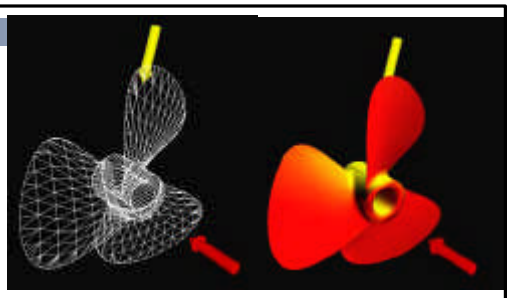
Ray Tracing

Illumination Model

- Given - light sources and objects
- Physics of real illumination is complex
  - Too complex for interactive graphics
- Create simplified models that give reasonable approximations
- Illumination models give us an approximate answer

Illumination Goals

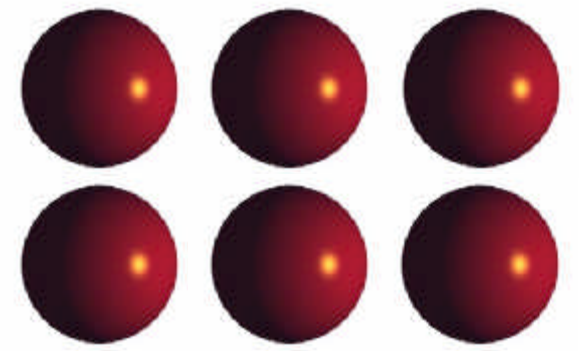


- Input
  - 3d objects
  - In 3D space
  - Light source(s) at specified location(s)
- Output
  - Need to calculate the intensity for any given point on the objects

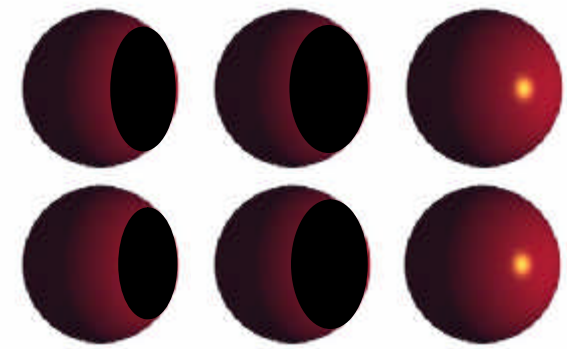
Local illumination

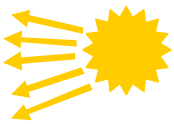
Local illumination

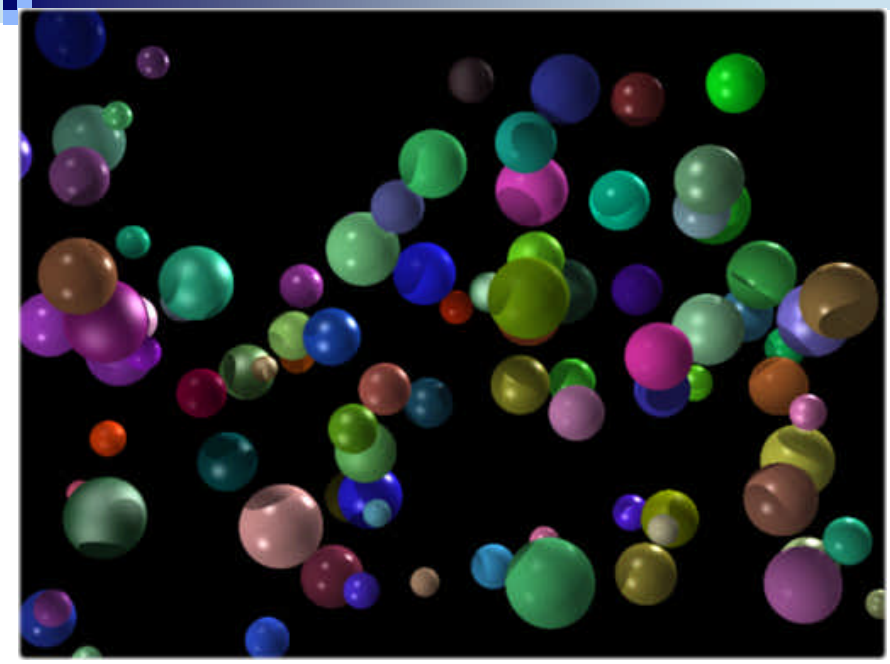
- Strong simplification: does not consider interaction between objects
- This model approximates how the light reflects from a surface under direct illumination
- Phong model is a famous local illumination



