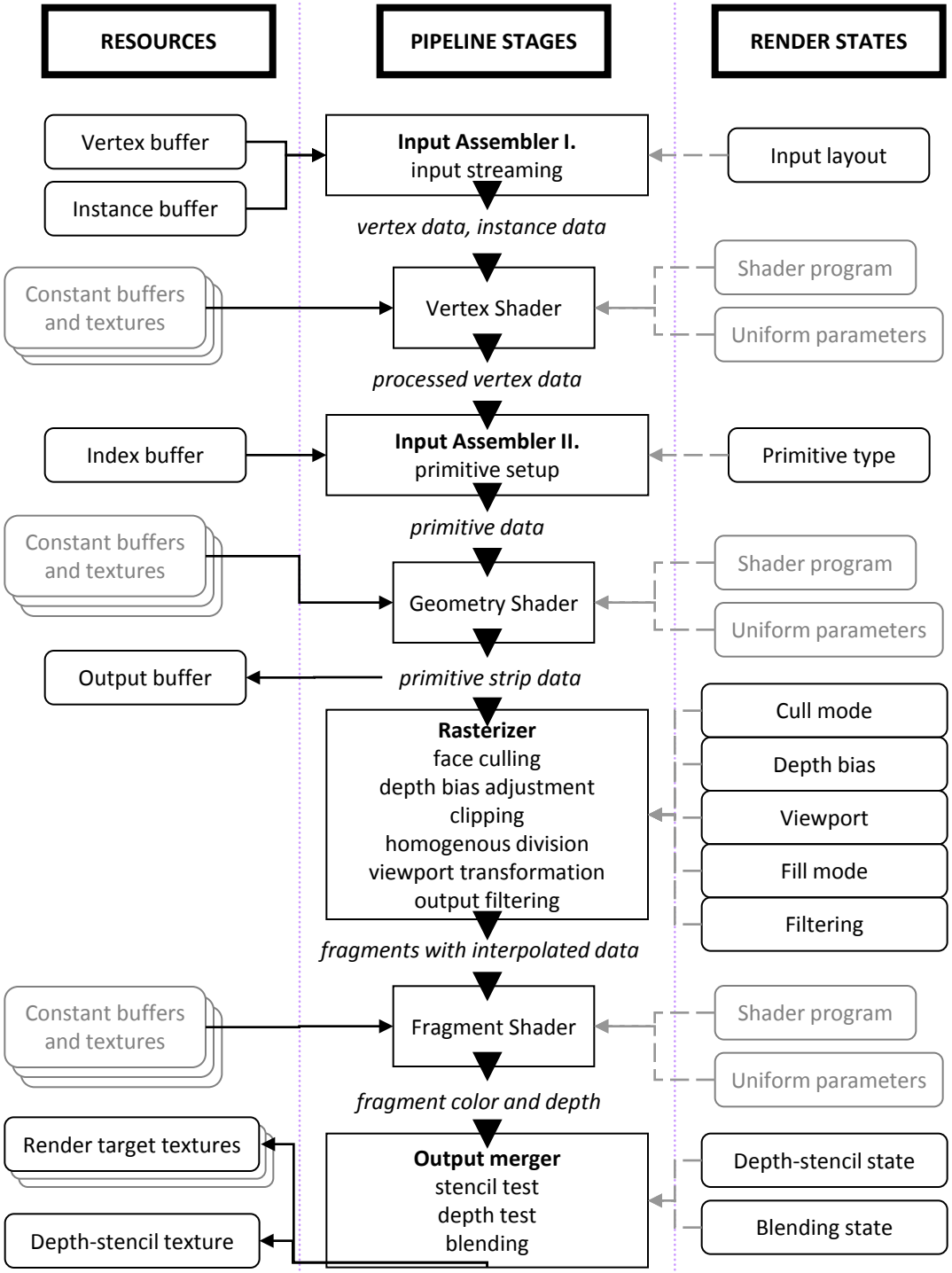
Direct3D pipeline

Grafikus játékok fejlesztése

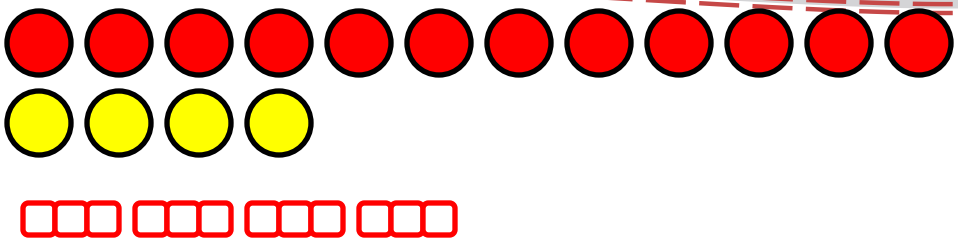
Szécsi László

2013.02.12. t03-pipeline


RESOURCES
PIPELINE STAGES
RENDER STATES
Vertex buffer
Instance buffer
Input Assembler I.
input streaming
Input layout
vertex data, instance data
Constant buffers and textures
Vertex Shader
Shader program
Uniform parameters
processed vertex data
Index buffer
Input Assembler II.
primitive setup
Primitive type
primitive data
Constant buffers and textures
Geometry Shader
Shader program
Uniform parameters
Output buffer
primitive strip data
Rasterizer
face culling
depth bias adjustment
clipping
homogenous division
viewport transformation
output filtering
Cull mode
Depth bias
Viewport
Fill mode
Filtering
fragments with interpolated data
Constant buffers and textures
Fragment Shader
Shader program
Uniform parameters
fragment color and depth
Render target textures
Depth-stencil texture
Output merger
stencil test
depth test
blending
Depth-stencil state
Blending state

