SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Seminarski rad iz kolegija:
Računalna Grafika

Tema:
HLSL

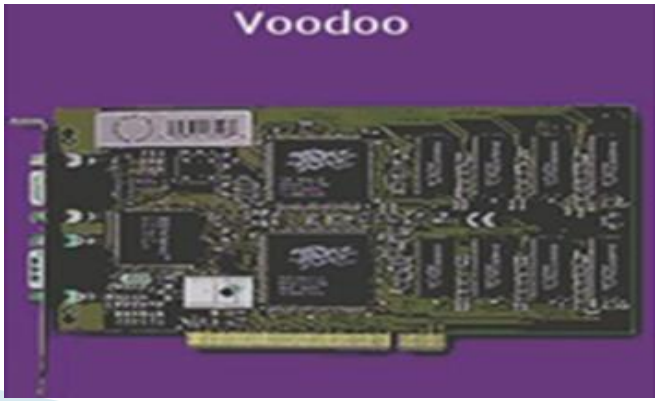
Bernard Brček Bedeković,
39587/10-R

U Varaždinu, 23.1.2012.

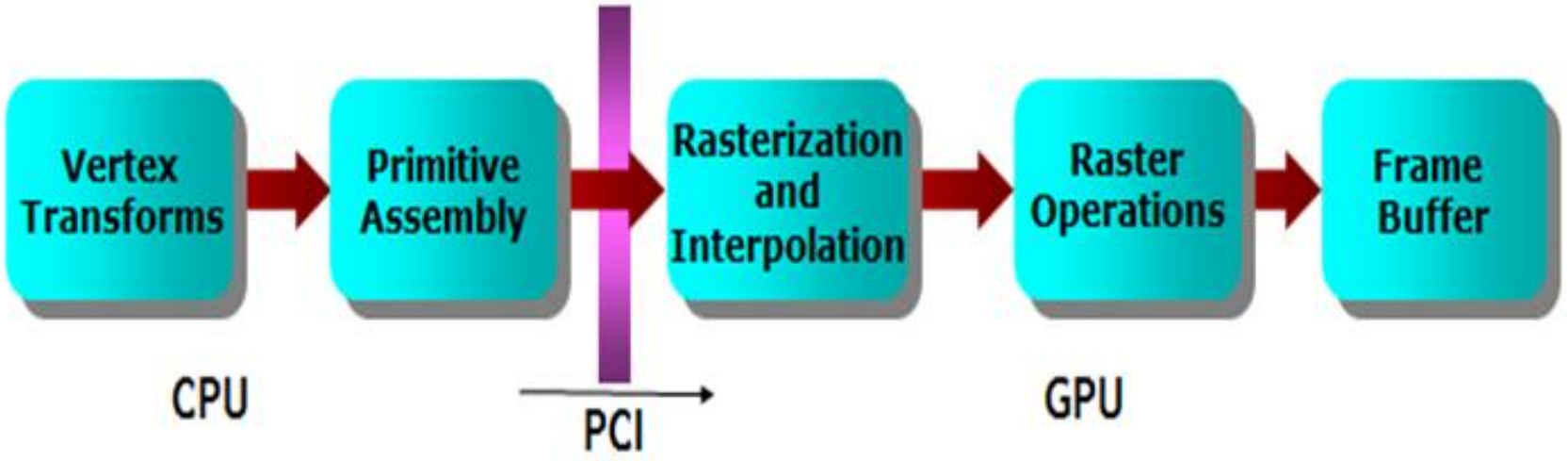
SADRŽAJ:

Uvod (1 / 9)

- 1. generacija grafičkih kartica sa 3D mogućnostima – 1996. godina
  - Nisu imale mogućnosti za transformaciju verteksa
  - Imale su mogućnosti za mapiranje tekstura i spremnik dubine
  - 3dfx Voodoo



Uvod (2/9)

