King Abdullah University of Science and Technology

KAUST

CS 380 - GPU and GPGPU Programming
Lecture 3: GPU Architecture 2

Markus Hadwiger, KAUST

Reading Assignment #2 (until Feb. 19)

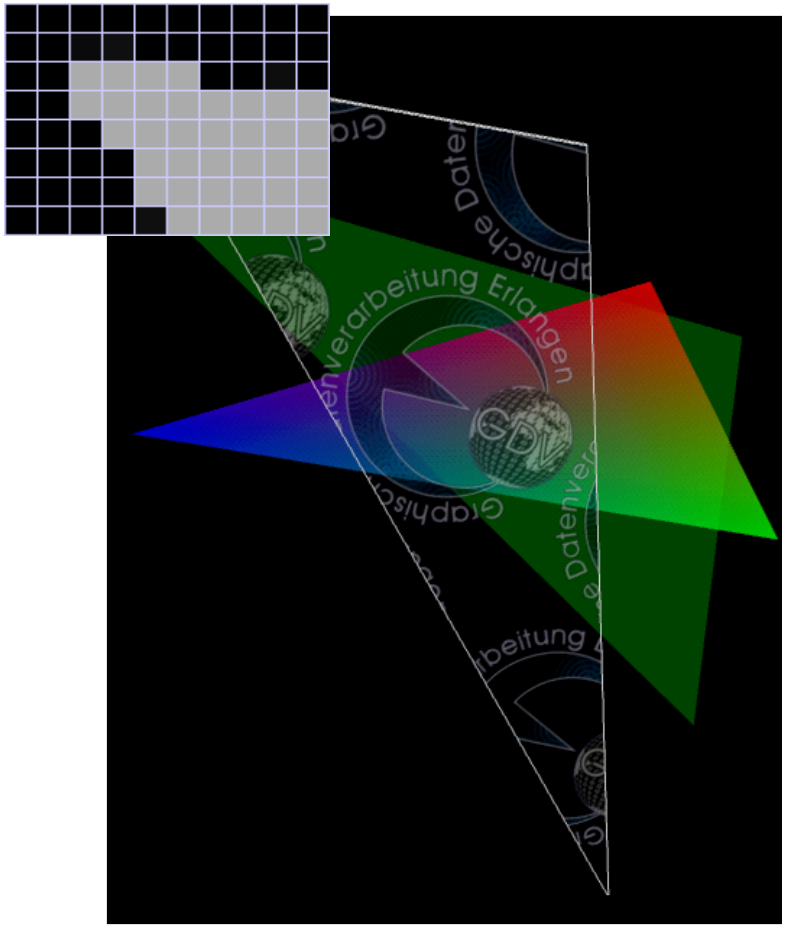
Read (required):

- GLSL book, chapter 4 (*The OpenGL Programmable Pipeline*)
- GPU Gems 2 book, chapter 30 (*The GeForce 6 Series GPU Architecture*) available online:

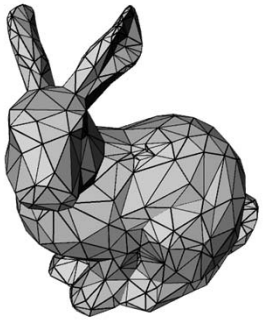
`http://download.nvidia.com/developer/GPU_Gems_2/GPU_Gems2_ch30.pdf`

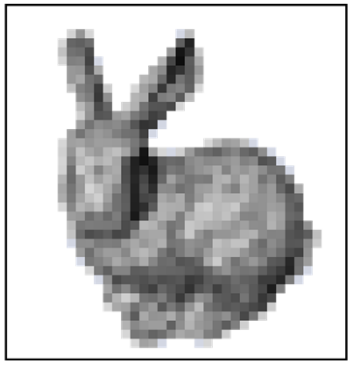
What can the hardware do?

