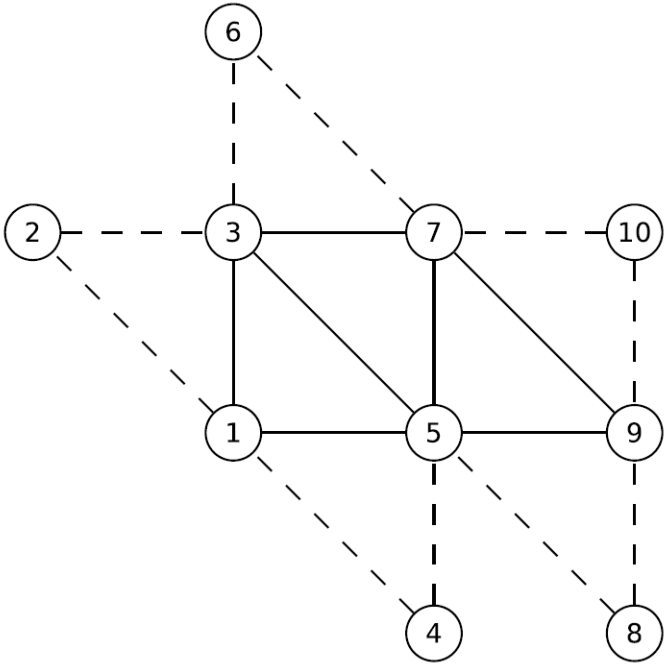
Trougao sa susedstvom

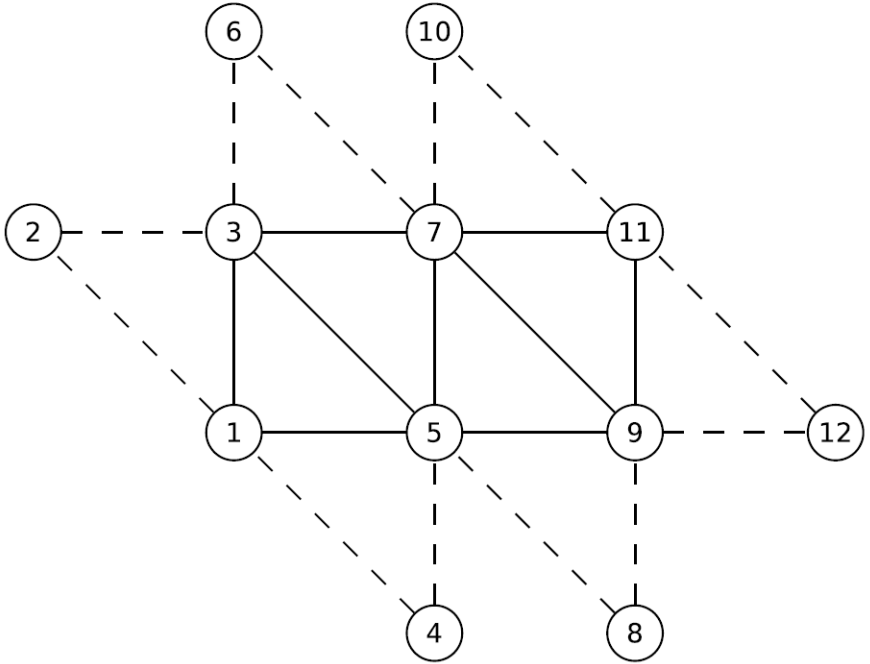
Traka trouglova sa susedstvom





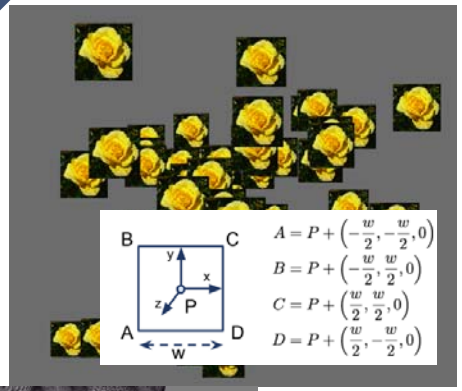
Traka trouglova sa susedstvom

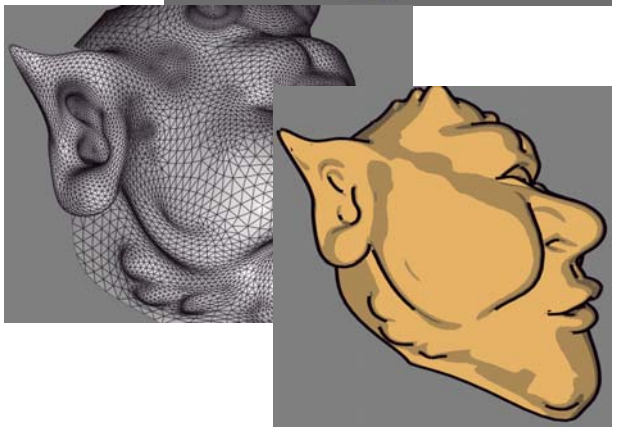




Primena geometry shader-a

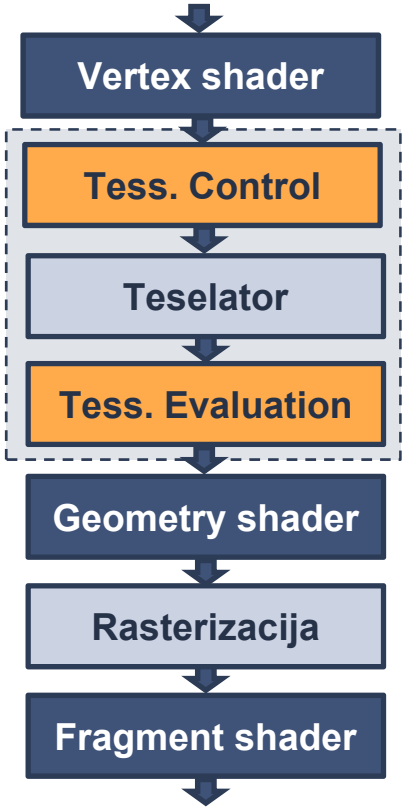
- Odbacivanje primitiva
- Tačkasti sprajtovi (*point sprites*) – prosleđuju se tačke, a GS generiše kvadrate (prostorne i teksturne koordinate) poravnate sa pogledom
- Crtanje žičanog preko senčenog modela – izračunava se rastojanje temena od naspramnih strana i interpolira u prostoru ekrana (*noperspective*), da bi FS na osnovu udaljenosti od ivica obojio piksele.
- Iscrtavanje silueta (stil crtanih filmova) – pomoću malih kvadrata poravnatih sa ivicama