Local illumination



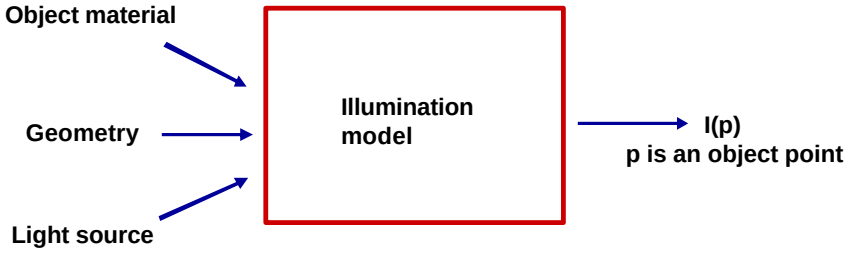




http://www.cs.unc.edu/~stewart/cg-images/index.html

Phong Model

- Re-cap



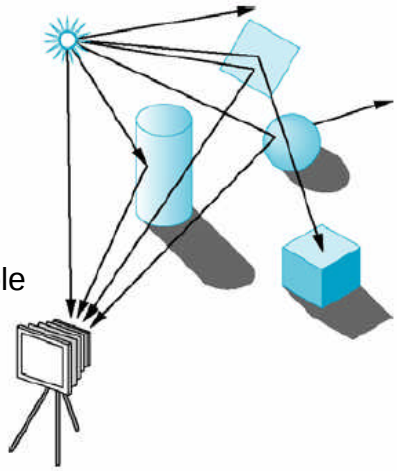
- Empirical model
- Ambient light simulates indirect lighting effects

Ray tracing

- Light rays
- Leave source
- Travel through environment
- Until they reach the eye or camera.

Rays reach the eye

- Directly from the source
- From interaction with a visible surface
- Multiple reflections
- After transmission through surfaces

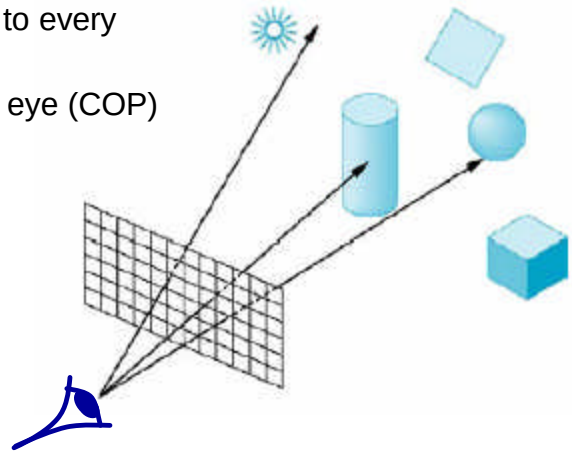


An Observation

- Most of the rays that leave the light source do not enter the system (the eye)
- No contribution to the image
- Following all rays leaving the light source
  - Light rays too numerous
  - Time consuming process
- Better idea: reverse the direction of the ray

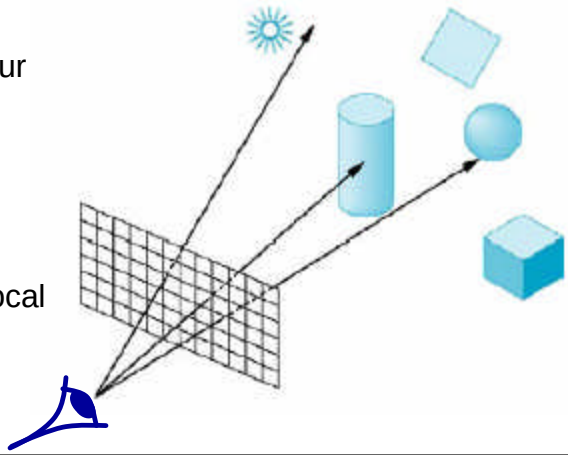
Ray Casting

- Reverse direction
- Follow those rays that start at the COP
- Aim: assign a color to every pixel
- Cast a ray from the eye (COP) through every pixel