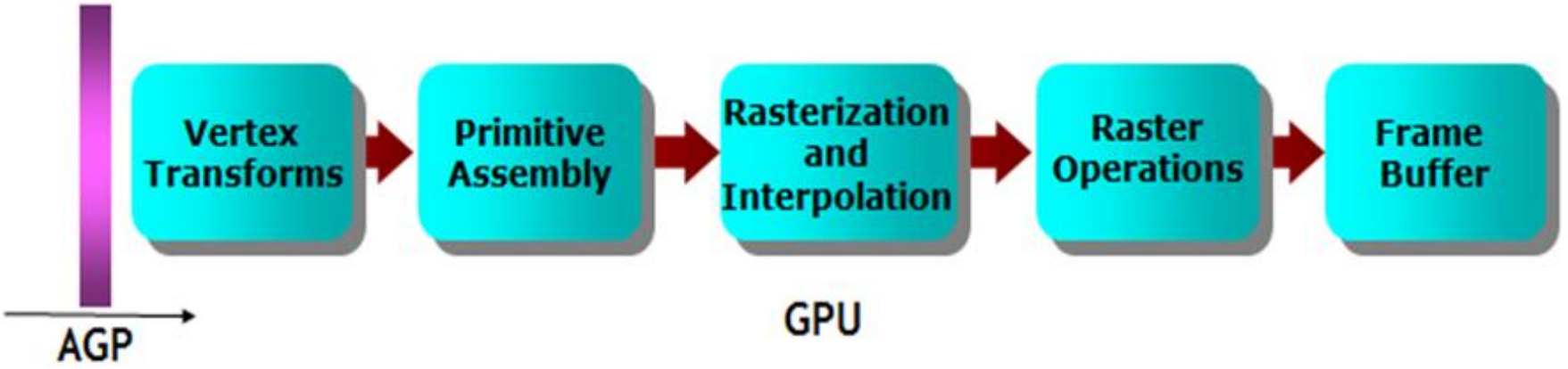

Vertex Transforms
Primitive Assembly
Rasterization and Interpolation
Raster Operations
Frame Buffer
CPU
PCI
GPU

Uvod (3/9)

- 2. generacija grafičkih kartica sa 3D mogućnostima (1998. godina)
  - Omogućuju transformacije verteksa i kalkulacije svijetla
  - Omogućuju miješanje tekstura
  - GeForce 256

GeForce 256

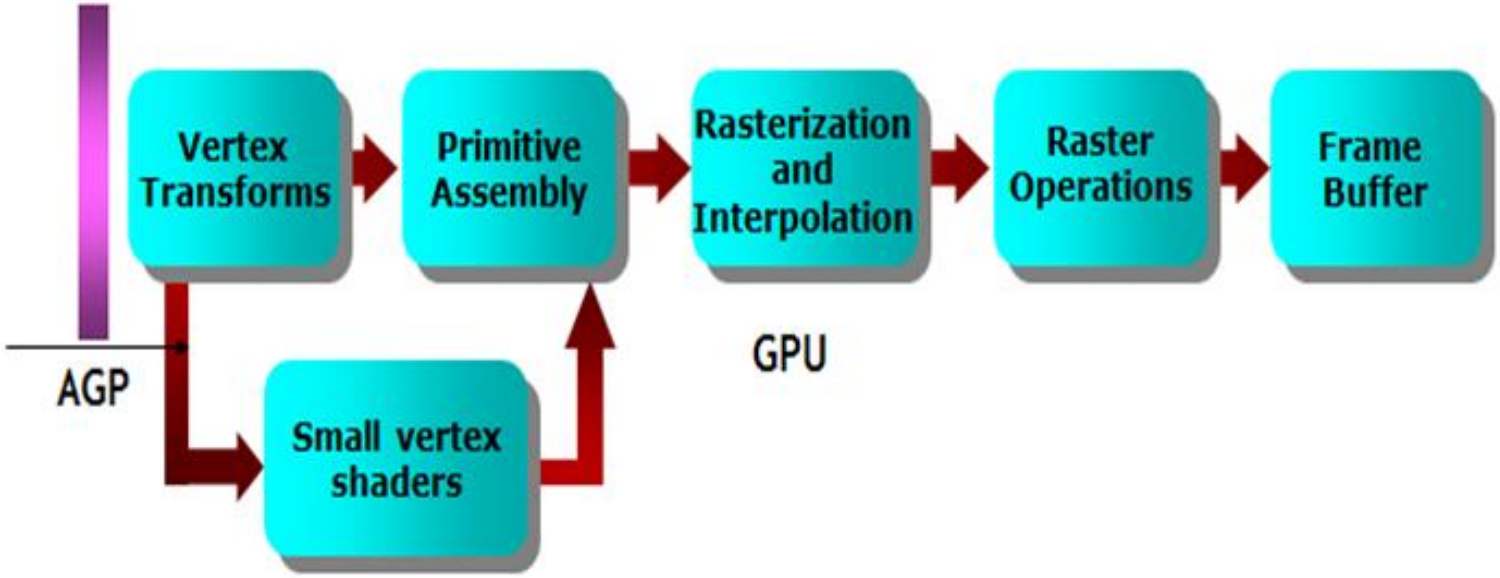
Uvod (4/9)


Vertex Transforms
Primitive Assembly
Rasterization and Interpolation
Raster Operations
Frame Buffer
AGP
GPU

Uvod (5/9)

- 3. generacija grafičkih kartica sa 3D mogućnostima (2001. godina)
  - Pružaju limitirane mogućnosti programiranja verteks cjevovoda
  - GeForce3

