Chapter 6 - Light, Materials, Appearance

• Types of light in nature and in CG
• Shadows
• Using lights in CG
• Illumination models
• Textures and maps
• Procedural surface descriptions
Light in Nature (Physics Refresher)

- Can be described as a electromagnetic wave
- Can also be described as a stream of photons
- Intensity drops with distance from the source
  - how?
    - __________________________
- Monochromatic (1 color) light has 1 frequency
- White light is a mixture of many frequencies
- Can be simulated for the human eye by adding Red, Green and Blue

- The human eye can discriminate a dynamic range of $1:2^{30}$ with adaptation or $1:2^{16}$ without [Seetzen et al., „High dynamic range display systems“, ACM Siggraph 2004]
- Film, digital cameras, and computer screens can only deal with less!

Point Light Sources

- have just a position in space
- emit light equally in all directions
- Intensity falloff with distance d is:
  \[ I = \frac{l_0}{(ad^2 + bd)} \]
  - this means that the falloff is less harsh than in nature. Why??

- Light source itself is invisible in the image
  - since points are infinitely small

- Shadows have sharp edges
- Shadows get bigger with distance from object
Spot Lights

- have a position and orientation in space
- have an opening angle and a parameter controlling the softness of the beam’s borders
- Intensity falloff with distance \( d \) is:
  \[ I = \frac{I_0}{(ad^2 + bd)} \]
  – this means that the falloff is less harsh than in nature. Why??
- Intensity falloff with angle depends on exact model
- light source itself invisible
- object shadows have sharp edges
- transition to surrounding shadow is soft.

Distant Light Source (a.k.a. the sun)

- size of the earth (radius) = 6.370 Km
- distance from earth to sun = 150.000.000 Km
- distance to the sun is practically equal for all points on earth
  - hence light falloff with distance is not noticeable for sunlight

- distant light source in 3DCG has only a direction and a fixed intensity
- good and neutral first step for lighting a scene!
- shadows should have sharp edges
Ambient Light

• Equivalent in nature:
  – light emitted from the entire sky
  – indirect light reflected from objects in the scene

• intensity is equal from all directions
• creates low contrast images by itself

• ambient light is a good way to light up harsh shadows
• combination with one distant light can already create a decent daylight simulation (sun + sky)

Area Light Sources

• described by object geometry and light intensity
• entire area emits light
• all natural light sources are of this kind
  – even a light bulb has a surface > 0

• shadows have soft edges
• light falloff with distance

• computationally difficult, take very long to render correctly
• can be simulated by many point light sources
• need global illumination techniques for correct rendering (see later)
The 3D Rendering Pipeline (our version for this class)

3D models in model coordinates → 3D models in world coordinates → 2D Polygons in camera coordinates → Pixels in image coordinates

Scene graph → Camera → Rasterization

Animation, Interaction → Lights
Chapter 6 - Light, Materials, Appearance

- Types of light in nature and in CG
- Shadows
- Using lights in CG
- Illumination models
- Textures and maps
- Procedural surface descriptions
Shadows in Nature

• Very important for spatial vision

• Artistically used in all art forms
  – drawing, painting
  – photography
  – cinematography

• Practically never really black

• Types of shadows in this image?

http://www.heise.de/imaqeine/Vz2PeXewMuStFADy2UvZXFsFUK/gallery/shadow-lines.jpg
Types of Shadow

• Object shadow
  – the shadow side of objects
  – exists in free space

• Cast shadow / drop shadow
  – the shadow cast onto another object (or the ground)
  – need another object or ground plane

• Shadow as the absence of light
  – no light source reaches this place
Cheating a Shadow (and a Reflection!)

• Try to guess how this simple VRML world creates shadows and reflections in real time!
Shadow Maps

• From the position of a light source, record a depth buffer
  – for each pixel in buffer, we know how far from the light it is

• For each rendered pixel in the camera image, check distance of its surface point to the light
  – if closer than shadow buffer: in this light
  – If further away: in the shadow of this light

• If scene or lights change, shadow map must be recalculated
Chapter 6 - Light, Materials, Appearance

- Types of light in nature and in CG
- Shadows

Using lights in CG
- Illumination models
- Textures and maps
- Procedural surface descriptions
Using Lights

• A few recipes to get started with lighting
• Really good lighting design is an art in itself
  – 3D animated movies hire full time light designers
Headlight

• Default light setup in VRML
• Light source in camera position
• Scene can be viewed from arbitrary directions
• Creates no visible drop shadows – why? or does it?

