CSE 167:
Introduction to Computer Graphics
Lecture #14: Environment Mapping

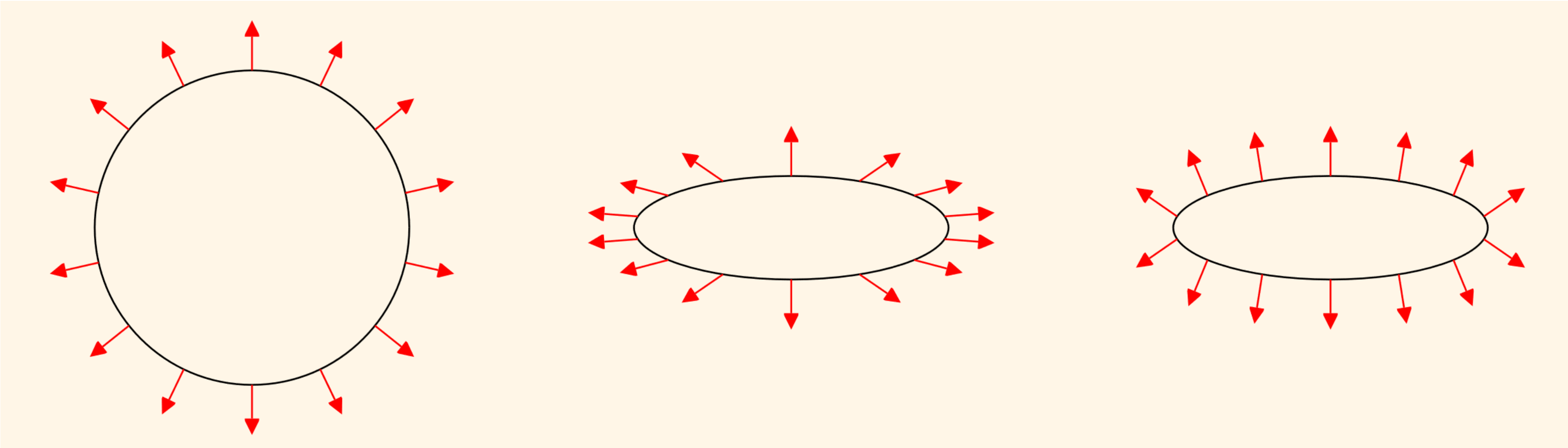
Jürgen P. Schulze, Ph.D.
University of California, San Diego
Fall Quarter 2017

Announcements

- Tomorrow: late grading project 3
- Monday night: midterm discussion
- Tuesday: midterm exam #2
- Vote for best robot!
  - Robots at: https://piazza.com/class/j7todrhr6a74af?cid=546
  - Vote at: https://piazza.com/class/j7todrhr6a74af?cid=582

Normal Transformation

- Why transform normal vectors with inverse transpose of a transformation matrix?



- Middle image: normal scaled like geometry gives wrong result

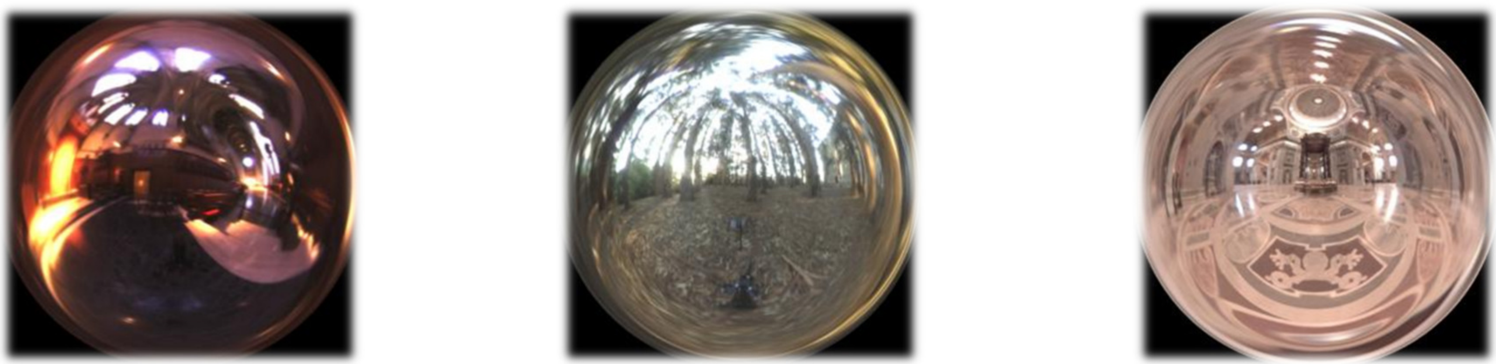
https://paroj.github.io/gltut/Illumination/Tut09%20Normal%20Transformation.html