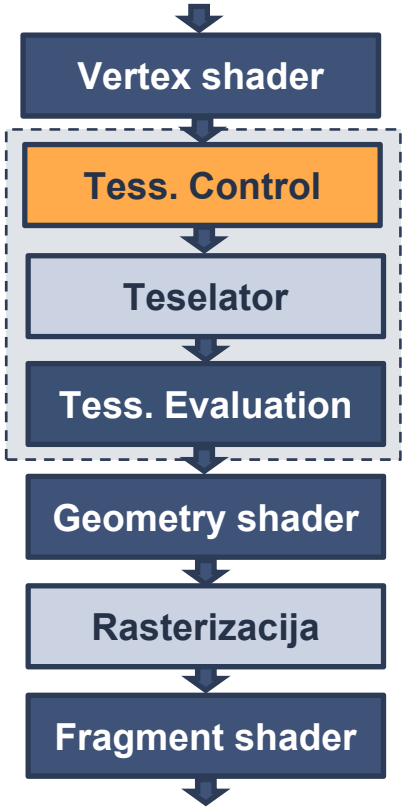
Tessellation shader

- **Teselacija** (popločavanje) je proces podele površine na podpovršine, bez preklapanja ili pukotina, čime se povećava složenost te površine
- Ovo je opcioni deo grafičkog protočnog sistema, koji se sastoji od tri koraka:
  - **Tessellation Control shader** (TCS) – programabilni korak koji upravlja procesom teselacije
  - **Teselator** (**generator primitiva**) – deo fiksne funkcionalnosti, koji formira temena „podeljene“ površine, na osnovu faktora definisanih u TCS
  - **Tessellation Evaluation shader** (TES) – programabilni korak koji modifikuje temena generisana teselatorom



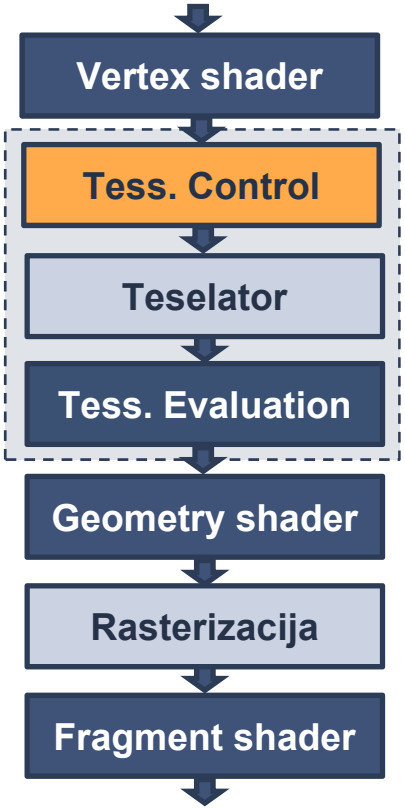
Tessellation Control shader (TCS)

- **Tessellation Control shader** prihvata ulazni peč (*patch*), obrađuje ga i emituje novi izlazni peč.
- **Peč** je novi tip primitive (TS se ne može koristiti ako su ulazne primitive tačke, linije ili trouglovi) koji sadrži skup temena na osnovu kojih se definišu nove primitive
- **Peč** ne predstavlja geometriju, već **kontrolne tačke**, čiji se broj unutar peča kreće od 1 do 32 (zavisno od implementacije i više)
- **TCS** se izvršava za **svako teme izlaznog peča** (obrađuje samo jedno teme peča), može pročitati atribute bilo kog temena ulaznog ili izlaznog peča, ali može upisati vrednosti *per-vertex* atributa samo odgovarajućeg temena izlaznog peča
- Osim *per-vertex*, postoje i *per-patch* atributi, koji su jedinstveni za čitav peč, i može ih upisati bilo koja instanca TCS datog peča

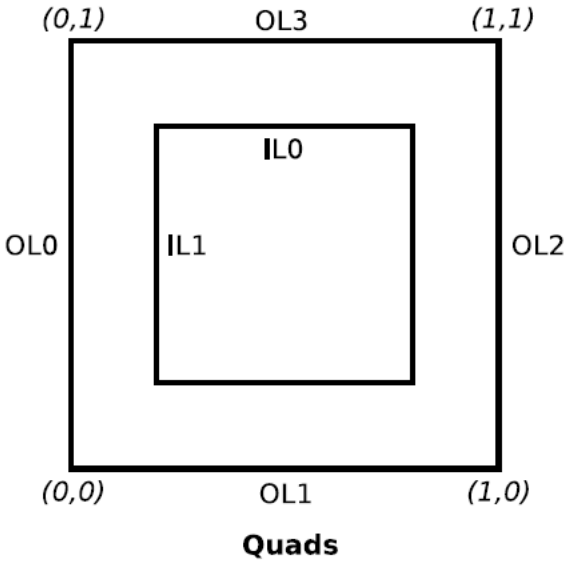


Tessellation Control shader (TCS)

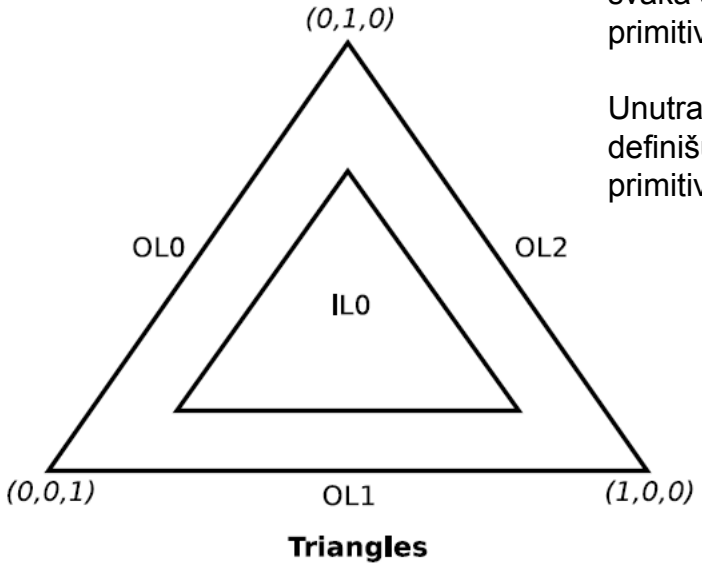
- TCS definiše koliko temena ima u izlaznom peču
- Pristup *per-vertex* atributima vrši se na osnovi ID-a instance TCS (*InvocationID*)
- Najznačajniji *per-patch* atributi su:
  - polje **unutrašnih nivoa teselacije** (*TessLevelInner*) i
  - polje **spoljašnjih nivoa teselacije** (*TessLevelOuter*)
- Nivoima teselacije se upravlja radom sledećeg korak u protočnom sistemu
- Ako je potrebna sinhronizacija instanci TCS, koriste se odgovarajuće sinhronizacione pozive (*barrier*), koje sprečavaju dalje izvršenje dok sve instance TCS za odgovarajući peč ne dođu do date barijere



Teselacija poligona



Quads



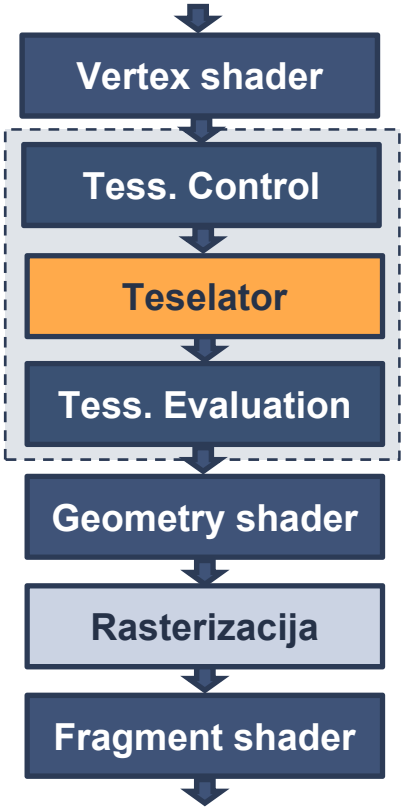
Triangles

Spoljašnji nivoi teselacije (OL) definišu na koliko podeoka se deli svaka spoljašnja strana početne primitive

Unutrašnji nivoi teselacije (IL) definišu kako se u unutrašnjosti deli primitiva

Teselator

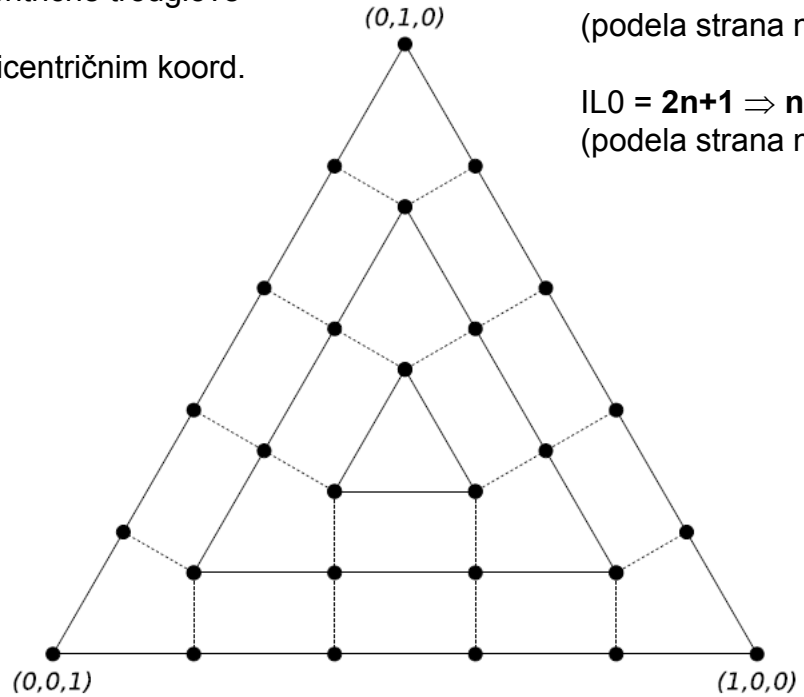
- Teselator (generator primitiva) preuzima izlazni peč od TCS i formira novi skup primitiva, ali samo ako je prisutan TES
- Ako nema TES, TPG samo prosleđuje ulaznu primitivu bez modifikacije
- **Tip teselacije** definiše se u **TES** i može biti: **izolinije**, **trouglovi** ili **četvorouglovi**
- Trouglove i četvorouglove deli na sitnije trouglove, a izolinije na kolekciju linijskih segmenata, organizovanih u trake (koje se protežu horizontalno preko okvirnog pravougaonika)
- Svako teme dobija normalizovane (u opsegu [0..1]) koordinate (u,v) ili (u,v,w)
- Izolinije i četvorouglovi dobijaju Dekartove (u,v), a trouglovi baricentrične (u,v,w) koordinate



Teselacija trougla

Podela na koncentrične trouglove

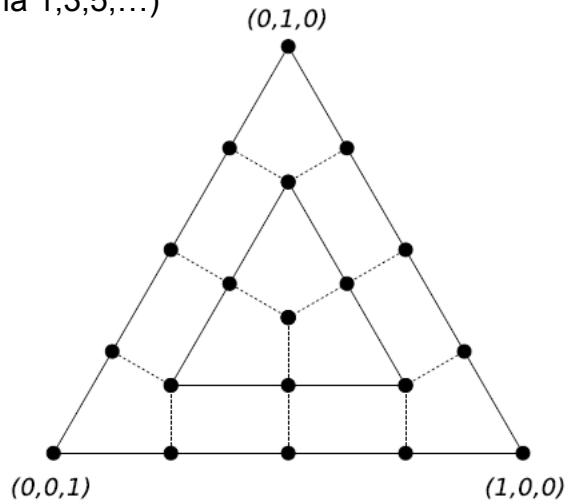
Centar je na baricentričnim koord.
(1/3, 1/3, 1/3)



