Ray Tracing Intro

CS 417 Lecture 31

Image order graphics

- Up until now, everything has been in terms of drawing objects one at a time into an image
- Now we begin looking at things in terms of computing the image one pixel at a time, finding the objects for each one
  - we examine the full collection of objects
  - leads to greater freedom in what to compute
  - also leads to slower performance for simple scenes

Light transport in brief

- Light comes from sources, reflects off surfaces, and gets to the eye

Light transport and ray tracing

- We are interested in light that gets to a particular pixel
- What parts of the scene could contribute?
  - answer: points along a particular ray

Light transport and ray tracing

- Ray tracing idea: answer question “what light arrives here” by analyzing where it might have come from and then looking there.
  - first level: look along ray to camera
  - second level: think of how light could get to surface
    - directly from sources
    - indirectly via reflection (we’ll only discuss mirror case)
    - indirectly via transmission (we’ll only discuss perfectly transparent case)
  - higher levels: recursive

Ray tracing effects

- Basic ray tracer: mirror reflections, sharp shadows
Ray tracing procedure

Fundamental operation: “how much light do I see if I look along this ray?”

Basic outline of ray tracer

- intersect(ray): find the first intersection with a surface along the ray.
- trace(ray): call intersect on to find out what surface is relevant, then call shade to compute a color for the ray.
- shade(point): evaluate a shading model (e.g. phong, lambert); call intersect or trace to get the needed information.
- Main loop: generate eye rays, call trace to ask what light intensity arrives along them.
Basic software components in RT

- Camera
  - generate eye rays
- Geometry modules
  - compute intersections
  - compute normal vectors
- Shaders (materials)
  - compute intensities based on lighting
- Main program
  - handle scene graph, transformations
  - perform parsing and image I/O

Ray tracing extensions

- Texture mapping
  - intersect needs to compute texture coordinates at the intersection point
- Distribution ray tracing
  - to achieve various kinds of blur
    - soft shadows
    - glossy reflections
    - motion blur
    - camera lens depth of field
  - use randomly perturbed rays

What is a ray?

- A half line
- Represented parametrically with a point and a direction:
  - \((p, d)\) denotes \(\{ p + td \mid t > 0 \}\)

Generating eye rays

- First way: using image plane
  - intuitive and simple
  - think of image plane actually sitting in scene
  - rays are defined by the eye point and the image plane point

Ray-object intersection: sphere

- Second way: using projection transform
  - integrates with pipeline graphics systems