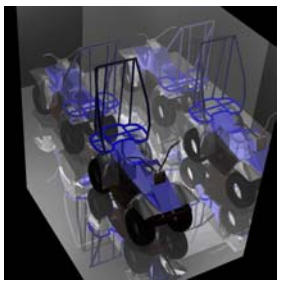
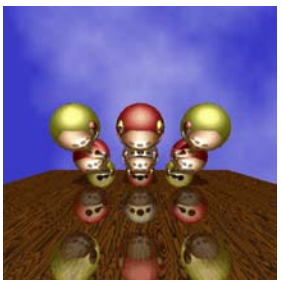
University of British Columbia

Chapter 12



Ray-Tracing

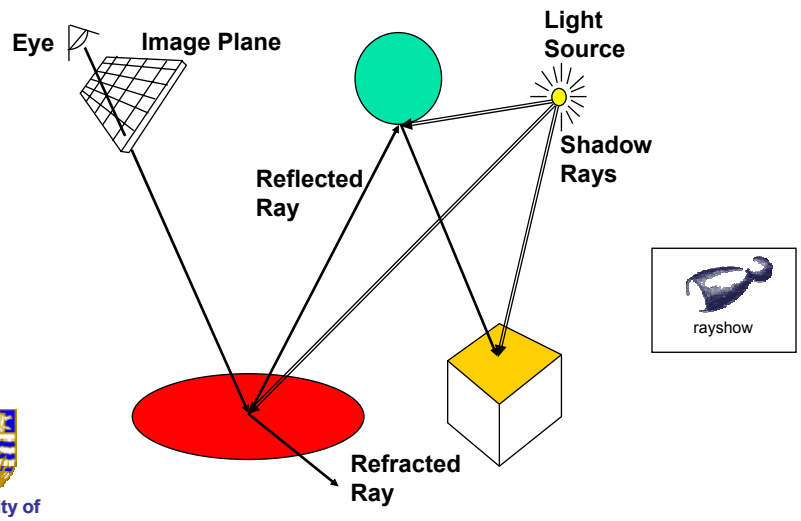


Global Illumination Models

- Simple shading methods simulate local illumination models
  - No object interaction
- To simulate global illumination models need more sophisticated & more computation-intensive algorithms
- Ray-tracing deals with
  - Reflectivity
  - Transparency
  - Shadows

University of British Columbia

Ray-Tracing Algorithm

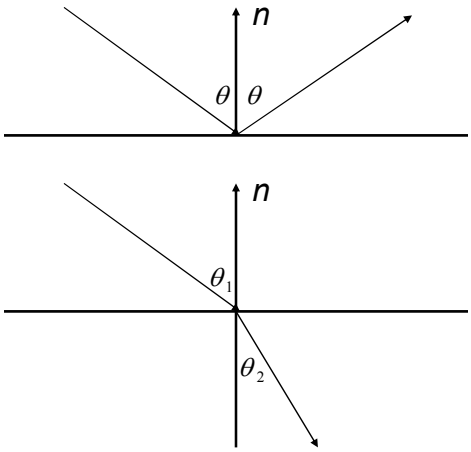


University of British Columbia

Reflection and Refraction

$$\frac{\sin\theta_1}{\sin\theta_2} = \frac{c_1}{c_2}$$

Snell's Law



University of British Columbia

Basic Ray-Tracing Algorithm

```
RayTrace(r,scene)
obj := FirstIntersection(r,scene)
if (no obj)  return BackgroundColor;
else begin
    if ( Reflect(obj) ) then
       reflect_color := RayTrace(ReflectRay(r,obj));
    else
       reflect_color := Black;
   if ( Transparent(obj) ) then
       refract_color := RayTrace(RefractRay(r,obj));
    else
       refract_color := Black;
   return Shade(reflect_color,refract_color,obj);
end;
```