• Creates rather „flat“ images
• Unnatural „flashlight“ look
• Good in combination with other setups for lighting up the scene
Daylight Simulation

- Sun
  - distant light source
  - warm color tint

- Sky
  - ambient light
  - cool color tint
  - can be simulated by directional light from opposite side

- creates a natural look
- can simulate daytimes (how??)
- can simulate sunny/cloudy weather (how??)
Simple Portrait Light Setup

• Borrows ideas from daylight
  – 1 main light source
  – direction: traditionally from top left
  – creates overall basic brightness

• One or several brighteners
  – from opposite sides
  – to light up shadows
  – sum of their brightness less than half of main light (why?)

• Basic setup for scenes viewed from just 1 direction
Sided Light

• Effect light known from movies
  – use only in addition to others
• Enhances object contours
• Placement behind the subject
  – not straight behind, but off-axis
  – positioning is difficult in real world
  – easier in graphics, but still:
  – highly position-dependent

• Can be used to clearly separate an object from the background.
• will highlight its silhouette.
Cheating with Light

• Light sources in computer graphics are invisible
  – only their effects on objects are visible!

• Can be positioned anywhere in a scene to light up dark areas

• Example on this slide is exaggerated!
Dramatic Lighting

• Combination of unnatural lights
  – coming from below
  – strong colors
  – mostly low key

• Unlit shadows can create mystery

• Can be supported by unnatural camera
  – from below
  – wide angle and close up
High Key, Low Key

• High Key: all colors in image are bright
  – start with very even lighting
  – frontal light will remove shadows
  – danger of saturated white
  – communicates light and cleanliness

• Low Key: all colors are very dark
  – often uses sided light
  – objects can be reduced to their contours
  – communicates e.g., mystery
Chapter 6 - Light, Materials, Appearance

• Types of light in nature and in CG
• Shadows
• Using lights in CG
  • Illumination models
• Textures and maps
• Procedural surface descriptions
Surfaces in Nature

• What does a surface do to light? (mini-Brainstorming)
The Rendering Equation [Kajiya ‘86]

\[ I_o(x, \vec{\omega}) = I_e(x, \vec{\omega}) + \int_{\Omega} f_r(x, \vec{\omega}', \vec{\omega}) I_i(x, \vec{\omega}') (\vec{\omega}' \cdot \vec{n}) d\vec{\omega}' \]

- \( I_o \) = outgoing light
- \( I_e \) = emitted light
- Reflectance Function
- \( I_i \) = incoming light
- angle of incoming light

- Describes all flow of light in a scene in an abstract way
- doesn't describe some effects of light:


\[ I_o = I_{amb} + I_{diff} + I_{spec} \]

• strong simplification and specialization of the situation
  – just 1 light source from a clear direction \( l \)
  – viewing direction is given as \( v \)

• only 3 components:
  – ambient component: reflection of ambient light source from and in all directions
  – diffuse component: diffuse reflection of the given light source in all directions
  – specular component: „glossy“ reflection creating specular highlights

Ambient Component

• $I_a = \text{Intensity of the ambient light source}$

• independent of any directions

• can simulate a „glowing in the dark“

• can be seen as the equivalent to emitted light $I_e$ in the rendering equation

$$I_{amb} = I_a k_a$$
Diffuse Component

• diffuse reflection is equal in all directions
• depends on the angle of incident light
  – light along the surface normal: maximum
  – light perpendicular to the normal: 0

• cosine function describes the energy by which a given area is lit, dep. on angle
  – hence, cosine is used here

• „Lambertian“ surface
• visual equivalent in nature: paper

\[ I_{diff} = I_i k_d \cos \phi = I_i k_d (\vec{l} \cdot \vec{n}) \]
Specular Reflection

• let \( r \) be the reflection of \( l \) on the surface

• specular reflection depends on the angle between \( v \) and \( r \)

• \( v = -r \): maximum

• \( v \) and \( r \) perpendicular: minimum

• function \( \cos^n \) behaves correctly
  – exponent \( n \) determines how wide the resulting specular highlight is
  – other functions could be used as well
Tweaking the Parameters

\[ I_o = I_{amb} + I_{diff} + I_{spec} = I_a k_a + I_i k_d (\vec{l} \cdot \vec{n}) + I_i k_s (\vec{r} \cdot \vec{v})^n \]

- Choose \( k_s = 0 \) for perfectly matte material
- Choose \( k_a > 0 \) to avoid harsh shadows
- Keep \( k_a \) small to avoid „glowing“ objects
- Add in some \( k_s > 0 \) to add gloss
- Adjust the size of specular highlights with \( n \)

- All of these calculations generalize to (RGB) color, of course!
The VRML Material Node