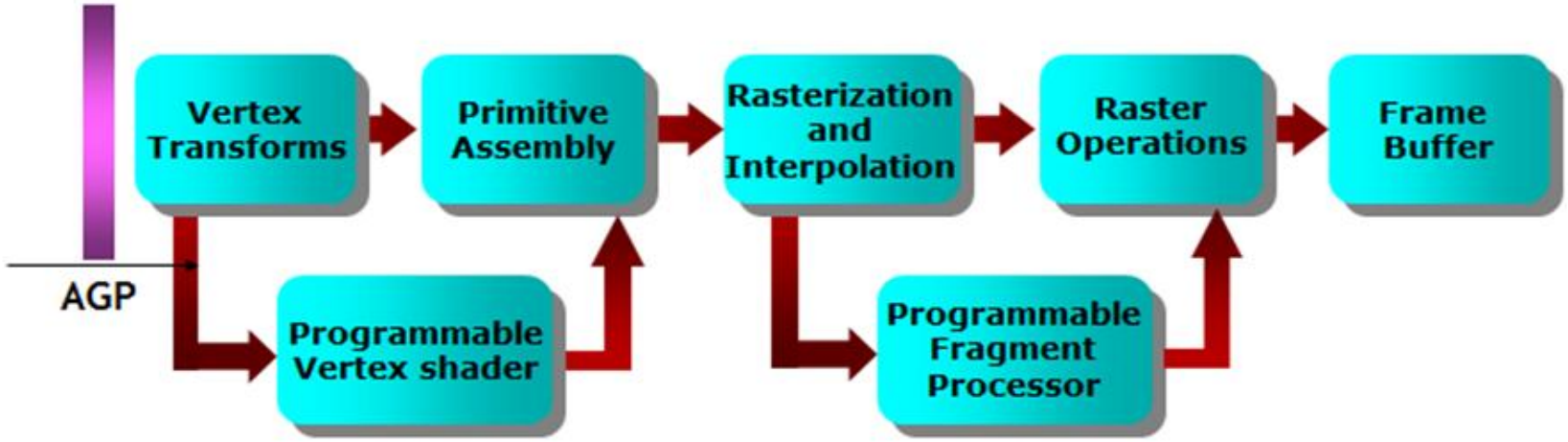
Uvod (6/9)


Vertex Transforms
Primitive Assembly
Rasterization and Interpolation
Raster Operations
Frame Buffer
AGP
Small vertex shaders
GPU

Uvod (7/9)

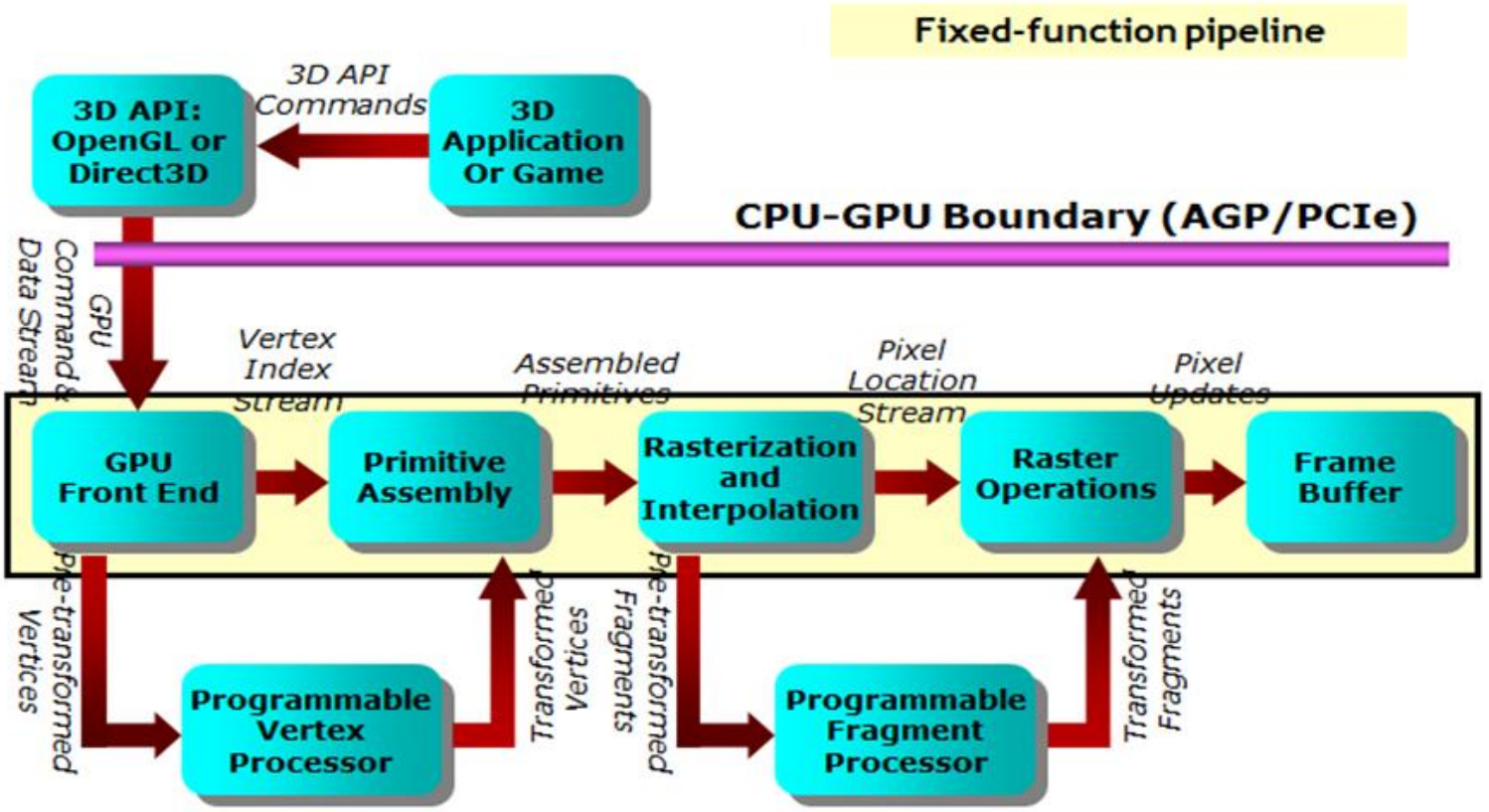
- 4. generacija grafičkih kartica sa 3D mogućnostima (2002. godina)
  - 1. generacija koja je se u potpunosti može programirati
  - GeForce FX