- **Rasterization**
  - Decomposition into fragments
  - Interpolation of color
  - Texturing
    - Interpolation/Filtering
    - Fragment Shading
- **Fragment Operations**
  - Depth Test (Z-Test)
  - Alpha Blending (Compositing)



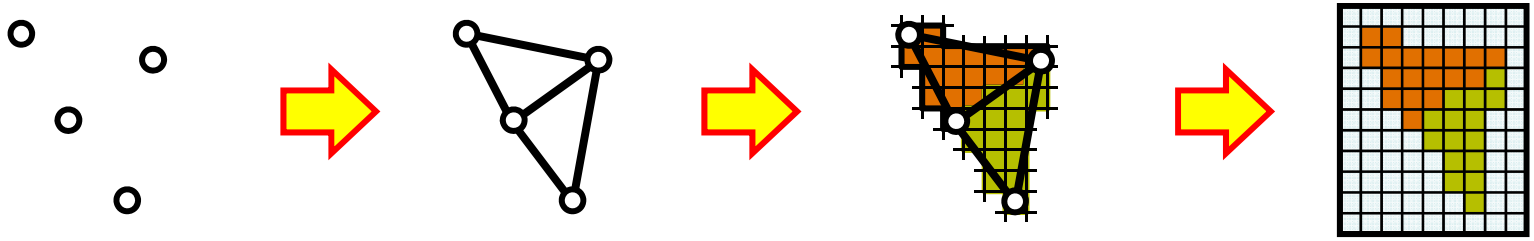
Graphics Pipeline





Scene Description

Raster Image



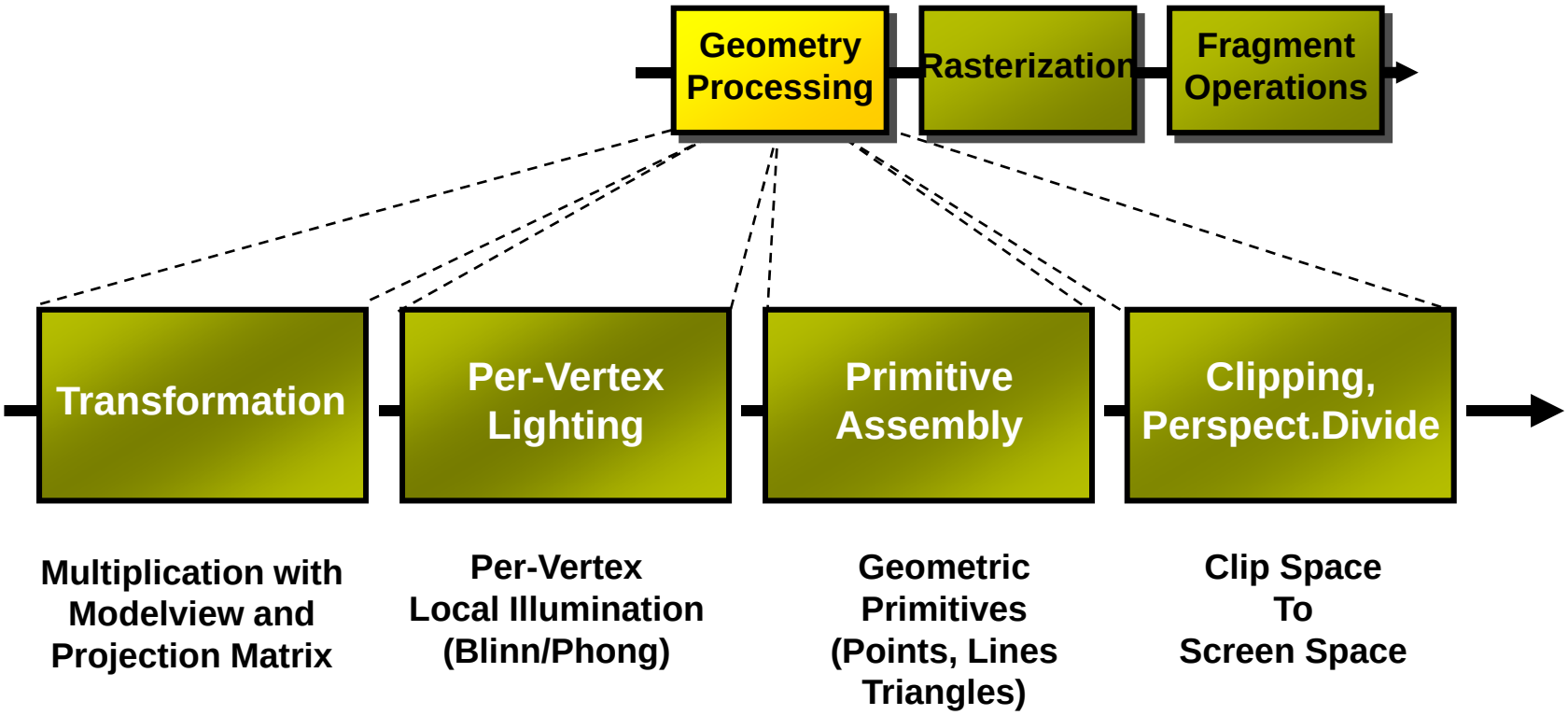
Vertices

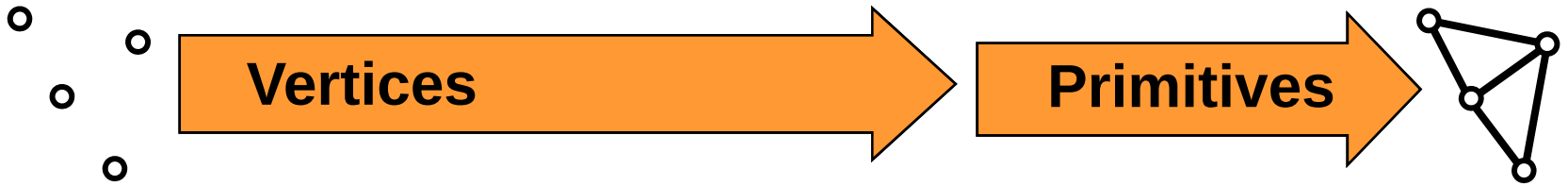
Primitives

Fragments

Pixels

Geometry Processing





Rasterization

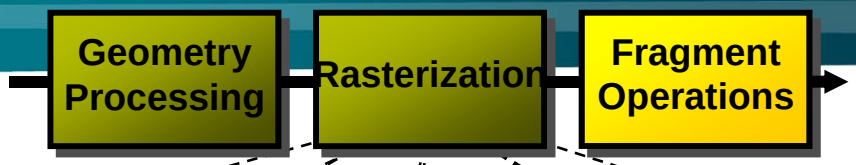
Decomposition of primitives into fragments

