CSE170 Introduction to Computer Graphics

ray tracing

Prof. Marcelo Kallmann
mkallmann@ucmerced.edu

Ray Tracing

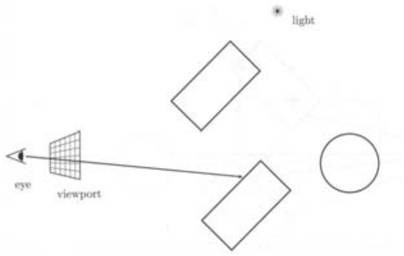
- Global technique for achieving
  - Lighting and shading
  - Hidden surface elimination
  - Reflection, refraction, transparency
  - Shadows
  - Etc
- Generates very realistic images
- But is very slow
  - Individual frames may take hours

Ray Tracing

- First Idea
  - Send a ray from the light source(s)
  - Reflect the ray through the scene
  - Until it hits the image plane

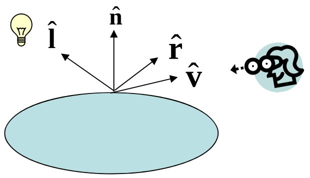
Ray Tracing

- Simplest Implementation
  - For every pixel in the image plane
    - Get closest intersection between the ray (sent from the center of the pixel) and the scene
    - Compute color for the intersection based on an illumination model (for example, Phong)

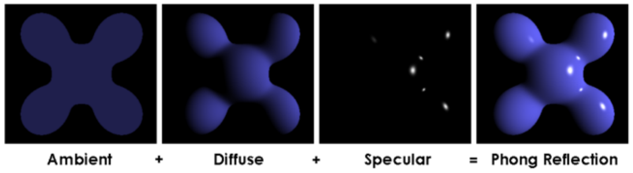


Ray Tracing

- Remembering Phong

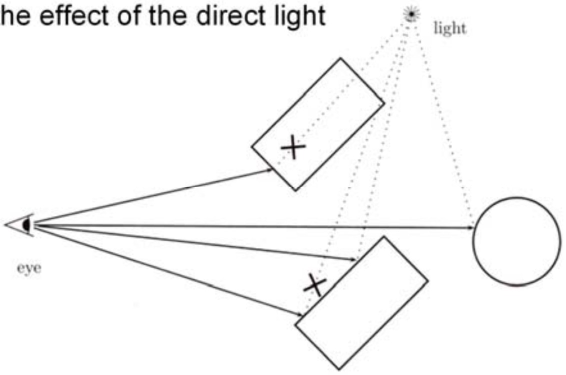


$$I = I_a k_a + I_d k_d (\hat{\mathbf{l}} \cdot \hat{\mathbf{n}}) + I_s k_s (\hat{\mathbf{v}} \cdot \hat{\mathbf{r}})^f$$



Ray Tracing

- Including shadows:
  - shadow feelers: for every hit, check if it can "see" the light source with another ray
    - And update a coefficient to cancel or not (0 or 1) the effect of the direct light

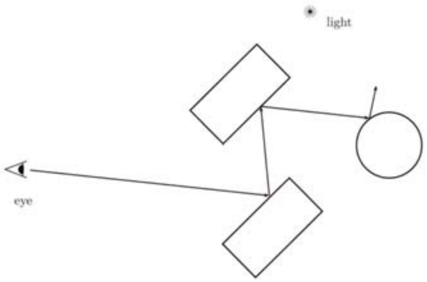


(Shadow feelers also sent to all light sources)

Ray Tracing

- Including reflection rays:
  - In the direction of the perfect specular reflection
  - Get the result of the reflected ray and add a portion of it (sigma in [0,1]) to the "first point hit"

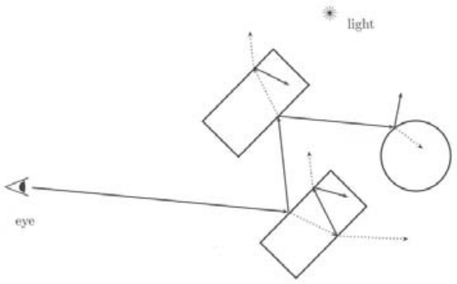
$$I = I_{local} + \sigma_{rglob} I_{reflect}$$



Ray Tracing

- Including transmission rays:
  - Similar to reflected rays, but for refractions, for generating transparency effects
  - Add its contribution to the equation:

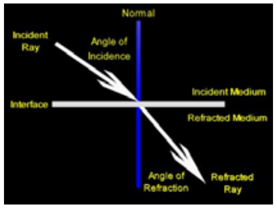
$$I = I_{local} + \sigma_{rglob} I_{reflect} + \sigma_{tglob} I_{xmit}$$

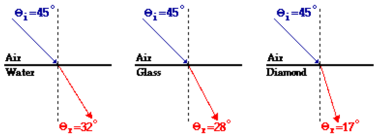


Ray Tracing

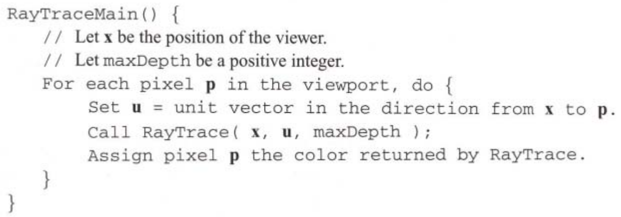
- How to compute refraction
  - Refraction depends on the amount of change in light speed
  - Different materials have different indices of refractions

(You can try different values and see what the material looks like)