GPU pipeline input

- vertex bufferek
- index buffer



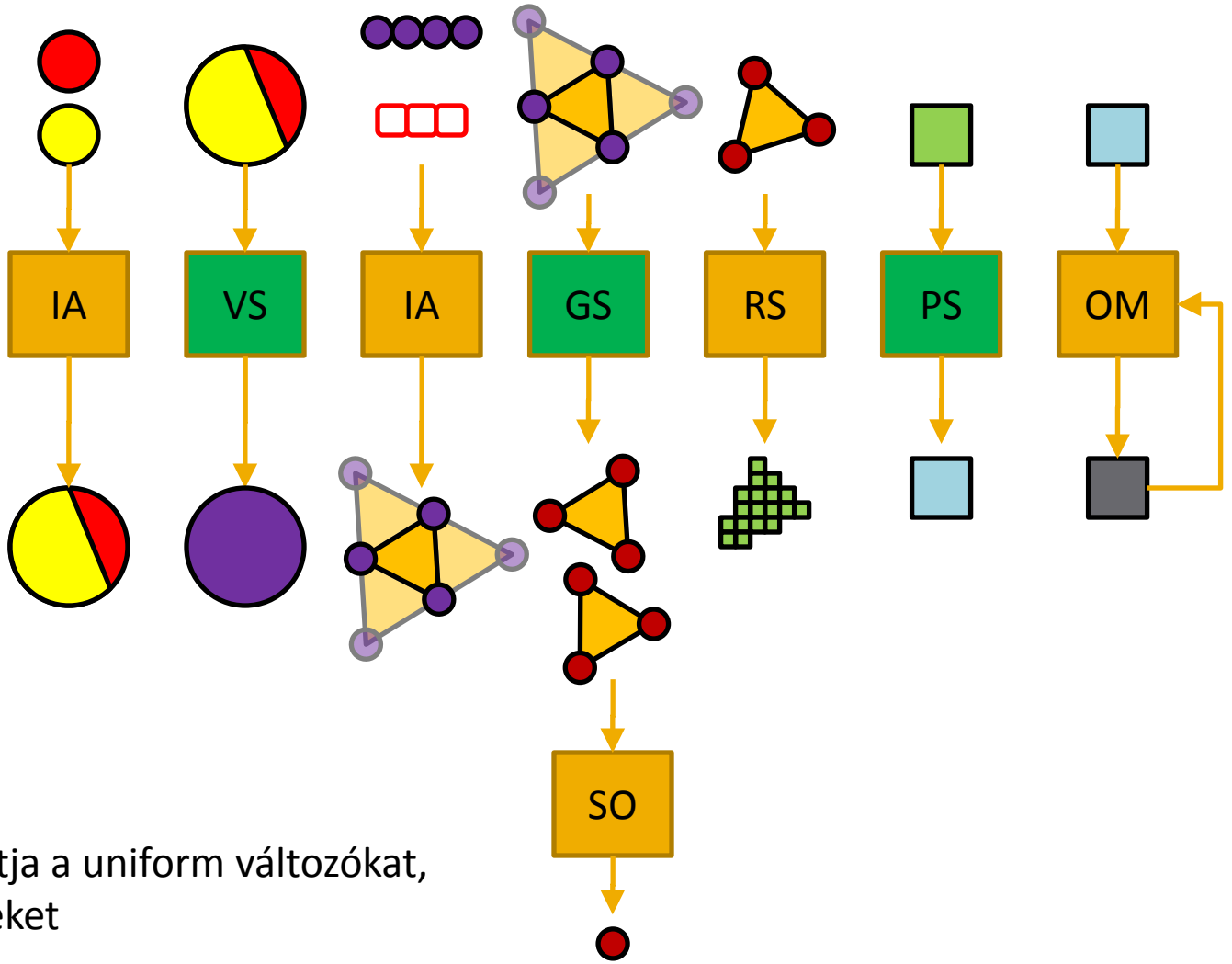
- rajzolási állapot
  - egyes pipeline-elemek működési beállításai
  - programozható elemek shader programjai
- erőforrások – globális memóriában adatok
  - globális (uniform) változók
  - textúrák
  - adatbufferek

} csak olvasható

GPU pipeline output

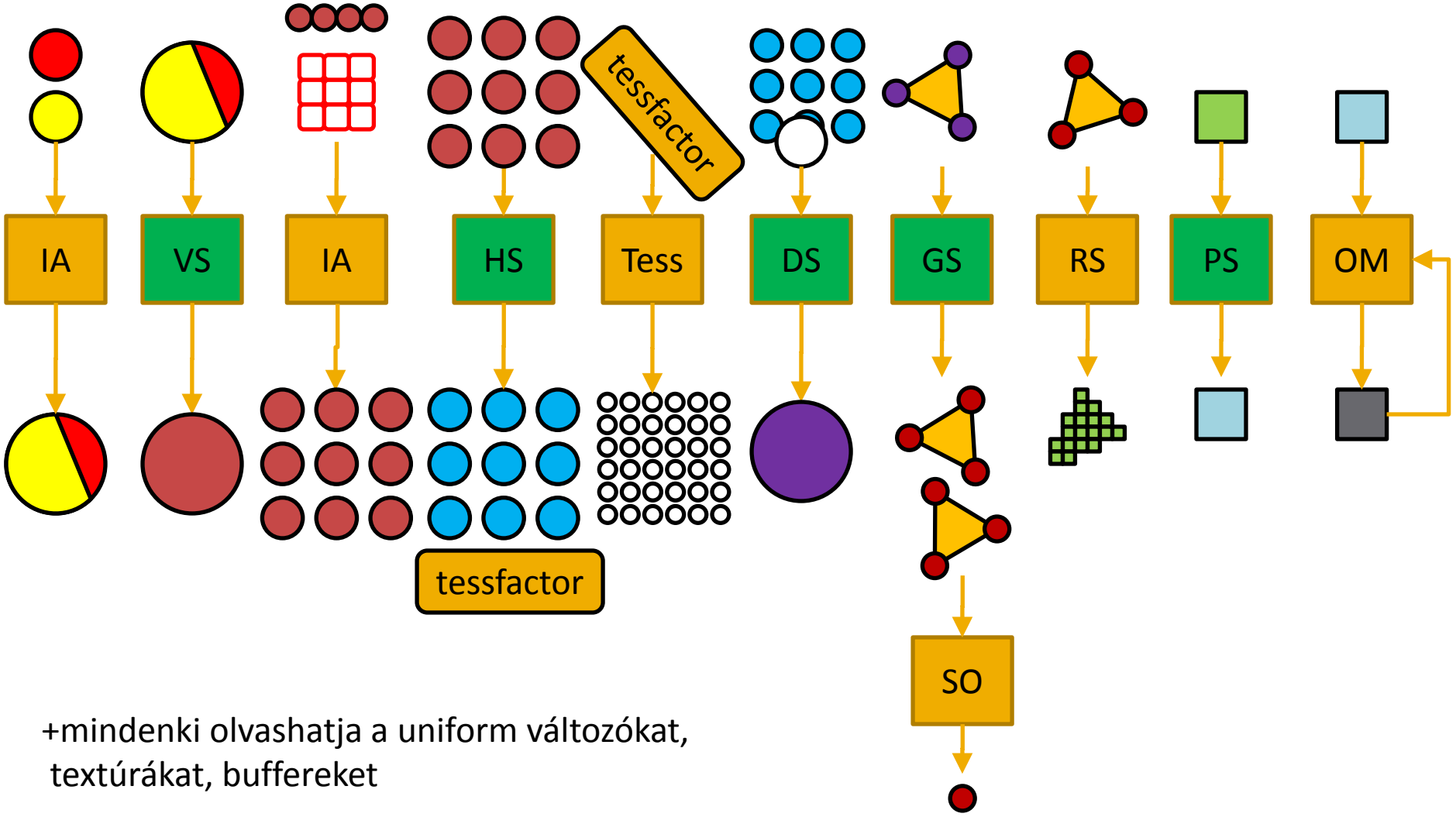
- kép
  - render target
    - frame buffer
    - textúra
  - mélységbuffer (+stencil)
- adatbuffer
  - stream out target
- read/write (GpGpu-hoz)
  - unordered access view

