D3D, DXUT primer

Grafikus játékok fejlesztése

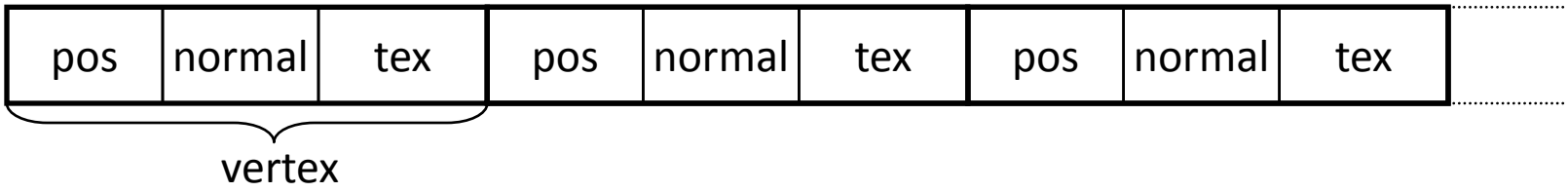
Szécsi László

2013.02.13. t01-system

Háromszögháló reprezentáció

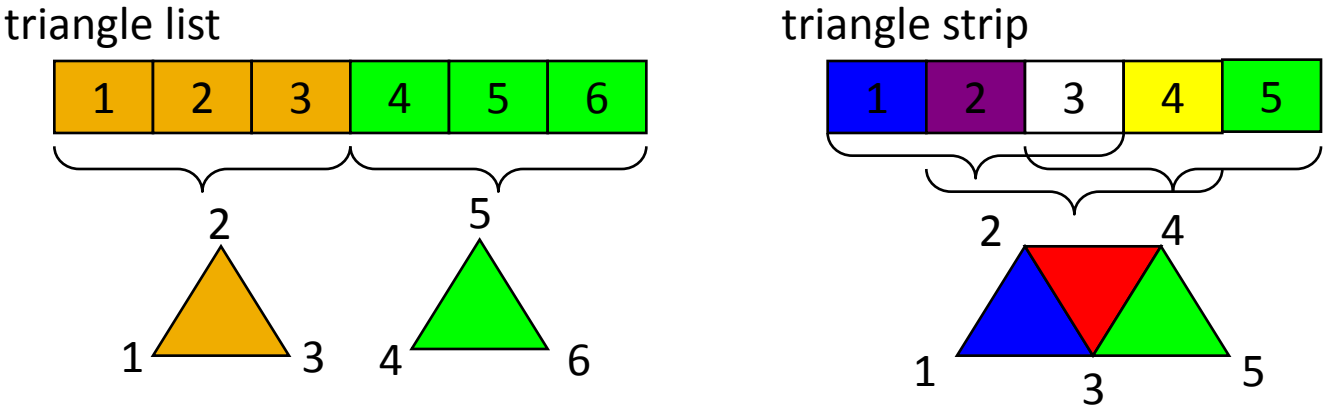
- Mesh
  - Vertex buffer
  - Index buffer

