Computergrafik

Matthias Zwicker
Universität Bern
Herbst 2016
Today

More shading

• Environment maps
• Reflection mapping
• Irradiance environment maps
• Ambient occlusion
• Reflection and refraction
• Toon shading
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 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 as light source at all locations 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?
Environment maps applications

- Use environment map as “light source”

Global illumination
[Sloan et al.]

Reflection mapping
Sphere & cube maps

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

Elevation
\( \theta \) const.
North pole
\( \theta = 90 \)
South pole
\( \theta = -90 \)

Elevation, azimuthal angle
\( \theta, \phi \)

Spherical map

Cube map

\([x, y, z]\)
Cube maps in OpenGL

Application setup

• Load, bind a cube environment map

```c
glBindTexture(GL_TEXTURE_CUBE_MAP, ...);
// the six cube faces
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);
```

• More details
  - “OpenGL Shading Language”, Randi Rost
  - “OpenGL Superbible”, Sellers et al.
  - Online tutorials
Cube maps in OpenGL

Look-up

- Given direction \((x,y,z)\)
- Largest coordinate component determines cube map face
- Dividing by magnitude of largest component yields coordinates within face
- Look-up function built into GLSL
  - Use \((x,y,z)\) direction as texture coordinates to `samplerCube`
Environment map data

- Also called „light probes“
  [http://www.debevec.org/Probes/](http://www.debevec.org/Probes/)

- Tool for high dynamic range data (HDR)
  [http://projects.ict.usc.edu/graphics/HDRShop/](http://projects.ict.usc.edu/graphics/HDRShop/)

- Pre-rendered light probes for games
  [http://docs.unity3d.com/Manual/LightProbes.html](http://docs.unity3d.com/Manual/LightProbes.html)
Today

More shading

- Environment maps
- Reflection mapping
- Irradiance environment maps
- Ambient occlusion
- Reflection and refraction
- Toon shading
Reflection mapping

- Simulate mirror reflection
- Compute reflection vector at each pixel using view direction and surface normal
- Use reflection vector to look up cube map
- Rendering cube map itself is optional
Reflection mapping in GLSL

**Vertex shader**

- Compute viewing direction for each vertex
- Reflection direction
  - Use GLSL built-in \texttt{reflect} function
- Pass reflection direction to fragment shader

**Fragment shader**

- Look-up cube map using interpolated reflection direction
  
  \begin{verbatim}
  in float3 refl;
  uniform samplerCube envMap;
  texture(envMap, refl);
  \end{verbatim}
Reflection mapping examples

• Approximation, reflections are not accurate
Today

More shading

- Environment maps
- Reflection mapping
- Irradiance environment maps
- Ambient occlusion
- Reflection and refraction
- Toon shading
Shading using environment map

- Assumption: distant lighting
  - Incident light is a function of direction, but not position

- Realistic shading requires
  - Take into account light from all directions
  - Include occlusion

“Illumination is a function of direction, but not position”
Mathematical model

- Assume Lambertian (diffuse) material, BRDF $k_d$
  - Ignore occlusion for now
- Illumination from point light sources
  \[ c = k_d \sum_i c_{l_i} (L_i \cdot n) \]
- Illumination from environment map using 
  hemispherical integral
  \[ c = k_d \int_{\Omega} c(\omega)(\omega \cdot n) d\omega \]
  - Directions $\omega$
  - Hemisphere of directions $\Omega$
  - Environment map, radiance from each direction $c(\omega)$
Irradiance environment maps

- Precompute *irradiance* as a function of normal
  \[ E(n) = \int_{\Omega} c(\omega)(\omega \cdot n) d\omega \]
  - Store as irradiance environment map

- Shading computation at render time
  - Depends only on normal, not position
  \[ c = k_d E(n) \]

Environment map  Irradiance map
Irradiance environment maps

Directional light

Environment illumination

Implementation

• Precompute irradiance map from environment
  - HDRShop tool, “diffuse convolution”
    [http://projects.ict.usc.edu/graphics/HDRShop/](http://projects.ict.usc.edu/graphics/HDRShop/)

• At render time, look up irradiance map using surface normal
  - When object rotates, rotate normal accordingly

• Can also approximate glossy reflection
  - Blur environment map less heavily
  - Look up blurred environment map using reflection vector
Today

More shading

• Environment maps
• Reflection mapping
• Irradiance environment maps
• Ambient occlusion
• Reflection and refraction
• Toon shading
Including occlusion

