Lighting

To do
• Continue to work on ray programming assignment
• Start thinking about final project

Course Outline
• 3D Graphics Pipeline
  Modeling
  (Creating 3D Geometry)
  Mesh; modeling; sampling; Interaction

  Rendering
  (Creating, shading images from geometry, lighting, materials)
  Ray tracing

This Lecture
• High-level overview and summary of ray tracing, global illumination
• Technical details of writing a ray tracer and more advanced topics
Realism
• (soft) shadows; reflections (mirrors and glossy); transparency (water, glass); Interreflection (color bleeding); Complex illumination (natural, area light); realistic materials (paints, glasses); and many more

Light
• Propagates along straight-line rays in empty space, stopping when it meets a surface
• Light bounces "like a billiard ball" from any shiny surface that it meets, following an "angle of incidence equals angle of reflection" model, or is absorbed by the surface, or some combination of the two (e.g., 40% absorbed, 60% reflected).
• Most apparently smooth surfaces, like the surface of a piece of chalk, are microscopically rough. These behave as if they were made of many tiny smooth facets, each following the previous rules; as a result, light hitting such a surface scatters in many directions (or is absorbed, as in the mirror-reflection)
• A pixel of a camera, or one of the cells in the eye that detects light, sums up (by integration) all the light that arrives at a small area over a small period of time. The value of the integral is the "sensor response" that corresponds to how much total light, based on the number of incident photons, the pixel (or cell) "saw".
• A pixel of a display can be adjusted to emit light of a specified intensity and color.

Outline
• Ray tracing
  • Pixel by pixel instead of object by object
  • Global illumination and rendering equation
Ray / object intersections

- Heart of ray tracer
  - One of the main initial research areas
  - Optimized routines for wide variety of primitives
- Various types of info
  - Shadow rays: Intersection / no intersection
  - Primary rays: point of intersection, material, normals
  - Texture coordinates
- Work out examples on board
  - Triangles, sphere, polygon, general implicit surface

Ray tracing transformed objs

- We have an optimized ray-sphere test
  - But we want to ray trace an ellipsoid...
- Solution: Ellipsoid transforms sphere
  - Apply inverse transform to ray, use ray-sphere
  - Allows for instancing (traffic jam of cars)
- Math details will be worked out in class

Acceleration

- Testing each object for each ray is slow
  - Fewer rays
    - Adaptive sampling, depth control
  - Generalized rays
    - Beam tracing, cone tracing, pencil tracing etc.
  - Faster intersections
    - Optimized ray-object intersections
    - Fewer intersections

Acceleration structures:
- Bounding Box
  - Spatial hierarchies (octree, kd tree, BSP trees)
Acceleration structures: grids

Interactive ray tracing
- Ray tracing historically slow
- Now viable alternative for complex scenes
  - Key is sublinear complexity with acceleration; need not process all triangles in scene
- Hardware implementation

Implementation
- Camera Ray Casting
- Ray-object intersections
- Ray-tracing transformed objects
- Lighting calculations
- Recursive ray tracing

Outline in Code
```
Image Raytrace (Camera cam, Scene scene, int width, int height) {
  Image image = new Image (width, height) ;
  for (int i = 0 ; i < height ; i++)
    for (int j = 0 ; j < width ; j++) {
      Ray ray = RayThruPixel (cam, i, j) ;
      Intersection hit = Intersect (ray, scene) ;
      image[i][j] = FindColor (hit) ;
    }
  return image ;
```

Ray Casting

Virtual Viewpoint
Virtual Screen
Ray intersects object: shade using color, lights, materials
Ray misses all objects: Pixel colored black
Multiple intersections: Use closest one

Shadows

Light source
Virtual Viewpoint
Virtual Screen
Shadow ray to light is blocked: object in shadow
Shadow ray to light is unblocked: object visible

Shadows: implementation caveats

- Numerical inaccuracy may cause intersection to be below surface (effects exaggerated in figure)
- Causing surface to incorrectly shadow itself
- Solution: move a little towards light before shooting shadow ray

Mirror reflections / refractions

Virtual Screen
Virtual Viewpoint
Objects
Generate reflected ray in mirror direction, Get reflections and refractions of objects
Recursive ray tracing
• For each pixel
  – Trace primary eye ray, find intersection
  – Trace secondary shadow ray(s) to all light(s)
    • Color = visible ? Illumination model : 0;
  – Trace reflected ray
    • Color += reflectivity * color of reflected ray

Caveats with recursion
• Reflection rays may be traced forever
• Solution: generally, set maximum recursion depth
• Same for transmitted rays (take refraction into account)

Finding Ray Direction
• Goal is to find ray direction for given pixel i and j
• Many ways to approach problem
  – Objects in world coordinate, find direction of each ray (we do this)
  – Camera in canonical frame, transform objects (OpenGL)
• Basic idea
  – Ray has origin (camera center) and direction
  – Find direction given camera parameters and i and j

Similar to gluLookAt
Physics of Light

- A simple model: Phong reflectance model
  - Little computational cost
  - Three distinct components of reflection:
    - Ambient (constant amount of light providing a gross simulation of inter-object reflection)
    - Diffuse (representing viewer-independent light reflected equally in all direction)
    - Specular (providing glossy highlights on shiny surfaces when the viewpoint is close to the reflection ray)
  - Independent diffuse and specular components

Ambient + diffuse + specular = result

Diffuse Layer
- For a solid-color material, this is a constant across the surface: (Cd, r, Cd, g, Cd)
- For a textured material, the pixmap and texture algorithm determine the color at each individual surface point

Innate color of the material's Specular layer
- Constant across the surface (Csr, Csg, Csb)

The reflection efficiency of the material's three distinct layers
- Ka: diffuse layer in reflecting ambient light
- Kb: diffuse layer in reflecting directional / geometric light
- Ks: specular layer in reflecting directional / geometric light

RGB trip of efficiency fractions: 0: no efficiency; 1: full efficiency