CSE 167:
Introduction to Computer Graphics
Lecture #16: Environment Mapping

Jürgen P. Schulze, Ph.D.
University of California, San Diego
Fall Quarter 2015

Announcements

- First blog entry due tonight at midnight
- Final project due December 10th at 8am
  - Presentations December 10th 8am-11am in CSE 1202

Midterm Statistics

Category	Midterm 1	Midterm 2
Exams Submitted	98	85
Average Score	52.8	66.4
Median Score	54	68.5
Highest Score	80	80
Lowest Score	27.5	41.5
70-80 Points	7	20
60-70 Points	26	25
50-60 Points	30	14
40-50 Points	17	6
30-40 Points	15	0
20-30 Points	3	0

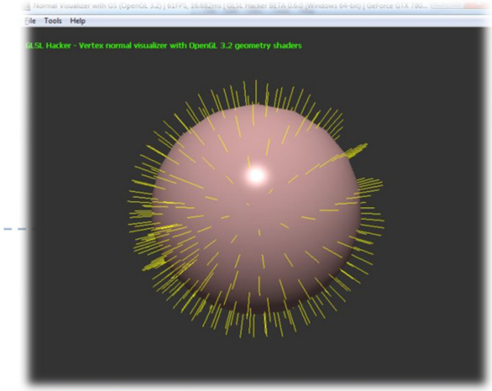
Visual Debugging

OpenGL error state: glGetError()

- OpenGL has an error state
- Use glGetError() to find location of error. It will clear the error flag.
- Then gluErrorString() to parse the error message

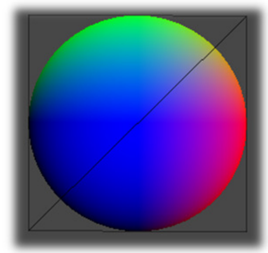
```
void printGLError(const char* msg)
{
  const GLenum err = glGetError();
  if(err != GL_NO_ERROR)
  {
    const char* str = (const char*)gluErrorString(err);
    cerr << “OpenGL error: " << msg << ", " << str << endl;
  }
}
```

Tips for Visual Debugging



Normal Visualizer

- Collisions, view frustum culling:
  - Show bounding boxes/spheres for all objects
- Problems with shading:
  - Display normal vectors on vertices as line segments pointing in the direction of the vector. Example: Normal Visualizer (pictured above).
  - Or interpret surface normals as RGB colors by shifting x/y/z range from -1..1 to 0..1.



Normal shading

- Display direction and other vectors:
  - Display normal vectors as described above.
- Objects don't get rendered:
  - Find out if they won't render or are just off screen by temporarily overwriting GL_MODELVIEW and GL_PROJECTION matrices with simpler ones, and/or zooming out by increasing the field of view angle.

OpenGL Debugging Tools

- Overview with many links:
  - https://www.opengl.org/wiki/Debugging_Tools
- Nvidia tools (Nsight and others):
  - https://developer.nvidia.com/gameworks-tools-overview

Lecture Overview

Advanced Shader Effects

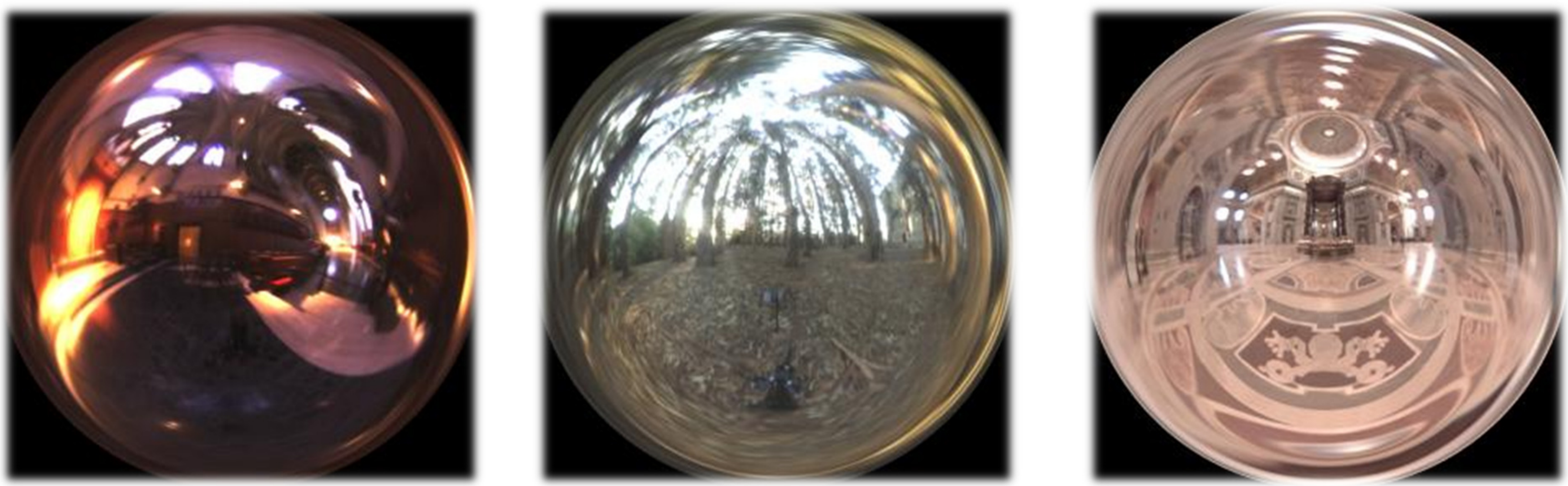
- Environment mapping
- Toon shading

More Realistic Illumination

- In the real world:
  At each point in scene light arrives from all directions
  - Not just from a few point light sources
  - → Global Illumination is a solution, but computationally expensive
- 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 by 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
- Approach uses same environment map at each point in scene
  → Approximation!

Applications for Environment Maps

- Use environment map as "light source"

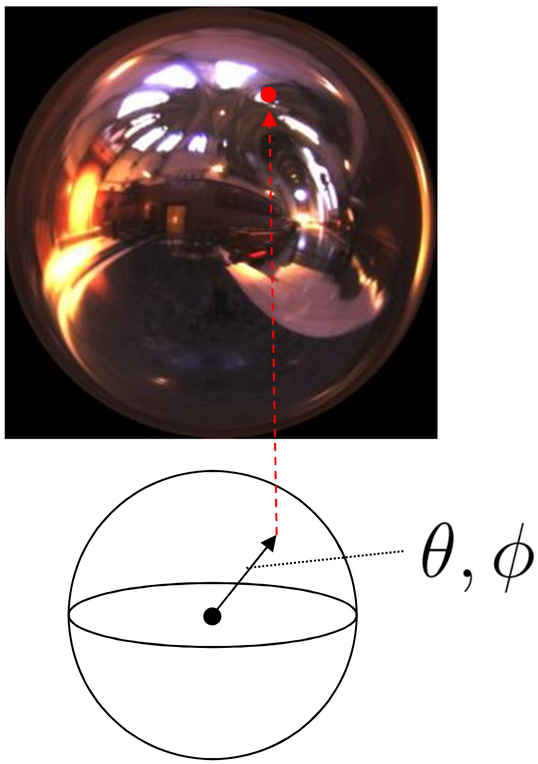
Global illumination with pre-computed radiance transfer [Sloan et al. 2002]

Reflection mapping [Terminator 2, 1991]

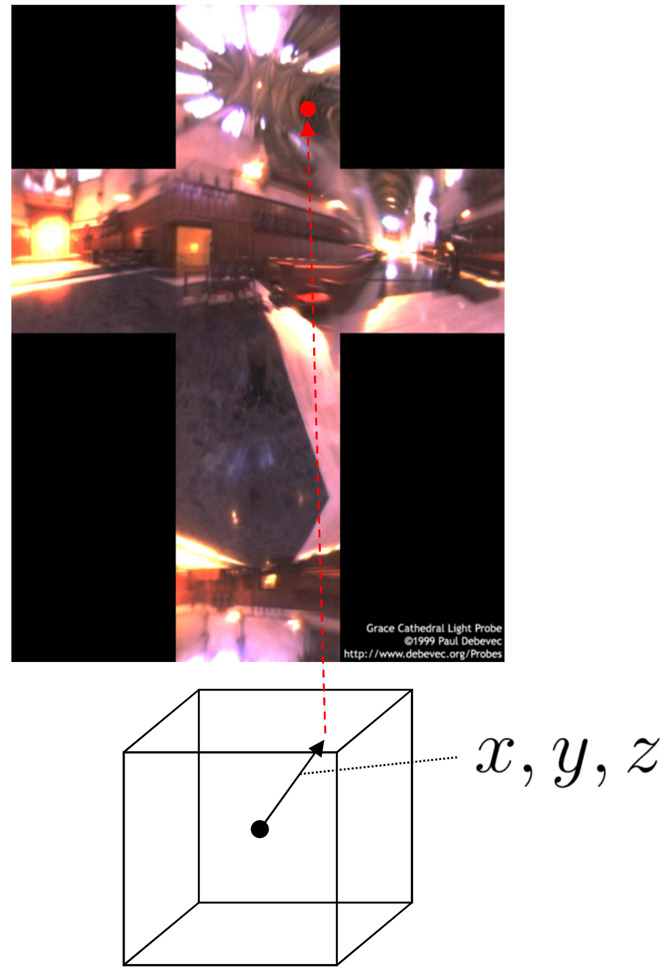
UCSD

Cubic Environment Maps

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



Spherical map



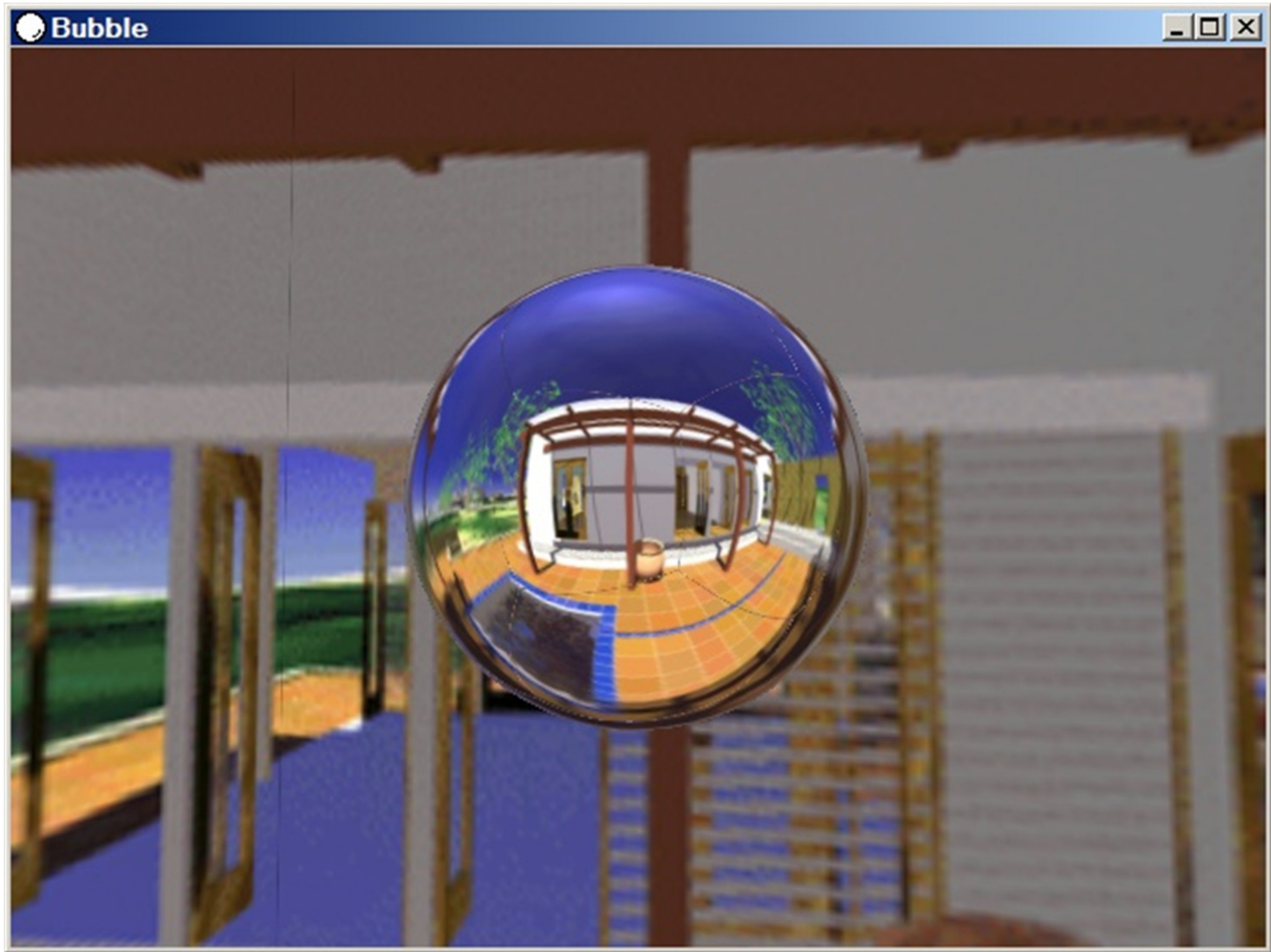
Cube map

UCSD

Cubic 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

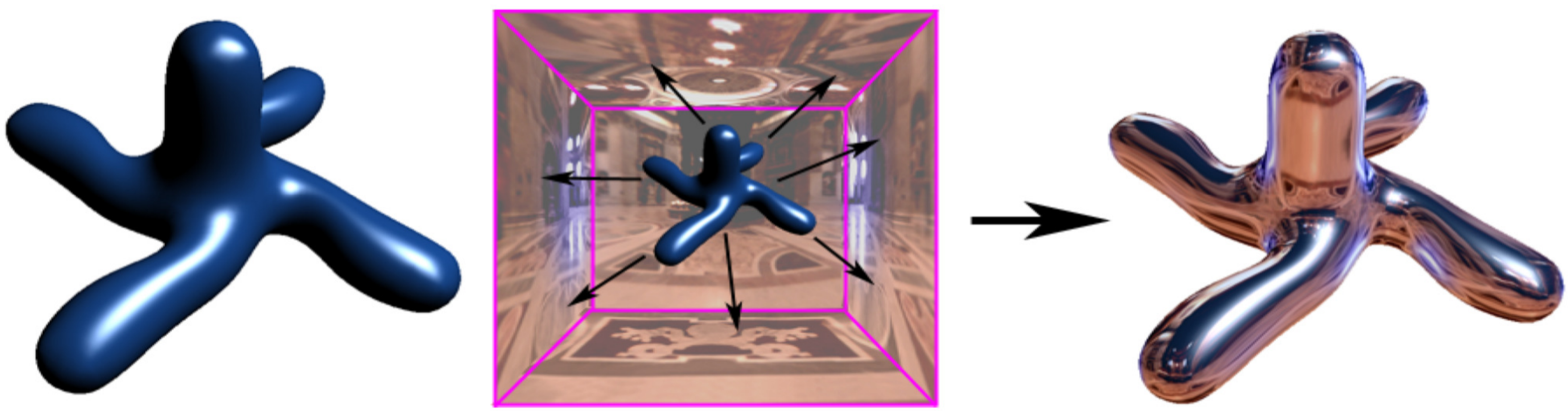
Cubic Environment Maps

Cube map look-up

- Given: light 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

- Simulates mirror reflection
- Computes reflection vector at each pixel
- Use reflection vector to look up cube map
- Rendering cube map itself is optional (application dependent)



Reflection mapping

Reflection Mapping in GLSL

Application Setup

- Load and 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

```
varying float3 refl;
uniform samplerCube envMap;
textureCube(envMap, refl);
```

Environment Maps as Light Sources

- Covered so far: shading of a specular surface

→ How do you compute shading of a diffuse surface?

Diffuse Irradiace Environment Map

- Given a 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 Irradiace Environment Map

- Two cubic environment maps:
  - Reflection map
  - Diffuse map

- Diffuse shading vs. shading w/diffuse map

Image source: http://http.developer.nvidia.com/GPUGems2/gpugems2_chapter10.html

UCSD