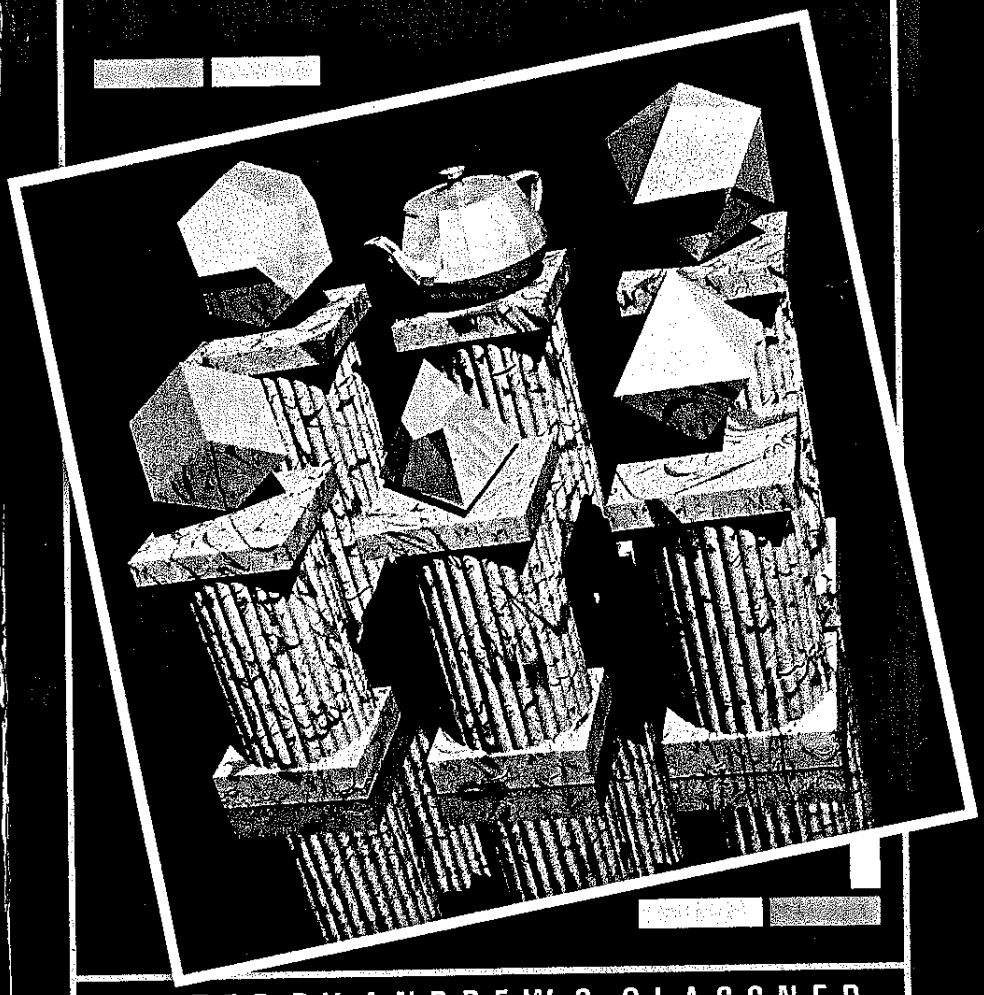
UNIVERSITY OF CALIFORNIA, SAN DIEGO
3 1822 02339 0560

AN INTRODUCTION TO RAY TRACING



EDITED BY ANDREW S. GLASSNER

DOCKET ALARM
Find authenticated court documents without watermarks at docketalarm.com.

An Introduction to Ray Tracing

Edited by
ANDREW S. GLASSNER

Xerox PARC
3333 Coyote Hill Road
Palo Alto CA 94304
USA

ACADEMIC PRESS
Harcourt Brace Jovanovich, Publishers
London · San Diego · New York · Berkeley · Boston
Sydney · Tokyo · Toronto

DOCKET ALARM
Find authenticated court documents without watermarks at docketalarm.com.

ACADEMIC PRESS LIMITED
24/28 Oval Road, London NW1 7DX

United States Edition Published by
ACADEMIC PRESS INC.
San Diego, CA 92101

Copyright © 1989 by
ACADEMIC PRESS LIMITED

All rights reserved. No part of this book may be reproduced in any form by photostat, microfilm, or any other means, without written permission from the publishers

British Library Cataloguing in Publication Data

An Introduction to ray tracing.
1. Computer systems. Graphic displays. Three-dimensional images.
I. Glassner, Andrew
006.6

ISBN 0-12-286160-4

Typeset by Mathematical Composition Setters Ltd, Salisbury
Printed in Great Britain at the University Press, Cambridge

DOCKET ALARM
Find authenticated court documents without watermarks at docketalarm.com.

2.6 Reflection Rays

If we look at a perfectly flat, shiny table, we will see reflections of other objects in the tabletop. We see those reflections because light is arriving at the tabletop from the other objects, bouncing off of the tabletop, and then arriving in our eye. For a fixed eyepoint, each position on the table has exactly one direction from which light can come that will be bounced back into our eye.