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
  - Sky boxes make for great environment maps

Capturing Environment Maps

- Environment map = surround panoramic image
- Creating 360 degrees panoramic images:
  - 360 degree camera
  - "light probe" image: take picture of mirror ball (e.g., silver Christmas ornament)



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

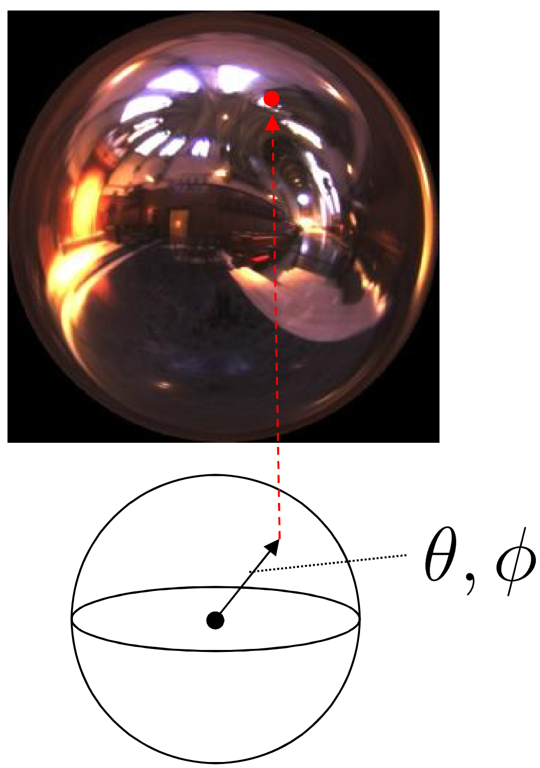
- Use environment map as "light source"

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

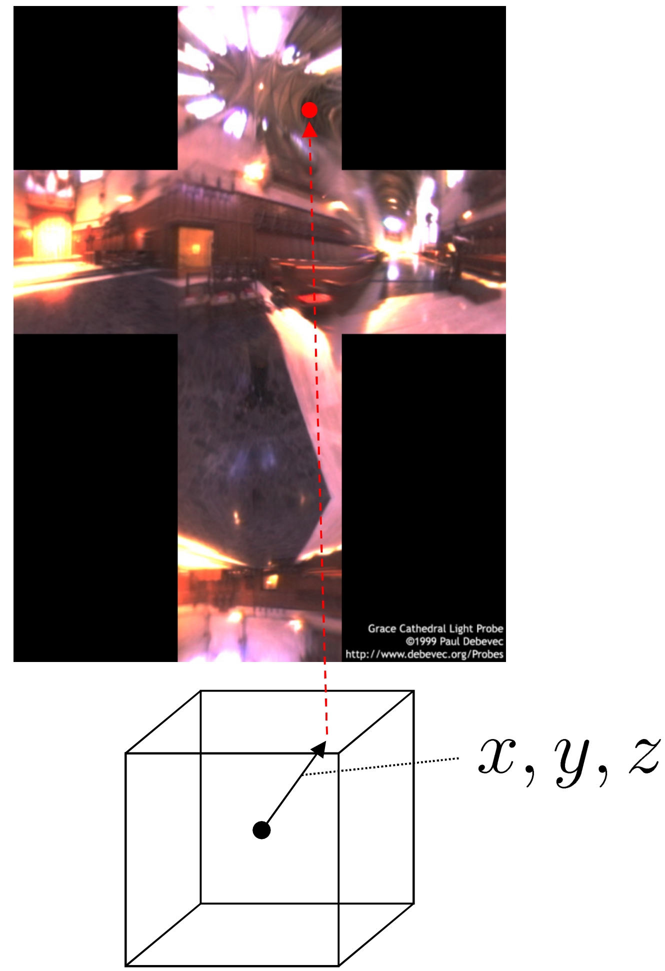
Reflection mapping [Georg-Simon Ohm University of Applied Sciences]

