Some Regularization Problems in Ray Tracing

John Amanatides
Don P. Mitchell

AT&T Bell Laboratories
Murray Hill, NJ 07974

1. Abstract

Ray tracers that render CSG models should consider issues of regularization and numerical accuracy. The special case of rays originating on surfaces (shadow probes, reflections, and refractions) present a regularization problem that is significant—even in ray tracers which are not explicitly based on the CSG scheme. An analysis of this problem yields a better solution than the epsilon tests incorporated in most ray tracers.

2. Introduction

Ray tracing is a simple and straightforward method of rendering images of solids described by the Constructive Solid Geometry scheme. This is because the problem of Boolean combinations of solids is reduced to one dimension.

Two complications arise, however. Set operations on solids should be regularized [Requicha80], and finite-precision machine arithmetic introduces error into the ray-intersection calculation. Simple proximity-regularization rules can be developed which tolerate some inaccuracy, but these rules can produce unwanted side effects.

Even in ray tracers which are not explicitly based on the CSG scheme, there is a regularization and accuracy problem that is important. That is the case of rays originating on the surface of a solid (for shadow calculation, reflection, or refraction).

For these rays, proximity-regularization rules are not unlike fixes in many ray tracers (which eliminate intersections too close to the origin of the ray), but they can still result in some bad pixels. We develop more robust heuristics to improve the effectiveness of these regularization rules.

3. Ray Intersection and Parametric Sequences

In ray tracers, the primary geometric computation is finding the intersection of rays with a solid. The intersection of geometric sets with CSG-defined solids can be computed by recursive divide-and-conquer algorithms [Tilove80], and this approach has been used to ray trace CSG models [Roth82].† The general idea is that if the intersection of a ray with two solids A and B is known, then the intersection with some Boolean combination (e.g., $A \cup B$, $A \cap B$, or $A - B$) can be inferred.

To describe such an algorithm in more detail, we begin by developing representations of rays and their subsets. Given an origin point O (in an affine space) and a direction vector $\bar{D}$, the points of a ray can be defined parametrically:

$$R = \{O + t\bar{D} \mid 0 \le t \le \infty\} \tag{1}$$

The intersection of a ray R with a solid S will be some subset of the ray, which can be represented by a *parametric sequence.* In our notation, this is a non-decreasing sequence of ray parameter values t_i corresponding to transitions between inside and outside the set $R \cap S$.

---

† Formal treatments like Tilove's are based on the concept of *classification,* rather than intersection. The classification of a ray R with respect to a solid S is a partitioning of the ray into RinS, RonS, and RoutS, which are points inside S, on its boundary, or outside the solid. We will simply discuss the problem in terms of ray intersection.
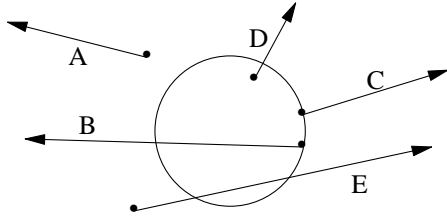
Figure 1. Rays Intersecting a Solid

Figure 1 illustrates a number of interesting cases of ray intersection. The circle can be thought of as representing a solid sphere or a hollow bubble in a unbounded solid. In both cases, the object is *closed,* meaning that the boundary is part of the solid. The table below gives parametric sequences corresponding to the ray intersections shown above.

Ray	Ray $\cap$ Sphere	Ray $\cap$ Bubble
A	( )	(0)
B	$(0, t_1)$	$(0, 0, t_1)$
C	$(0, 0)$	(0)
D	$(0, t_1)$	(t_1)
E	(t_1, t_2)	$(0, t_1, t_2)$

Table 1. Parametric Sequences Associated with Ray Intersections

In our notation, the first element of a parametric sequence always represents the first transition from outside $R \cap S$ to inside. Thus, when ray E intersects the bubble, the first inside section begins at the ray origin where $t = 0$. In this case, $[0, t_1]$ represents an inside segment, the open interval (t_1, t_2) is outside, and $[t_2, \infty]$ is inside.

The two simplest parametric sequences are ( ), which represents an empty intersection, and (0), which represents the whole ray.

