33._Möller-Trumbore_algorithm

15 April 2024 22:50

Source: 1. Ray Tracing Essentials, Part-1 to 7
By Nefi Alarcon, NVIDIA

2. **3D Computer Graphics Primer: Ray-Tracing as an Example**

From <https://www.scratchapixel.com/lessons/3d-basic-rendering/introduction-to-ray-tracing/implementing-the-raytracing-algorithm.html>

Möller-Trumbore Algorithm :-

A faster ray-triangle intersection algorithm.
(1997)

Using barycentric coordinates equation,

$$P = wA + uB + vC \quad \text{--- (1)}$$

also $\quad w + u + v = 1$

$$\Rightarrow w = 1 - u - v \quad \text{--- (2)}$$

Using (2) in (1)

$$P = (1-u-v)A + uB + vC$$

$$P = A - Au - Av + uB + vC$$

$$\boxed{P = A + u(B-A) + v(C-A)} \quad \text{--- (3)}$$

$(B-A) \rightarrow$ AB edge

$(C-A) \rightarrow$ AC edge

Also, $\quad P = O + t\vec{D}$

using in (3)

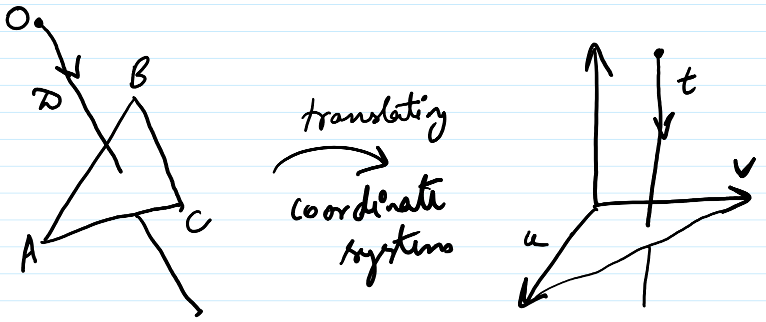
$$O + t\vec{D} = A + u(B-A) + v(C-A)$$

$$\boxed{O - A = -t\vec{D} + u(B-A) + v(C-A)}$$

$$\underbrace{\begin{bmatrix} -D & (B-A) & (C-A) \end{bmatrix}}_{\text{known}} \underbrace{\begin{bmatrix} t \\ u \\ v \end{bmatrix}}_{\text{unknowns.}} = \underbrace{(O-A)}_{\text{known.}}$$

where $-\vec{D}$, $(B-A)$, $(C-A)$, $(O-A)$ are vectors

$\underbrace{t}_{\substack{\text{distance}\\\text{from ray to}\\\text{intersection point}}}, \underbrace{u, v}_{\text{barycentric coordinates}}$ are scalar quantities



Using Cramer's Rule,

$$\begin{bmatrix} t \\ u \\ v \end{bmatrix} = \frac{1}{\left| -D \quad B-A \quad C-A \right|} \begin{bmatrix} \left| O-A \quad B-A \quad C-A \right| \\ \left| -D \quad O-A \quad C-A \right| \\ \left| -D \quad B-A \quad O-A \right| \end{bmatrix}$$

$$|A\,B\,C| = (C \times A) \cdot B$$

$$= \frac{1}{(D \times (C-A)) \cdot (B-A)} \begin{bmatrix} ((C-A) \times (O-A)) \cdot (B-A) \\ ((C-A) \times -D) \cdot (O-A) \\ ((O-A) \times -D) \cdot (B-A) \end{bmatrix}$$

$\Rightarrow$ t, u, v can be calculating using cross and dot products between vertices of the triangle, origin and the ray direction.

Advantage :-

Plane equations need not be computed on the fly or stored.

Fast, Minimum Storage Ray/Triangle Intersection

Tomas Möller
Prosolvia Clarus AB
Chalmers University of Technology
E-mail: `tompa@clarus.se`

Ben Trumbore
Program of Computer Graphics
Cornell University
E-mail: `wbt@graphics.cornell.edu`

Abstract

We present a clean algorithm for determining whether a ray intersects a triangle. The algorithm translates the origin of the ray and then changes the base of that vector which yields a vector $(t\ u\ v)^T$, where t is the distance to the plane in which the triangle lies and (u, v) represents the coordinates inside the triangle.

One advantage of this method is that the plane equation need not be