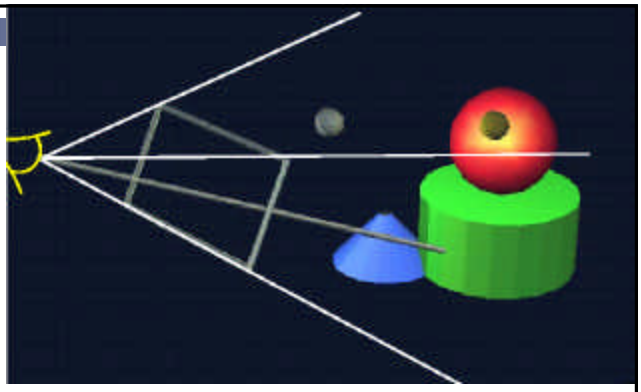
Possible cases for the cast Ray

- Intersect a surface
  - Compute shade for point of intersection
- Go off to infinity
  - Background colour
- Result is same as local illumination



Ray Casting

- Pinhole camera
- Perspective
- Hidden surface removal



For every pixel in viewplane

Project a ray (from COP through each pixel) into the scene

Find nearest intersection point (checking every object in scene)

Compute colour of pixel

Ray Cast

```
Procedure RayCast()
{
    for y = -1 to 1 by 2/(Y-1) do {
        for x = -1 to 1 by 2/(X-1) do {
            ray origin = [0,0,0];
            ray direction = [x, y, 1];
            ColourPixel (Render(ray));
        }
    }
}

Function Render (ray)
{
    object = QueryScene(ray);
    if (object==NIL)
        return backgroundColour;
    return Shade (object, ray);
}
```

```
Function QueryScene(ray)
{
    closest = NIL;
    distance = INFINITY;
    For each object in sceneList do {
        if intersect (ray, object)
            if (object.distance<distance) {
                closest = object;
                distance = object.distance;
        }
    }
    return closest;
}
```

Ray tracing

- Like ray casting but at each intersection point start new rays recursively.

Shadow checker

Reflected ray

Refracted ray

Pseudo Code of Recursive Ray Tracer

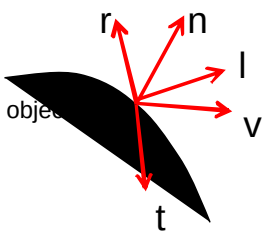
```
colour c = trace (point p, vector d, int step)
{
color local, reflected, transmitted;
point q;
normal n;

if (step > max)
      return(background_color);
q = intersect(p, d, status);
if(status == light_source)
      return(light_source_color);
if(status == no_intersection)
      return(background_color);

n = normal(q);
r = reflect(q,n);
t = transmit(q,n);
local = phong(q,n,r);
reflected = trace(q,r,step+1);
transmitted = trace(q,t,step+1);
return(local+reflected+transmitted);
}
```

Vectors

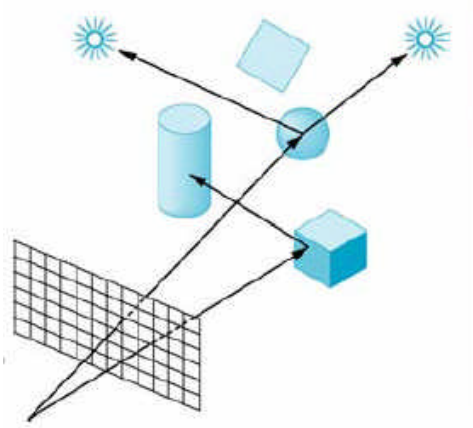
- unit normal vector **n**
- unit vector **v**, from **P** to the viewer
- unit vector **l**, from **P** to the light
- unit vector **t** (transmit)
- unit vector **r** (reflect)