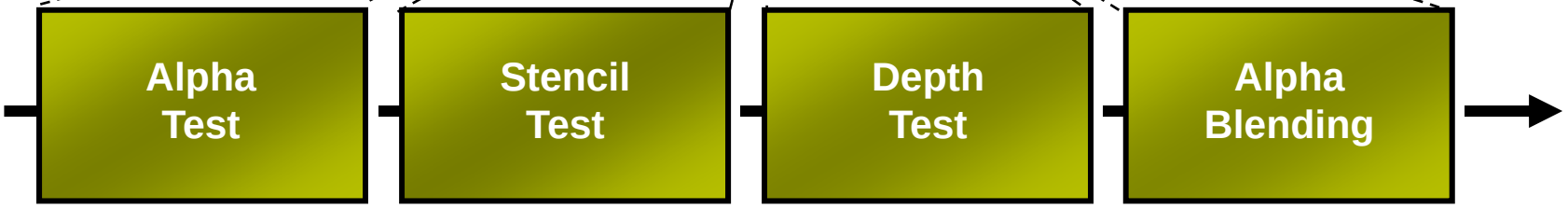
Interpolation of texture *coordinates*

Filtering of texture color

Combination of primary color with texture color

Fragment Operations



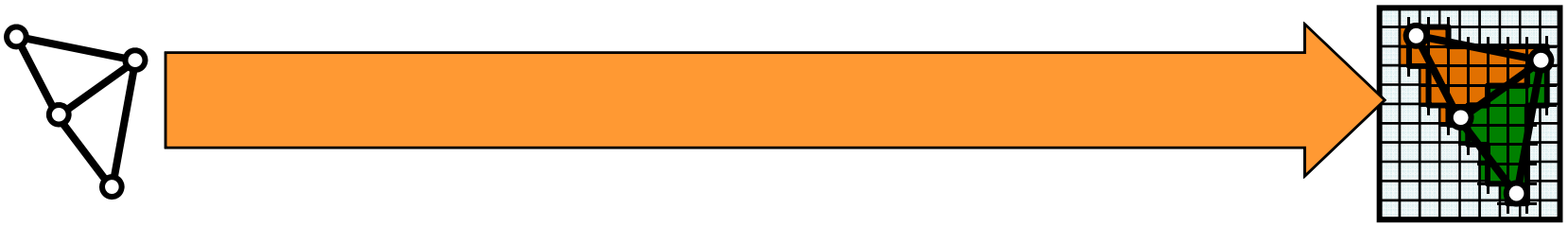


Discard all fragments within a certain alpha range

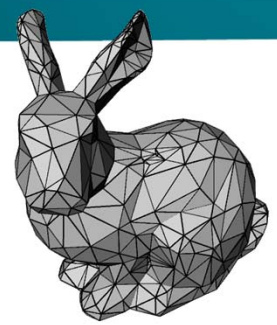
Discard a fragment if the stencil buffer is set