Material {
  exposedField SFFloat ambientIntensity 0.2 # [0,1]
  exposedField SFColor diffuseColor 0.8 0.8 0.8 # [0,1]
  exposedField SFColor emissiveColor 0 0 0 # [0,1]
  exposedField SFFloat shininess 0.2 # [0,1]
  exposedField SFColor specularColor 0 0 0 # [0,1]
  exposedField SFFloat transparency 0 # [0,1]
}

shininess in VRML is multiplied by 128 to produce n in the lighting model.
Chapter 6 - Light, Materials, Appearance

• Types of light in nature and in CG
• Shadows
• Using lights in CG
• Illumination models

• Textures and maps
• Procedural surface descriptions
Textures and Maps

• one of the simplest and oldest ways to achieve good looking objects with simple geometry
• texture design is a very complex task, needs a lot of imagination!
• idea: use a bitmap image, shrink wrap around the object
• use bitmap contents for object surface color: image map
  – can be used for other parameters, e.g., normal, elevation, transparency, reflection
• problem: what does shrink wrap mean exactly?
Texture Coordinates and UV Mapping

• each texture is mapped to a 1x1 square
• each object defines u,v coordinates
  – such that u,v are both between 0 and 1

• straightforward for geometric primitives
  – different possibilities
  – conventions exist

• not so easy for polygon models
  – can be defined per vertex
  – ...but who wants to do this?
  – simplifications: shrink a sphere onto the object
    • works fine with convex objects
    • always tricky for complicated objects
Texture Filtering

- During rasterization, for each rendered pixel of the textured object we need to look up a color value from the texture
  - will almost always fall between texture pixels (texels)
  - texture may have too much resolution: sampling or integration
  - texture may have too little resolution: interpolation
- naive approach: pick the nearest neighbor pixel
  - leads to blocky textures
- better approach: bilinear filtering
  - pick the 4 neighboring pixels and linearly interpolate
- Mip map: image pyramid with image scaled to 1/4 area in each step
  - eliminates excessive integration over pixels
- trilinear filtering: find the 2 best levels of the mip map and interpolate within and between them
Bump Mapping

• texture file is only greyscale
• grey value determines the elevation of the surface
  – e.g., black = dent, white = bulge

• can simulate complex 3D surface structure on very simple geometry
• often used together with image maps to enhance realism
• only modifies surface color, not silhouette!

• introduced by Jim Blinn in 1978
  – related and improved techniques with similar look in use today: normal mapping, displacement mapping

http://en.wikipedia.org/wiki/Bump_mapping
Environment Maps

• maps show the environment of the object
  – inside out view, 360 degrees in all directions
  – can be represented as 6 sides of a cube
  – can be photographed in a real environment

• can be used to calculate appropriate reflections
  – problem: ________________________________

• can also be used for lighting
  – record map in real environment
  – light a 3D model with it
  – this model will seem as if lit in the real environment
  – useful for combining real and virtual objects
Chapter 6 - Light, Materials, Appearance

• Types of light in nature and in CG
• Shadows
• Using lights in CG
• Illumination models
• Textures and maps
• Procedural surface descriptions
Procedural Surface Descriptions

• programming languages for surface descriptions
• can influence various stages of the rendering pipeline
  – in particular: can implement textures and the phong model
  – but also much more...
• can describe real 3D structures
  – not just surface color
• state of the art in high end 3D graphics
  – e.g., RenderMan, used in PIXAR movies
  – also in OpenGL, DirectX
• detailed implementation varies depending on the platform
• in OpenGL: vertex shaders and fragment shaders
  – fragments = parts of an object that cover 1 screen pixel

OpenGL: Vertex and Fragment Shaders

- A vertex shader can do the following:
  - transform the vertex position using the modelview and projection matrices
  - transform normals, and if required normalize them
  - generate and transform texture coordinates
  - lighting per vertex or compute values for lighting per pixel
  - color computation

- A fragment shader can do the following:
  - compute colors, and texture coordinates per pixel
  - apply a texture
  - fog computation
  - compute normals if you want lighting per pixel

- This, and more details at: http://www.lighthouse3d.com/opengl/gls/
Links, Various

• You‘ve been doing 3DCG too long if ...when people ask you, "What's up?", you reply "Y": http://www.deakin.edu.au/~agoodman/scc308/toolong.html

• Detailed class material from one of the world‘s leading groups: http://graphics.stanford.edu/courses/

• Compact overviews from the wisdom of the masses ;-) : http://en.wikipedia.org/wiki/Rendering_(computer_graphics)


• Some nice tutorials related to this class: http://www.lighthouse3d.com/