Ray-Tracing

CPSC 314

CAD Raytraced Image of Audi R8C

Raytracing

University of British Columbia

Raytracing

University of British Columbia

© Michiel van de Panne

© Michiel van de Panne
**Overview**

**So far**
- projective rendering (hardware)
- radiosity

**Ray-Tracing**
- simple algorithm for software rendering
- extremely flexible
- well suited to transparent and specular objects
- global illumination (*)
- partly physics-based: geometric optics

---

**Ray-Tracing**

\[
\text{raytrace( ray )} \quad \{
\text{find closest intersection}
\text{cast shadow ray, calculate colour_local}
\text{colour_reflect} = \text{raytrace( reflected_ray )}
\text{colour_refract} = \text{raytrace( refracted_ray )}
\text{colour} = k1*\text{colour_local} + \text{colour_reflect} + \text{colour_refract}
\text{return( colour )}
\}
\]

- “raycasting” : only cast first ray from eye
Ray-Tracing

Ray Termination Criteria:

- ray hits a diffuse surface
- ray exits the scene
- threshold on contrib. towards final pixel colour
- maximum recursion depth

Ray-Tracing – Generation of Rays

Camera Coordinate System

Ray in 3D Space:

\[ R_{i,j}(t) = C + t \cdot (P_{i,j} - C) \]

where \( t = 0 \ldots \infty \)

Task:

- Given an object \( o \), find ray parameter \( t \), such that \( R_{i,j}(t) \) is a point on the object
  - Such a value for \( t \) may not exist
- Intersection test depends on geometric primitive

Ray Intersections

Sphere at origin:

- Implicit function:
  \[ S(x, y, z) : x^2 + y^2 + z^2 = r^2 \]

- Ray equation:
  \[
  R_{i,j}(t) = C + t \cdot v_{i,j} = \begin{pmatrix} c_x \\ c_y \\ c_z \end{pmatrix} + t \cdot \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} = \begin{pmatrix} c_x + t \cdot v_x \\ c_y + t \cdot v_y \\ c_z + t \cdot v_z \end{pmatrix}
  \]
Ray Intersections

**To determine intersection:**

- Insert ray $R_{i,j}(t)$ into $S(x,y,z)$:
  
  $$(c_x + t \cdot v_x)^2 + (c_y + t \cdot v_y)^2 + (c_z + t \cdot v_z)^2 = r^2$$

- Solve for $t$ (find roots)
  - *Simple quadratic equation*

---

Ray Intersections

**Triangles:**

$$p = (1-\beta-\gamma)a + \beta b + \gamma c$$

Figure 10.5: The ray hits the plane containing the triangle at point $p$.

---

Ray Tracing

**Triangle Intersection (cont.)**

Cramer's rule gives us

$$\beta = \frac{i(ei-hf)+k(gf-di)+l(dh-eg)}{M}$$

$$\gamma = \frac{i(ak-jb)+h(jc-ol)+g(bl-ke)}{M}$$

$$t = -\frac{f(ak-jb)+e(jc-ol)+d(bl-ke)}{M}$$

where

$$M = a(ei-hf) + b(gf-di) + c(dh-eg).$$

---

Ray Tracing

**Triangle intersection (cont.): check bounds**

Check $0 \leq B \leq 1$

$0 \leq \gamma \leq 1$

$0 \leq 1-\beta-\gamma \leq 1$

$\gamma > 0$
Ray Transformation:

- For intersection test, it is only important that ray is in the same coordinate system as object representation.
- Transform ray into object coordinates:
  - Transform camera point and ray direction by the inverse of the model/view matrix.
- Shading has to be done in world coordinates (where light sources are given):
  - Transform object space intersection point to world coordinates.
  - Thus have to keep both world and object-space ray.