CSE 167:
Introduction to Computer Graphics
Lecture #12: Environment Mapping

Jürgen P. Schulze, Ph.D.
University of California, San Diego
Spring Quarter 2016

Announcements

- This Friday: Project 3 late grading
- Next Friday: Project 4 due

OpenGL Memory Management: Buffer Usage Hints

void glBufferData(GLenum target, GLsizeiptr size, const GLvoid * data, GLenum **usage**);

usage is a hint to the GL implementation as to how a buffer object's data store will be accessed. This enables the GL implementation to make more intelligent decisions that may significantly impact buffer object performance. It does not, however, constrain the actual usage of the data store. usage may be one of these:

- **GL_STATIC_DRAW**: The data store contents will be modified once and used many times as the source for GL drawing commands.
- **GL_DYNAMIC_DRAW**: The data store contents will be modified repeatedly and used many times as the source for GL drawing commands.

```
static void LoadTriangle()
{
    // make and bind the VAO
    glGenVertexArrays(1, &gVAO);
    glBindVertexArray(gVAO);

    // make and bind the VBO
    glGenBuffers(1, &gVBO);
    glBindBuffer(GL_ARRAY_BUFFER, gVBO);

    // Put the three triangle vertices into the VBO
    GLfloat vertexData[] = {0.0f, 0.8f, 0.0f, -0.8f,-0.8f, 0.0f, 0.8f,-0.8f, 0.0f,
    };
    glBufferData(GL_ARRAY_BUFFER, sizeof(vertexData), vertexData, GL_STATIC_DRAW);
    glEnableVertexAttribArray(0);
    glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 0, NULL);

    // unbind the VBO and VAO
    glBindBuffer(GL_ARRAY_BUFFER, 0);
    glBindVertexArray(0);
}
```

Lecture Overview

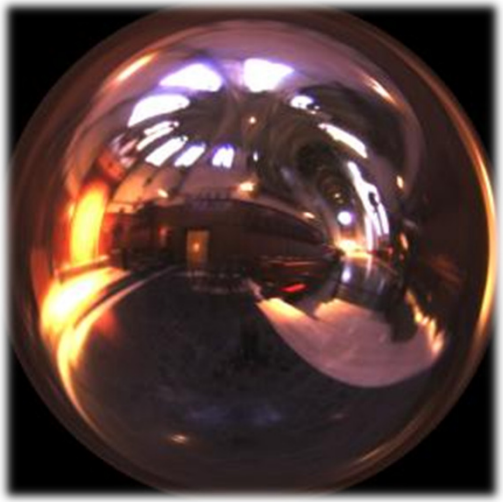
- Environment mapping

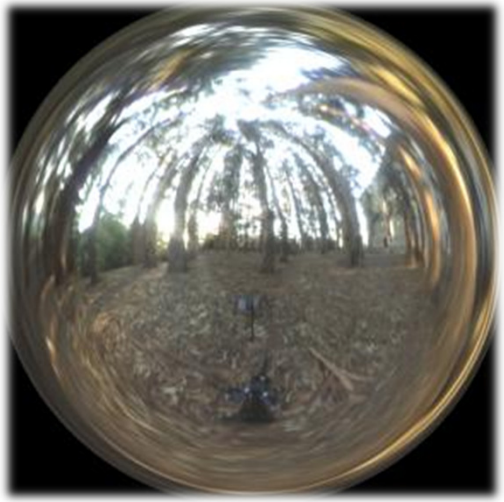
More Realistic Illumination

- 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

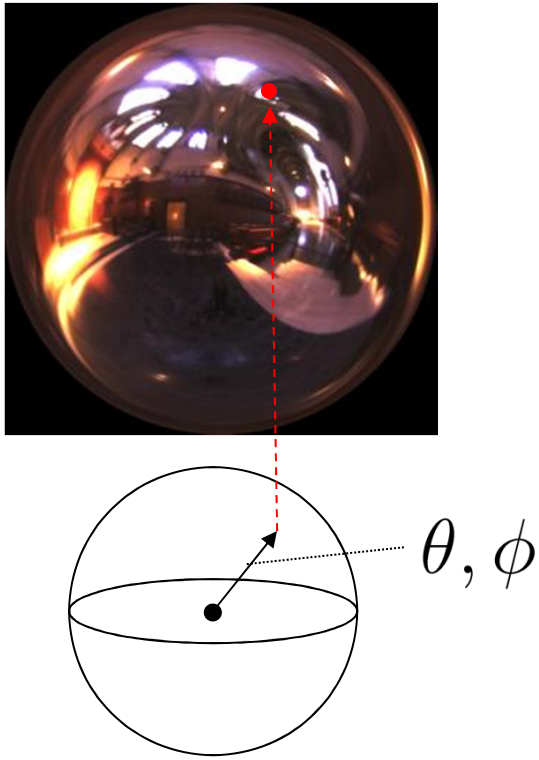
- Use environment map as "light source"