IL0 = OL0 = OL1 = OL2 = 5

IL0 = **2n** ⇒ **n** konc. trougao + teme u centru
(podela strana na 2,4,6,…)

IL0 = **2n+1** ⇒ **n+1** konc. trougao
(podela strana na 1,3,5,…)

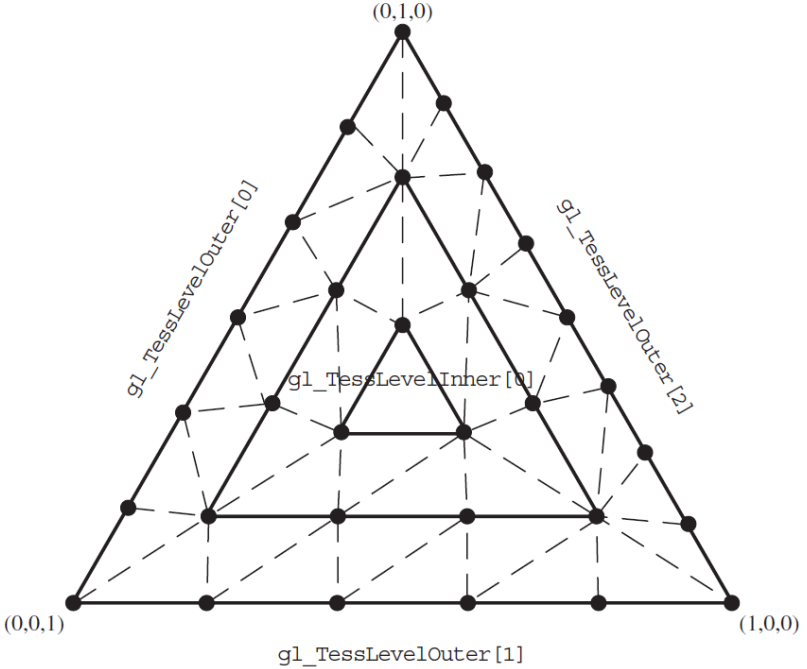


IL0 = OL0 = OL1 = OL2 = 4

Teselacija trougla

- gl_TessLevelOuter[0] = 6;
- gl_TessLevelOuter[1] = 5;
- gl_TessLevelOuter[2] = 8;

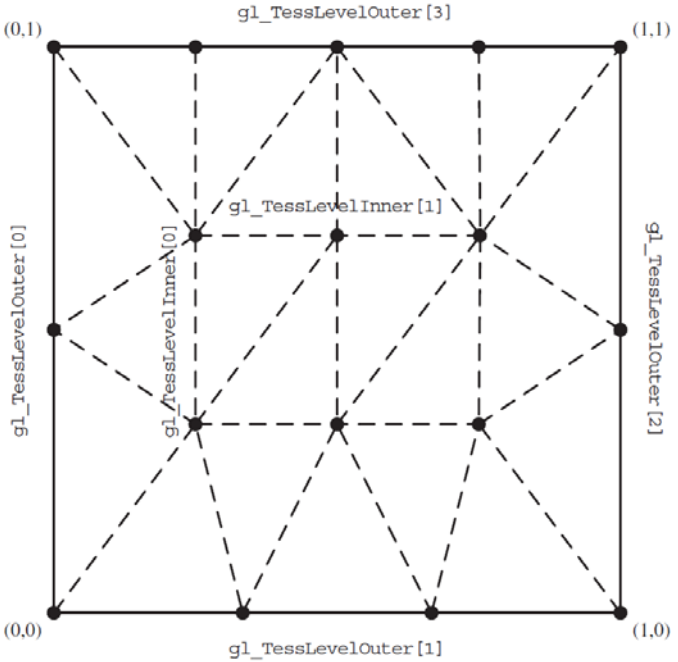
- gl_TessLevelInner[0] = 5;