An unusual case is the intersection of ray C with the sphere. The sequence $(0, 0)$ indicates that a single-point interval $[0, 0]$ is inside the intersection (because the ray origin is on the boundary), and the interval $(0, \infty]$ is outside.

In Roth's divide-and-conquer algorithm, the intersection of a ray with a Boolean combination $A \square B$ is inferred by merging the parametric sequences of the intersections with A and B [Roth82].

Earlier ray tracers also rendered CSG models, but with less efficient algorithms based on point intersection [Goldstein71]. In its simplest form, the point-intersection algorithm must find all primitive intersections and sort them. At that point, it has often done about the same amount of work as Roth's divide-and-conquer algorithm. It must then make the additional effort of calculating point intersections to find the closest primitive intersection that is part of the CSG model.

4. Regular Closure and ON/ON Ambiguity

A formal analysis of solid representation introduces some complications in the divide-and-conquer intersection algorithm. Not all sets of points in space are what we naturally think of as "solid objects". A collection of isolated points, lines or surfaces are not examples of solids, because they have no volume. Even connected subsets with a volume can be unsatisfactory. For example, the object in Figure 2 has a dangling face:
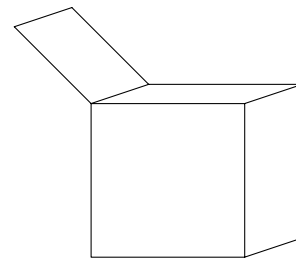

Figure 2. Solid with Dangling Face

Using elementary concepts from topology, a mathematical definition can be found which matches our intuitive notion of a proper solid [Requicha80]. A set of points in space $S \subset \mathbf{R}^3$ is a *regular solid,* if it is equal to its regular closure. The regular closure of a set is the closure of its interior:

$$S \equiv \overline{S^\circ} \tag{2}$$

This is equivalent to removing the boundary† of the set and then closing it—by adding a new boundary. In the case of the object pictured in Figure 2, removing the boundary of the set would remove the dangling

---

† A point is on the *boundary* of a set if every neighborhood of the point is partially inside the set and partially out.

face and leave an open cube. Adding a new boundary would leave a closed cube, which is what we want.

Regular solids will always be made from Boolean combinations of regular solids if the Boolean operations are "regularized", as for example:

$$A \cap^* B \equiv \overline{(A \cap B)^\circ} \qquad (3)$$

In addition to maintaining regular closure, another important formal consideration is *ON/ON ambiguity.* Imagine two blocks, *A* and *B*, which can make face-to-face contact in two ways:
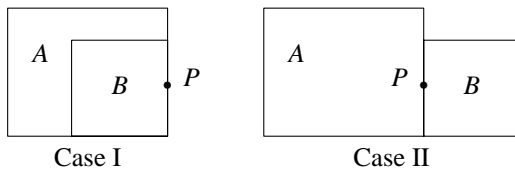

Case I   Case II

Figure 3. Point on Common Boundary of Two Objects

If the point *P* were in the interior or exterior of all primitive solids, then it is straightforward to decide whether it is in some Boolean combination of solids. However, when *P* is on the boundary of two solids, there is ambiguity about the result of regularized Boolean combinations.

Consider the example of $A \cap B$. If we ignore regularization, then this intersection is always nonempty at point *P*. But the value of $A \cap^* B$ is nonempty in case I and empty in case II. This is because, in case II, the nonregularized intersection results in a dangling face.

5. Some Topological Issues in Ray Tracing

Two classes of problems in ray tracing will be discussed in this paper. First, there are topological issues that would need to be considered even if the ray tracing computation were performed with perfect accuracy. Secondly, there are problems that arise from the limited accuracy of machine arithmetic. The purely topological issues will be discussed in this section.

