Introduction to Programmable GPUs

**Vertex Shader - Instructions**
- Arithmetic Operations on 4-vectors:
  - ADD, MUL, MAD, MIN, MAX, DP3, DP4
- Operations on Scalars
  - RCP (1/x), RSQ (1/x²), EXP, LOG
- Specialty Instructions
  - DST (distance: computes length of vector)
  - LIT (quadratic falloff term for lighting)
- Later generation:
  - Loops and conditional jumps

**Vertex & Fragment Shader**
- massively parallel computing by parallelization
- same shader is applied to all data (vertices or fragments) – SIMD (single instruction multiple data)
- parallel programming issues:
  - main advantage: high performance
  - main disadvantage: no access to neighboring vertices/fragments

**Vertex Shader**
- per-vertex computation (transform & lighting):
  - model and view transform
  - perspective transform
  - texture coordinate transform
  - per-vertex lighting

**Programmable Pipeline**
- so far:
  - have discussed rendering pipeline as specific set of stages with fixed functionality

**Programmable Pipeline**
- now: programmable rendering pipeline!

**GPUs vs CPUs**
- 800 GFLOPS vs 80 GFLOPS
- 86.4 GB/s vs 8.4 GB/s

**Vertex Shader - Applications**
- deformable surfaces: skinning
- different parts have different rigid transformations
- vertex positions are blended
- used in facial animations – many transformations!

**Fragment Shader - Applications**
- Performs all per-fragment computation:
  - texture mapping
  - fog
- input (interpolated over primitives by rasterizer):
  - texture coordinates
  - color
- output:
  - fragment color

**Vertex Shader**
- Input:
  - vertex position and normal (sometimes tangent)
  - multi-texture coordinates
  - modelview, projection, and texture matrix
  - vertex material or color
  - light sources – color, position, direction etc.

**Fragment Shader**
- Performs all per-fragment computation:
  - texture mapping
  - fog
- Input (interpolated over primitives by rasterizer):
  - texture coordinates
  - color
- Output:
  - fragment color

**Real-Time Graphics**
- Virtua Fighter 1995 (Sega Corporation)
- Dead or Alive 3 2001 (Force Corporation, Abec (NVIDIA))
- Dawn 2003 (NVIDIA Corporation)
- Medusa 2008 (NVIDIA Corporation)

**Shading languages**
- Cg (C for Graphics – NVIDIA)
- GLSL (GL Shading Language – OpenGL)
- HLSL (High Level Shading Language – MS Direct3D)
Cg History

// compute the diffuse term
float3 diffuse = normalize(normalWorld.xyz) * lightDiffuse * materialDiffuse;

Material: animated teapot

DEMO

GPGPU Applications

DEMO