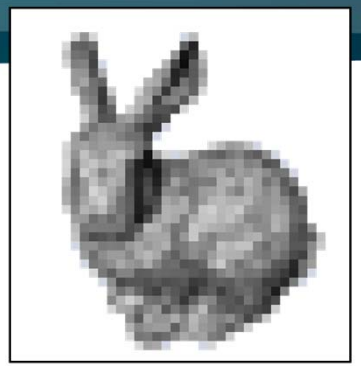
Discard all occluded fragments

Combination of primary color with texture color



Graphics Pipeline





Scene Description

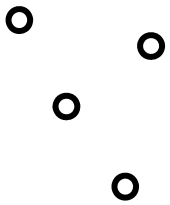
Programmable Pipeline

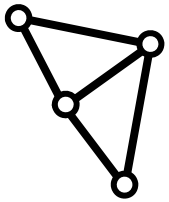
Raster Image

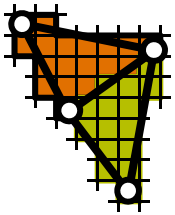
Vertex Shader

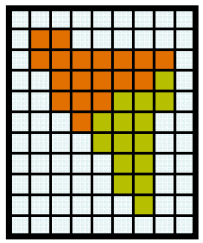
Fragment Shader

Fragment Operations









Vertices

Primitives

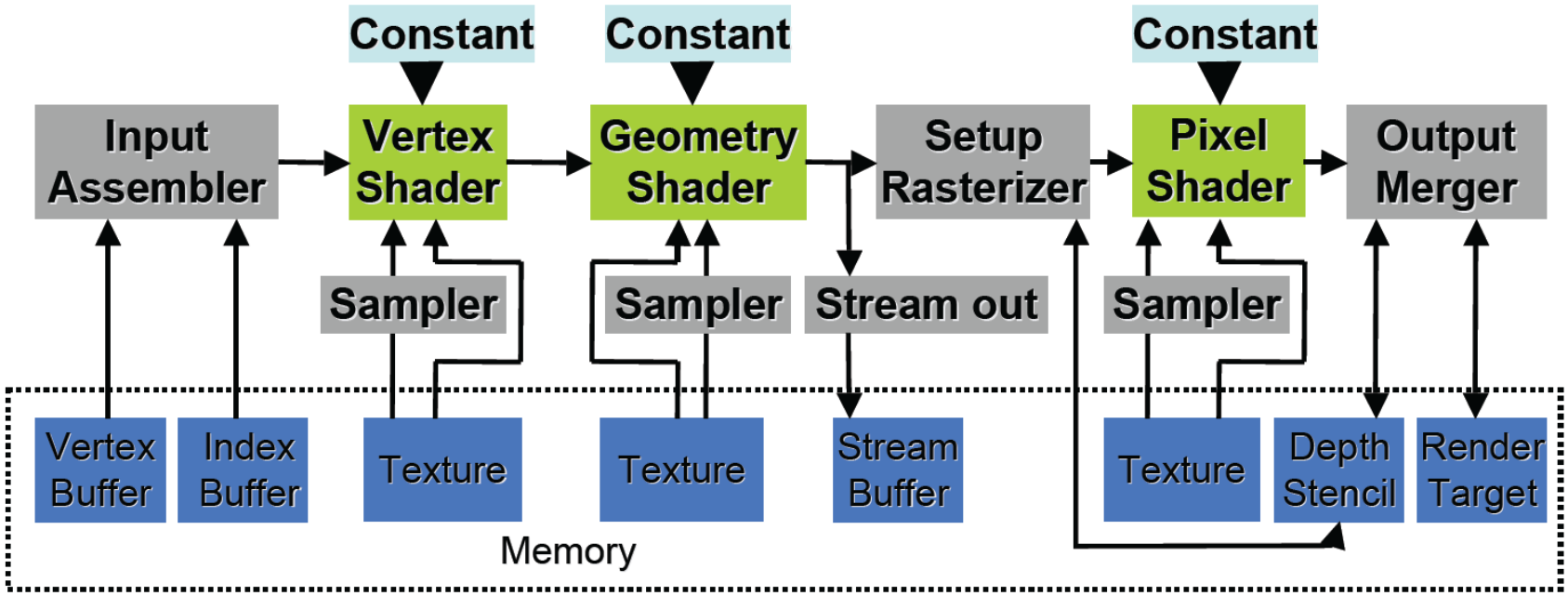
Fragments

Pixels

Direct3D 10 Pipeline (~OpenGL 3.2)

