Shading and Illumination

CSC 292:
Computer Graphics

Randal C. Nelson

Issues

- Want methods for determining the color of each pixel to provide (more or less) realistic-looking and/or informative depictions of objects in the scene.
- *Illumination Model*: Mathematical model of how light interacts with objects.
- *Shading Model*: How illumination model is applied to an object representation.
- Much more complex physical processes than visible surface determination.
- Accurate models of actual physical process of light interaction often do not exist, or are too complex to be practical.
- Much of what is used is a collection of "hacks" and empirical models that have proven to be useful and/or efficient.

Ambient Light

- Each surface/object has constant color, independent of angle, position etc.
- Corresponds to effect of illumination coming equally from all directions, called *ambient light*.
- Can be modeled by equation $I = I_a k_a$.
- I is intensity on the screen.
- I_a is the intensity of the ambient light.
- k_a is a constant between 0 and 1 called the *ambient reflection coefficient*, which is a *material property*.
- Not realistic.
- Not very informative.
- Sometimes used to account for unmodeled residual.

Diffuse Reflection

- Now consider a *point light source*.
- Brightness of object will vary, depending on angle and distance.
- Simplest model is *Lambertian* or *diffuse* reflection.
- Surface appears equally bright in all directions.
- Modeled by equation $I = I_p k_d \cos\theta$ where θ is the angle of incidence.
- For normalized vectors, $\cos\theta$ can be represented as the dot product $\bar{N} \cdot \bar{L}$ of the surface normal $\bar{N}$ and the lighting direction $\bar{L}$.
- Can give a somewhat harsh appearance, so sometimes an ambient term is added.

Light-Source Attenuation

- More distant objects often appear darker or less distinct due to lower light and atmospheric attenuation.
- Physically based $1/d^2$ does not work well - falloff tends to be either too fast or too slow to look good.
- Empirically useful compromise for an *attenuation factor* is

$$f_{att} = \min\left(\frac{1}{c_1 + c_2 d + c_3 d^2}, 1\right).$$

- Provides reasonable, if not physically correct depth cueing when used together with ambient and diffuse reflection.

$$I = I_a k_a + f_{att}(\bar{N} \cdot \bar{L}).$$

Colored Light and Surfaces

- Colored illumination can be represented by describing incident light in terms of three color components (e.g. RGB).
- An object's color can then be by giving three reflectivities, one for each color component, and writing three equations.
- Not completely correct, but often produces reasonable results.

Specular Reflection

- Highlights or "shiny" effects produced by non-uniform reflection from surfaces.
- Perfectly specular surface is like a mirror, reflecting a point source as a point.
- More useful situation is partial specularity where a point source is reflected as some sort of blur.
- The *Phong illumination model* approximates this effect with a term of the form $W(\theta)\cos^n \alpha$ where α is the angle between the viewing direction, and the direction of perfect specular reflection.
- n is typically somewhere between 1 and a few hundred.
- $W(\theta)$ is often set to a constant, the *specular reflection coefficient*.

Improving the Light Source

- Can vary intensity of light with direction from the source.
- In the *Warn model*, this directionality is given by $\cos^n \gamma$ where γ is the angle from the central (most intense) direction.
- Can implement "shutters" to restrict light to a certain region of space.
- Multiple point sources can also be used - just sum their effects.
- Extended sources require more complex methods.

Shading for Polygons

Constant Shading

- Apply illumination model once per polygon to obtain color for whole region.
- Give "faceted effect".
- Fast and useful if modeled objects are actually polyhedral.
- Undesirable if polygons are being used to model curved surface.
- Increasing number of facets is not as effective as might be thought because of *Mach banding*.

Gouraud Shading

- Interpolate intensity values across polygon using values obtained from applying reflectance model at vertices.
- Normal at vertices can be approximated as the mean of normals of polygons intersecting there.
- Use multiple normals if we want edge to be visible.
- Interpolation is along scan lines between values on edges.
- Faceted appearance greatly reduced, but not always completely eliminated, especially in regions with strong contrasts.
- Highlights sharper than individual polygons don't work well.

Phong Shading

- Interpolate surface normals instead of intensity values across polygon.
- Apply illumination model using interpolated normal at each point during scan conversion.
- Computationally more expensive.
- Gives better results, especially for highlights.

Problems with Interpolated Shading

- Polygonal silhouette.
- Perspective distortion since interpolation is performed on projected image.
- Orientation dependence (solve by using triangles).
- Problems if two adjacent polygons fail to share a vetex.
- Insufficient sampling can produce unrepresentative vertex normals (essentially a form of aliasing).
- Despite all this, polygonal systems form the core of most rendering systems.

Texture Mapping

- Any surface parameterized by two coordinates (e.g. pixels, polygons, bicubics) can be mapped to a planar image called a *texture map*.
- Values of reflectance coefficients (e.g. color) can be obtained from the texture map prior to scan-converting a pixel, rather than being constant.
- This provides a good way to obtain realistic, fine detail.
- Aliasing dealt with by having multiple texels within a pixel.

- Often for polygons, vertices are mapped, and intermediate coordinate values are interpolated.
- Problems with perspective.
- Problems with seams (no distance-preserving map for a sphere)
- Replacing reflectance values with a normal perturbation function produces a *bump map*.

Shadows

- Determine which surfaces can be "seen" from a light source.
- Consider point sources.
- At each pixel, combine ambient term with directional term for all light sources visible from that pixel.
- So how do we (efficiently) determine light source visibility?