Uvod (8/9)


AGP
Vertex Transforms
Primitive Assembly
Rasterization and Interpolation
Raster Operations
Frame Buffer
Programmable Vertex shader
Programmable Fragment Processor

Uvod (9/9)

- Fixed-function pipeine VS. Programmable pipeline



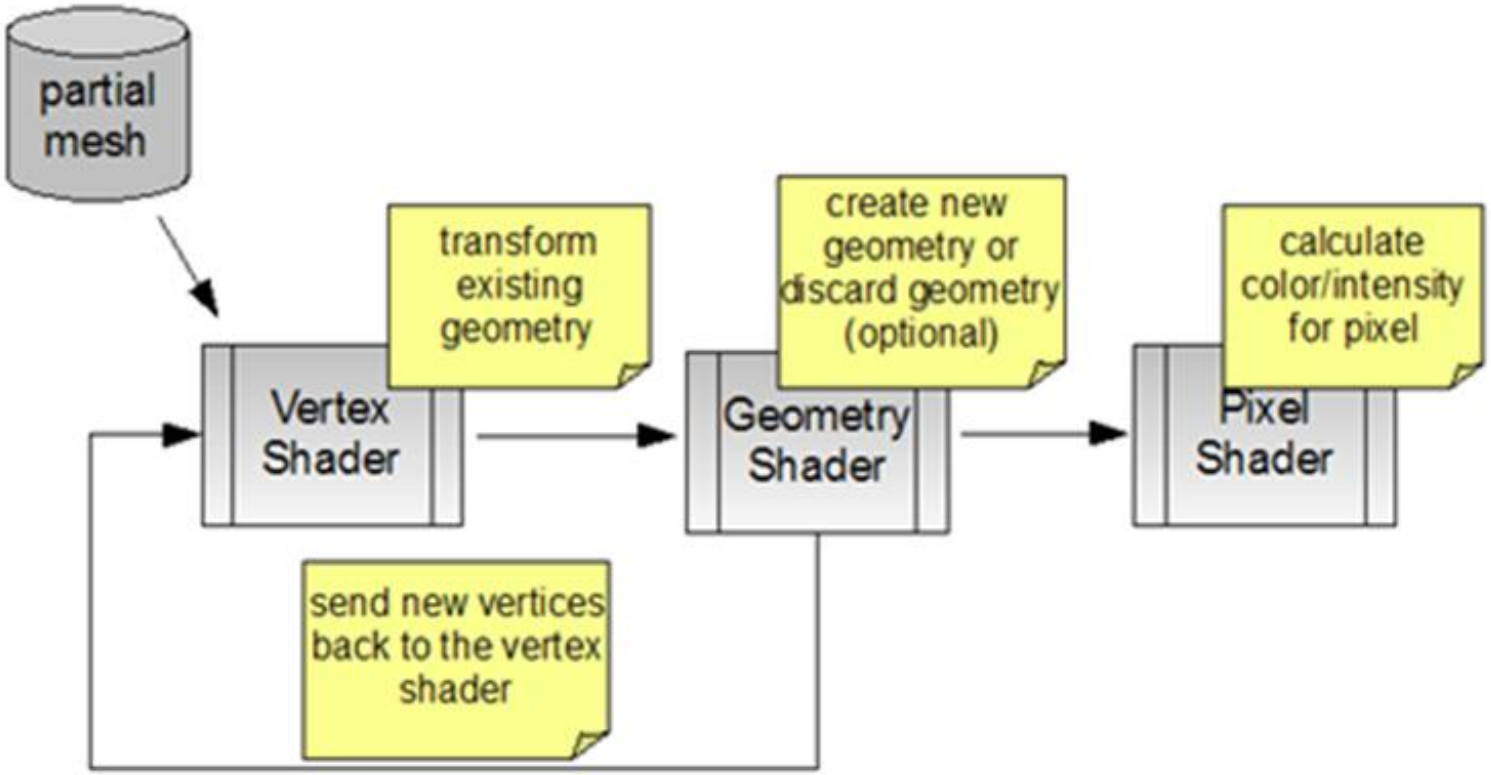
HLSL

- Shader
  - mali računalni program koji služi za programiranje programibilnog grafičkog cjevovoda
  - Određuje kako su 3D objekti iscrtani na ekran i na koji način svijetla utječu na objekte