- Vertex buffer
  - csúcs-rekordok tömbje



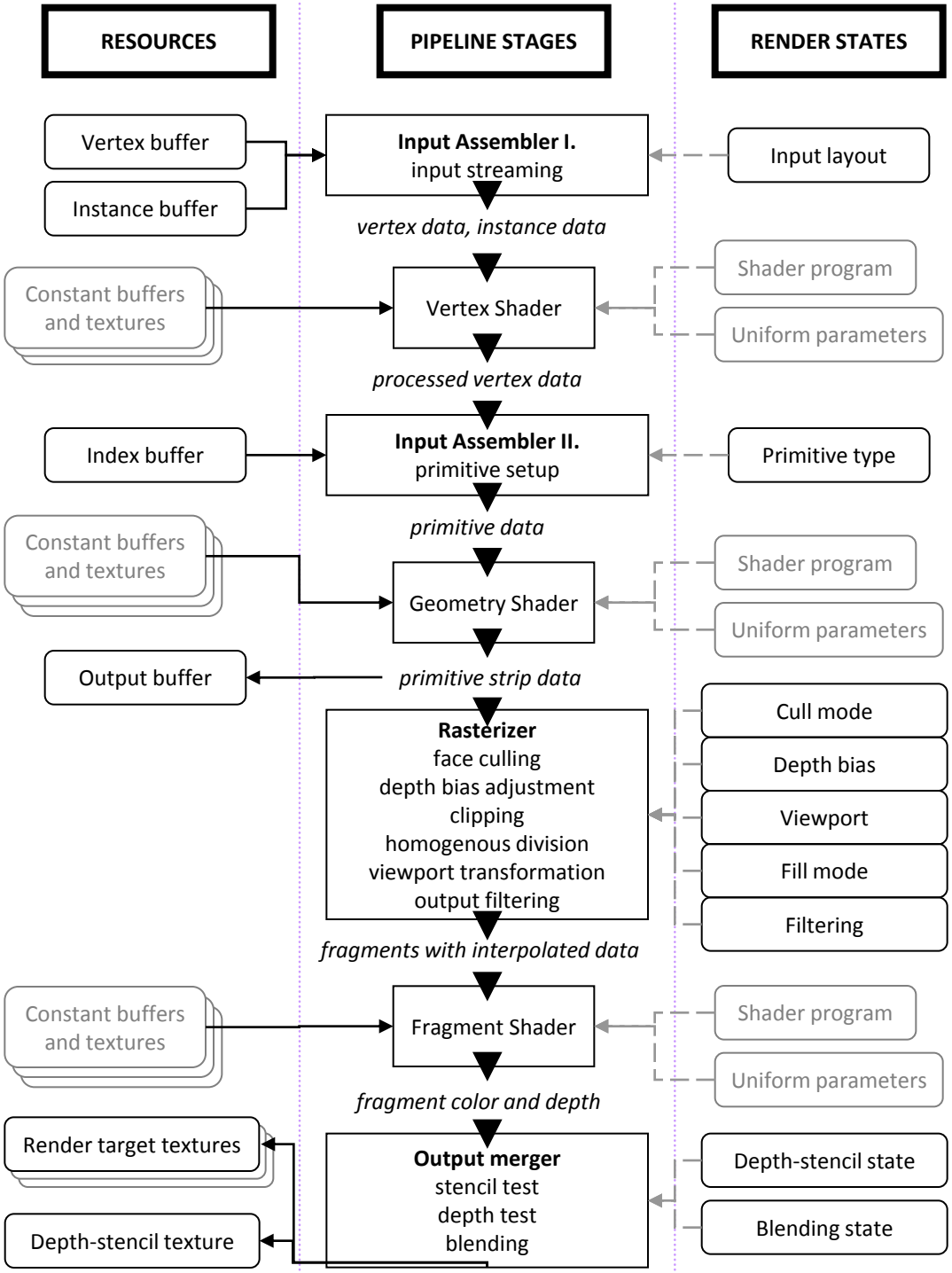
Primitívek összeállítása

- non-indexed
  - vertex bufferben egymás után következők



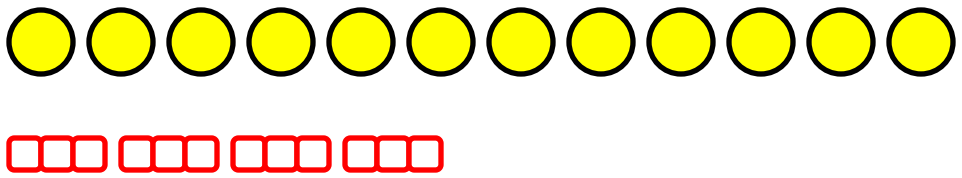
Indexelt primitívek

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


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

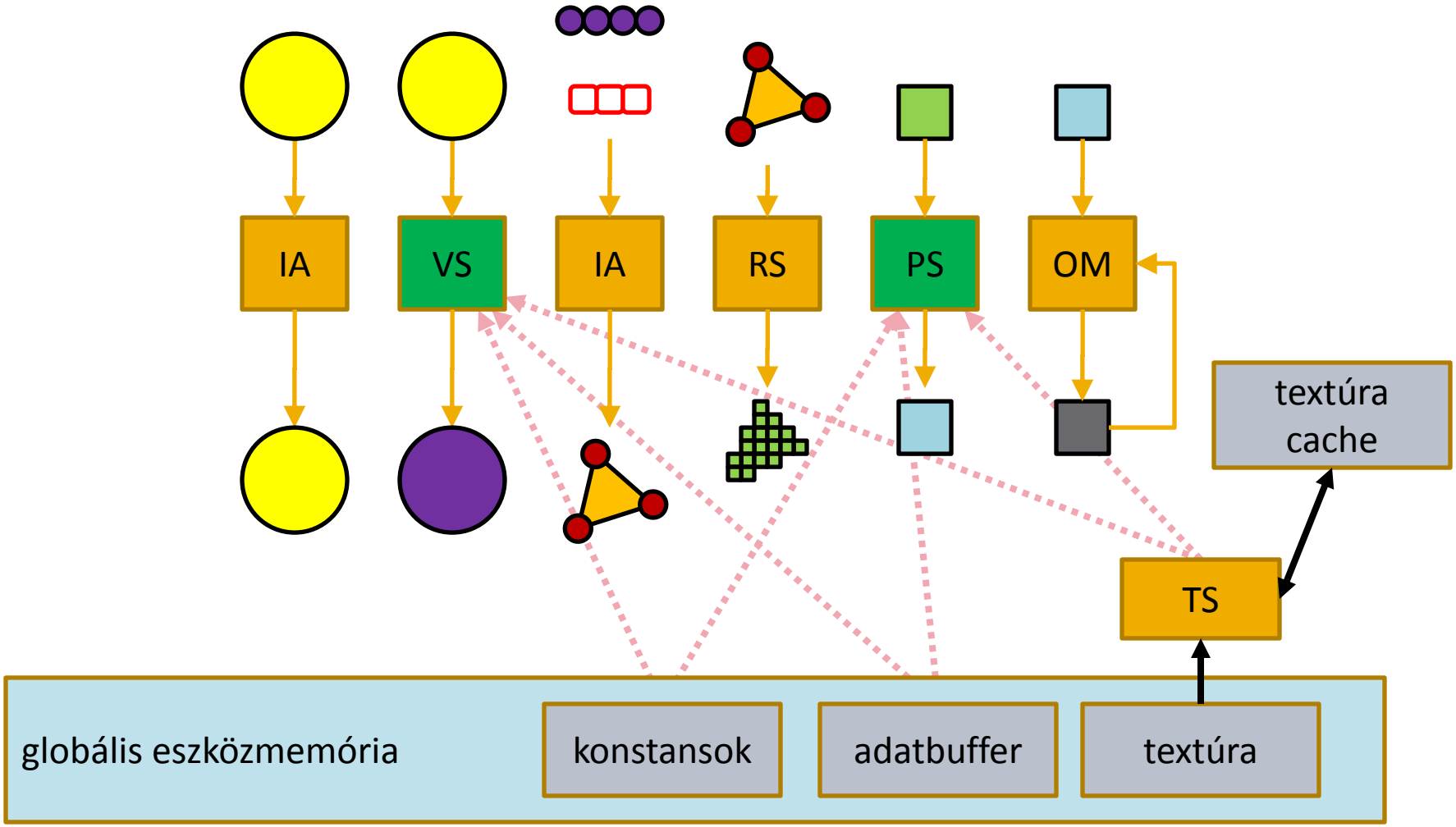


csak olvasható

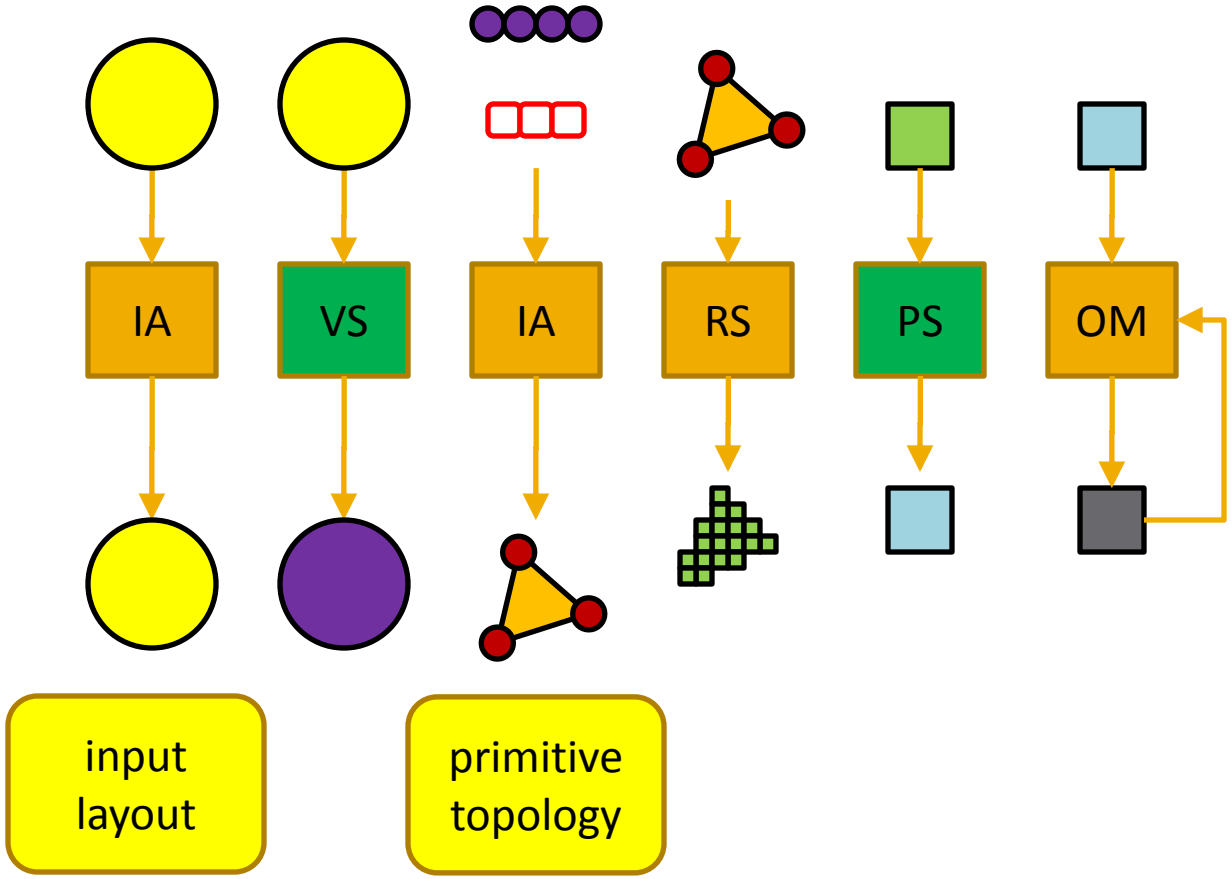
GPU pipeline output

- kép
  - render target
    - frame buffer
    - textúra
- stb…

Minimális GPU pipeline



Minimális rajzolási állapot



A pipeline vezérlése

- erőforrások allokálása
  - vertex buffer, index buffer, textúrák
- rajzolási állapot beállítása
  - ..., culling, blending, ...
- shader programok, uniform változók
  - HLSL forrás alapján
- működés indítása
  - draw call

Direct3D API - device

- eszköz [device]
  - a grafikus kártya memóriájának absztrakciója
  - erőforrások kezelésére szolgáló felület