Scan Line Generation of Shadows

- Augment scan-line algorithm to include shadow processing.
- Basically keep track of polygon edges that might cast shadows.
- Preprocess by projecting all polygons onto a sphere (or cube) surrounding light source.
- Pairs whose extents do not overlap can be eliminated.
- Compute shadow projections for remainder.
- Keep track of entry and exit of shadows using parallel shadow scan.

Two Pass Object-Precision Algorithm

- Use same visible surface algorithm twice.
- From light source viewpoint, find visible polygons.
- Merge these with original polygons.
- From observer viewpoint, render original polygons using ambient lighting model.
- Add directional term from illuminated polygons.
- Deal with multiple light sources by adding components.

Two Pass Z-buffer Algorithm

- Determine z-buffer from viewpoint of the light source.
- Run z-buffer algorithm from viewpoint of observer.
- Whenever a pixel is determined to be visible, transform the coordinates into the light's coordinate system, and check whether it is farther from the light than the value in the light's z-buffer.
- If so, then something is blocking it, and it should be shadowed.
- Need to be a little careful about self-shadowing, aliasing.

Transparency

- Representing transparency is sometimes desirable, either for clarity or increased realism.
- Both *refractive* and *nonrefractive* models are used.
- Refractive methods model the bending of light that occurs when it passes from one medium to another.
- Nonrefractive methods are less realistic, but easier to implement, and often useful.

Nonrefractive Transparency

In *interpolated transparency* the shade seen when an opaque polygon is seen through a transparent one is determined by interpolating between the two shades.

$$I_\lambda = (1 - k_{t1})I_{\lambda 1} + k_{t1}I_{\lambda 2}$$

- k_{t1} is the *transmission coefficient* of the transparent polygon 1, and represents the fraction of the light passing through.
- Can be thought of as a mesh model of transparency.
- Can be iterated.

The *screen door* transparency model literally implements a mesh model by turning on a certain fraction of the pixels.

- Easy to use in z-buffer systems.
- Can have serious aliasing problems.

In *Filtered Transparency* polygons are treated as transparent filters that selectively pass different wavelengths.

$$I_\lambda = I_{\lambda 1} + k_t O_{t\lambda} I_{\lambda 2}$$

- $O_{t\lambda}$ represents polygon 1's *transparency color*.
- Can be iterated back to front.

Refractive transparency

- Takes into account the bending of light when passing from one medium to another.
- Reason that lenses focus light etc.
- Basic mathematical model is *Snell's Law*.

$$\frac{\sin\theta_i}{\sin\theta_t} = \frac{\eta_{t\lambda}}{\eta_{i\lambda}}$$

- η represents the *index of refraction* of the various materials.
- Usually implemented using ray tracing techniques.

Interobject Reflections

- Light bounces between objects, both specularly and diffusely.
- Modeling this realistically requires ray-tracing or radiosity methods.
- Useful hack for approximating mirror-like specular reflection is *Reflectance map*.
- Basically determine what the world looks like in any direction around an object of interest by projecting down onto a sphere (or cube) surrounding object.
- This is used as a texture map, indexed by the angle obtained by reflecting the viewing direction about the surface normal at every point on the shiny object.

Physically Based Illumination Models

- Defined in terms of *Radiant Intensity*.
- Torrance-Sparrow *micro-facet* model is dominant paradigm.
- Can get quite complicated, depending on what is modeled.

Global Illumination Models

- Much light in a scene does not come directly from a point source.
- May be reflected, transmitted from other object in various ways.
- Light sources may be extended.
- So far, our only way of dealing with this has been to "fake" it using the ambient term.
- True situation is very complicated, as light is reflected back and forth many times between objects.
- Two techniques: *recursive ray tracing* and *radiosity methods* provide more principled (if not completely satisfactory) ways of approaching the problem.

Recursive Ray Tracing

- Simple ray tracing determines visible surfaces by shooting a set of rays from the viewing point into the scene, and determining the first object that each one hits.
- Shadowing can be accomplished by shooting a ray towards each light source from each visible point in the scene to see if it hits anything before reaching the source.
- We can also trace a reflected (specular) and a transmitted (refracted) ray from each visible point, which may in turn intersect another object, which itself may be producing light through reflection and transmission...
- Recursive ray tracing is the process of following this all out (to some termination condition).

- Lots of work, as number of rays can increase exponentially.
- Lots of hacks for attempting to limit the work required.
- Can't be done in real time at present; minutes or hours per frame.
- Works well for point sources, pure specular reflection, and clear transparency (hence prevalence of glass spheres in ray-traced images).
- Dealing with diffuse reflection, extended sources, and translucency required casting many rays (or a cone or beam) of rays from each point, with resulting increase in complexity.

Radiosity Methods

- Basic idea is to model each polygonal patch and exended light source as a region that redirects the light falling on it from other patches by some reflectance rule (and for sources, emits light as well).
- Under simplification that emission from each patch is lambertian (diffuse) the problem is to find the a set of radiosities for each patch that represents a self-consistent solution to all these equations.
- Generally expressed by calculating coupling coefficients or *form factors* for each pair of patches representing how much of the energy from one patch illuminates the other.
- This results in a set of simultaneous equations that can be solved iteratively.

- Various tricks for computing form factors efficiently
- Gives good results for diffuse illumination, but specularity is not handled.
- Directionality can be incorporated by keeping track of amount of light traveling in various direction, at large increase in computational cost.
- A useful, though not completely principled approach is to combine ray tracing and radiosity, using radiosity to obtain diffuse illumination, and ray tracing for highly specular components.
- This still leaves some middle ground that is hard to get at in a principled manner.