- Postojeći jezici:
  - Stanford rea-time shading language
  - HLSL – DirectX
  - GLSL – OpenGL
  - Cg – NVIDIA (DirectX i OpenGL)

Vrste

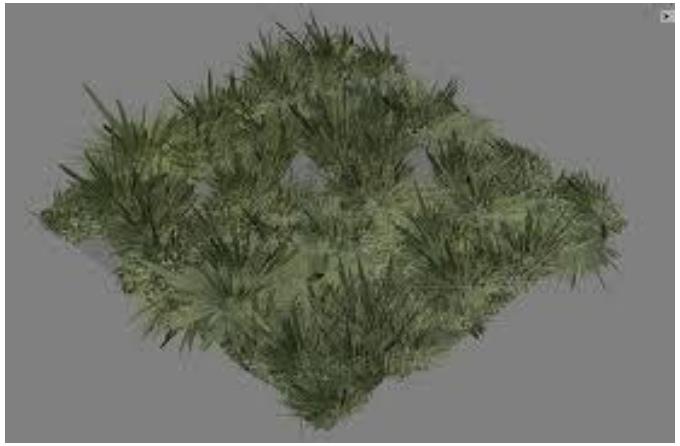
- Geometry shader
  - kreiranje nove geometrije
- Vertex shader
  - transformacija verteksa 3D objekta
  - kalkulacija svijetla
- Pixel shader
  - izračunavanje boje piksela na ekranu

Geometry shader (1 / 2)



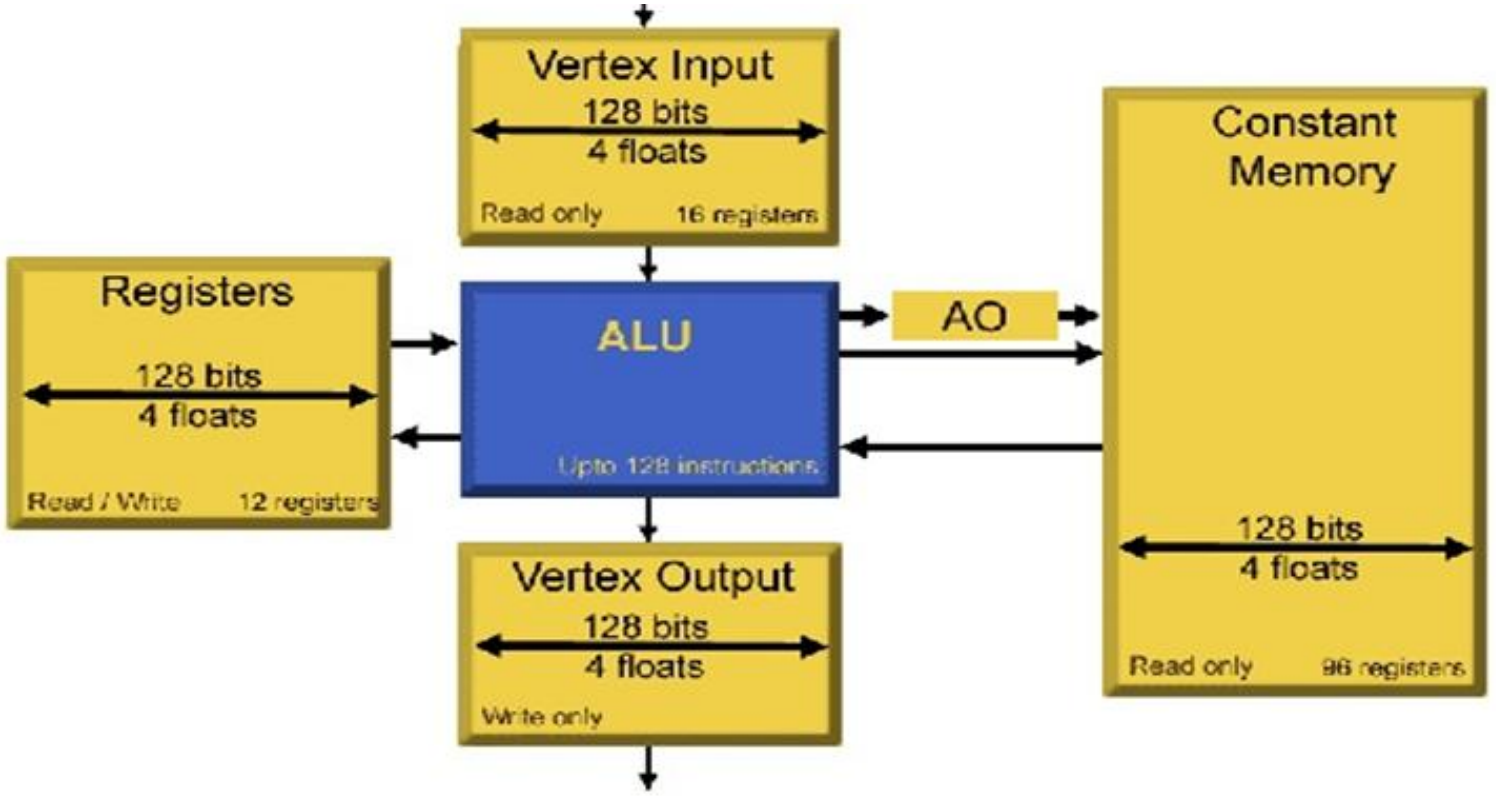
Geometry shader (2/2)

