Computer Graphics 10 - Ray tracing

Tom Thorne

Slides courtesy of Taku Komura
www.inf.ed.ac.uk/teaching/courses/cg
Overview

- Ray tracing overview
- Ray trees
- Intersections
  - Spheres
  - Planes
  - Polygons
- Bounding volumes
  - Bounding volume hierarchies
Ray tracing (Appel ’68)

- One of the most popular methods used in 3D computer graphics to render an image
- Different from the rasterisation-based approach
- Good at simulating specular effects, producing shadows
- Also used as a function for other global illumination techniques
Ray tracing

- Tracing the path taken by a ray of light through the scene

- Rays are cast to each pixel. They are reflected, refracted, or absorbed whenever they intersect objects
Procedure

- Rays that miss the objects are coloured as the background
Procedure

- When a ray hits an object...
Procedure

- Check for shadowing:
  - Cast a shadow ray towards each light source
Shadow rays

- If shadow ray hits another object, only apply ambient lighting
- Otherwise perform local Phong illumination
Shadow rays
Reflected rays

- Also generate a reflected ray, and test for intersections with the scene
Reflected rays

- If the reflected ray intersects an object, apply local illumination at intersection point, and return result to original intersection point
Refracted rays

- If the object is transparent, calculate refracted ray based on Snell’s law

\[ T = rI + (w - k)n \]
\[ r = \frac{n_1}{n_2} \]
\[ w = -(I \cdot n)r \]
\[ k = \sqrt{1 + (w - r)(w + r)} \]
Refracted rays

- As with reflection, calculate local illumination of intersection of refracted ray, and return to original intersection
Ray tracing outline

- Shadow ray
- Reflection ray
- Refraction ray
- Combine contributions from each ray:

\[ I = I_{local} + k_r R + k_t T \]
Ray Tree (Whitted '80)

- Reflection and refraction rays are recursively cast on hitting a surface
- Performed to some depth and then returned to the previous hits
Test scene

- Ray tree of depth 1. Mirror and teapot are reflective but no reflected ray is cast
Test scene

- Ray tree of depth 2. Reflection of mirror and teapot have no reflections on them!
Test scene

- Ray tree of depth 3. Reflection of mirror on reflected teapot has no reflection.
Test scene

- Ray tree of depth 4. No reflection on teapot in reflection of mirror on teapot in mirror.
Test scene

- Ray tree of depth 5...
Test scene

- Ray tree of depth 6...
Test scene

- Ray tree of depth 7...
Ray trees on a specular surface

- Compute the colour of each ray:

\[
I = I_{local} + K_r R + K_t T
\]
\[
R = I'_{local} + K'_r R' + K'_t T'
\]
\[
R'' = I''_{local} + K''_r R'' + K''_t T''
\]
\[\vdots\]

- In one single equation:

\[
I = I_{local} + K_r (I'_{local} + K'_r (I''_{local} + K''_r (I'''_{local} + K'''_r (\ldots))))
\]
Stopping

- Need to decide when to stop:
  - When we hit a completely diffuse surface
  - On specular surfaces at some fixed depth
  - Once the product of coefficients falls below a threshold

\[ I = I_{local} + K_r (I'_{local} + K'_r (I''_{local} + K''_r (I'''_{local} + K'''_r (...)))) \]

\[ K_r K'_r K''_r K'''_r ... < \text{threshold} \]

Examples
Complexity?

- Ray tracing - at a resolution of $w$ by $h$, and $N$ triangles, $O(\ ?)$
- Rasterisation - with $V$ vertices and $N$ triangles, $O(\ ?)$
Overview

• Ray tracing overview
• Ray trees

• **Intersections**
  • Spheres
  • Planes
  • Polygons

• Bounding volumes
  • Bounding volume hierarchies
Parametric representation of rays

- Ray is a line from some origin $e$ in direction $d$. E.g. starting at the camera in the direction of the pixel, or starting on the surface in the direction of reflection or refraction.