Shadows

Question:

- Is the point of intersection illuminated?
- Compute a shadow ray from the point to the source

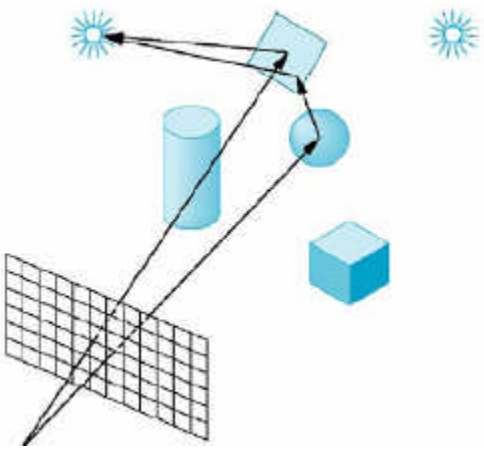


Condition:

- If shadow ray intersects a surface before it meats the source the point is not illuminated

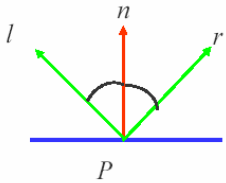
Reflection

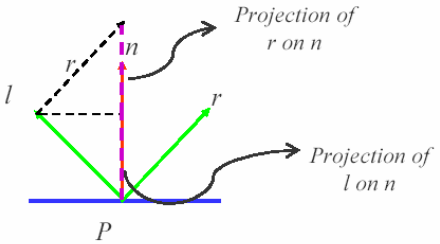
- Some surfaces are highly reflective
- Follow the reflected rays until
  - Intersects a source
  - Goes off to the infinity
- Recursive algorithm



How to Compute **r** – reflection vector

- Have:
  - l (from point P to light source)
  - n (surface normal)
- The angle of incidence is equal to the angle of reflection
  - l, n, r lie in the same plane
- solving we obtain r = 2(n·l)n-l





Refraction/transmission

The light of the source partially

- Absorbed
- Transmitted
- Reflected

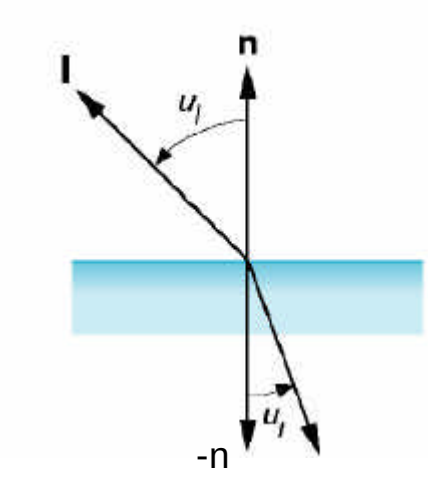
Necessary tasks

- Compute the intensity value using standard shading model
- Cast a ray in the direction of a perfect reflection (recursive call)
- Cast a ray in the direction of the transmitted ray (recursive call)
- Termination condition?

Direction of Transmitted Ray (from here)

Snell's law:

$$\frac{\sin u_l}{\sin u_r} = \eta$$



Intersecting A Ray With A Sphere

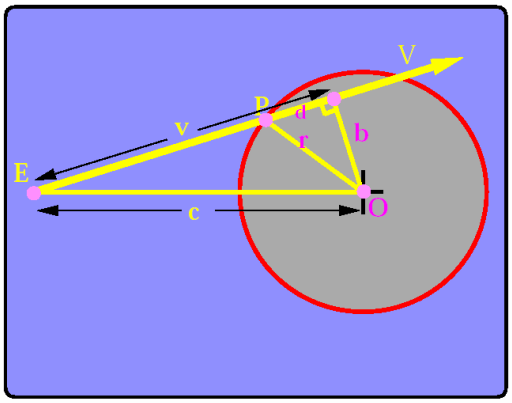
$v^2 + b^2 = c^2$

$d^2 + b^2 = r^2$

$d = \sqrt{r^2 - (c^2 - v^2)}$ ----(1)

$c = \|EO\|$

V is the unit vector in the direction of the ray.