For example, *Figure 9* shows a photon of light bouncing around a scene, ending up finally passing through the screen and into the eye. On its last bounce, the photon hit point P and then went into the eye. Photon B also hit point P, but it was bounced (or reflected) into a direction that didn't end up going into our eye. So for that eyepoint and that object, only a photon travelling along the path marked A could have been reflected into our direction of interest.

When we wish to find what light is reflected from a particular point into the direction of the incident ray, we find the *reflected ray* (or *reflection ray*) for that point and direction; this is the ray that can carry light to the surface that will be perfectly reflected into the direction of the incident ray. To find the color of the reflected ray, we follow it backwards to find from which object it began. The color of the light leaving that object along the line of the reflected ray is the color of that reflected ray. When we know the reflected ray's color, we can contribute it to any other light leaving the original surface struck by the incident ray.

Note the peculiar terminology of backward ray tracing: light arrives along the reflected ray and departs along the incident ray.

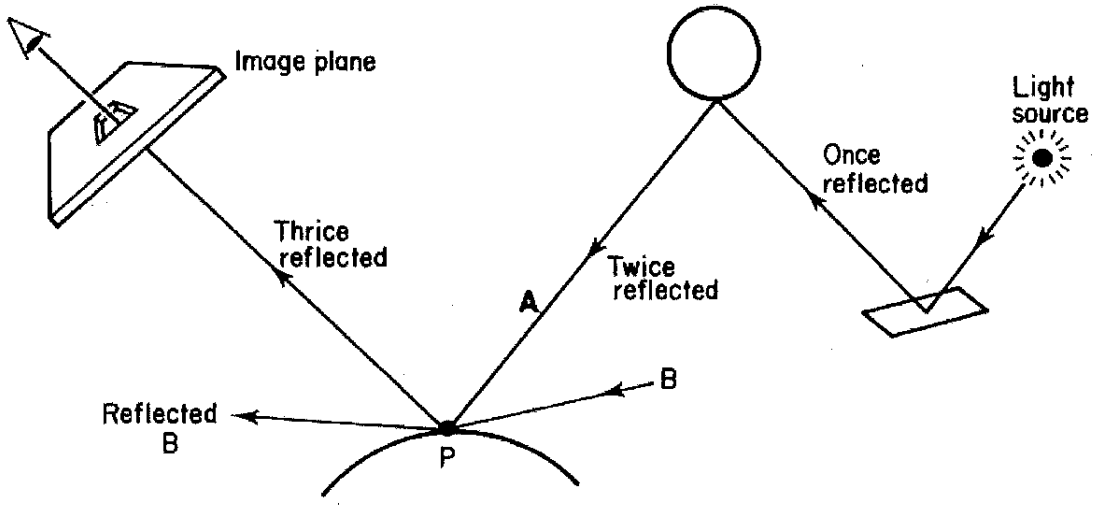


Fig. 9. The color of perfectly reflected light is dependent on the color of the object and the color of the incoming light that bounces off in the direction we care about. For example, at point P we want to know the color of the light

DOCKET ALARM Find authenticated court documents without watermarks at docketalarm.com.

```
for each entity, A, from which rays can originate do begin
  for each object B do begin
    d = lower bound on distance between A and B;
    S = direction bound for rays from A which hit B;
    for each direction cell of A which intersects S do begin
      insert B into the cell's sorted candidate list
      according to the distance d;
      endfor
    endfor
  endfor
```

Fig. 21. The pre-processing algorithm of the 'ray coherence' algorithm. The sorting operation can be performed by insertion, as shown here, or after all the candidate lists have been formed.

Figure 21, is greater than the distance to a known point of intersection. This is the distance interval optimization yet again.

An outline of the pre-processing algorithm which creates the candidate lists is shown in *Figure 21*. It is assumed that bounding volumes are all spheres or all convex polyhedra. The direction bound, S, will be a unit vector and an angle (or cosine) defining a cone in the case of bounding spheres and a spherical convex hull in the case of bounding polyhedra.

6.4 Ray Classification

The ray classification algorithm, described by Arvo and Kirk [3], does not use explicit direction cubes except in the special case of first-generation rays. The data structure used to accelerate the intersection process for other rays is closely tied to the concept of a direction cube, however. Ray classification is based upon the observation that rays in three-space have five degrees of freedom and correspond to the points of $R^3 \times S^2$, where S^2 is the unit sphere in three-space. The algorithm proceeds by partitioning the five-dimensional space of rays into small neighborhoods, encoded as 5-D hypercubes, and associating a complete list of candidate objects with each. A hypercube represents a collection of rays with similar origins and similar directions, and its associated candidate list contains all objects which are hit by any of these rays (neglecting occlusion). To intersect a ray with the environment, we locate the hypercube which contains the 5-D equivalent of the ray and test only the objects in the associated candidate list.

Rays with a given dominant direction can be conveniently encoded as 5-tuples, (x, y, z, u, v), where the first three elements specify the origin of the ray, and the last two are the UV direction coordinates obtained from a face of the direction cube. Any ray in three-space can be specified by such a 5-tuple

DOCKET ALARM Find authenticated court documents without watermarks at docketalarm.com.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.