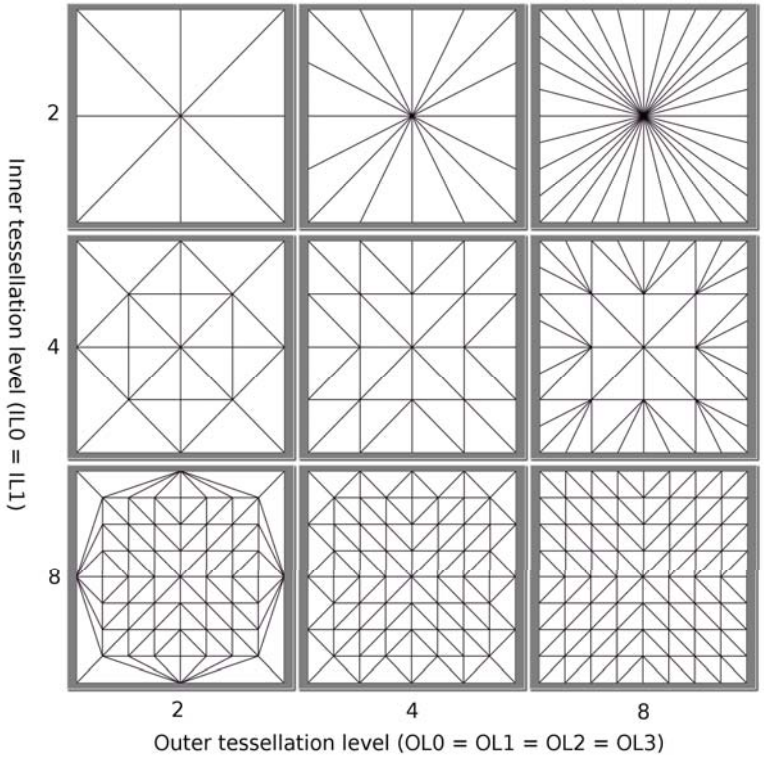
Teselacija četvorougla

- gl_TessLevelOuter[0] = 2.0;
- gl_TessLevelOuter[1] = 3.0;
- gl_TessLevelOuter[2] = 2.0;
- gl_TessLevelOuter[3] = 4.0;

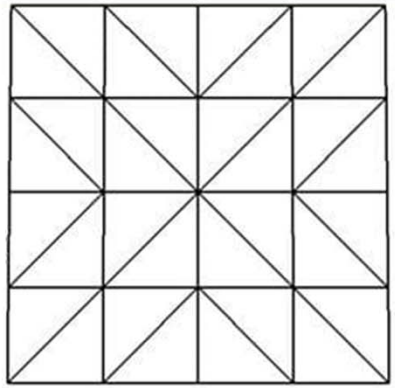
- gl_TessLevelInner[0] = 4.0;
- gl_TessLevelInner[1] = 3.0;

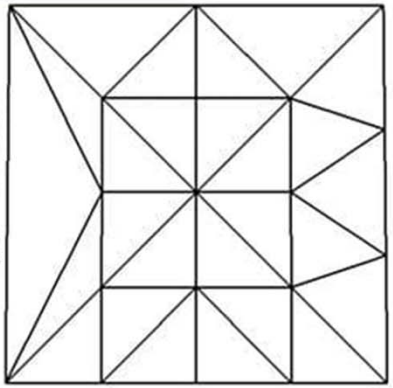


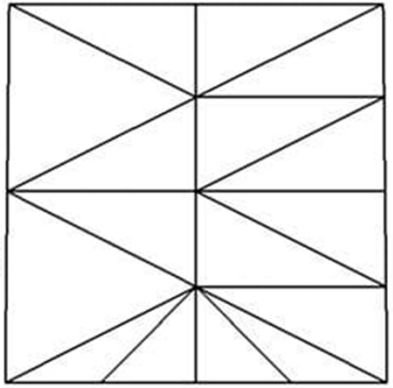
Teselacija četvorougla

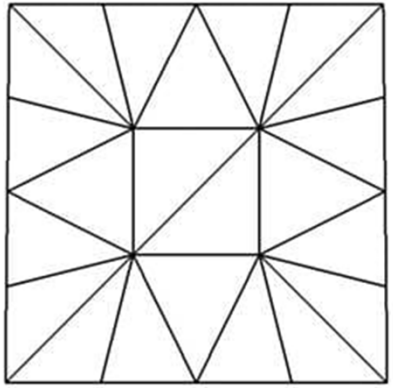


Teselacija četvorougla – vežba



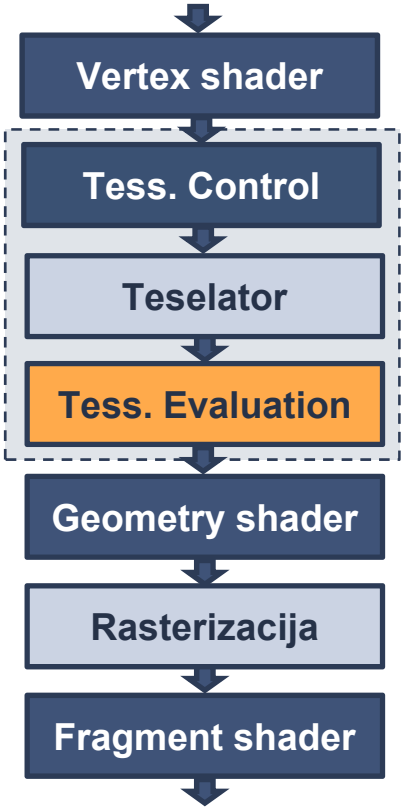






Tessellation Evaluation shader (TES)