- Primjene
  - Generiranje trave na terenu
  - Slapovi
  - Krzno



Vertex shader

- Arhitektura



Verzije

Vertex shader version	VS 1.1[6]	VS 2.0[3][6]	VS 2.0a[3][6]	VS 3.0[4][6]	VS 4.0[5]
# of instruction slots	128	256	256	≥ 512	4096
Max # of instructions executed	Unknown	65536	65536	65536	65536
Instruction predication	No	No	Yes	Yes	Yes
Temp registers	12	12	13	32	4096
# constant registers	≥ 96	≥ 256	≥ 256	≥ 256	16x4096
Static Flow Control	???	Yes	Yes	Yes	Yes
Dynamic Flow Control	No	No	Yes	Yes	Yes
Dynamic Flow Control Depth	No	No	24	24	Yes
Vertex Texture Fetch	No	No	No	Yes	Yes
# of texture samplers	N/A	N/A	N/A	4	128
Geometry instancing support	No	No	No	Yes	Yes
Bitwise Operators	No	No	No	No	Yes
Native Integers	No	No	No	No	Yes

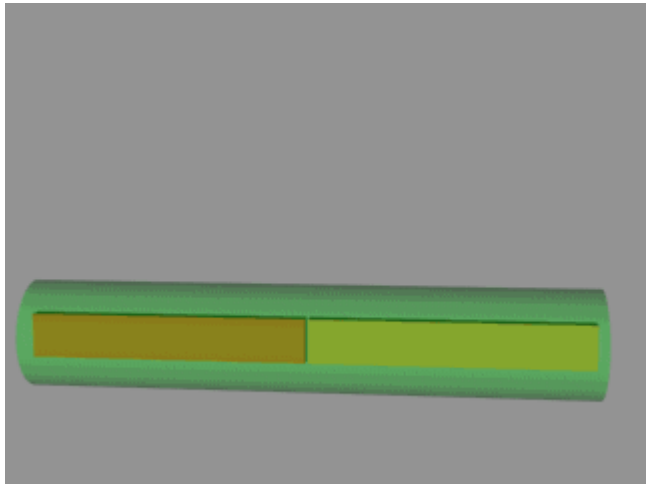
Glavna funkcija

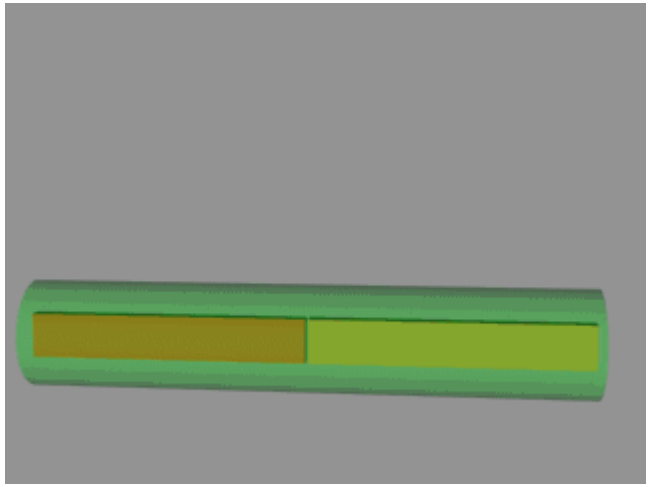
- World View Projection transformacija
- World
  - transformira 3D objekt sa inicijalne pozicije na poziciju koja je definirana translacijom, rotacijom i skalom
- View
  - transformira vertekse na poziciju koja je relativna u odnosu na poziciju kamere
- Projection
  - perspektivna ili ortogonalna projekcija