GPU pipeline tesszellátor nélkül



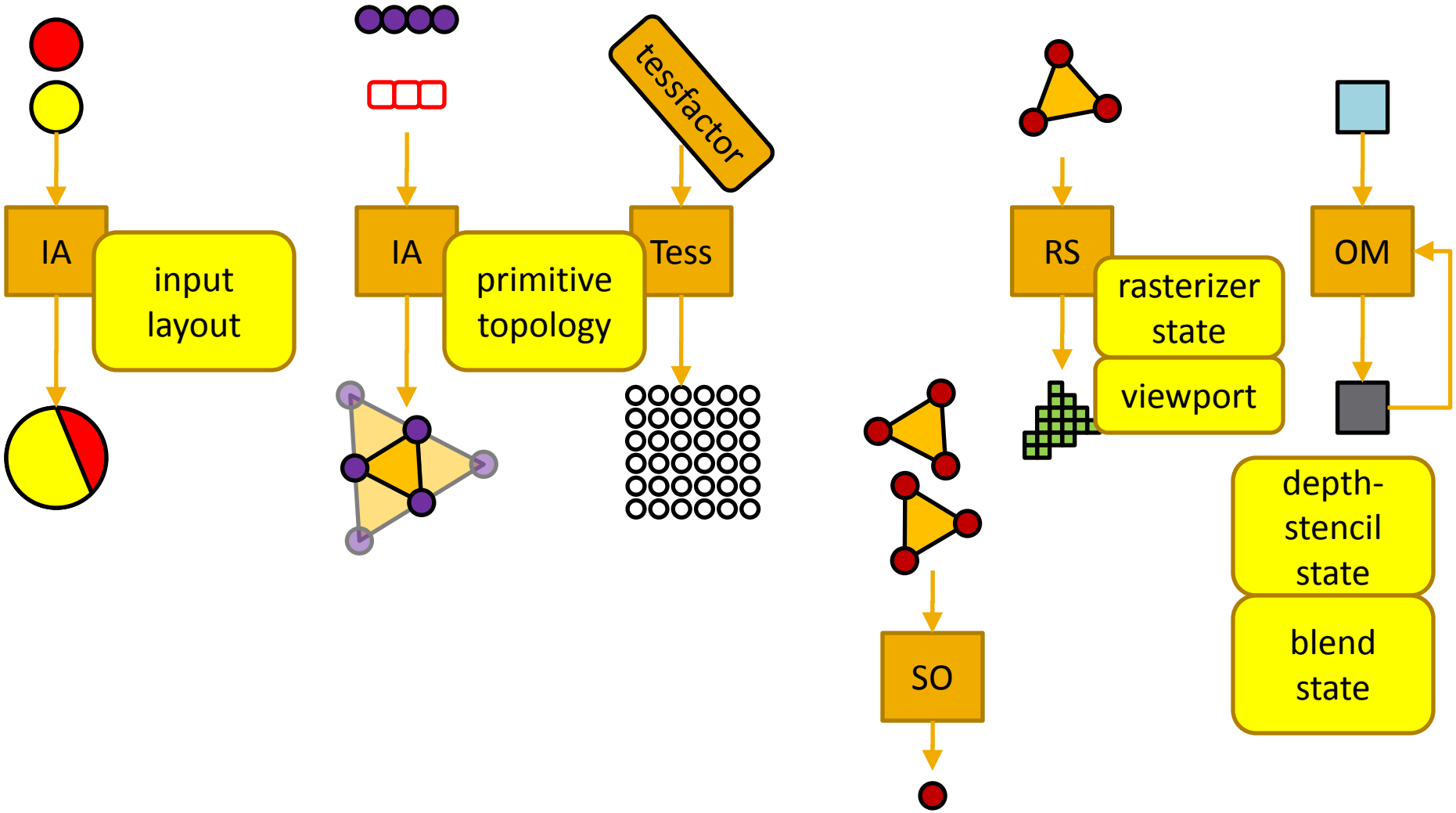
+mindenki olvashatja a uniform változókat,
textúrákat, buffereket

GPU pipeline tesszellátorral



+mindenki olvashatja a uniform változókat,
textúrákat, buffereket

Rajzolási állapot



Lépések I

- vertexek összeállítása
- Vertex Shader
  - model trafó – hol van a vertex világkoordinátában
  - árnyalás
  - view és proj trafó – hova esik a vertex a képernyőn
- primitívek összeállítása
  - vertexek összeválogatása
    - vertex bufferbeli sorrend alapján
    - index bufferbeli indexek alapján