- **Tessellation Evaluation shader** izračunava poziciju i druge atribute temena generisanih teselatorom (generatorom primitiva)
- Ono što je VS za temena ulaznih primitiva, to je TES za temena generisana teselacijom
- TES obrađuje **samo jedno teme** novoformirane primitive
- TES može očitati
  - **atribute** bilo kog temena ulaznog **peča** i
  - **teselacione koordinate** (relativna pozicija unutar primitive koja je teselirana)

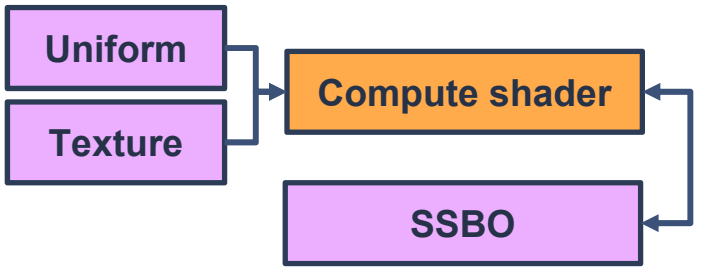


Primena tessellation shader-a

- Teselacija prostornih linija i površi – dodavanje detalja u skladu sa matematičkim modelom
- Upravljanje nivoom detalja (Level-of-Detail, LoD) – smanjenje nivoa detalja sa rastojanjem od posmatrača

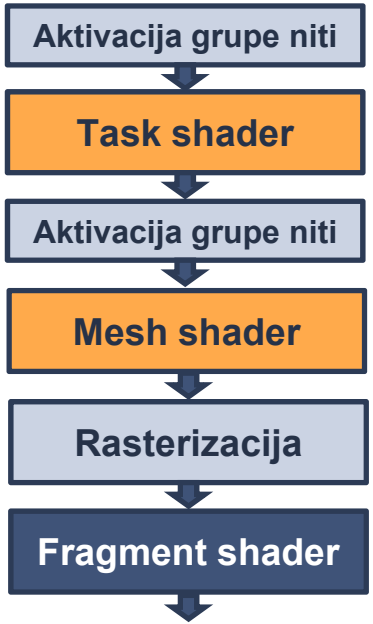
Compute shader (CS)

- **Compute shader** (CS) se izvršava potpuno nezavisno od ostalih koraka u protočnom sistemu
- Ima pristup skoro svim resursima kao i ostali šejderi, ali nema izlaza vezanih za fiksnu funkcionalnost
- Nije deo grafičkog protočnog sistema i njegov rad je vidljiv kroz modifikaciju tekstura, bafera i sl. Objekata
- Koristi se za negrafičko paralelno izračunavanje
- Organizuje posao u okviru radnih grupa (*workgroups*), od kojih je svaka kolekcija instanci šejdera



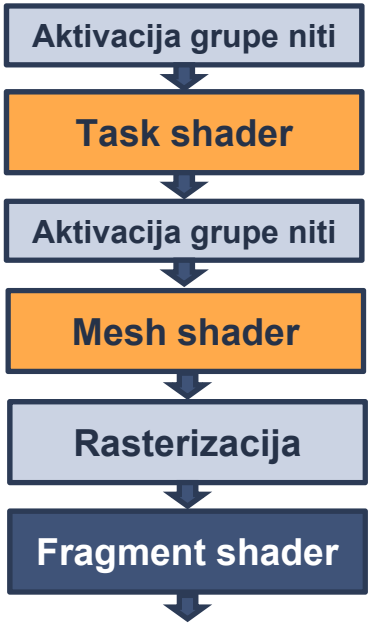
Task i Mesh shader

- Alternativni protočni sistem koji u potpunosti zamenjuje standardni (TS+MS ⇔ VS+TCS+TES+GS)
- **Task shader** (TS) je opcioni korak koji upravlja generisanjem poslova
- Pojedinačne instance TS (max. 32) organizovane su u radne grupe
- TS radna grupa određuje koliko MS radnih grupa će biti kreirano, a svaka instanca TS, na osnovu **ID-a radne grupe** i **ID-a instance** i pristupa SSBO, da li će se odgovarajuća MS radna grupa aktivirati (praktično vrši *culling*)
- ID „preživelih“ MS radnih grupa prosleđuju se sledećem koraku
- TS radna grupa može proslediti i druge podatke MS radnim grupama preko izlaznih promenljivih tipa **taskNV**