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

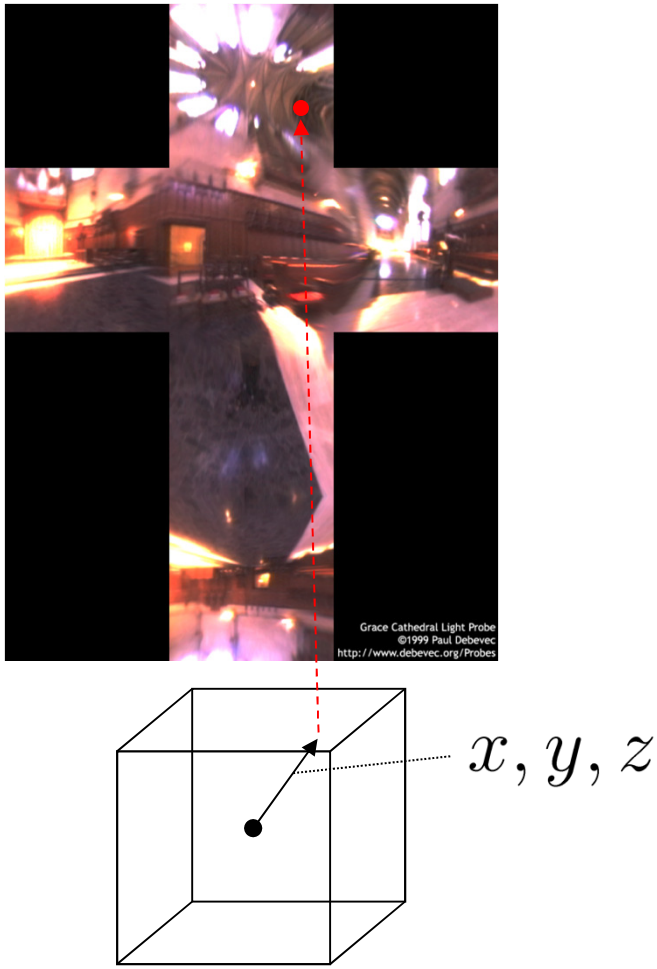
Reflection mapping [Terminator 2, 1991]