New geometry shader stage:

- Vertex -> geometry -> pixel shaders
- Stream output after geometry shader



Courtesy David Blythe, Microsoft

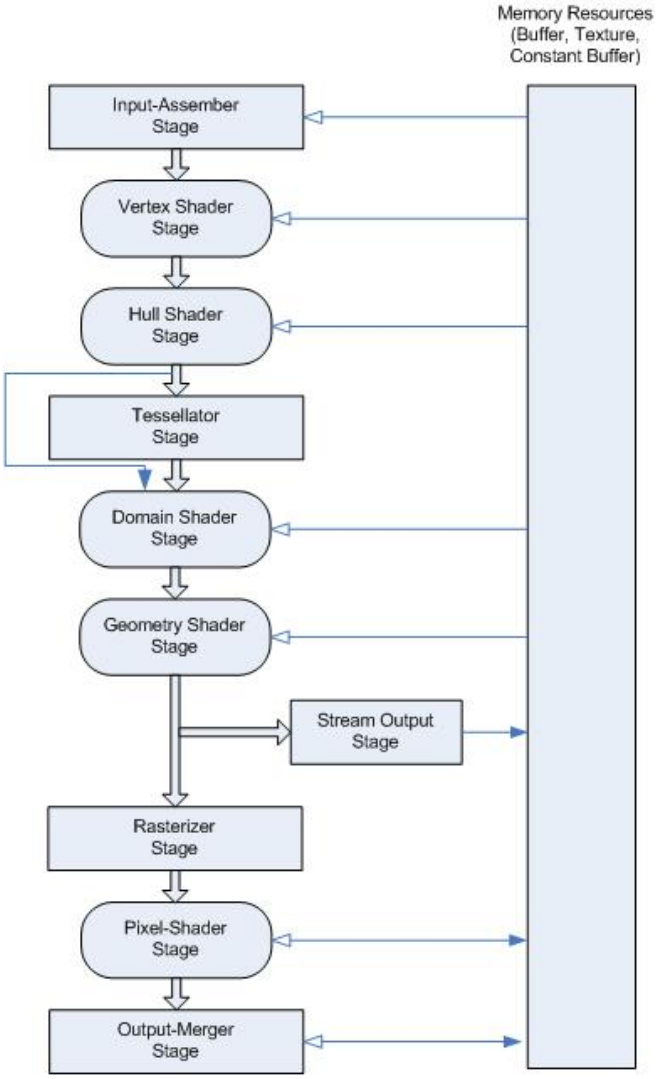
Direct3D 11 Pipeline (~OpenGL 4.0)

New tessellation stages

- Hull shader
  (OpenGL: *tessellation control*)
- Tessellator
  (OpenGL: *tessellation primitive generator*)
- Domain shader
  (OpenGL: *tessellation evaluation*)

- Trend of adding new stages
likely to continue...

- ... or full flexibility such as
in Intel MIC (Larrabee) architecture?