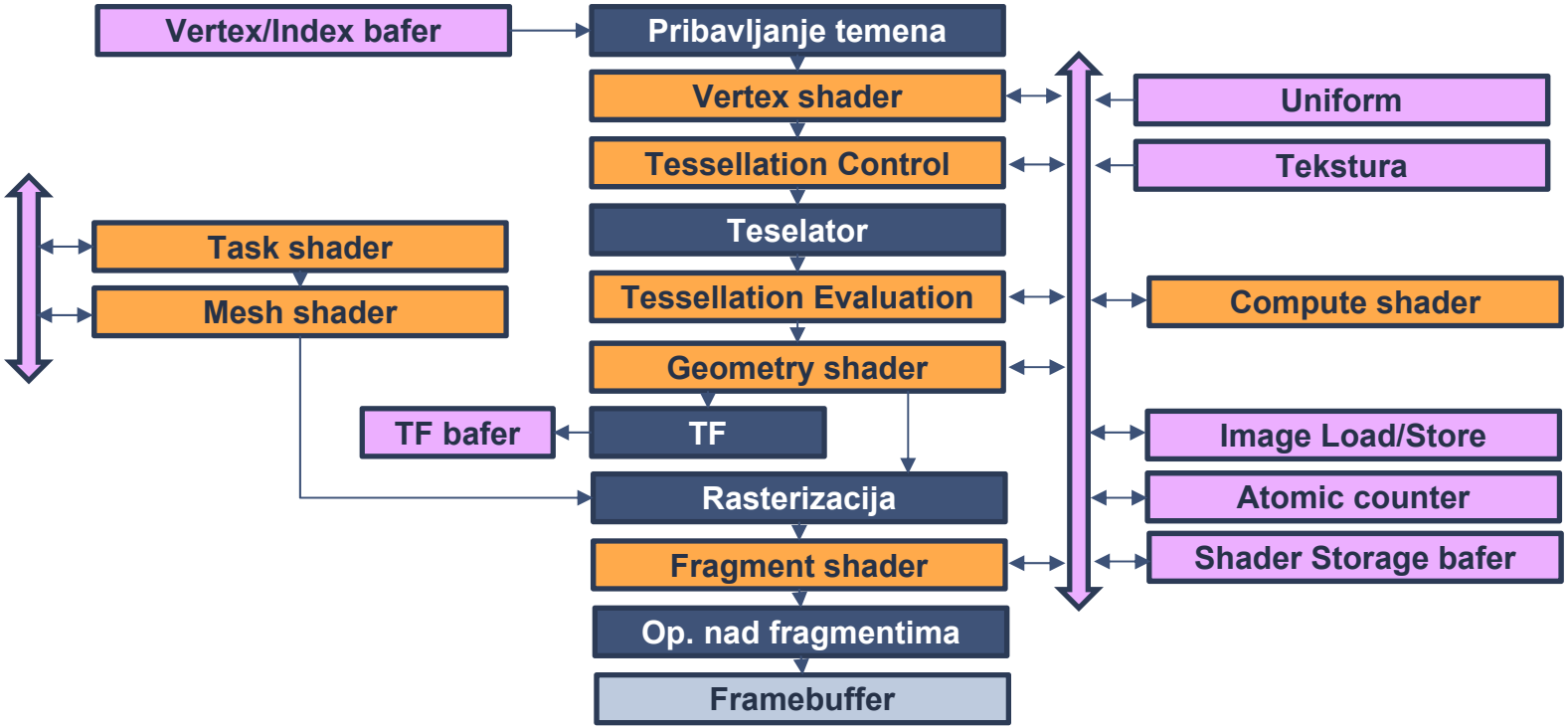


Task i Mesh shader

- **Mesh shader** (MS) je obavezni korak koji generiše geometriju
- Svaka instanca u MS radnoj grupi (max. 32) ima pristup ugrađenim promenljivama koje određuju radnu grupu i instancu, kao i izlaznim promenljivama TS
- Svaka MS radna grupa emituje:
  - **Broj generisanih primitiva**
  - Kolekciju **atributa temena** (svako teme u mešu ima skup ugrađenih i korisnički-definisanih izlaznih promenljivih i blokova)
  - Kolekcije **atributa primitiva** (*per-primitive* – svaka od generisanih primitiva ima ugrađene i korisnički-def. Prom.)
  - **Polje** vrednosti **indeksa** temena (svaka izlazna primitiva ima skup od 1, 2 ili 3 indeksa koji određuju temena u izlaznom mešu koja formiraju tu primitivu)



3D grafički protočni sistem



PITANJA