Lépések II

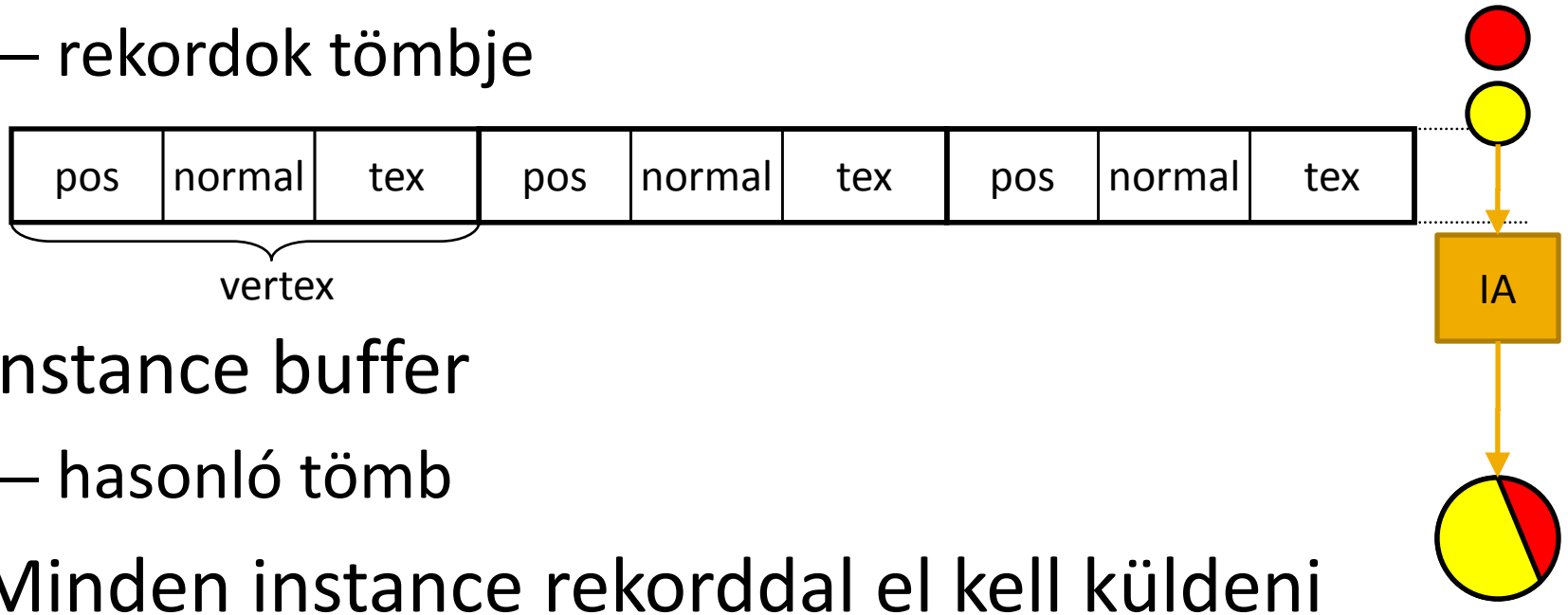
- tesszelláció
  - patch primitívekből
    - ami vezérlőpontokkal (vertexekkel) adott
  - lineáris primitívek előállítása
    - vonal, háromszög
- Geometry Shader
  - primitívek módosítása: kb fix számú és kevés kimeneti primitív
  - primitívek szűrése (kidobni is lehet)
  - vertex bufferbe írás (stream out)

Lépések III

- raszterizálás
  - hátsólap-eldobás
  - vágás
  - homogén osztás
  - viewport trafó
- Pixel shader
  - pixel szín meghatározása
- z-teszt
- blending

Vertexek összeállítása

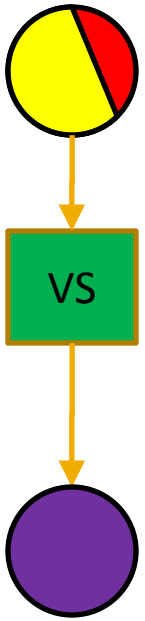
- Vertex buffer
  - rekordok tömbje



- Instance buffer
  - hasonló tömb
- Minden instance rekorddal el kell küldeni minden vertex rekordot a vertex shadernek

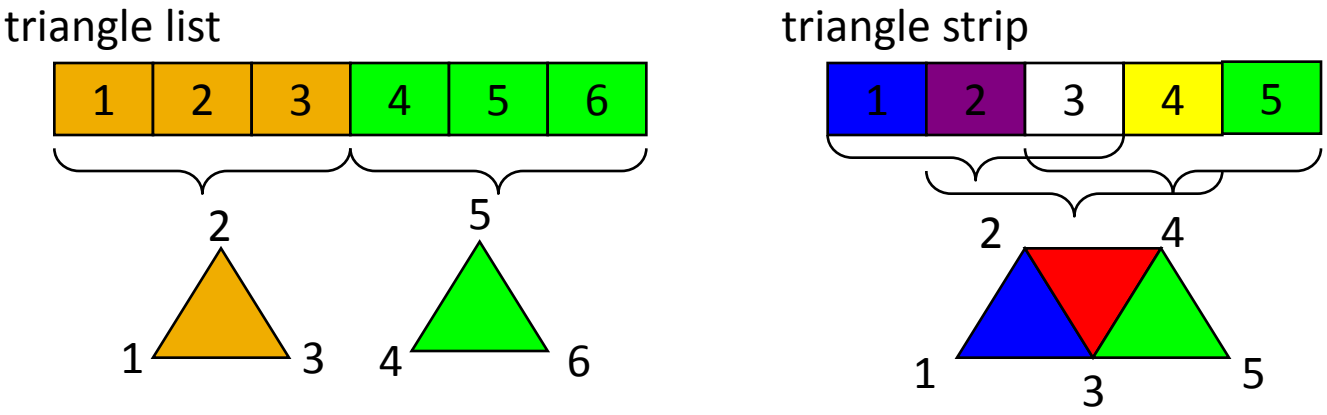
Vertex shader

- bejövő adat
  - uniform: model, view, proj mátrixok, fények, ...
  - minden vertexre más: vertex és instance elemek [pozíció, normál, szín, texcoord]
- kimenő adat
  - pozíció homogén normalizált képernyő koordinátákban
  - árnyalás eredménye [szín]
  - bármi más [normál, texcoord]



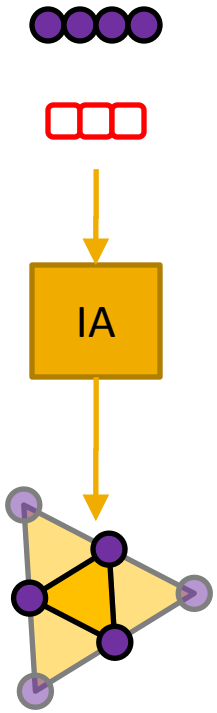
Primitívek összeállítása

- non-indexed
  - vertex bufferben egymás után következők



Indexelt primitívek

- Index buffer
  - egészek tömbje
- Indexed primitive
  - index bufferben egymás utáni következő indexekhez tartozó vertexek
- list, strip
- előny
  - kényelmes
  - nem kell strip hogy gyors legyen