University of British Columbia

Sub-Routines

- ReflectRay(r,obj) – computes reflected ray (use obj normal at intersection)
- RefractRay(r,obj) - computes refracted ray
  - Note: ray is inside obj
- Shade(reflect_color,refract_color,obj) – compute illumination given three components

University of British Columbia

Ray-Object Intersections

- Kernel of ray-tracing $\Rightarrow$ must be extremely efficient
- Usually involves solving a set of equations

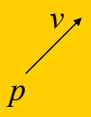
Example: Ray-Sphere intersection

ray: $x(t) = p_x + v_x t,\ \ y(t) = p_y + v_y t,\ \ z(t) = p_z + v_z t$

(unit) sphere: $x^2 + y^2 + z^2 = 1$

quadratic equation in t :

$$0 = (p_x + v_x t)^2 + (p_y + v_y t)^2 + (p_z + v_z t)^2 - 1$$
$$= t^2 (v_x^2 + v_y^2 + v_z^2) + 2t(p_x v_x + p_y v_y + p_z v_z)$$
$$+ (p_x^2 + p_y^2 + p_z^2) - 1$$



University of British Columbia

Ray-Object Intersections

- Efficient for
  - Primitives – Box, Sphere, etc..
  - Quadrics
  - Polygons
  - Volumetric Data
- Problematic for free-form surfaces
- Subdivision?

University of British Columbia

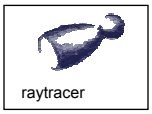
More About Ray-Tracing

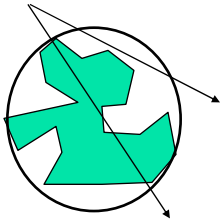
- Algorithm above has a BUG….
- Does not terminate
- Termination Criteria
  - No intersection
  - Contribution of secondary ray attenuated below threshold – each reflection/refraction attenuates ray
  - Maximal depth is reached

University of British Columbia

Optimized Ray-Tracing

- Basic algorithm simple but VERY expensive
- Optimize…
  - Reduce number of rays traced
  - Reduce number of ray-object intersection calculations
- Methods
  - Bounding Boxes
  - Spatial Subdivision
    - Visibility & Intersection
  - Tree Pruning





University of British Columbia

Simulating Shadows

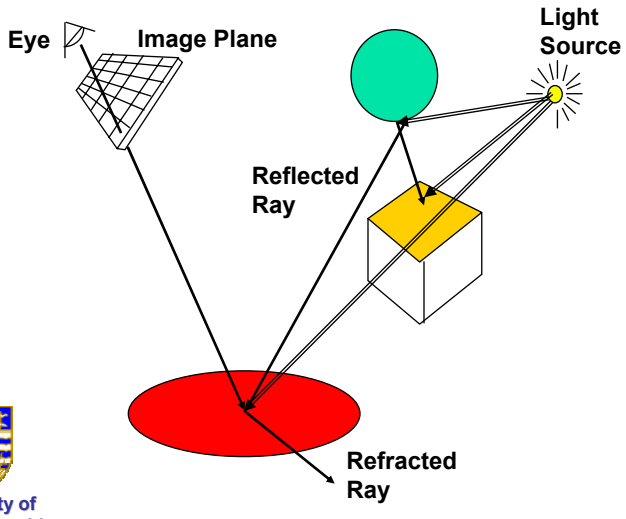
- Trace ray from each ray-object intersection point to light sources
  - If the ray intersects an object in between ⇒ point is shadowed from the light source

```
shadow = RayTrace(LightRay(obj,r,light));

return Shade(shadow,reflect_color,refract_color,obj);
```

University of British Columbia

Ray-Tracing With Shadows



University of British Columbia

Advanced Phenomena

- Can (not allways efficiently) simulate
  - Soft Shadows
  - Fog
  - Frequency Dependent Light (diamonds & prisms)
  - Barely handle S*DS*
    - S – Specular
    - D - diffuse

University of British Columbia