- Given an object represented by an implicit surface we can find the value of $t$ at which the ray intersects the object.

- Knowing $t$ at the intersection we can calculate the coordinates of the intersection.

$$r(t) = e + td$$
Implicit representation of spheres

We can represent a sphere using an implicit equation of the form $f(p) = 0$.

A sphere is defined by $(x - s_x)^2 + (y - s_y)^2 + (z - s_z)^2 = r^2$, so for a sphere with center at coordinates $s$ and of radius $r$:

$$(p - s) \cdot (p - s) - r^2 = 0$$
Ray/sphere intersection

To find the intersection of a ray with a sphere, we substitute \( r(t) = e + td \) into the implicit equation for a sphere:

\[
(e + td - s) \cdot (e + td - s) - r^2 = 0
\]

\[
(d \cdot d)t^2 + 2d \cdot (e - s)t + (e - s) \cdot (e - s) - r^2 = 0
\]

This is a quadratic equation in \( t \), e.g. \( at^2 + bt + c = 0 \), and so we can find the solutions for \( t \) using:

\[
t = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}
\]
Ray/sphere intersection

This gives us the solution for \( t \) as:

\[
t = \frac{-2d \cdot (e - s) \pm \sqrt{(2d \cdot (e - s))^2 - 4(d \cdot d)((e - s) \cdot (e - s) - r^2)}}{2(d \cdot d)}
\]

With the number of solutions determined by the value in the square root.

- If \( b^2 - 4ac > 0 \) there are two intersections of the ray with the sphere
- If \( b^2 - 4ac = 0 \) the ray grazes the sphere and there is a single intersection
- If \( b^2 - 4ac < 0 \) the ray misses the sphere completely.
16.6 Ray/Sphere Intersection

Figure 16.5. The left image shows a ray that misses a sphere and consequently $b^2 - c < 0$. The middle image shows a ray that intersects a sphere at two points ($b^2 - c > 0$) determined by the scalars $t_1$ and $t_2$. The right image illustrates the case where $b^2 - c = 0$, which means that the two intersection points coincide.

The last step comes from the fact that $d$ is assumed to be normalized, i.e., $d \cdot d = ||d||^2 = 1$. Note surprisingly, the result in equation is a polynomial of the second order, which means that if the ray intersects the sphere, it does so at up to two points (see Figure 16.5). If the solutions to the equation are imaginary, then the ray misses the sphere. If not, the two solutions $t_1$ and $t_2$ can be inserted into the ray equation to compute the intersection points on the sphere.

The resulting Equation 16.10 can be written as a quadratic equation:

$$t^2 + 2tb + c = 0$$

where $b = d \cdot (o - c)$ and $c = (o - c) \cdot (o - c) - r^2$. The solutions of the second-order equation are shown below:

$$t = -b \pm \sqrt{b^2 - c}.$$  \hfill (16.12)

Note that if $b^2 - c < 0$, then the ray misses the sphere and the intersection can be rejected and calculations avoided (e.g., the square root and some additions). If this test is passed, both $t_0 = -b - \sqrt{b^2 - c}$ and $t_1 = -b + \sqrt{b^2 - c}$ can be computed. To find the smallest positive value of $t_0$ and $t_1$, an additional comparison needs to be executed.

If these computations are instead viewed from a geometric point of view, then better rejection tests can be discovered. The next subsection describes such a routine.

For the other quadrics, e.g., the cylinder, ellipsoid, cone, and hyperboloid, the mathematical solutions to their intersection problems are almost as straightforward as for the sphere. Sometimes, however, it is necessary to bound a surface (for example, usually you do not want a cylinder to be infinite, so caps must be added to its ends), which can add some complexity to the code.

$$r(t) = o + dt$$
Implicit representation of planes

A plane can be described by the implicit equation

\[(p - s) \cdot n = 0\]

where \(s\) is a point on the plane, and \(n\) is the normal vector to the plane. Points \(p\) satisfying this equation lie on the plane.