Primitívtopológiák I

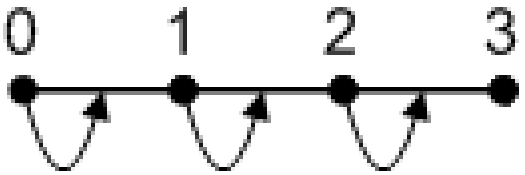
Point List

Line List

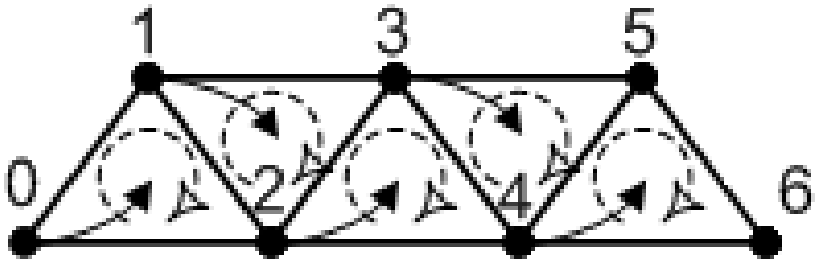
Triangle List

Primitívtopológiák II

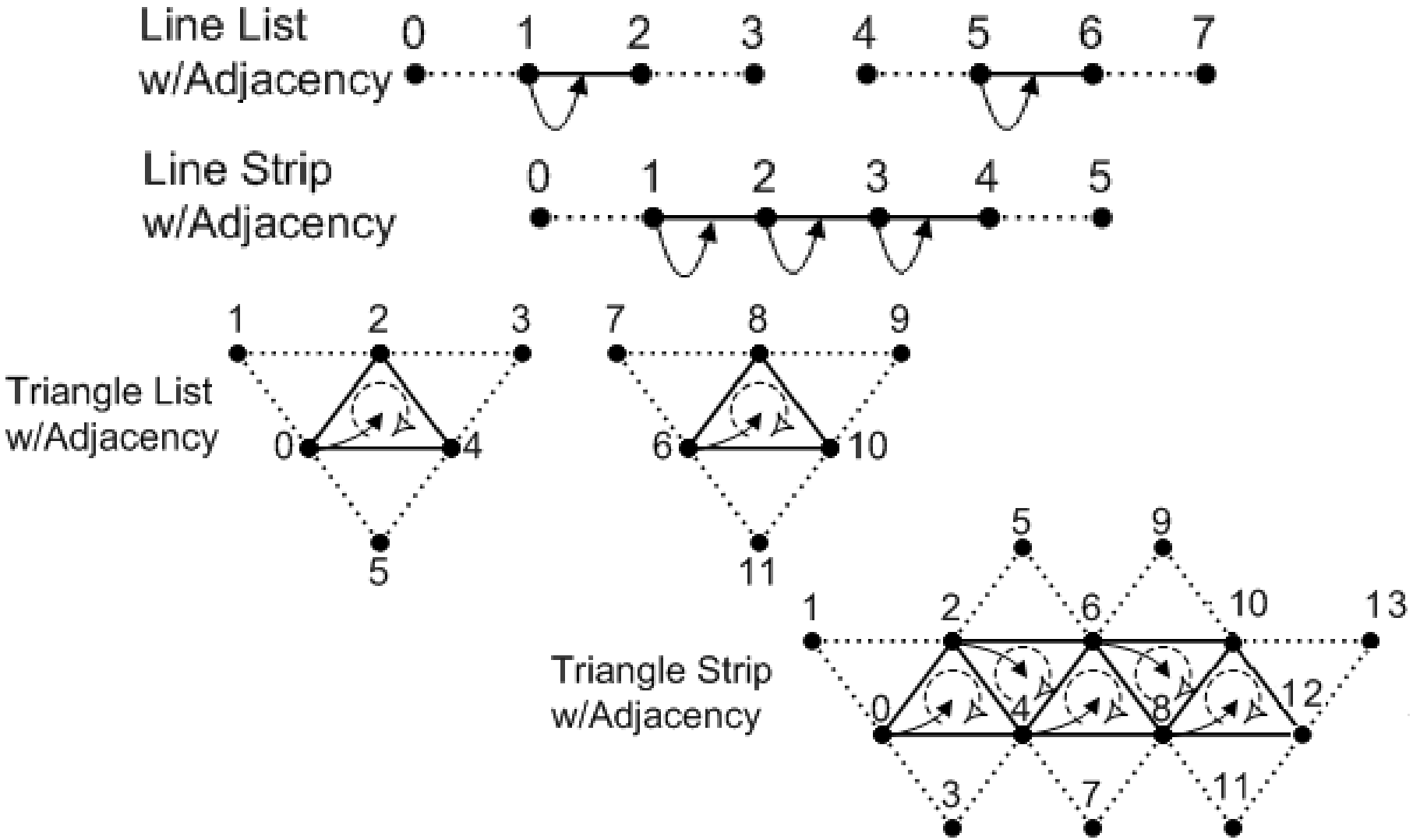
Line Strip



Triangle Strip



Primitívtopológiák III

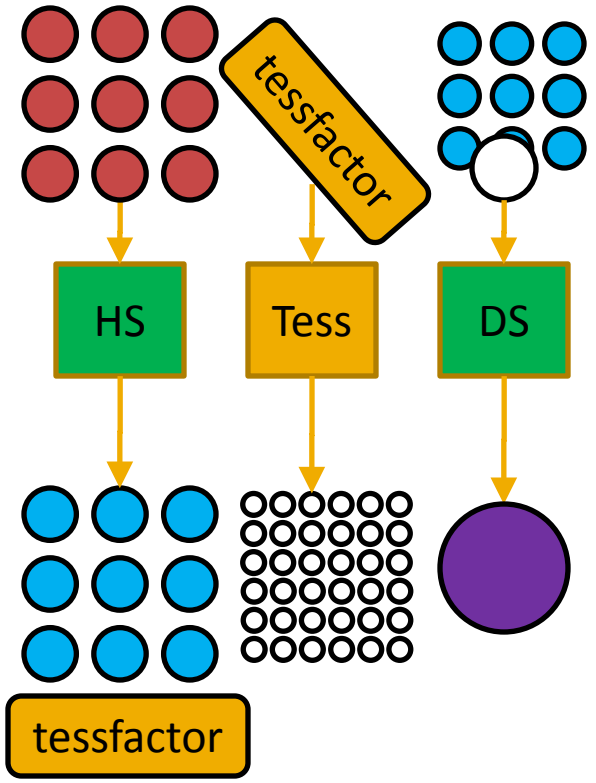


Primitívtopológiák IIII

- Tesszellátorhoz
  - 1-control-point patch list
  - …
  - 32-control-point patch list
- Olyan mint a triangle list, csak nem 3 elemű
- Domain shaderben úgy értelmezzük őket ahogy akarjuk