  - ID3D11Device interface

ID3D11Device* device;

Direct3D API - context

- kontextus [context]
  - a pipelineelemek állapotának absztrakciója
  - rajzolási állapot beállítására, rajzolásra szolgáló felület
  - alapvetően egy van belőle egy devicehoz
    - immediate context
    - deferred context többszálú működéshez jó
  - ID3D11DeviceContext interface

ID3D11DeviceContext* context;

DXUT

- DirectX utility toolkit
  - olyan mint az OpenGLhez a GLUT
  - oprendszer funkciók elrejtése (ablaknyitás pl.)
  - sokkal többet tud
  - ezért bonyolultabb
- de a magja ugyanaz
  - általunk írt callback függvények regisztrálása
  - vezérlés átadása
  - eseményekkor meghívja a mi callback függvényeinket

Eszköz-események

- létrejött [CreateDevice]
  - program indulásakor
- új ablak [ResizedSwapChain]
  - induláskor, új oprendszer-összerendeléskor (pl. átméretezés után)
- régi ablak eltűnik [ReleasingSwapChain]
  - hiba esetén, az oprendszer-felület változásakor (pl. átméretezéskor)
- megszűnt [DestroyDevice]

Erőforrás-gazdálkodás

- create, resized
  - létrehozás
- destroy, releasing
  - felszabadítás
- típusok
  - vertex, index buffer — ID3D11Buffer
  - textúra — ID3D11Texture2D
    - cube texture, 3D texture

Erőforrásszerű állapotcsomagok

- ezeket is a device metódusaivla hozzuk létre és a context metódusaival választjuk ki
- input layout ID3D11InputLayout
  - VB elemek és VS inputok összrendelése
- vertex shader ID3D11VertexShader
  - program
- pixel shader ID3D11PixelShader
  - program
- RasterizerState, BlendState, DepthStencilState

Erőforrás-kezelési módok (usage)

- immutable
  - létrehozáskor inicializálható, utána csak olvasható
- dynamic
  - rátölthetünk új adatot akár minden frameben
- default
  - nem tölthetünk rá, de szerepelhet a pipelineban mint kimenet (pl. render-to-texture)
- staging
  - CPU memóriában van, átmásolható bele egy default erőforrás tartalma ha olvasni akarjuk

Melyik usage mikor kell?

