Lighting, Illumination, and Shading

CPSC 314

The Rendering Pipeline

Geometry Database → Model/View Transform. → Lighting → Perspective Transform. → Clipping

Scan Conversion → Texturing
Depth Test → Blending

Frame-buffer

Rasterization
Fragment Processing
Homogeneous Coordinates

**Homogeneous representation of points:**

- Add an additional component $w=1$ to all points
- All multiples of this vector are considered to represent the same 3D point
- All points are represented as column vectors

\[
\begin{pmatrix}
  x \\
  y \\
  z \\
  1
\end{pmatrix}
= \begin{pmatrix}
  x \\
  y \\
  z \\
  w
\end{pmatrix} = \begin{pmatrix}
  x' \\
  y' \\
  z' \\
  w
\end{pmatrix}, \quad \forall w \neq 0
\]

Projective Rendering Pipeline

<table>
<thead>
<tr>
<th>object OCS</th>
<th>world WCS</th>
<th>viewing VCS</th>
<th>clipping CCS</th>
</tr>
</thead>
<tbody>
<tr>
<td>modeling transformation</td>
<td>viewing transformation</td>
<td>projection transformation</td>
<td>perspective divide</td>
</tr>
<tr>
<td>OCS - object/model coordinate system</td>
<td>WCS - world coordinate system</td>
<td>VCS - viewing/camera/eye coordinate system</td>
<td>CCS - clipping coordinate system</td>
</tr>
<tr>
<td>CCS - clipping coordinate system</td>
<td>NDCS - normalized device coordinate system</td>
<td>DCS - device/display/screen coordinate system</td>
<td></td>
</tr>
</tbody>
</table>
Perspective Projection

**Example:**
- Assume image plane at \( z = -1 \)
- A point \([x, y, z, I]^T\) projects to \([\frac{-x}{z}, \frac{-y}{z}, \frac{-z}{z}, I]\) = \([x, y, z, -z]^T\)

![Diagram of perspective projection]

**Analysis:**
- This is a special case of a general family of transformations called *projective transformations*
- These can be expressed as 4x4 homogeneous matrices!
  - *E.g. in the example:*

\[
\begin{bmatrix}
    x \\
    y \\
    z \\
    1
\end{bmatrix} =
\begin{bmatrix}
    1 & 0 & 0 & 0 \\
    0 & 1 & 0 & 0 \\
    0 & 0 & 1 & 0 \\
    0 & 0 & -1 & 0
\end{bmatrix}
\begin{bmatrix}
    x \\
    y \\
    z \\
    1
\end{bmatrix} =
\begin{bmatrix}
    -\frac{x}{z} \\
    -\frac{y}{z} \\
    -1 \\
    1
\end{bmatrix}
\]
**Projective Transformations**

**OpenGL Convention**

Camera coordinates vs. NDC:

- **Frustum**
  - Camera coordinates: $x$, $y$, $z$,
  - NDC: $x'$, $y'$, $z'$
  - $x' = x$, $y' = y$, $z' = z f / n$

**Perspective Matrices in OpenGL**

**Perspective Matrices:**
- `glFrustum( left, right, bottom, top, near, far )`
  - Specifies perspective xform (near, far are always positive)
- `glOrtho( left, right, bottom, top, near, far )`

**Convenience Functions:**
- `gluPerspective( fovy, aspect, near, far )`
  - Another way to do perspective
- `gluLookAt( eyeX, eyeY, eyeZ, centerX, centerY, centerZ, upX, upY, upZ )`
  - Useful for viewing transform
The Rendering Pipeline

Geometry Processing

Geometry Database → Model/View Transform. → Lighting → Perspective Transform. → Clipping

Scan Conversion → Texturing → Depth Test → Blending

Rasterization → Fragment Processing

Frame-buffer

© Wolfgang Heidrich
Illumination

Goal
Model interaction of light with matter in a way that appears realistic and is fast

- Phenomenological reflection models
  - *Ignore real physics, approximate the look*
  - *Simple, non-physical*
  - *Phong, Blinn-Phong*