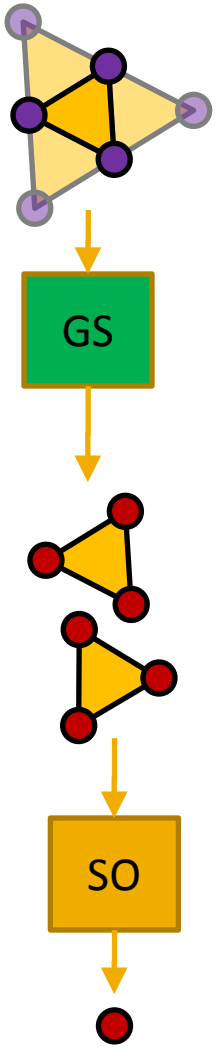
Tesszellátor

- Vertexek a vezérlőpontok
- Hull shader
  - vezérlőpontok módosítása
  - tesszellációs faktorok számítása
- Tesszellátor
  - legyárt egy háromszöghálót
- Domain shader
  - a háromszögháló vertexeinek feltöltése adatokkal
  - pl. pozíció számítása a vezérlőpontok súlyozásával



Geometry shader

- Primitívek vertextömbjét kapja
  - elemszám topológiafüggő
  - ha adjacenciát tartalmaz, akkor azokat is megkapja
    - de a tesszellátor ilyen topológiát nem tud előállítani, ha van tesszellátor akkor adjacencia nincs
- Primitívfolyamba írhat
  - max. elemszámot meg kell adni
  - lassú ha ez a szám nagy



Folyamkimenet (Stream Out)

- Geometry shader kimenete egy vertex bufferbe irányítva
- Pipeline többi része nem is kell hogy működjön
- VB később menetre köthető
- Pl. részecskerendszer-szimuláció
  - részecskék a vertexek
  - GS számítja az új pozíciójukat
  - ping-pong a két VB között

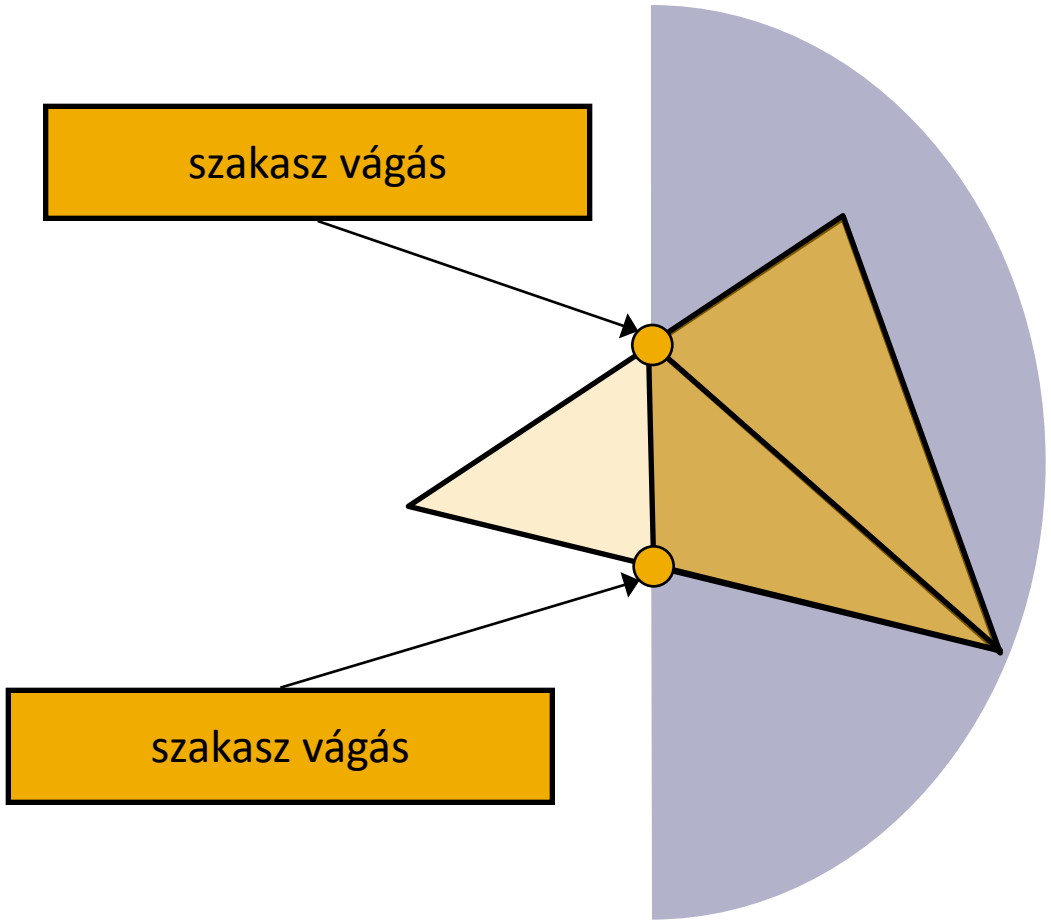
Raszterizáló egység: lapeldobás

- körüljárási irány (képernyőn) alapján eldobhatunk lapokat
- ha a modellünkben (index bufferben) konzisztens a háromszögek körüljárási iránya, ezzel eldobhatjuk a test hátsó lapjait
  - a belsejét úgysem látjuk, ha poliéder

