CSE167: Introduction to Computer Graphics

Lecture #16: Environment Mapping

Jürgen Schulze
University of California, San Diego
Fall 2009
Announcements

• Today:
  - CAPE
  - TA evaluation today
Final Project

- Problem description is on-line
- **Must** be done in teams of two
- ~300 word application design description due Monday Nov 30. Email to:
  - Last names A-Le: Jason (j.kimball AT ucsd.edu)
  - Last names Li-Z: Daniel (d.tenedor AT ucsd.edu)
- Project due **Thursday, Dec 10**: presentation in CSE 1202, 3-6pm
- No late submissions accepted!
StarCAVE Tour

- Location: Atkinson Hall, 1st floor
- 18 Dell XPS PCs with Quad Core Intel CPUs
- CentOS 5.2 Linux
- Dual Nvidia Quadro 5600 graphics cards per node
- 34 JVC HD2k projectors (1920x1080 pixels): ~34 megapixels per eye
- Passive stereo with circular polarization filters
- 15 screens, ~8 x 4 feet each
- Floor projection
- Optical, wireless tracking system
- Software: COVISE
- Programming Language: C++

Tour Date:
- Thursday, Nov 19, 4:00-5:30pm

Location:
Immersive Visualization Laboratory
1st floor Atkinson Hall
Turn right at main entrance
Mathematical Continuity

- **C\(^0\) continuity:**
  - Curve segments are connected

- **C\(^1\) continuity:**
  - \(C^0\) & 1st-order derivatives agree at joints
  - Tangents/normals are \(C^0\) continuous
  - Important for smooth shading

- **C\(^2\) continuity:**
  - \(C^1\) & 2nd-order derivatives agree at joints
  - Tangents/normals are \(C^1\) continuous
  - Important for high quality reflections
Geometric Continuity

- **G0:**
  - curve segments are connected
  - same as C0

- **G1:**
  - G0 & 1st-order derivatives are proportional at joints
  - Proportional = same direction but may have different magnitudes
  - Weaker than C1

- **G2:**
  - G1 & 2nd-order derivative proportional at joints
Lecture Overview

Advanced shader effects

- Environment mapping
More realistic illumination

- In real world, at each point in scene light arrives from all directions
  - Not just from point light sources

- Environment maps
  - Store “omni-directional” illumination as images
  - Each pixel corresponds to light from a certain direction
Capturing environment maps

- “360 degrees” panoramic image
- Instead of 360 degrees panoramic image, take picture of mirror ball (light probe)

Light probes
[Paul Debevec, http://www.debevec.org/Probes/]
Environment maps as light sources

Simplifying assumption

• Assume light captured by environment map is emitted from infinitely far away

• Environment map consists of directional light sources
  - Value of environment map is defined for each direction, independent of position in scene

• Use single environment map at each point in the scene

• Approximation!
Environment maps as light sources

- How do you compute shading of a diffuse surface using an environment map?
- What is more expensive to compute, shading a diffuse or a specular surface?
Diffuse Irradiance Environment Map

- Given scene with $k$ directional lights, light directions $d_1...d_k$ and intensities $i_1...i_k$, illuminating a diffuse surface with normal $n$ and color $c$
- Pixel intensity $B$ is computed as:
  \[
  B = c \sum_{j=1..k} \max(0, d_j \cdot n)i_j
  \]
- Cost of computing $B$ proportional to number of texels in environment map!
- -> Precomputation of diffuse reflection
- Observations:
  - All surfaces with normal direction $n$ will return the same value for the sum
  - The sum is dependent on just the lights in the scene and the surface normal
- Precompute sum for any normal $n$ and store result in a second environment map, indexed by surface normal
- Second environment map is called *diffuse irradiance environment map*
- Allows to illuminate objects with arbitrarily complex lighting environments with single texture lookup
Diffuse Irradiance Environment Map

- Cube map with diffuse map

- Diffuse shading vs. shading w/diffuse map

Environment maps applications

- Use environment map as “light source”

Global illumination [Sloan et al.]

Reflection mapping
Cube environment maps

- Store incident light on six faces of a cube instead of on a sphere

Spherical map

Cube map
Cube vs. Spherical Maps

• Advantages of cube maps:
  - More even texel sample density causes less distortion, allowing for lower resolution maps
  - Easier to dynamically generate cube maps for real-time simulated reflections
Bubble Demo

http://download.nvidia.com/downloads/nZone/demos/nvidia/Bubble.zip
Cube environment maps

Cube map look-up

- Given direction \((x, y, z)\)
- Largest coordinate component determines cube map face
- Dividing by magnitude of largest component yields coordinates within face
- In GLSL
  - Use \((x, y, z)\) direction as texture coordinates to `samplerCube`
Reflection mapping

- Simulate mirror reflection
- Compute reflection vector at each pixel
- Use reflection vector to look up cube map
- Rendering cube map itself is optional
Reflection mapping in GLSL

Application setup

• Load, bind a cube environment map
  
  ```
  glBindTexture(GL_TEXTURE_CUBE_MAP, ...);
  glTexImage2D(GL_TEXTURE_CUBE_MAP_POSITIVE_X,...);
  glTexImage2D(GL_TEXTURE_CUBE_MAP_NEGATIVE_X,...);
  glTexImage2D(GL_TEXTURE_CUBE_MAP_POSITIVE_Y,...);
  ...
  glEnable(GL_TEXTURE_CUBE_MAP);
  ```
Reflection mapping in GLSL

Vertex shader

• Compute viewing direction
• Reflection direction
  - Use `reflect` function
• Pass reflection direction to fragment shader

Fragment shader

• Look-up cube map using interpolated reflection direction
  ```glsl
  varying float3 refl;
  uniform samplerCube envMap;
  textureCube(envMap, refl);
  ```
Next time

- Toon Shading
- Shadow mapping
- Shadow volumes