Primjene (1 /2)

- Skinning – tehnika koja se koristi za animiranje 3D likova
- Vertex blending
- Izračunavanje micanja biljaka pod utjecajem vjetra

Primjene (2/2)





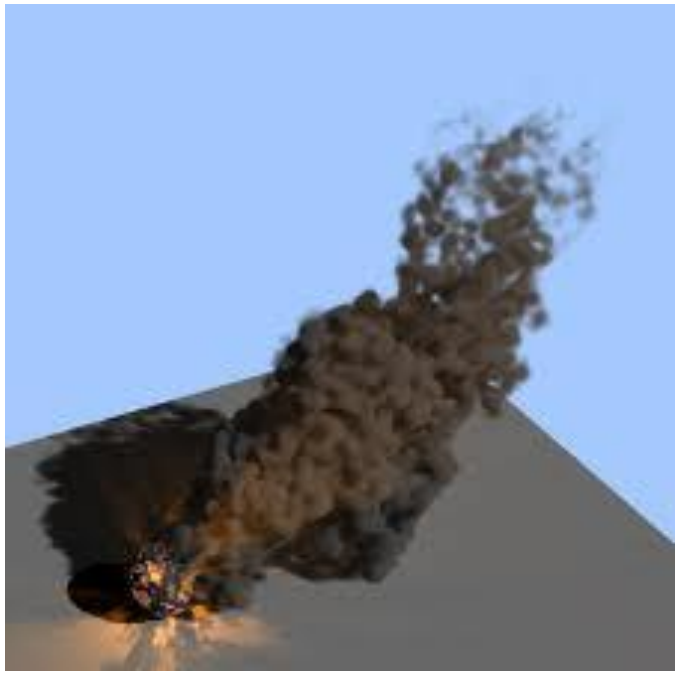
- Nakon verteks shader-a sa verteksima se događaju slijedeće operacije:
- Clipping
- Backface cull
- Perspective division (Homogenous divide)
- Viewport transform
- Rasterizer

Piksel shader

Pixel shader version	1.0 to 1.3[2]	1.4[2]	2.0[2][3]	2.0a[2][3]	2.0b[2][3]	3.0[2][4]	4.0[5]
Dependent texture limit	4	6	8	Unlimited	8	Unlimited	Unlimited
Texture instruction limit	4	6*2	32	Unlimited	Unlimited	Unlimited	Unlimited
Position register	No	No	No	No	No	Yes	Yes
Instruction slots	8+4	8+4	32 + 64	512	512	≥ 512	≥ 65536
Executed instructions	8+4	6*2+8*2	32 + 64	512	512	65536	Unlimited
Texture indirections	4	4	4	Unlimited	4	Unlimited	Unlimited
Interpolated registers	2 + 8	2 + 8	2 + 8	2 + 8	2 + 8	10	32
Instruction predication	No	No	No	Yes	No	Yes	No
Index input registers	No	No	No	No	No	Yes	Yes
Temp registers	2	6	12 to 32	22	32	32	4096
Constant registers	8	8	32	32	32	224	16x4096
Arbitrary swizzling	No	No	No	Yes	No	Yes	Yes
Gradient instructions	No	No	No	Yes	No	Yes	Yes
Loop count register	No	No	No	No	No	Yes	Yes
Face register (2-sided lighting)	No	No	No	No	No	Yes	Yes
Dynamic flow control	No	No	No	No	No	24	Yes
Bitwise Operators	No	No	No	No	No	No	Yes
Native Integers	No	No	No	No	No	No	Yes

Primjene

- Jednostavno teksturiranje
- Cell shading
- Normal mapping
- Čestice
- Alpha blending



Verzije

- DirectX 8
  - Shader Model 1 – jezik sličan asembleru
- DirectX 9
  - Shader Model 2
  - Shader model 3
- DirectX 10
  - Shader Model 4
- DirectX 11
  - Shader Model 5

Struktura (1 / 5)

- Kod se nalazi unutar .fx datoteke
- Datoteka se sastoji od slijedećih dijelova
  - Deklaracija varijabli
  - Verteks input struktura
  - Vertex output struktura
  - Vertex shader
  - Pixel shader
  - Tehnike
    - Prolazi

Struktura (2/5)

- Podržani tipovi podataka:
  - Buffer
  - Scalar
    - Bool, int, uint, half, float, double
  - Vector
    - Npr. float3 ili vector <float, 3>
  - Matrix
    - Npr. Int1x1, double2x2, float3x3 ili matrix <float,2,2>
  - Sampler, Shader, Texture
  - Struct
  - User defined
    - Npr. typedef vector <int, #> int#;

Struktura (3/5)

```
float var1;       // Variable Declaration
float var2;
...

struct VSInput    // Structure defining what data your vertex shader needs passed into it
{
};
struct VSOutput   // Structure defining what data you vertex shader outputs to the pixel shader.
{
};

// The vertex shader
VSOutput VertexShader( VSInput inputData )
{
}
// The pixel shader
float4 PixelShader( VSOutput inputData )
{
}

// The technique gives you a way of selecting which shaders you want to use and in
// what combination in case you have more in one file. This technique will expand
// to include several different passes referencing different vertex and pixel shaders
// when you get into more advanced shaders.
technique Diffuse
{
   pass Pass0
   {
      VertexShader = compile vs_1_1 VertexShader();
      PixelShader = compile ps_1_1 PixelShader();
   }
}
```

Struktura (4/5)

```
struct VS_INPUT
{
    float4 vPosition : POSITION;
    float3 vNormal : NORMAL;
    float4 vBlendWeights : BLENDWEIGHT;
};

struct VS_OUTPUT
{
    float4  vPosition : POSITION;
    float4  vDiffuse : COLOR;

};
```

Struktura (5/5)

```
VS_OUTPUT VS_Skinning_Example(const VS_INPUT v, uniform float len=100)
{
    VS_OUTPUT out;

    // Skin position (to world space)
    float3 vPosition =
        mul(v.vPosition, (float4x3) mWld1) * v.vBlendWeights.x +
        mul(v.vPosition, (float4x3) mWld2) * v.vBlendWeights.y +
        mul(v.vPosition, (float4x3) mWld3) * v.vBlendWeights.z +
        mul(v.vPosition, (float4x3) mWld4) * v.vBlendWeights.w;
    // Skin normal (to world space)
    float3 vNormal =
        mul(v.vNormal, (float3x3) mWld1) * v.vBlendWeights.x +
        mul(v.vNormal, (float3x3) mWld2) * v.vBlendWeights.y +
        mul(v.vNormal, (float3x3) mWld3) * v.vBlendWeights.z +
        mul(v.vNormal, (float3x3) mWld4) * v.vBlendWeights.w;

    // Output stuff
    out.vPosition    = mul(float4(vPosition + vNormal * Len, 1), mTot);
    out.vDiffuse  = dot(vLight,vNormal);

    return out;
}
```

Semantika (1/2)

- Služi za povezivanje izlaza iz jedne faze i ulaza u drugu fazu unutar shader programa
- Podržane vertex shader semantike
  - Input
    - BINORMAL[n], BLENDINDICES[n], BLENDWEIGHT[n], COLOR[n], NORMAL[n], POSITION[n], TANGENT[n], TEXCOORD[n], POSITIONT, PSIZE[n]
  - Output
    - COLOR[n], FOG, POSITION[n], PSIZE, TESSFACTOR[n], TEXCOORD[n]

Semantika (2/2)

- Podržane pixel shader semantike
- Input
  - COLOR[n], TEXCOORD[n], VFACE, VPOS
- Output
  - COLOR[n], DEPTH[n]

Funkcije

- Abs, acos, atan, atan2, ceil, clamp, cos, dot, lerp, min, max itd.
- Cijeli popis na http://msdn.microsoft.com/en-us/library/ff471376%28v=vs.85%29.aspx

Literatura (1 /4)

- http://www.cis.upenn.edu/~suvenkat/700/lectures/2/Lecture2.ppt
- http://knol.google.com/k/hlsl-shaders#
- http://digitseven.com/shadersintro.aspx
- http://developer.amd.com/media/gpu_assets/ShaderX2_IntroductionToHLSL.pdf
- http://msdn.microsoft.com/en-us/library/bb509647%28v=vs.85%29.aspx#PS

Literatura (2/4)

- http://msdn.microsoft.com/en-us/library/windows/desktop/bb944006%28v=vs.85%29.aspx
- http://xengine.sourceforge.net/pdf/Programmable%20Graphics%20Pipeline%20Architectures.pdf
- http://www.neatware.com/lbstudio/web/hlsl.html

Literatura (3/4)

- http://www.toymaker.info/Games/html/effects_files.html
- http://developer.amd.com/media/gpu_assets/FixedFuncShader.pdf
- http://www.flipcode.com/archives/Geometry_Skinning_Blending_and_Vertex_Lighting-Using_Programmable_Vertex_Shaders_and_DirectX_80.shtml
- http://developer.amd.com/documentation/articles/pages/7112007172.aspx

Literatura (4/4)

- http://developer.download.nvidia.com/shaderlibrary/webpages/shader_library.html
- http://msdn.microsoft.com/en-us/library/ff471376%28v=vs.85%29.aspx