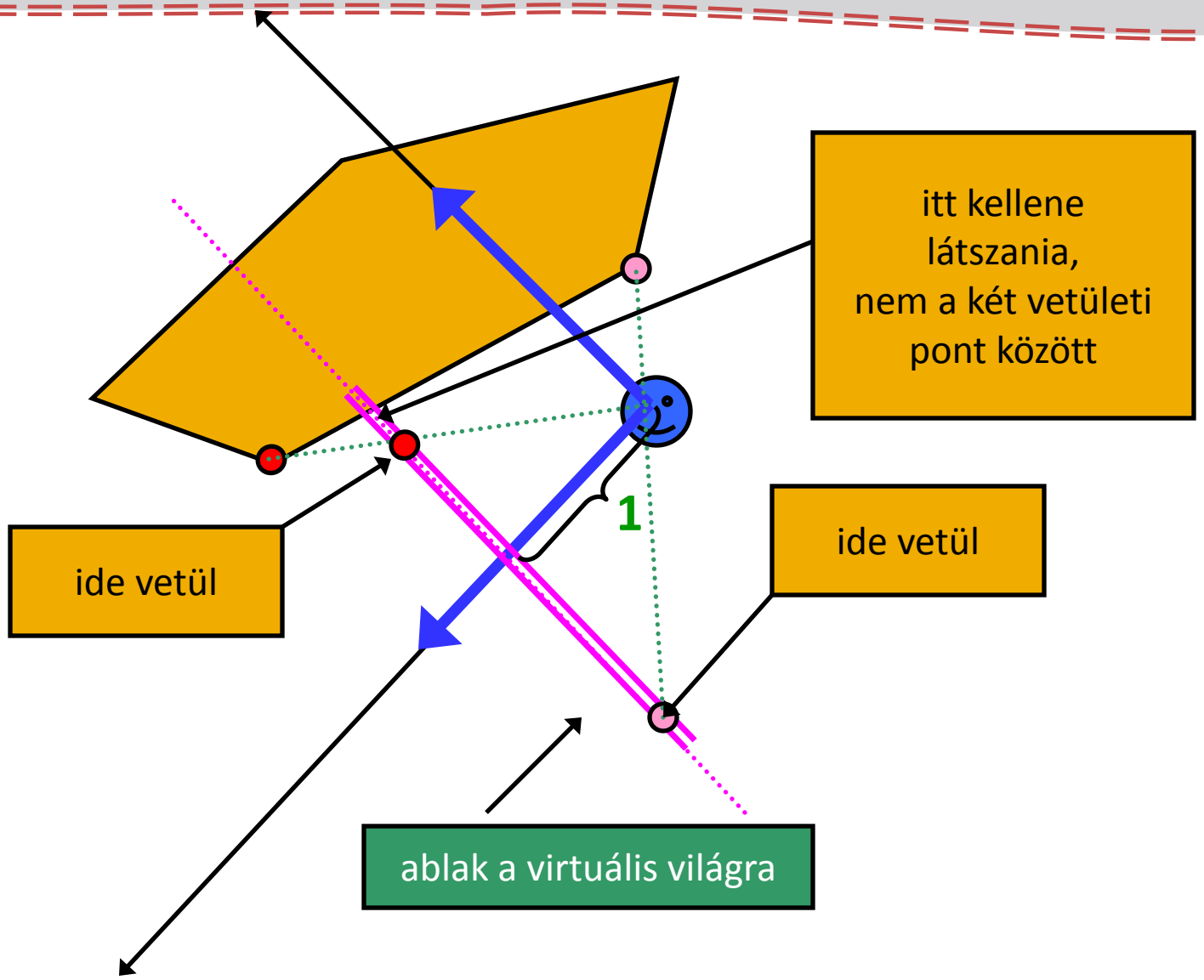
Raszterizáló egység: vágás

- ami kilógna a képernyőről, vágjuk le
- hsz vágás szakasz vágásra visszavezethető
  - szakasz vágás = pont vágás + metszéspont számítás
- normalizált képernyőkoordinátákban
  - [-1, -1, 0] [1, 1, 1] téglatestre vágunk [Descartes]
- Átfordulási probléma
  - a homogén osztás előtt kell vágni

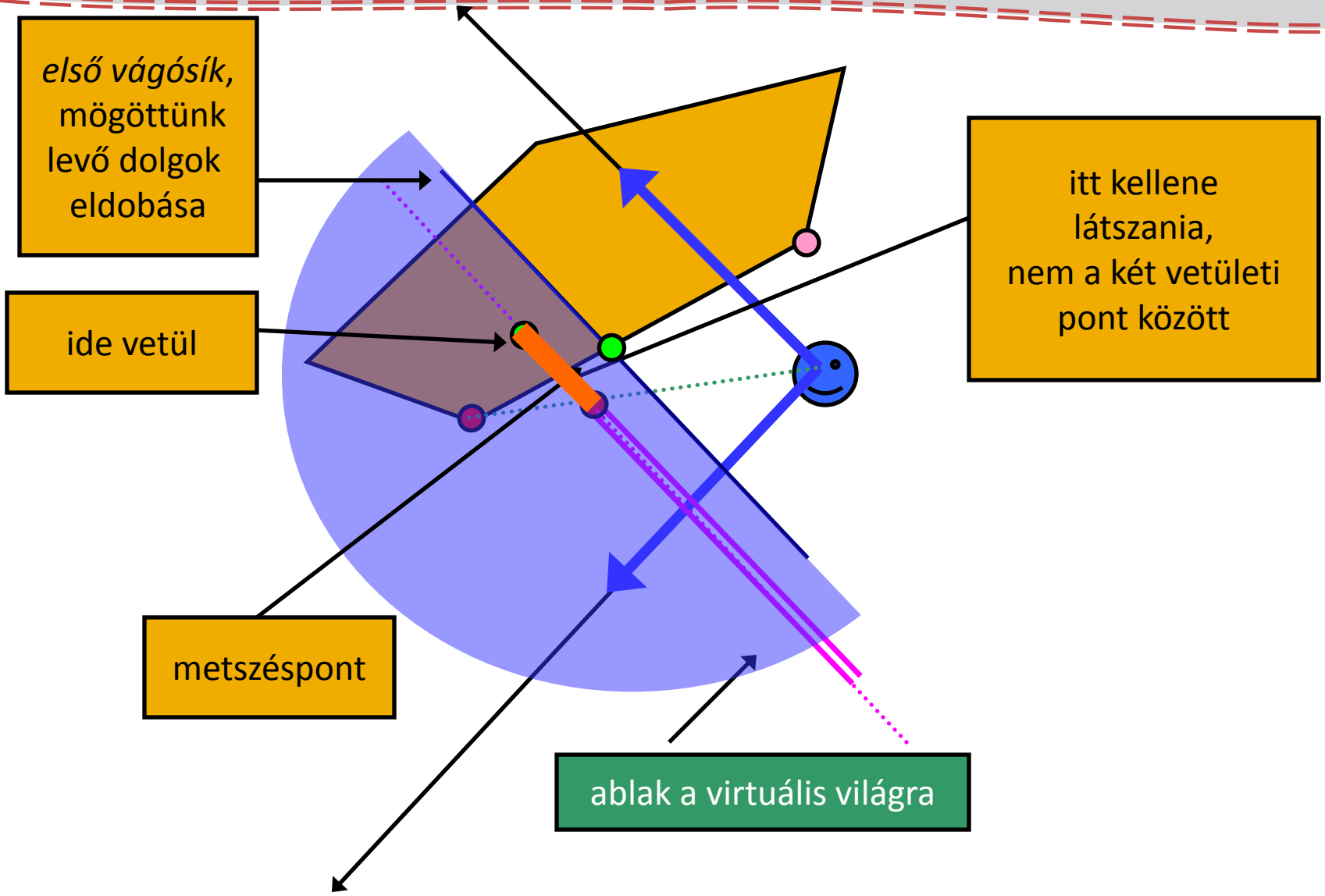
Háromszög-vágás



Átfordulási probléma



Átfordulási probléma

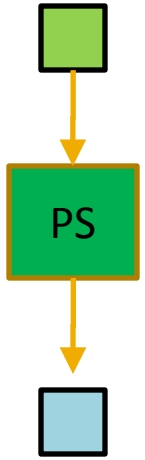


Raszterizáció

- homogén osztás
- viewport trafó: pixel koordináták
- lineáris interpoláció
  - vertex output adatok (kivéve pozíció) interpolálása minden kitöltendő pixelhez
  - pixel shader indítása minden pixelszín meghatározásához

