Ray Tracing
Light Reflection, Illumination
Hierarchies, Transforms, Advanced Rendering

CS 4620 Lecture 36
Adding microgeometry
Classic reflection behavior

- Ideal specular (mirror)
- Glossy specular
- Lambertian
Broad modeling approaches

- **Empirical expressions**
  - a long and glorious history…
  - you know these: Phong, Ward, Kajiya, etc.

- **Microfacet models**
  - a geometric optics approach for surface reflection
  - based on statistical averaging over microgeometry

- **Other geometric-optics surface models**
  - including Oren-Nayar and other diffuse models
  - also several grooved-surface models

- **Subsurface scattering models**
  - Hanrahan-Kreuger; diffusion models
Cook-Torrance BRDF Model

• A *microfacet* model
  – surface modeled as random collection of planar facets
  – an incoming ray hits exactly one facet, at random

• Key input: probability distribution of facet angle
Facet Reflection

- $H$ vector used to define facets that contribute
  - $L$ and $V$ determine $H$; only facets with that normal matter
  - reflected light is proportional to number of facets

![Facet Reflection Diagram](image)
Cook-Torrance BRDF Model

- “Specular” term (really glossy, or directional diffuse)

\[ f_r(n, l, v) = \frac{F(l, h)D(h)G(l, v, h)}{4|n \cdot l||n \cdot v|} \]
Cook-Torrance BRDF Model

\[ f_r(n, l, v) = \frac{F(l, h) D(h) G(l, v, h)}{4|n \cdot l||n \cdot v|} \]
Facet Distribution

- $D$ function describes distribution of $H$
- Popular choice is due to Beckmann
  - derivation based on Gaussian random surface
  - for the purposes of this model we take it as given

$$D(h) = \frac{e^{-\tan^2(h, n) \cdot m^2}}{\pi m^2 \cos^4(h, n)}$$
Cook-Torrance BRDF Model

\[ f_r(n, l, v) = \frac{F(l, h)D(h)G(l, v, h)}{4|n \cdot l||n \cdot v|} \]

- Fresnel reflectance for smooth facet
  - more light reflected at grazing angles
Cook-Torrance BRDF Model

\[ f_r(n, l, v) = \frac{F(l, h)D(h)G(l, v, h)}{4|n \cdot l||n \cdot v|} \]
Masking and Shadowing

- Many options; C-T chooses simple 2D analysis:

\[ G(l, v, h) = \min \left[ 1, \frac{2(n \cdot h)(n \cdot v)}{v \cdot h}, \frac{2(n \cdot h)(n \cdot l)}{v \cdot h} \right] \]
Model vs. measurement: aluminum

Measured

Model

[Torrance & Sparrow 1967]
Rob Cook’s vases

Carbon  Red Rubber  Obsidian  Lunar Dust  Olive Drab  Rust
Bronze  Tungsten  Copper  Tin  Nickel  Stainless Steel

[Cook & Torrance 1981]
Sources of illumination

• Point sources
  – energy emanating from a single point
• Directional sources
  – aka. point sources at infinity
• Area sources
  – energy emanating from an area of surface
• Environment illumination
  – energy coming from far away
• Light reflected from other objects
  – leads to global illumination
Light reflection: full picture

- all types of reflection reflect all types of illumination
  - diffuse, glossy, mirror reflection
  - environment, area, point illumination

incident distribution (function of direction)

reflected distribution (function of direction)
Implementing a bvol hierarchy

- A BoundedSurface can contain a list of Surfaces
- Some of those Surfaces might be more BoundedSurfaces
- Voilà! A bounding volume hierarchy
  - And it’s all still transparent to the renderer
BVH construction example
BVH ray-tracing example
Ray-slab intersection

\[ p_x + t_{x_{\text{min}}} d_x = x_{\text{min}} \]

\[ t_{x_{\text{min}}} = \frac{(x_{\text{min}} - p_x)}{d_x} \]

\[ p_y + t_{y_{\text{min}}} d_y = y_{\text{min}} \]

\[ t_{y_{\text{min}}} = \frac{(y_{\text{min}} - p_y)}{d_y} \]
Intersecting intersections

- Each intersection is an interval
- Want last entry point and first exit point

\[ t_{x\text{enter}} = \min(t_{x\text{min}}, t_{x\text{max}}) \]
\[ t_{x\text{exit}} = \max(t_{x\text{min}}, t_{x\text{max}}) \]
\[ t_{y\text{enter}} = \min(t_{y\text{min}}, t_{y\text{max}}) \]
\[ t_{y\text{exit}} = \max(t_{y\text{min}}, t_{y\text{max}}) \]
\[ t_{\text{enter}} = \max(t_{x\text{enter}}, t_{y\text{enter}}) \]
\[ t_{\text{exit}} = \min(t_{x\text{exit}}, t_{y\text{exit}}) \]
Building a hierarchy