Cubic Environment Maps

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



Spherical map

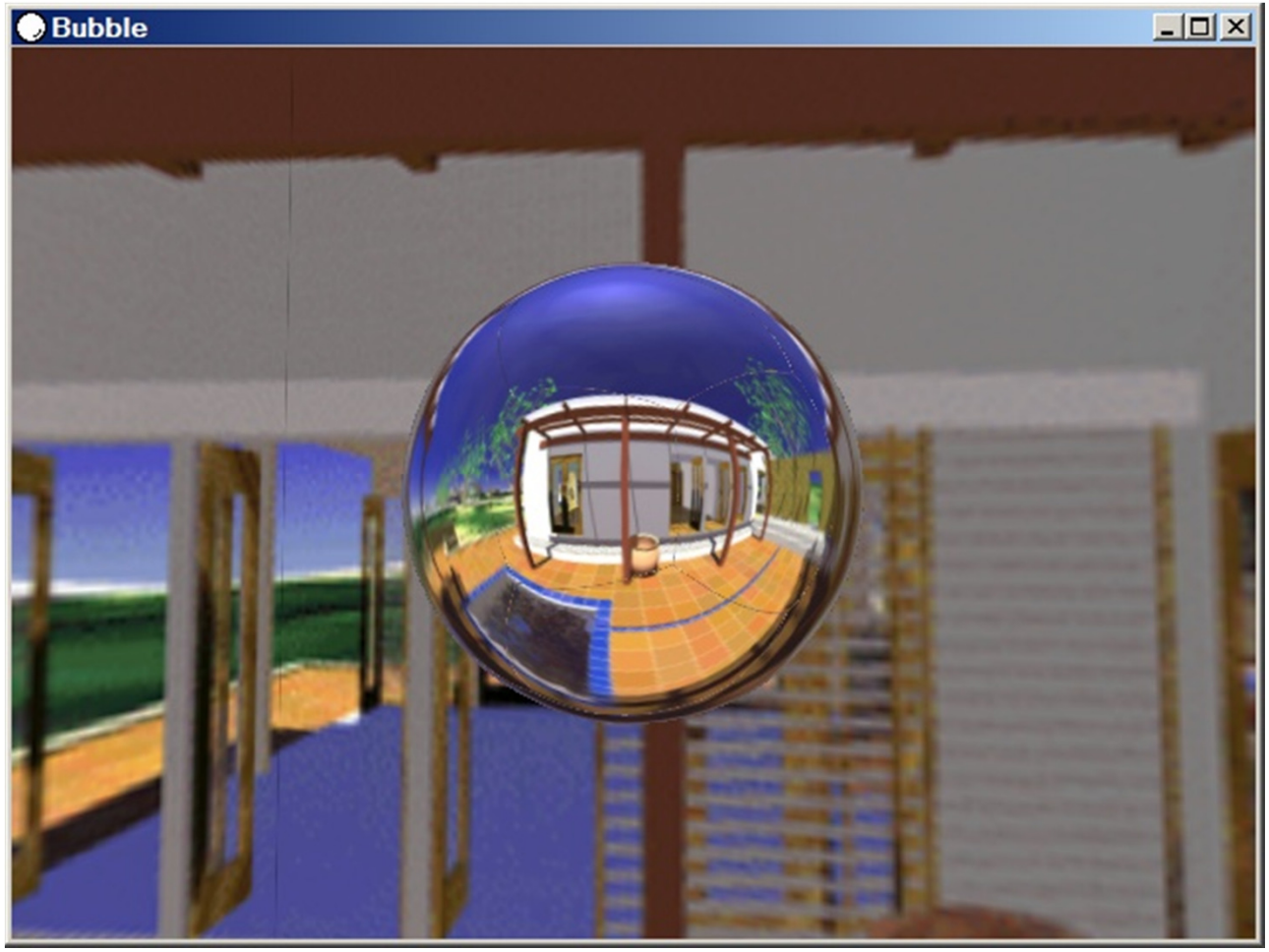


Cube map

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

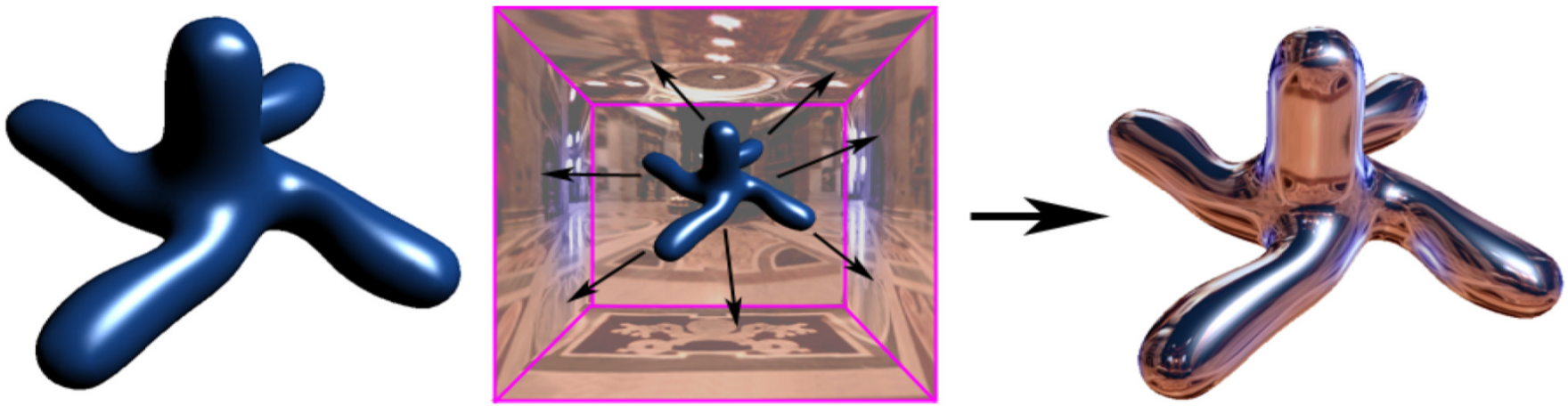
Cubic Environment Maps

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