Ray Tracing

John C. Hart
CS 418
Interactive Computer Graphics
Environment Mapped Bump Mapping

env. map
What’s Wrong with this Picture?
What’s Wrong with this Picture?

1. Reflection doesn’t meet boat
What’s Wrong with this Picture?

1. Reflection doesn’t meet boat
2. Reflection behind the boat
What’s Wrong with this Picture?

1. Reflection doesn’t meet boat
2. Reflection behind the boat
3. Environment map magnified
How Can We Do This?
How Can We Do This?  Ray Tracing
Ray Tracing v. Rasterization

- **Rasterization**
  - For each primitive
  - For each pixel
  - Render pixel

- **Ray Tracing**
  - For each pixel
  - For each primitive
  - Render pixel
Pixels in World Coords

- aspect ratio $a = w/h$
- focal length $d = 1/\tan(\text{fovy}/2)$

$$\text{ll} = \text{eye} + d \text{ l} - a \text{ v} - \text{u}$$

for (j = 0; j < VRES; j++) {
    for (i = 0; i < HRES; i++) {
        \begin{align*}
        \mathbf{p} &= \text{ll} + 2\mathbf{av} \cdot (\text{double})i/\text{HRES} + 2\mathbf{u} \cdot (\text{double})j/\text{VRES}; \\
        \text{color} &= \text{TraceRay} \left( \text{Ray} \left( \text{eye}, \mathbf{p} - \text{eye} \right) \right); \\
        \text{plot} \left( i, j, \text{color} \right); \\
        \end{align*}
    }
}
TraceRay

- TraceRay($r = (\mathbf{o}, \mathbf{d})$) returns the intensity of light arriving at the ray anchor $\mathbf{o}$ in the opposite direction ($-\mathbf{d}$)

- Invoked with ray parameter only
  - Better if object database is global
  - Best if TraceRay is a member function of object database

- Returns intensity across the visible spectrum
  - e.g. an RGB triple

```cpp
Color TraceRay(Ray r, int depth) {
    Color c = background;
    if (!depth) return c;
    if ((hit = Intersect(r)) != NULL) {
        hit->depth = depth - 1;
        c = hit->Shade();
    }
    return c;
}
```
TraceRay

- TraceRay(\(r = (o,d)\)) returns the intensity of light arriving at the ray anchor \(o\) in the opposite direction (-\(d\))
- Invoked with ray parameter only
  - Better if object database is global
  - Best if TraceRay is a member function of object database
- Returns intensity across the visible spectrum
  - e.g. an RGB triple

```
Color TraceRay(Ray r, int depth) {
    Color c = background;
    if (!depth) return c;
    if ((hit = Intersect(r)) != NULL) {
        hit->depth = depth - 1;
        c = hit->Shade();
    }
    return c;
}
```
Intersection Computation

• Parametric ray: \( \mathbf{r}(t) = \mathbf{o} + t \mathbf{d} \)
  - \( t \geq 0 \)
  - Since \( \|\mathbf{d}\| = 1 \), \( t \) is distance along ray

• Implicit object: \( f(\mathbf{x}) = 0 \)
  - \( f(\mathbf{x}) > 0 \) outside, \( f(\mathbf{x}) < 0 \) inside
  - Or vice-verse, doesn’t matter

• Intersection occurs when \( f(\mathbf{r}(t)) = 0 \)
  - Let \( F(t) = f(\mathbf{r}(t)) \)
  - Real function of one real variable
  - Intersection \( \equiv \) root finding
Sphere Intersection

\[ f(x) = (x - c) \cdot (x - c) - r^2 \]
\[ f(r(t)) = (o + t \cdot d - c) \cdot (o + t \cdot d - c) - r^2 \]
\[ = d \cdot d \ t^2 + 2 (o-c) \cdot d \ t + (o-c) \cdot (o-c) - r^2 \]

\[ t = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A} \]

\[ A = d \cdot d \ (= 1 \text{ if } d \text{ unit length}) \]
\[ B = 2 (o-c) \cdot d \]
\[ C = (o-c) \cdot (o-c) - r^2 \]

D = B* B - 4*A*C;
if (D < 0.0) return NULL;
rootD = sqrt(D);
t0 = 0.5*(-B - rootD)/A;
t1 = 0.5*(-B + rootD)/A;
if (t0 >= 0)
    hit->t = t0, return hit;
if (t1 >= 0)
    hit->t = t1, return hit;
return NULL;

Hit position \( x = o + t \cdot d \)  
Hit normal \( n = (x - c)/\|x - c\| \)
Shading

• Hit point \( \mathbf{x} = \mathbf{o} + t\mathbf{d} \)
• Shadows
  – Create light vector
  – Trace ray to light source
• Mirrors
  – Reflect a ray about normal
  – Trace reflected ray to determine reflection color
• Glass
  – Use normal to refract ray
  – Trace refracted ray to determine transmitted color