Cubic Environment Maps

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



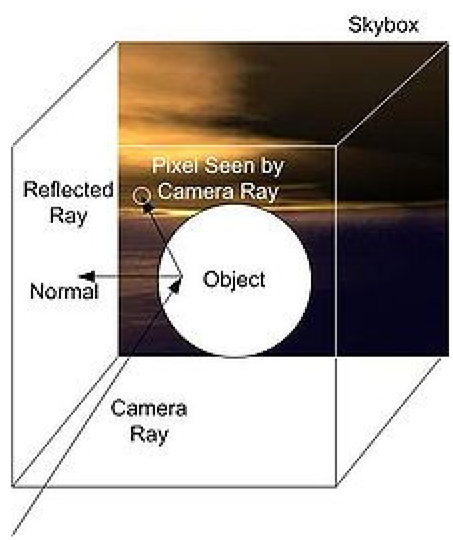
Spherical map



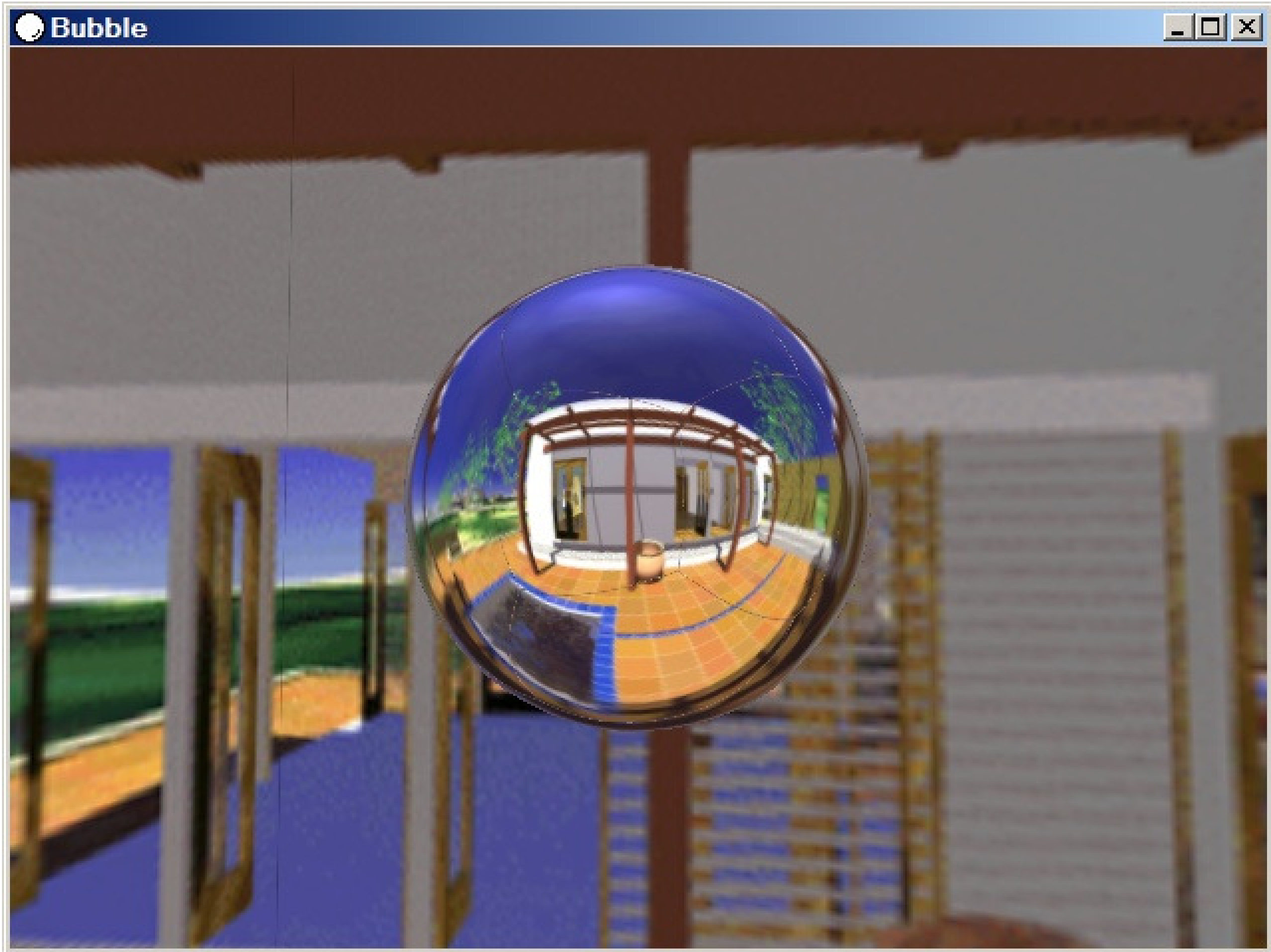
Cube map

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

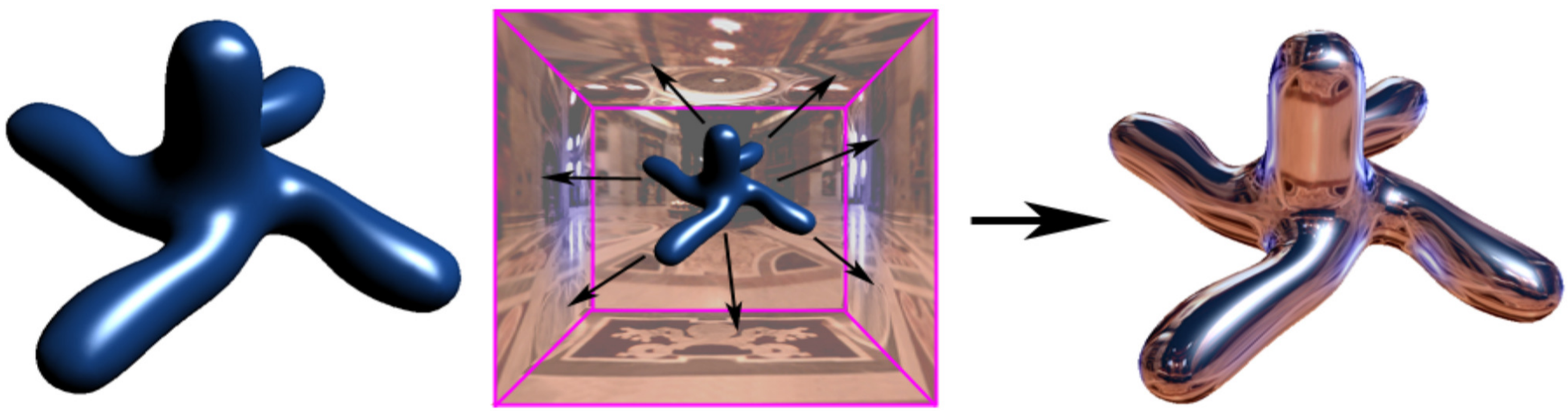
Cube map look-up

- 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`

UCSD

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

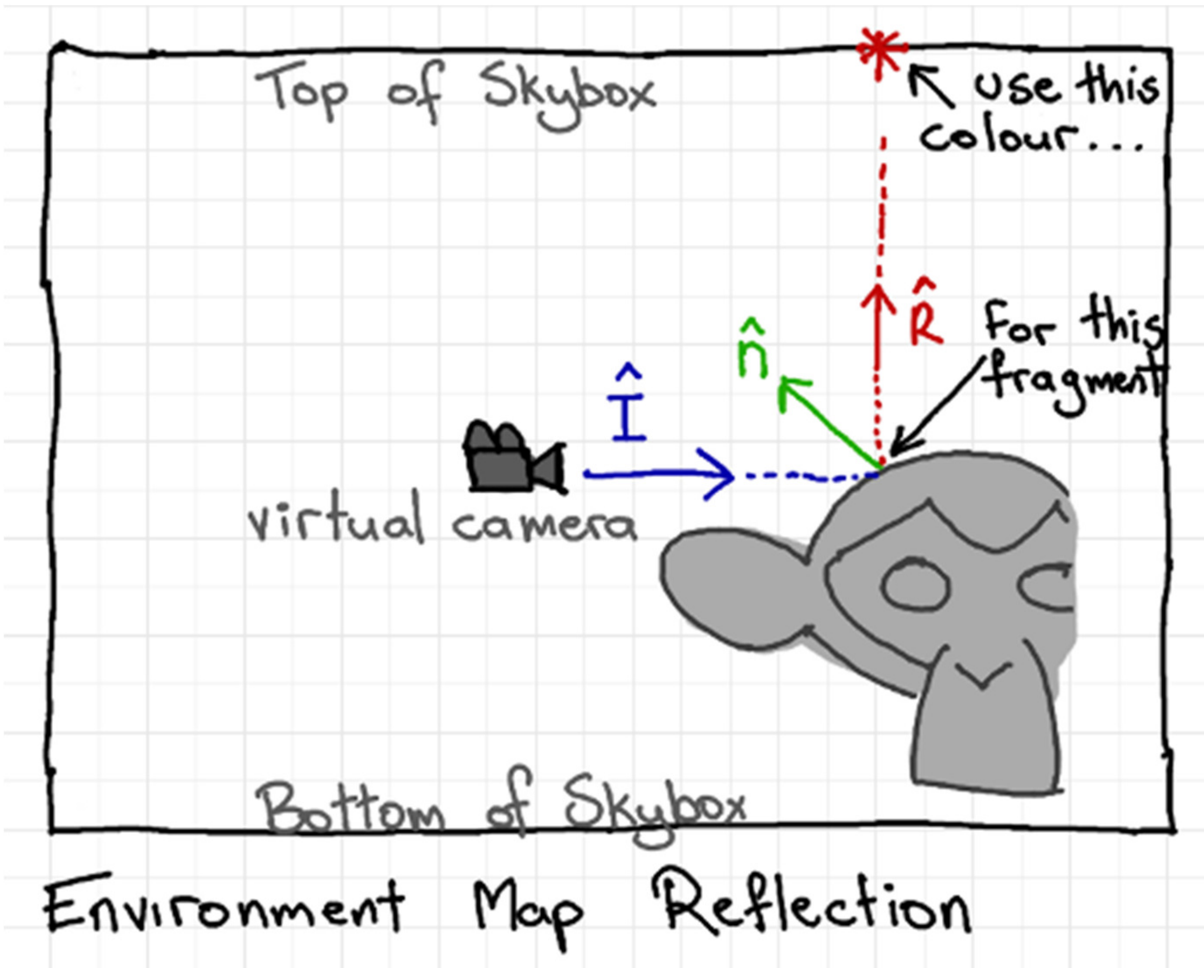
Reflection Mapping in GLSL

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);
```

Environment Mapping: Concept



Source: http://antongerdelan.net/opengl/cubemaps.html

UCSD

Environment Mapping: Vertex Shader

```
#version 400

in vec3 vp; // positions from mesh
in vec3 vn; // normals from mesh
uniform mat4 P, V, M; // proj, view, model matrices
out vec3 pos_eye;
out vec3 n_eye;

void main()
{
  pos_eye = vec3(V * M * vec4(vp, 1.0));
  n_eye = vec3(V * M * vec4(vn, 0.0));
  gl_Position = P * V * M * vec4(vp, 1.0);
}
```

Environment Mapping: Fragment Shader

```
#version 400

in vec3 pos_eye;
in vec3 n_eye;
uniform samplerCube cube_texture;
uniform mat4 V; // view matrix
out vec4 frag_colour;

void main()
{
  // reflect ray around normal from eye to surface
  vec3 incident_eye = normalize(pos_eye);
  vec3 normal = normalize(n_eye);

  vec3 reflected = reflect(incident_eye, normal);
  // convert from eye to world space
  reflected = vec3(inverse(V) * vec4(reflected, 0.0));

  frag_colour = texture(cube_texture, reflected);
}
```

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