For points \(a, b\) on the plane:

\[n = (a - s) \times (b - s)\]

\[(p - s) \cdot ((a - s) \times (b - s)) = 0\]
Ray/plane intersections

To calculate the intersection of a ray with a plane we substitute the equation for the points on the ray into the implicit plane equation:

\[(e + td - s) \cdot n = 0\]
\[(e - s) \cdot n + td \cdot n = 0\]
\[t = \frac{(s - e) \cdot n}{d \cdot n}\]

In the case where \(d \cdot n = 0\) the ray is parallel to the plane, and so does not intersect it.
Ray/triangle intersection

First perform intersection with the plane:

\[ t = \frac{(s - e) \cdot n}{d \cdot n} \]

Then test if the point \( r(t) = e + td \) lies within the triangle.
Projection onto primary planes

To make things simpler, we project the triangle onto one of the planes corresponding to a pair of axes (xy, yz or xz).

- We chose the plane on which the triangle has the largest projection, using the normal vector $n$.
- The largest component of $n$ is dropped e.g. if $|n_y|$ is the largest we project onto the xz plane, dropping the $y$ coordinate.
Projection onto primary planes

After projection to a 2D plane we can test for a point being inside the triangle using barycentric coordinates:

\[
\begin{align*}
\alpha &= \frac{f_{P_1P_2}(x, y)}{f_{P_1P_2}(x_0, y_0)} \\
\beta &= \frac{f_{P_2P_0}(x, y)}{f_{P_2P_0}(x_1, y_1)} \\
\gamma &= \frac{f_{P_0P_1}(x, y)}{f_{P_0P_1}(x_2, y_2)},
\end{align*}
\]

where

\[
f_{pq}(x, y) = (y_q - y_p)x - (x_q - x_p)y + x_q y_p - y_q x_p
\]
Overview

• Ray tracing overview

• Ray trees

• Intersections
  • Spheres
  • Planes
  • Polygons

• Bounding volumes
  • Bounding volume hierarchies
Bounding volumes

- We want to reduce number of ray-object intersections to test
- Use bounding volumes:
  - Test for an intersection with bounding volume
  - Only test intersection with objects inside volume if we intersect the bounding volume
- Boxes, spheres
Hierarchical structures

- Enclose objects in hierarchical bounding volumes
- Octrees, KD-trees
- Bounding volume hierarchies
Bounding volume hierarchy

- Give each object a bounding volume
- The bounding volume does not partition
- The bounding volumes can overlap each other
- The volume higher in the hierarchy contains their children
- If a ray misses a bounding volume, no need to check for intersection with children
- If we intersect a bounding volume, check intersection with children
Producing the hierarchy

- Find bounding box of objects
Producing the hierarchy

- Find bounding box of objects
- Split into two groups
Producing the hierarchy

- Find bounding box of objects
- Split into two groups
- Recurse
Producing the hierarchy

- Find bounding box of objects
- Split into two groups
- Recurse
Producing the hierarchy

- Find bounding box of objects
- Split into two groups
- Recurse
Producing the hierarchy

- Find bounding box of objects
- Split into two groups
- Recurse
Where to split?

- At midpoint
- Sort and put half on each side
Computing intersections

```c
intersect(node, ray, hits) {
    if( intersectp(node->bound, ray)
        if( leaf(node) )
            intersect(node->prims, ray, hits)
        else
            for each child
                intersect(child, ray, hits)
}
```
Summary

- Simple but computationally expensive
- Easily includes reflection, refraction and shadows
- Calculating intersections is main bottleneck
- Reduce the number of intersection calculations using a bounding volume hierarchy
References

- Shirley Chapter 4 (Ray tracing)
- Shirley Chapter 12.3 (12.3.1, 12.3.2) (Spatial Data Structures)
- Foley Chapter 15.10 (Visible-surface ray tracing), 16.11, 16.12 (Global illumination, Recursive ray tracing)
- Akenine-Möller Chapter 16.6, 16.8 (Ray/Sphere intersection, Ray/Triangle intersection)