- immutable
  - betöltött, fix dolgok, modellek, textúrák
- dynamic
  - mozgó dolgok, részecskerendszerek
- default
  - GPU outputként is szereplő elemek
- staging
  - vészhelyzetben

Erőforrások kötési módjai (bind)

- vertex buffer
  - pipeline bemenet
- index buffer
  - pipeline bemenet
- shader resource
  - shader olvashat belőle
- render target
  - kép kerülhet bele
- …

Vertex buffer létrehozása

```
D3D11_BUFFER_DESC desc;
desc.Usage = D3D11_USAGE_IMMUTABLE;
desc.BindFlags = D3D11_BIND_VERTEX_BUFFER;
desc.ByteWidth = sizeof(D3DXVECTOR3) * 3;
desc.StructureByteStride = sizeof(D3DXVECTOR3);

D3DXVECTOR3 vertexPositionArray[3] = {
    D3DXVECTOR3(0, 0, 0.5),
    D3DXVECTOR3(0, 1, 0.5),
    D3DXVECTOR3(1, 0, 0.5) };
D3D11_SUBRESOURCE_DATA initData;
initData.pSysMem = vertexPositionArray;

pd3dDevice->CreateBuffer(
                &desc, &initData, &vertexBuffer);
```

Vertex Shader létrehozása

```
const char* vertexShaderCode =
  "float4 vsIdle(float4 pos :POSITION )
  :SV_Position {return pos;}";
ID3DBlob* vertexShaderByteCode;
D3DX11CompileFromMemory(vertexShaderCode,
  strlen(vertexShaderCode), NULL, NULL, NULL,
  "vsIdle", "vs_5_0", 0, 0, NULL,
  &vertexShaderByteCode, NULL, NULL);

pd3dDevice->CreateVertexShader(
  vertexShaderByteCode->GetBufferPointer(),
  vertexShaderByteCode->GetBufferSize(), NULL,
  &vertexShader);
```

Input layout létrehozása

```
D3D11_INPUT_ELEMENT_DESC positionElement;
positionElement.AlignedByteOffset = 0;
positionElement.Format =
  DXGI_FORMAT_R32G32B32_FLOAT;
positionElement.InputSlot = 0;
positionElement.InputSlotClass =
  D3D11_INPUT_PER_VERTEX_DATA;
positionElement.SemanticName = "POSITION";
positionElement.SemanticIndex = 0;
pd3dDevice->CreateInputLayout(
      &positionElement, 1,
      vertexShaderByteCode->GetBufferPointer(),
      vertexShaderByteCode->GetBufferSize(),
      &inputLayout);
```

Erőforrások felszabadítása

- COM objektumok
- referenciaszámlált
- createValami növeli a referenciszámlálót
- ha végeztünk vele
  - valami->Release();
  - csökkenti a referenciaszámlálót
  - ha máshol sem kell már fel lesz szabadítva

Rajzolás

- OnFrameRender esemény
- context metódusainak hívásával
- be kell állítani
  - render target
  - render state
  - shaderek, uniform paraméterek
  - textúrák
  - vertex, index buffer
- draw call

Rajzolás

ekkorát kell lépni a következő vertexhez

```
unsigned int stride = sizeof(D3DXVECTOR3);
unsigned int offset = 0;
context->IASetVertexBuffers(0, 1,
      &vertexBuffer, &stride, &offset);
context->IASetPrimitiveTopology(
      D3D11_PRIMITIVE_TOPOLOGY_TRIANGLELIST);
context->IASetInputLayout(inputLayout);


context->VSSetShader(vertexShader, NULL, 0);
context->PSSetShader(pixelShader,  NULL, 0);

context->Draw(3, 0);
```

innen kezdve

1 db VB

elemek értelmezése

ennyi vertexet

innen kezdve