- At each point, environment is partially occluded by geometry
- Add light only from un-occluded directions

Visualization of un-occluded directions
Including occlusion

Visibility function $V_x(\omega)$

- Binary function of direction $\omega$
- Indicates if environment is occluded
- Depends on position $x$

Environment map

Visibility functions

$V_{x0} = 1$

$V_{x1} = 0$

$V_{x0} = 0$

$V_{x1} = 1$
Mathematical model

- Diffuse illumination with visibility

\[ c = k_d \int_{\Omega} V_x(\omega)c(\omega)(\omega \cdot n) d\omega \]

- Ambient occlusion

  - “Fraction” of environment that is not occluded from a point \( x \)
  
  - Scalar value

\[ a_x = \int_{\Omega} V_x(\omega)(\omega \cdot n) d\omega \]

- Approximation: diffuse shading given by irradiance weighted by ambient occlusion

\[ c = k_d a_x E(n) \quad E(n) = \int_{\Omega} c(\omega)(\omega \cdot n) d\omega \]
Ambient occlusion

http://en.wikipedia.org/wiki/Ambient_occlusion
Implementation

• **Precomputation** (off-line, before rendering)
  - Compute ambient occlusion on a per-vertex basis
  - Using ray tracing
  - Free tool that saves meshes with per-vertex ambient occlusion
    [http://www.xnormal.net/](http://www.xnormal.net/)

• **Caution**
  - Basic pre-computed ambient occlusion does not work for animated objects
Shading integral

- Ambient occlusion with irradiance environment maps is a crude approximation to the general shading integral:

\[ c(\omega_o) = \int_{\Omega} V_x(\omega_i)c(\omega_i)f(\omega_o, \omega_i)(\omega_i \cdot \mathbf{n})\,d\omega_i \]

- BRDF for (non-diffuse) material \( f(\omega_o, \omega_i) \)

\[
\begin{align*}
\text{Incident directions } & \omega_i \\
\text{Hemisphere } & \Omega \\
\text{Reflected radiance } & c(\omega_o) \\
\text{Outgoing direction } & \omega_o
\end{align*}
\]
Shading integral

- Accurate evaluation is expensive to compute
  - Requires numerical integration
- Many tricks for more accurate and general approximation than ambient occlusion and irradiance environment maps exist
  - Spherical harmonics shading
Note

• Visually interesting results using combination (sum) of diffuse shading with ambient occlusion and reflection mapping

Today

More shading

- Environment maps
- Reflection mapping
- Irradiance environment maps
- Ambient occlusion
- Reflection and refraction
- Toon shading
Reflection & refraction
• Light rays that travel from one medium to another are bent.

• To the viewer, object at location $x$ appears to be at location $y$. 

![Refraction diagram](image)
Index of refraction

http://en.wikipedia.org/wiki/Refractive_index

- Speed of light depends on medium
  - Speed of light in vacuum \( c \)
  - Speed of light in medium \( v \)
- Index of refraction \( n = \frac{c}{v} \)
  - Air 1.00029
  - Water 1.33
  - Acrylic glass 1.49
- “Change in phase velocity leads to bending of light rays”
Snell’s law


- Ratio of sines of angle of incidence $\theta_1$ and refraction $\theta_2$ is equal to opposite ratio of indices of refraction $n_1, n_2$

$$\frac{\sin \theta_1}{\sin \theta_2} = \frac{n_2}{n_1}$$

- Vector form in 3D

$$\mathbf{r} = \frac{n_1}{n_2} \mathbf{v} + \left( \frac{n_1}{n_2} \cos \theta_1 + \cos \theta_2 \right) \mathbf{n}$$

- Viewing, refracted direction $\mathbf{v}, \mathbf{r}$
- Normal vector $\mathbf{n}$
Total internal reflection

http://en.wikipedia.org/wiki/Total_internal_reflection

- Angle of refracted ray
  \[ \theta_2 = \arcsin \left( \theta_1 \frac{n_1}{n_2} \right) \]

- Critical angle
  \[ \theta_c = \frac{n_2}{n_1} \]

- If \( \theta_1 = \theta_c \) we get \( \theta_2 = \pi/2 \), refracted ray is parallel to interface

- If \( \theta_1 > \theta_c \) we have total internal reflection (light ray does not cross interface between media)
Fresnel equations

http://en.wikipedia.org/wiki/Fresnel_equations

- When light travels from one medium to another, both reflection and refraction may occur.