- Top Down vs Bottom Up
- Top down
  - Make bbox for whole scene, then split into (maybe 2) parts
    - Recurse on parts
    - Stop when there are just a few objects in your box
- Bottom Up
  - Expensive, but optimal
  - Good for static (maybe)
Building a hierarchy

• How to partition?
  – Ideal: clusters
  – Practical: partition along axis
    • Center partition
      – Less expensive, simpler
      – Unbalanced tree
    • Median partition
      – More expensive
      – More balanced tree
    • Surface area heuristic
      – Model: expected cost of ray intersection
      – Generally produces best-performing trees
BVH Intersection

• Trace ray with root node

• If intersection, trace rays with ALL children
  – If no intersection, eliminate tests with all children
Regular space subdivision

- An entirely different approach: uniform grid of cells
Regular grid example

- Grid divides space, not objects
Traversing a regular grid
Non-regular space subdivision

• *k*-d Tree
  – subdivides space, like grid
  – adaptive, like BVH
Implementing acceleration structures

- Conceptually simple to build acceleration structure into scene structure
- Better engineering decision to separate them
Transforming objects

• In modeling, we’ve seen the usefulness of transformations
  – How to do the same in RT?
• Take spheres as an example: want to support transformed spheres
  – Need a new Surface subclass
• Option 1: transform sphere into world coordinates
  – Write code to intersect arbitrary ellipsoids
• Option 2: transform ray into sphere’s coordinates
  – Then just use existing sphere intersection routine
Intersecting transformed objects
Implementing RT transforms

• Create wrapper object “TransformedSurface”
  – Has a transform $T$ and a reference to a surface $S$
  – To intersect:
    • Transform ray to local coords (by inverse of $T$)
    • Call surface.intersect
    • Transform hit data back to global coords (by $T$)
      – Intersection point
      – Surface normal
      – Any other relevant data (maybe none)
Groups, transforms, hierarchies

• Often it’s useful to transform several objects at once
  – Add “SurfaceGroup” as a subclass of Surface
• Has a list of surfaces
• Returns closest intersection
  – Opportunity to move ray intersection code here to avoid duplication
• With TransformedSurface and SurfaceGroup you can put transforms below transforms
  – Voilà! A transformation hierarchy.
A transformation hierarchy

- Common optimization: merge transforms with groups
Instancing

• Transform objects several ways
  – Many models have repeated subassemblies
    • Mechanical parts (wheels of car)
    • Multiple objects (chairs in classroom, …)
  – Nothing stops you from creating two TransformedSurface objects that reference the same Surface
    • Allowing this makes the transformation tree into a DAG
      – (directed acyclic graph)
    • Mostly this is transparent to the renderer
With instancing

```
Transform
  Group: car
    Surface: body
    Transform
    Transform
    Transform
      Group: wheel
      ...```

...
Advanced Ray Tracing
Basic ray tracing

• Many advanced methods build on the basic ray tracing paradigm
• Basic ray tracer: one sample for everything
  – one ray per pixel
  – one shadow ray for every point light
  – one reflection ray, possibly one refraction ray, per intersection
Discontinuities in basic RT

- Perfectly sharp object silhouettes in image
  - leads to aliasing problems (stair steps)
- Perfectly sharp shadow edges
  - everything looks like it’s in direct sun
- Perfectly clear mirror reflections
  - reflective surfaces are all highly polished
- Perfect focus at all distances
  - camera always has an infinitely tiny aperture
- Perfectly frozen instant in time (in animation)
  - motion is frozen as if by strobe light
The Blue Umbrella

- Latest Pixar short
- Made partly to showcase new more photorealistic rendering
  - much of it based on the ideas in this lecture

worth a look: 
http://rainycitytales332.tumblr.com
Soft shadows
Cause of soft shadows

point lights cast hard shadows
Cause of soft shadows

area lights cast soft shadows
Glossy reflection
Cause of glossy reflection

smooth surfaces produce sharp reflections
Cause of glossy reflection

rough surfaces produce soft (glossy) reflections
Depth of field

REAL CAMERAS
Cause of focusing effects

what lenses do (roughly)
Cause of focusing effects

point aperture produces always-sharp focus
Cause of focusing effects

finite aperture produces limited depth of field
Motion blur
Cause of motion blur

single image point

moving object

image point sees different object points at different times
Creating soft shadows

• For area lights: use many shadow rays
  – and each shadow ray gets a different point on the light
• Choosing samples
  – general principle: start with uniform in square
Creating glossy reflections

• Jitter the reflected rays
  – Not exactly in mirror direction; add a random offset
  – Can work out math to match Phong exactly
  – Can do this by jittering the normal if you want
Depth of field

- Make eye rays start at random points on aperture
  - always going toward a point on the focus plane
Motion blur

• Caused by finite shutter times
  – strobing without blur
• Introduce time as a variable throughout the system
  – object are hit by rays according to their position at a given time
• Then generate rays with times distributed over shutter interval