Ray Tracing Complex Scenes

Timothy L. Kay and James T. Kajiya

presented by Lei Yuan
Introduction

• Published in August 1986
• Purpose:
  To develop a more efficient algorithm for computing ray-object intersection.
Previous Methods

• Bounding Volume
• Space Partition
• Hierarchy Tree
Overview

• New Type of Bounding Volume
  ◆ Arbitrarily tight to the object
  ◆ Easy to compute intersection

• New Traverse Algorithm
  For a given ray, the objects are queried in an efficient order regardless of the hierarchy.
Bounding Volume

• Parallelopipeds constructed of planes
• *Implicit equation:* $Ax + By + Cz - d = 0$
• *Normal* $Pi = \{A, B, C\}$, $d$: $d$ units to the origin
Figure 3: Bounding an object using various normals
Bounding Volume

- To avoid the cost of memory and computation

Figure 4: Objects bounded by a fixed set of normals
Bounding Volume

• Polyhedra: collection of vertices

1. Transformation to world coordinates:

\[
\begin{pmatrix}
  x_j^i \\
  y_j^i \\
  z_j^i
\end{pmatrix} = M \begin{pmatrix}
  x_j \\
  y_j \\
  z_j
\end{pmatrix} + T.
\]

2. Projection to Normal $\Pi_i$:

\[
d_{ij} = (\hat{P}_i)^T \begin{pmatrix}
  x_j^i \\
  y_j^i \\
  z_j^i
\end{pmatrix}.
\]

3. For each $\Pi_i$, find the min($d_{ij}$) and max($d_{ij}$)
Bounding Volume

• Implicit surfaces: Points satisfy a function
• Implicit Function: \( g(x, y, z) = 0 \)
• Let \( f(x, y, z) \) be the function of \( d \) value.

\[
f \left( \begin{array}{c} x \\ y \\ z \end{array} \right) = \hat{P}_i^T \left( M \left( \begin{array}{c} x \\ y \\ z \end{array} \right) \right).
\]

• We want to find extreme value of \( f() \) using Lagrange Multipliers.
Bounding Volume

- Example: Sphere

\[ f \left( \begin{pmatrix} x \\ y \\ z \end{pmatrix} \right) = P_i^T \left( M \left( \begin{pmatrix} x \\ y \\ z \end{pmatrix} \right) \right). \]

\[ \Rightarrow \quad f \left( \begin{pmatrix} x \\ y \\ z \end{pmatrix} \right) = (P_i^T M) \left( \begin{pmatrix} x \\ y \\ z \end{pmatrix} \right). \]

- Let

\[ \begin{pmatrix} A \\ B \\ C \end{pmatrix} = (P_i^T M)^T = M^T P_i, \]

\[ \Rightarrow \quad f \left( \begin{pmatrix} x \\ y \\ z \end{pmatrix} \right) = Ax + By + Cz. \]

- Constrains: \( g(x, y, z) = x^2 + y^2 + z^2 - 1 = 0. \)
Bounding Volume

- Lagrange Multiplier

When \( \begin{pmatrix} A \\ B \\ C \end{pmatrix} = \lambda \begin{pmatrix} 2x \\ 2y \\ 2z \end{pmatrix} \), \( f(x, y, z) \) has extremes. Then we substitute it back:

\[
    f \left( \begin{pmatrix} x \\ y \\ z \end{pmatrix} \right) \bigg|_{x^2+y^2+z^2-1=0} = \pm \sqrt{A^2 + B^2 + C^2}.
\]

Then we get 
\[(d_i^\text{near}, d_i^\text{far}) = (T \cdot \hat{P}_i - |M^T \hat{P}_i|, T \cdot \hat{P}_i + |M^T \hat{P}_i|).\]
Bounding Volume

• Compound objects

Figure 5: The bounding volume of two bounding volumes
Ray Volume Intersection

- Compute all the intervals
- Ray intersects Planes:

\[ \mathbf{R} = \hat{\mathbf{a}}t + \mathbf{b} \quad t = \frac{d_i - \hat{\mathbf{P}}_i \cdot \mathbf{b}}{\hat{\mathbf{P}}_i \cdot \hat{\mathbf{a}}}. \]

- For a given ray:

\[ \mathbf{S} = \hat{\mathbf{P}}_i \cdot \mathbf{b} \quad T = \frac{1}{\hat{\mathbf{P}}_i \cdot \hat{\mathbf{a}}} \cdot \quad t = (d_i - S)T. \]
Traverse Algorithm

• Initialize:
  1. We are given a ray
  2. Compute dot products and reciprocals
  3. Let $t$ be the nearest distance
     Initially: $t=+\infty$
  4. Let $p$ points to the nearest object hit so far
     Initially: $p=null$;
Traverse Algorithm

• Traverse

While heap is non-empty and distance to top node < t
  Extract candidate from heap
  If the candidate is a leaf
    Compute ray-object intersection
    If ray hits candidate and distance < t then
      \[ t = \text{distance} \]
      \[ p = \text{candidate} \]
    Endif
  Else
    For each child of the candidate
      Compute ray-bounding volume intersection
      If the ray hits the bounding volume
        Insert the child into the heap
      Endif
    Endfor
  Endif
Endwhile
Traverse Algorithm

- Priority Queue (Heap)
  Sort according to the Estimated d value
  = the nearest d value
  Compute in Ray Volume intersection.

Advantage:
  The search order is derived when the traverse is in progress regardless of the Tree’s Structure
Result

• Significantly faster
  It runs 2.6 times faster than Glassher’s method when reproducing a same model.