Programmable GPUS

Last Time?
- Planar Shadows
- Projective Texture Shadows
- Shadow Maps
- Shadow Volumes
  - Stencil Buffer

Today
- Modern Graphics Hardware
- Shader Programming Languages
- Gouraud Shading vs. Phong Normal Interpolation
- Many “Mapping” techniques

Modern Graphics Hardware
- High performance through
  - Parallelism
  - Specialization
  - No data dependency
  - Efficient pre-fetching

Programmable Graphics Hardware
- Geometry and pixel (fragment) stage become programmable
  - Elaborate appearance
  - More and more general-purpose computation (GPU hacking)

Misc. Stats on Graphics Hardware
- 2005
  - 4-6 geometry units, 16 fragment units
  - Deep pipeline (~400 stages)
- NVIDIA GeForce 9 (Feb 2008)
  - 32-64 cores, 512 MB/1GB memory
- ATI Radeon R700 (2008)
  - 480 stream processing units
- NVIDIA GeForce GTX 480 (2010)
  - 480 cores, 1536 MB memory
  - 2560x1600 resolution
- ATI Radeon HD 7900 (2012)
  - 2048 processors, 3GB memory
- NVIDIA GeForce GTX 680 (2012)
  - 1536 cores, 2040 MB memory
Today

- Modern Graphics Hardware
- **Shader Programming Languages**
  - Cg design goals
  - GLSL examples
- Gouraud Shading vs. Phong Normal Interpolation
- Many “Mapping” techniques

---

Emerging & Evolving Languages

- Inspired by Shade Trees [Cook 1984] & Renderman Shading Language [1980’s]:
  - RTSL [Stanford 2001] – real-time shading language
  - Cg [NVIDIA 2003] – “C for graphics”
  - GLSL [OpenGL ARB 2004] – OpenGL 2.0
  - Optix [NVIDIA 2009] – Real time ray tracing engine for CUDA
- General Purpose GPU computing
  - CUDA [NVIDIA 2007]
  - OpenCL (Open Computing Language) [Apple 2008] for heterogeneous platforms of CPUs & GPUs

---

Cg Design Goals

- Ease of programming
- Portability
- Complete support for hardware functionality
- Performance
- Minimal interference with application data
- Ease of adoption
- Extensibility for future hardware
- Support for non-shading uses of the GPU

*“Cg: A system for programming graphics hardware in a C-like language” Mark et al. SIGGRAPH 2003*

---

Cg Design

- Hardware is changing rapidly [2003]… no single standard
- Specify “profile” for each hardware
  - May omit support of some language capabilities (e.g., texture lookup in vertex processor)
- Use hardware virtualization or emulation?
  - “Performance would be so poor it would be worthless for most applications”
  - Well, it might be ok for general purpose programming (not real-time graphics)

---

Cg compiler vs. GPU assembly

- Can inspect the assembly language produced by Cg compiler and perform additional optimizations by hand
  - Generally once development is complete (& output is correct)
- Using Cg is easier than writing GPU assembly from scratch

---

(Typical) Language Design Issues

- Parameter binding
- Call by reference vs. call by value
- Data types: 32 bit float, 16 bit float, 12 bit fixed & type-promotion (aim for performance)
- Specialized arrays or general-purpose arrays
  - float4 x vs. float x[4]
- Indirect addressing/pointers (not allowed…)
- Recursion (not allowed…)
Today

• Modern Graphics Hardware
• Shader Programming Languages
  – Cg design goals
  – GLSL examples
• Gouraud Shading vs. Phong Normal Interpolation
• Many “Mapping” techniques

GLSL example: checkerboard.vs

Remember Gouraud Shading?

• Instead of shading with the normal of the triangle, we’ll shade the vertices with the average normal and interpolate the shaded color across each face
  – This gives the illusion of a smooth surface with smoothly varying normals

Phong Normal Interpolation

• Interpolate the average vertex normals across the face and compute per-pixel shading
  – Normals should be re-normalized (ensure length=1)

• Before shaders, per-pixel shading was not possible in hardware (Gouraud shading is actually a decent substitute!)

Today

• Modern Graphics Hardware
• Shader Programming Languages
• Gouraud Shading vs. Phong Normal Interpolation
• Many “Mapping” techniques
  – Bump Mapping
  – Displacement Mapping
  – Environment Mapping
  – Light Mapping
  – Normal Mapping
  – Parallax Mapping
  – Parallax Occlusion Mapping
Bump Mapping

- Use textures to alter the surface normal
  - Does not change the actual shape of the surface
  - Just shaded as if it were a different shape

Another GLSL example: orange.vs

- Treat a greyscale texture as a single-valued height function
- Compute the normal from the partial derivatives in the texture

Another GLSL example: orange.fs

Bump Mapping

- Variation on Bump Mapping:
  Use an RGB texture to directly encode the normal

Normal Mapping

- Variation on Bump Mapping:
  Use an RGB texture to directly encode the normal
What's Missing?

• There are no bumps on the silhouette of a bump-mapped or normal-mapped object

• Bump/Normal maps don’t allow self-occlusion or self-shadowing

Today

• Modern Graphics Hardware
• Shader Programming Languages
• Gouraud Shading vs. Phong Normal Interpolation
• Many “Mapping” techniques
  – Bump Mapping
  – Displacement Mapping
  – Environment Mapping
  – Light Mapping
  – Normal Mapping
  – Parallax Mapping
  – Parallax Occlusion Mapping

Displacement Mapping

• Use the texture map to actually move the surface point
• The geometry must be displaced before visibility is determined

Parallax Mapping

• Displace the texture coordinates for each pixel based on view angle and value of the height map at that point
• At steeper view-angles, texture coordinates are displaced more, giving illusion of depth due to parallax effects

Image from: Geometry Caching for Ray-Viewing Displacement Maps EGRW 1996 Matt Pharr and Pat Hanrahan

Displacement Mapping

Image from: Geometry Caching for Ray-Viewing Displacement Maps EGRW 1996 Matt Pharr and Pat Hanrahan

Displacement Mapping

"Detailed shape representation with parallax mapping", Kaneko et al. ICAT 2001

Parallax Mapping

a.k.a. Offset Mapping or Virtual Displacement Mapping
Parallax Occlusion Mapping

- Brawley & Tatarchuk 2004
- Per pixel ray tracing of the heightfield geometry
- Occlusions & soft shadows

http://developer.amd.com/media/gpu_assets/
Tatarchuk-ParallaxOcclusionMapping-Sketch-print.pdf

Today

- Modern Graphics Hardware
- Shader Programming Languages
- Gouraud Shading vs. Phong Normal Interpolation
- Many “Mapping” techniques
  - Bump Mapping
  - Displacement Mapping
  - Environment Mapping
  - Light Mapping
  - Normal Mapping
  - Parallax Mapping
  - Parallax Occlusion Mapping

Environment Maps

- We can simulate reflections by using the direction of the reflected ray to index a spherical texture map at "infinity".
- Assumes that all reflected rays begin from the same point.

Environment Mapping Example

Terminator II

What's the Best Chart?

Texture Maps for Illumination

- Also called "Light Maps"
Questions?

Image by Henrik Wann Jensen
Environment map by Paul Debevec

Reading for Today:

• Chris Wyman, "An Approximate Image-Space Approach for Interactive Refraction", SIGGRAPH 2005

Readings for Friday:

Choose: