Lecture 16:
Hardware Rendering
and Projects

Fall 2004
Kavita Bala
Computer Science
Cornell University

Announcements

• Project proposal due Oct 26

• Contact me if you are still unsure
New Programmable GPUs

- Pipelined and parallel
  - Current pipeline 600-800 stages deep!

- Branching/looping??

- Floating point arithmetic

- Programmable Vertex and Shader programs

- Essentially writing assembly/C code

New OpenGL

Vertices → Vertex Shader → Transformed vertices → Rasterization → Fragments → Fragment Shader → Shaded fragments → Composite

Frame Buffer → Texture Memory

© Kavita Bala, Computer Science, Cornell University
Key Hardware Capabilities

• Z-Buffering
• Accumulation Buffer
• Antialiasing
• Transparency/Compositing
• Stencil Buffer
• Filtered Texturing

Texture Mapping

Images courtesy Tito Pagan
Many types of Texture Maps

• Texture modulates diffuse coefficients in shading model

• Textures can modulate
  – Normals: bump mapping and normal mapping
  – Positions: displacement mapping
  – Lighting: environment mapping

Environment Map

• Want to compute reflections of environment on surfaces
  – Planar surfaces?
  – Curved surfaces

• Assumptions:
  – Environment Map represents objects at infinity

• Index into EM using reflection vector
Environment Mapping

• EM gives reflections in curved surfaces
  – Not very good for flat surfaces

Env Map Algorithm

• Generate 2D environment map
  – Spherical, cubical, paraboloid

• For each pixel on a reflective object
  – Find N on surface of object
  – Compute R from V and N: \( R = V - 2(N \cdot V)N \)
  – Index into EM using R
  – Modulate pixel color
Cube Mapping

- The norm on modern hardware
- Place camera in center of the environment
- Project environment onto cube sides
  - 90 degree field of view
  - Cost?

Picking the cube map

- Compute R
  - Don’t need to normalize it
- Pick the largest component (magnitude)
  - What does it mean?
- Scale other two components to [0,1]
Looking up EM

- If triangle spans multiple EM faces?
- Per-pixel based

Sphere Maps

- Assume viewing is from infinity
- Capture reflections
  - Creation uses photographs or ray tracing or warping
Sphere Mapping Example

- Environment map $\rightarrow$ radiance
- Filter this map $\rightarrow$ irradiance (diffuse lighting)
- Fast diffuse and ambient (just a lookup, or eqn)
Filtered Reflection Mapping

• Blur EM for gloss

Lobe Filtering has problems

© Kavita Bala, Computer Science, Cornell University
Techniques to Render with EM

- Ambient Occlusion
- Structured Importance Sampling

Use hardware for better illumination

- Multi-pass rendering
- Multi-texture rendering
  – Dependent texture reads
Multi-Pass Texturing

• Limits to what hardware can do in 1 pass

• So multi-pass texturing
   – Each pass does some part of shading
   – Outputs a “fragment”: rgb, alpha, z
   – Add or blend with previous pass

• For example
   – 1<sup>st</sup> pass: diffuse
   – 2<sup>nd</sup> pass: specular

Why multi-pass?

• Scalable

Quake III Engine
1. Passes 1-4: accumulate bump map
2. Pass 5: Diffuse lighting
3. Pass 6: Base texture
4. Pass 7: Specular lighting
5. Pass 8: Emissive lighting
6. Pass 9: Volumetric lighting
Multitexturing

- Modern hardware can apply multiple texture values in each pass
- Series of texture stages

Interpolated Vertex value

Stage 0

Texture Value

Stage 1

Texture Value

© Kavita Bala, Computer Science, Cornell University
Multitexture Example: Light Maps

- Two separate textures
  - Material and lighting
  - Can be different resolutions

\[ X \]

Light Maps

- Light Maps used in games
- Cost: extra texture read
- Benefit:
  - Can use it to capture global illumination
  - Can store different resolutions of textures
  - Maybe animate texture coordinates

© Kavita Bala, Computer Science, Cornell University
Dependent Texture Reads

- Introduced in 1999

- Number of passes proportional to the longest "chain" of operations you need

- Dependent texture reads helps
  - Can read a texture
  - Transform it
  - And then read another texture based on transformed value!
  - Much more efficient
Reflections and Normal Maps

Environment Map Bump Mapping (EMBM)

GPU Rendering

- Rendering high-quality illumination on GPUs is getting more effective

- Attempts at
  - Ray tracing on GPUs
  - Photon mapping on GPUs
  - …
Future Topics in Course

• Shadows
  – Shadow maps
  – Shadow volumes
  – Soft shadows

• Many lights
  – Rendering environment maps

• NPR

Shadow Maps

• Introduced by Lance Williams (SIGGRAPH 1978)
Using the Shadow Map

• When scene is viewed, check viewed location in light's shadow buffer
  – If point's depth is (epsilon) greater than shadow depth, object is in shadow

For each pixel, compare distance to light with the depth stored in the shadow map

Shadow Volumes

• Crow 1977

• Can cast shadows onto curved surfaces
Soft Shadows

- Soft shadows appear natural
- Hard to get soft shadows in hardware
- Slow in software

Soft Shadows: Heckbert/Herf

2 x 2 samples

Images courtesy of Michael Herf and Paul Heckbert
Soler and Sillion

- Shadows as convolution

Penumbra Maps

Uses fragment program

Figure 1: Using a standard shadow map results in hard shadows (left); add a penumbra map to get soft shadows (right). Using a 10k polygon dragon model for the shadow and a 30k polygon model to render, we get 14.5 fps at 1624x1024.

© Kavita Bala, Computer Science, Cornell University
Environment Map Sampling

Other topics: NPR

• Non-photorealistic lighting model
Future Topics in Course

- High-complexity rendering
  - Points

- Image-based Rendering

Other topics: Point-based Rendering

- Use points instead of polygons

- Much more compact and robust

- How to render?
  - Splat points in hardware
Other topics: Image-Based Rendering

• Use photographs to capture complex scenes

Project Ideas

• Rendering:
  – Photon mapping
  – BRDF factorization for sampling
  – Shadow algorithms for soft shadows
  – Sampling and rendering with environment maps
  – Silhouette finding and rendering with modes

• NPR
  – Silhouette finding
  – Contour finding
Project Ideas

• High-complexity rendering
  – Point-based rendering

• Texture for complexity
  – Texture synthesis

• Acceleration structures
  – Support for dynamics