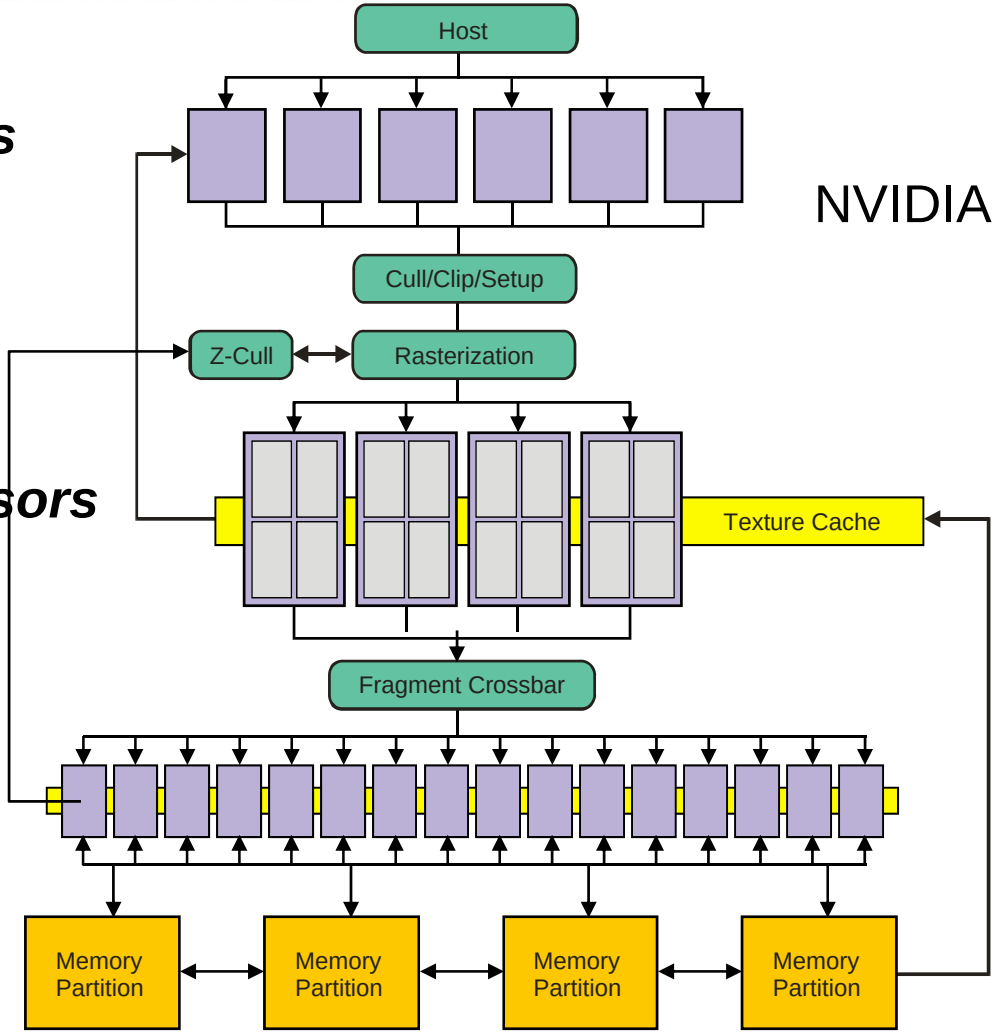


GPU Structure Before Unified Shaders

Vertex Processors

Fragment Processors

Memory Access
Z-Compare and Blending



Example
NVIDIA Geforce 6/7,
2004, 2005

A diffuse reflectance shader

```
sampler mySamp;
Texture2D<float3> myTex;
float3 lightDir;

float4 diffuseShader(float3 norm, float2 uv)
{
  float3 kd;
  kd = myTex.Sample(mySamp, uv);
  kd *= clamp( dot(lightDir, norm), 0.0, 1.0);
  return float4(kd, 1.0);
}
```

Independent, but no explicit parallelism

Compile shader

1 unshaded fragment input record

```
sampler mySamp;
Texture2D<float3> myTex;
float3 lightDir;

float4 diffuseShader(float3 norm, float2 uv)
{
  float3 kd;
  kd = myTex.Sample(mySamp, uv);
  kd *= clamp ( dot(lightDir, norm), 0.0, 1.0);
  return float4(kd, 1.0);
}
```

```
<diffuseShader>:
sample r0, v4, t0, s0
mul  r3, v0, cb0[0]
madd r3, v1, cb0[1], r3
madd r3, v2, cb0[2], r3
clmp r3, r3, l(0.0), l(1.0)
mul  o0, r0, r3
mul  o1, r1, r3
mul  o2, r2, r3
mov  o3, l(1.0)
```

1 shaded fragment output record

Thank you.