- Physically based reflection models
  - *Simulate physics*
  - *BRDFs: Bidirectional Reflection Distribution Functions*

Photorealistic Illumination

[electricimage.com]
©Wolfgang Heidrich
Photorealistic Illumination

Fast Local Illumination
Illumination

- Transport of energy from light sources to surfaces & points
  - Includes direct and indirect illumination

Components of Illumination

Two components
- Light sources and surface properties

Light sources (or emitters)
- Spectrum of emittance (i.e., color of the light)
- Geometric attributes
  - Position
  - Direction
  - Shape
- Directional attenuation
- Polarization
Components of Illumination

**Surface properties**
- Reflectance spectrum (i.e., color of the surface)
- Subsurface reflectance
- Geometric attributes
  - Position
  - Orientation
  - Micro-structure

Illumination as Radiative Transfer

**Radiative heat transfer analogon**
- Substitute light for heat
- Light as packets of energy (photons)
  - *Particles not waves*
- Model light transport as packet flow
Light Transport Assumptions

**Geometrical optics:**
- Light is photons not waves)
- No diffraction
- No polarization (some sunglasses)
  - Light of all orientations gets through
- No interference (packets don’t interact)
  - Which visual effects does this preclude?

Light Transport Assumptions II

**Color approximated by discrete wavelengths**
- Quantized approx of dispersion (rainbows)
- Quantized approx of fluorescence (cycling vests)

**No propagation media (surfaces in vacuum)**
- No
  - Atmospheric scattering (fog, clouds)
  - Refraction (mirages)
  - Gravity lenses
- But methods exist for all these effects

**Superposition (lights can be added)**
- No nonlinear reflection models
  - Pretty good assumption (only few non-linear materials)
Light Sources and Materials

**Appearance depends on**
- Light sources, locations, properties
- Material (surface) properties
- Viewer position

**Local illumination**
- Compute at material, from light to viewer

**Global illumination (later in course)**
- Ray tracing: from viewer into scene
- Radiosity: between surface patches

Illumination in the Rendering Pipeline

**Local illumination**
- Only models light arriving directly from light source
- No interreflections and shadows
  - *Can be added through tricks, multiple rendering passes*

**Light sources**
- Simple shapes

**Materials**
- Simple, non-physical reflection models
Light Sources

Types of light sources

- Directional/parallel lights
  - E.g. sun
  - Homogeneous vector
- (Homogeneous) point lights
  - Same intensity in all directions
  - Homogeneous point
- Spot lights
  - Limited set of directions
  - Point + direction + cutoff angle

Area lights:

- Light sources with a finite area
- Can be considered a continuum of point lights
- Not available in many rendering systems
Light Sources

*ambient lights*
- no identifiable source or direction
- hack for replacing true global illumination
  - (light bouncing off from other objects)

Ambient Light Sources
- Scene lit only with an ambient light source

- Light Position
  - Not Important
- Viewer Position
  - Not Important
- Surface Angle
  - Not Important
**Directional Light Sources**

- Scene lit with directional and ambient light

**Point Light Sources**

- Scene lit with ambient and point light source
Light Sources

Geometry: positions and directions
- Standard: world coordinate system
  - Effect: lights fixed wrt world geometry
  - Demo: http://www.xmission.com/~nate/tutors.html
- Alternative: camera coordinate system
  - Effect: lights attached to camera (car headlights)
- Points and directions undergo normal model/view transformation

illumination calculations: camera coods

Types of Reflection
- Specular (a.k.a. mirror or regular) reflection causes light to propagate without scattering.
- Diffuse reflection sends light in all directions with equal energy.
- Mixed reflection is a weighted combination of specular and diffuse.
Types of Reflection

- *retro-reflection* occurs when incident energy reflects in directions close to the incident direction, for a wide range of incident directions.

- *gloss* is the property of a material surface that involves mixed reflection and is responsible for the mirror-like appearance of rough surfaces.