- Fresnel equations describe the fraction of intensity of light that is reflected and refracted.
  - Depend on polarization of light ($R_s$, $R_p$ in plots).
Fresnel equations

- Fresnel equations are relatively complex to evaluate
- In graphics, often use Schlick’s approximation
  
  - Ratio $F$ between reflected and refracted light
  - Indices of refraction $n_1, n_2$

\[
F = f + (1 - f)(1 - v \cdot n)^5
\]
\[
f = \frac{\left(1.0 - \frac{n_1}{n_2}\right)^2}{\left(1.0 + \frac{n_1}{n_2}\right)^2}
\]
Implementation

• Accurate implementation requires ray tracing

• For interactive graphics, approximation using environment maps
  - Use reflected and refracted rays to look up environment map
  - Use Fresnel equations to determine fraction of reflected and refracted light
  - Does not take into account geometry after first bounce (i.e., surface intersection)
  - Assumes illumination is infinitely far away
const float Eta = 0.67; // Ratio of indices of refraction (air -> glass)
const float FresnelPower = 10.0; // Controls degree of reflectivity at grazing angles

const float F = ((1.0 - Eta) * (1.0 - Eta)) / ((1.0 + Eta) * (1.0 + Eta));

varying vec3 Reflect;
varying vec3 Refract;
varying float Ratio;

void main(void)
{
    vec4 ecPosition = gl_ModelViewMatrix * gl_Vertex;
    vec3 ecPosition3 = ecPosition.xyz / ecPosition.w;

    vec3 i = normalize(ecPosition3);
    vec3 n = normalize(gl_NormalMatrix * gl_Normal);

    Ratio = F + (1.0 - F) * pow((1.0 - dot(-i, n)), FresnelPower);

    Refract = refract(i, n, Eta);
    Refract = vec3(gl_TextureMatrix[0] * vec4(Refract, 1.0));

    Reflect = reflect(i, n);
    Reflect = vec3(gl_TextureMatrix[0] * vec4(Reflect, 1.0));

    gl_Position = ftransform();
}
Fragment shader

• Application needs to set up cube map

```glsl
varying vec3 Reflect;
varying vec3 Refract;
varying float Ratio;

uniform samplerCube cubemap;

void main(void)
{
    vec3 refractColor = vec3 (textureCube(cubemap, Refract));
    vec3 reflectColor = vec3 (textureCube(cubemap, Reflect));

    vec3 color = mix(refractColor, reflectColor, Ratio);

    gl_FragColor = vec4(color, 1.0);
}
```

CAUTION: need to update this for OpenGL 3 compliance
Chromatic dispersion


• Phase velocity (i.e., index of refraction) in many media depends on wavelength/frequency
  - Dispersive media
• Different colors refract at different angles
Chromatic dispersion

- In the context of camera lenses, chromatic aberration
  - Try to use *achromatic* lenses
Implementation

• Approximate dispersion by using three different ratios of indices of refraction for R,G,B channels
  - Glass: 0.65, 0.67, 0.69
• Perform separate look-ups for R,G,B channels in environment map
Today

More shading

• Environment maps
• Reflection mapping
• Irradiance environment maps
• Ambient occlusion
• Reflection and refraction
• Toon shading
Toon shading

- Simple cartoon style shader
- Emphasize silhouettes
- Discrete steps for diffuse shading, highlights
- Sometimes called CEL shading
  
  http://en.wikipedia.org/wiki/Cel-shaded-animation
Toon shading

• Silhouette edge detection
  - Compute dot product of viewing direction $v$ and normal $n$

    $$edge = \max(0, \mathbf{n} \cdot \mathbf{v})$$

  - Use 1D texture to define edge ramp

    uniform sample1D edgeramp; e=texture1D(edgeramp,edge);
Toon shading

- Compute diffuse and specular shading
  \[ \text{diffuse} = \mathbf{n} \cdot \mathbf{L} \quad \text{specular} = (\mathbf{n} \cdot \mathbf{h})^s \]

- Use 1D textures diffuseramp, specularramp to map diffuse and specular shading to colors

- Final color
  \[
  \begin{align*}
  \text{uniform sampler1D diffuseramp; } \\
  \text{uniform sampler1D specularramp; } \\
  c &= e \ast (\text{texture(diffuse,diffuseramp)} + \text{texture(specular,specularramp)});
  \end{align*}
  \]
Tools

- Nvidia developer page
Next time

- Bump mapping, shadows