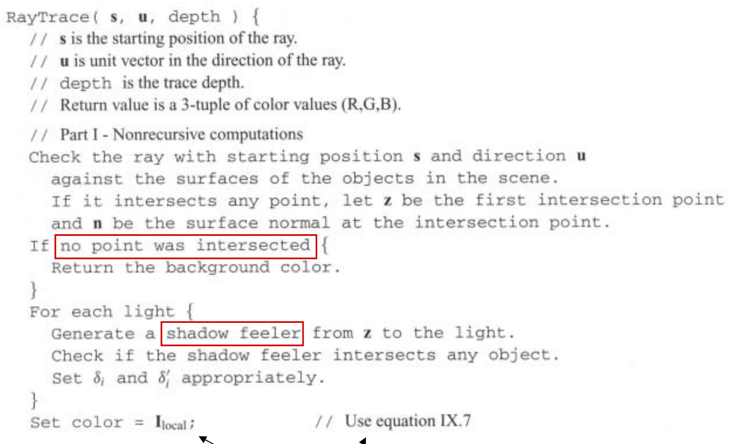
Ray Tracing: implementation



```
RayTraceMain() {
    // Let x be the position of the viewer.
    // Let maxDepth be a positive integer.
    For each pixel p in the viewport, do {
        Set u = unit vector in the direction from x to p.
        Call RayTrace( x, u, maxDepth );
        Assign pixel p the color returned by RayTrace.
    }
}
```

Reference: 3-D computer graphics, Samuel R. Buss

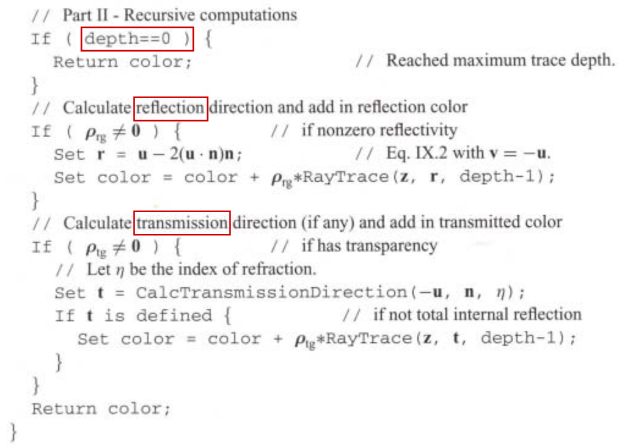
Ray Tracing



```
RayTrace( s, u, depth ) {
    // s is the starting position of the ray.
    // u is unit vector in the direction of the ray.
    // depth is the trace depth.
    // Return value is a 3-tuple of color values (R,G,B).
    // Part I - Nonrecursive computations
    Check the ray with starting position s and direction u
        against the surfaces of the objects in the scene.
        If it intersects any point, let z be the first intersection point
        and n be the surface normal at the intersection point.
    If no point was intersected {
        Return the background color.
    }
    For each light {
        Generate a shadow feeler from z to the light.
        Check if the shadow feeler intersects any object.
        Set $\delta_i$ and $\delta_i'$ appropriately.
    }
    Set color = $\mathbf{I}_{\mathrm{local}}$;           // Use equation IX.7
```

Phong equation, modified by the shadow feeler coefficient

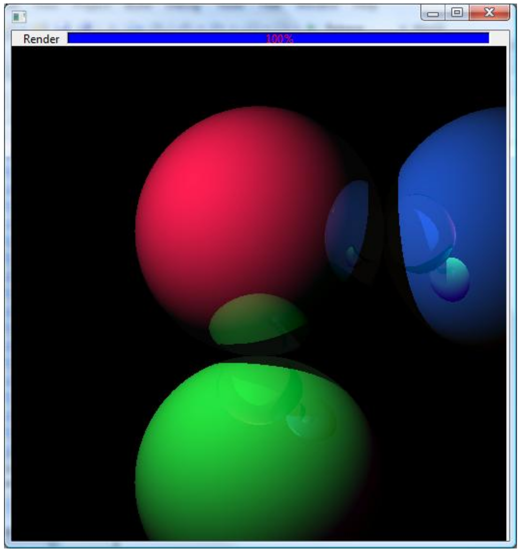
Ray Tracing



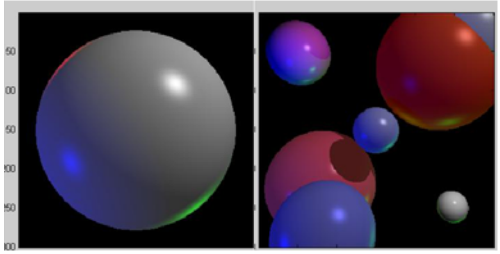
```
    // Part II - Recursive computations
    If ( depth==0 ) {
        Return color;                        // Reached maximum trace depth.
    }
    // Calculate reflection direction and add in reflection color
    If ( $\rho_{rg} \neq 0$ ) {                   // if nonzero reflectivity
        Set r = $\mathbf{u} - 2(\mathbf{u}\cdot\mathbf{n})\mathbf{n}$;              // Eq. IX.2 with $\mathbf{v} = -\mathbf{u}$.
        Set color = color + $\rho_{rg}$*RayTrace(z, r, depth-1);
    }
    // Calculate transmission direction (if any) and add in transmitted color
    If ( $\rho_{tg} \neq 0$ ) {                   // if has transparency
        // Let $\eta$ be the index of refraction.
        Set t = CalcTransmissionDirection($-\mathbf{u}$, n, $\eta$);
        If t is defined {                    // if not total internal reflection
            Set color = color + $\rho_{tg}$*RayTrace(z, t, depth-1);
        }
    }
    Return color;
}
```

Examples

By Robert Backmann, CSE170 2009



Examples


50
00
50
00
50
00