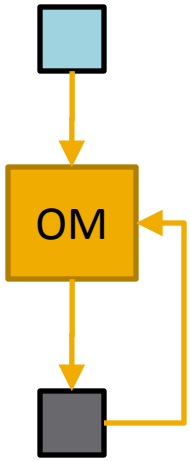
Pixel shader

- bejövő adat
  - vertex shader kimenő adatai lineárisan interpolálva
  - pixel koordináták
- kimenő adat
  - pixel szín [RGBA]
  - mélység, ha felül akarjuk írni a háromszög csúcsainak z-jéből interpolált eredetit
    - ha nem, akkor a z-teszt előrehozható a pixel shader elé



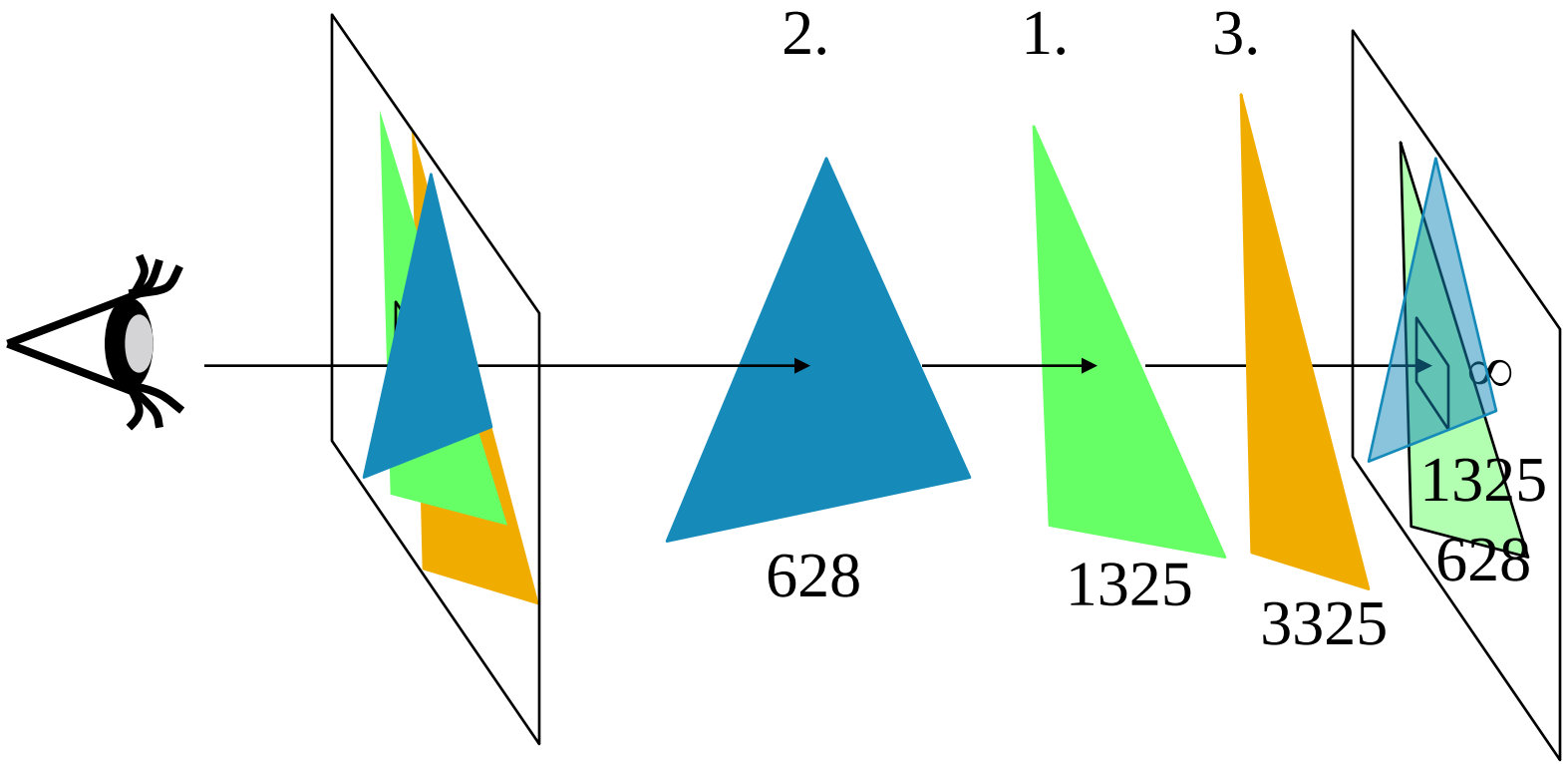
Kimeneti műveletek

- bufferek
  - célterület (render target)
    - frame buffer
    - textúra
  - mélység-stencil buffer
    - textúra
- műveletek
  - mélység teszt
  - stencil teszt
  - keverés [alfa blending]



Z-teszt

- a 3D takarási probléma megoldására



Stencil teszt

- a stencil buffer a z-buffer pár használatlan bitje
- beállítódik valamire, ha a pixelbe rajzoltunk valamit
- később feltételként szabhatjuk a pixel felülírására a stencil buffer értéket
- pl. kirajzoljuk a tükör téglalapját, utána a tükörben látható objektumokat csak a tükör pixeleire rajzoljuk

Keverés [alpha blending]

- A célterületen már meglevő érték [dest] és az újonnan számított szín [src] kombinálása
- mindkettőhöz megadható egy súly (0, 1, srcalpha, dstalpha, 1-alpha ...)
- megadható a függvény (add, subtract, min, max ...)
  - átlátszóság: hátulról előre rajzolás + blending
  - src * srcalpha + dst * (1- srcalpha)