Reflectance Distribution Model

*Most surfaces exhibit complex reflectances*

- Vary with incident and reflected directions.
- Model with combination

\[
\text{specular} + \text{glossy} + \text{diffuse} = \text{reflectance distribution}
\]
Surface Roughness

- at a microscopic scale, all real surfaces are rough
- cast shadows on themselves
- “mask” reflected light:

Notice another effect of roughness:
- Each “microfacet” is treated as a perfect mirror.
- Incident light reflected in different directions by different facets.
- End result is mixed reflectance.
  - Smoother surfaces are more specular or glossy.
  - Random distribution of facet normals results in diffuse reflectance.
Physics of Diffuse Reflection

**Ideal diffuse reflection**
- Very rough surface at the microscopic level
  - *Real-world example: chalk*
- Microscopic variations mean incoming ray of light equally likely to be reflected in any direction over the hemisphere
- Reflected intensity only depends on light direction!

---

Lambert’s “Law”

Intuitively: cross-sectional area of the “beam” intersecting an element of surface area is smaller for greater angles with the normal.
Computing Diffuse Reflection

- Depends on angle of incidence: angle between surface normal and incoming light
  \[ I_{\text{diffuse}} = k_d \cdot I_{\text{light}} \cdot \cos \theta \]
- In practice use vector arithmetic
  \[ I_{\text{diffuse}} = k_d \cdot I_{\text{light}} \cdot (n \cdot l) \]
- Always normalize vectors used in lighting
  - \( n \), \( l \) should be unit vectors
- Scalar (B/W intensity) or 3-tuple or 4-tuple (color)
  - \( k_d \): diffuse coefficient, surface color
  - \( I_{\text{light}} \): incoming light intensity
  - \( I_{\text{diffuse}} \): outgoing light intensity (for diffuse reflection)

Diffuse Lighting Examples

- Lambertian sphere from several lighting angles:

- need only consider angles from 0° to 90°
Specular Reflection

*Shiny surfaces exhibit specular reflection*
- Polished metal
- Glossy car finish

*Specular highlight*
- Bright spot from light shining on a specular surface

*View dependent*
- Highlight position is function of the viewer’s position

---

Physics of Specular Reflection

- At the microscopic level a specular reflecting surface is very smooth
- Thus rays of light are likely to bounce off the microgeometry in a mirror-like fashion
- the smoother the surface, the closer it becomes to a perfect mirror

© Wolfgang Heidrich
Optics of Reflection

Reflection follows Snell’s Law:
- Incoming ray and reflected ray lie in a plane with the surface normal
- Angle the reflected ray forms with surface normal equals angle formed by incoming ray and surface normal

\[ \theta_{\text{light}} = \theta_{\text{reflection}} \]

Non-Ideal Specular Reflectance
- Snell’s law applies to perfect mirror-like surfaces, but aside from mirrors (and chrome) few surfaces exhibit perfect specularity
- How can we capture the “softer” reflections of surface that are glossy, not mirror-like?
- One option: model the microgeometry of the surface and explicitly bounce rays off of it

or...
Empirical Approximation

- We expect most reflected light to travel in direction predicted by Snell’s Law

- But because of microscopic surface variations, some light may be reflected in a direction slightly off the ideal reflected ray

- As angle from ideal reflected ray increases, we expect less light to be reflected

Empirical Approximation

Angular fall off

how might we model this fall off?
**Phong Lighting**

*Most common lighting model in computer graphics*

– (Phong Bui-Tuong, 1975)

\[ I_{\text{specular}} = k_s I_{\text{light}} (\cos \phi)^{n_{\text{shiny}}} \]

- \( n_{\text{shiny}} \): purely empirical constant, varies rate of falloff
- \( k_s \): specular coefficient, highlight color
- no physical basis, works ok in practice

**Phong Lighting: The \( n_{\text{shiny}} \) Term**

- Phong reflectance term drops off with divergence of viewing angle from ideal reflected ray