$v = \mathbf{EO}.\mathbf{V}$; then from (1):

$disc = r^2 - ((\mathbf{EO}.\mathbf{EO}) - v^2)$;

if (disc <= 0) then no intersection;

else {

$d = \sqrt{disc}$;

$P = \mathbf{E} + (v-d)\mathbf{V}$;

}

Intersecting A Ray With A Triangle

Normal to the plane

$\mathbf{n} = (\mathbf{b}-\mathbf{a}) \times (\mathbf{b}-\mathbf{c})$

For a point p in the plane:

$\mathbf{n}.(\mathbf{p}-\mathbf{b}) = (\mathbf{b}-\mathbf{a}) \times (\mathbf{b}-\mathbf{c}).(\mathbf{p}-\mathbf{b})=0$

(i.e the angle between the plane and the normal should be 90)

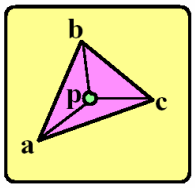
For the intersection tests the ray is presented as an origin plus a ray direction scaled by distance along the ray. The point of intersection is given by:

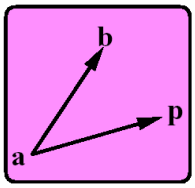
$\mathbf{p} = \mathbf{u} + \mathbf{v}t$

substituting this into the plane equation:

$\mathbf{n}.(\mathbf{u} + \mathbf{v}t-\mathbf{b}) = 0$

$t = \mathbf{n}.(\mathbf{b}-\mathbf{u})/\mathbf{n}.\mathbf{v}$





If $\mathbf{n}.\mathbf{v} = 0$ then ray is parallel to the plane

otherwise we can find **p** but still have to test if it is inside the triangle.

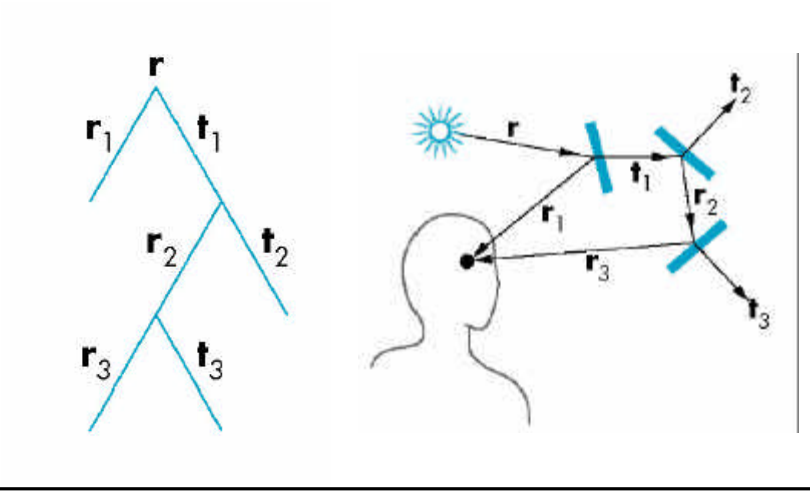
This is easily done using a cross product to tell us on which side of each edge the point lies.

In other words the following cross products must have the same sign:

$(\mathbf{b}-\mathbf{a})\times(\mathbf{p}-\mathbf{a}).\mathbf{n}$
$(\mathbf{c}-\mathbf{b})\times(\mathbf{p}-\mathbf{b}).\mathbf{n}$
$(\mathbf{a}-\mathbf{c})\times(\mathbf{p}-\mathbf{c}).\mathbf{n}$

Ray Trace

- At each surface, additional rays may be generated



The Ray Tree

Surface properties of object define the amount carried by each ray. Recursion is terminated when the value drops below some threshold, or after a preset number of intersections.

Shadow Feelers
One per light source
(Extended light sources require more samples)