*what does this term control, visually?*

Viewing angle – reflected angle
Phong Examples

- varying $I$
  - 
- varying $n_{\text{shiny}}$
  - 

Calculating Phong Lighting

- compute cosine term of Phong lighting with vectors

$$I_{\text{specular}} = k_S I_{\text{light}} (v \cdot r)^n_{\text{shiny}}$$

- $v$: unit vector towards viewer/eye
- $r$: ideal reflectance direction (unit vector)
- $k_S$: specular component
  - highlight color
- $I_{\text{light}}$: incoming light intensity

*how to efficiently calculate $r$?*
Computing the Reflected Direction

Specular/Glossy

- Computing reflection direction $r_1$ of $l$
  - $n$ and $l$ are unit length!

\[ r_1 = 2(n \cdot l) \cdot n - l \]

Phong Lighting: Intensity Plots

<table>
<thead>
<tr>
<th>Phong</th>
<th>$\rho_{ambient}$</th>
<th>$\rho_{diffuse}$</th>
<th>$\rho_{specular}$</th>
<th>$\rho_{total}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\phi = 60^\circ$</td>
<td><img src="image1" alt="Image" /></td>
<td><img src="image2" alt="Image" /></td>
<td><img src="image3" alt="Image" /></td>
<td><img src="image4" alt="Image" /></td>
</tr>
<tr>
<td>$\phi = 25^\circ$</td>
<td><img src="image5" alt="Image" /></td>
<td><img src="image6" alt="Image" /></td>
<td><img src="image7" alt="Image" /></td>
<td><img src="image8" alt="Image" /></td>
</tr>
<tr>
<td>$\phi = 0^\circ$</td>
<td><img src="image9" alt="Image" /></td>
<td><img src="image10" alt="Image" /></td>
<td><img src="image11" alt="Image" /></td>
<td><img src="image12" alt="Image" /></td>
</tr>
</tbody>
</table>
**Alternative Model**

*Blinn-Phong model (Jim Blinn, 1977)*

- Variation with better physical interpretation
  - \( h \): halfway vector; \( r \): roughness

\[
I_{out}(x) = k_t \cdot (h \cdot n)^{1/r} \cdot I_{in}(x); \text{ with } h = (l + v) / 2
\]

**Light Source Falloff**

*Quadratic falloff*

- Brightness of objects depends on power per unit area that hits the object
- The power per unit area for a point or spot light decreases quadratically with distance
Light Source Falloff

**Non-quadratic falloff**

- Many systems allow for other falloffs
- Allows for faking effect of area light sources
- OpenGL / graphics hardware
  - $I_o$: intensity of light source
  - $x$: object point
  - $r$: distance of light from $x$

\[
I_{in}(x) = \frac{1}{ar^2 + br + c} \cdot I_0
\]

Lighting Review

**Lighting models**

- Ambient
  - *Normals don’t matter*
- Lambert/diffuse
  - *Angle between surface normal and light*
- Phong/specular
  - *Surface normal, light, and viewpoint*
Lighting in OpenGL

**Light source: amount of RGB light emitted**
- Value represents percentage of full intensity
  - E.g., (1.0,0.5,0.5)
- Every light source emits ambient, diffuse, and specular light

**Materials: amount of RGB light reflected**
- Value represents percentage reflected
  - E.g., (0.0,1.0,0.5)

**Interaction: multiply components**
- Red light (1,0,0) x green surface (0,1,0) = black (0,0,0)

```cpp
glLightfv(GL_LIGHT0, GL_AMBIENT, amb_light_rgba );
glLightfv(GL_LIGHT0, GL_DIFFUSE, dif_light_rgba );
glLightfv(GL_LIGHT0, GL_SPECULAR, spec_light_rgba );
glLightfv(GL_LIGHT0, GL_POSITION, position);
glEnable(GL_LIGHT0);

glMaterialfv( GL_FRONT, GL_AMBIENT, ambient_rgba );
glMaterialfv( GL_FRONT, GL_DIFFUSE, diffuse_rgba );
glMaterialfv( GL_FRONT, GL_SPECULAR, specular_rgba );
glMaterialfv( GL_FRONT, GL_SHININESS, n );
```
Coming Up

**Thursday:**
- Shading (Part 1)

**Tuesday:**
- Shading (Part 2)
- Quiz 1