Parametric sequences represent subsets of a ray, and the ray is a one-dimensional subset of $\mathbf{R}^3$. In this 1-D subspace (with its relative topology), regular closure will have the effect of removing isolated points. It

would also remove isolated single-point gaps, but they should not arise from regularized Boolean operations on closed sets. 1-D regular closure can be implemented by a *rewrite rule* for parametric sequences. The rule is simply to delete adjacent pairs of identical parameter values in a sequence. For example:

$$(\cdots t_{i-1}, t_i, t_i, t_{i+1}, \cdots) \qquad (4)$$

$$\rightarrow (\cdots t_{i-1}, t_{i+1}, \cdots)$$

Regularization in a 1-D subspace along the ray intersection is not equivalent to intersecting a ray with a regularized solid. That is, given a ray *R* and a solid *S*:

$$R \cap \overline{S^\circ} \neq \overline{(R \cap S)^\circ} \qquad (5)$$

The following figure shows examples of where 1-D regularization works and where it fails.
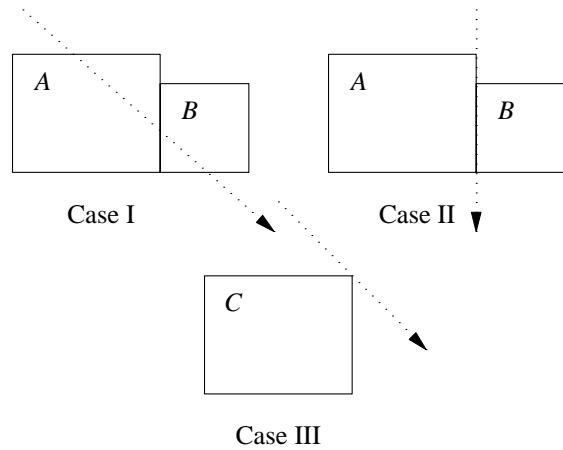

Case I   Case II

Case III

Figure 4. Ray Intersecting with Boundaries and Corners

In case I, a ray passes thru the common boundary between *A* and *B*. Assume that we wish to find the intersection of the ray with $A \cap B$. A properly designed divide-and-conquer intersection routine would yield a merged parametric sequences of the form (t_1, t_1) representing the passage of the ray thru the dangling face in $A \cap B$, and 1-D regularization would reduce this to the empty sequence ( ). That is correct, because in this case $A \cap^* B$ would be the empty set.

In case II, the ray lies *on* the common boundary. The

parametric sequence for the ray intersection with $A \cap B$ is now a finite segment (t_1, t_2), and 1-D regularization will not change it. This is incorrect; the sequences should be ( ) as in case I.

In case III, the ray passes thru a vertex point. The parametric sequence for the ray intersection would be (t_1, t_1), and 1-D regularization would reduce this to the empty sequence ( ). That is incorrect, because the corner of C is a legitimate part of the regular solid.

The failures of 1-D regularization are due to ON/ON ambiguity. It would be possible to correct these failures by augmenting the parametric sequences with 3-D *neighborhood representations* [Reqicha85]. Case II could be corrected by including 3-D edge-neighborhood representations (i.e., the contents of sectors of the cylinder surrounding the ray segment). Case III would require more complex vertex-neighborhood information, in order to be corrected. Case I succeeded because the parametric sequence implicitly contained the necessary face-neighborhood information.

Managing full 3-D neighborhood information in parametric sequences would greatly increase the complexity of the ray-intersection calculation. We agree with Roth that it is not practical or necessary to do this [Roth82]. The cases where 1-D regularization fails are improbable (e.g., a ray lying on a dangling face). In addition, these failure cases should not have a major effect on the appearance of an antialiased image because they affect a zero-measure subset of the view-plane.

Finally, we would like to point out an important special case of 1-D regularization. In ray tracers supporting Whitted's shading model [Whitted80], shadows, specular reflection, and refraction are simulated by casting secondary rays from points of intersection on the surface of objects. When these rays start from a surface and go outward, their intersections will result in parametric sequences of the form:

$$(0, 0, \cdots) \tag{6}$$

As in Figure 1 (rays B and C), a pair of zeros begins the sequence because the ray intersects the solid that it originates on. This self intersection is not desired (we don't want a surface to shadow itself), and 1-D regularization conveniently removes the pair of zeros.

6. Some Numerical-Accuracy Issues in Ray Tracing

Of course, ray tracing programs do not perform calculations with perfect accuracy. The use of finite-precision machine arithmetic introduces errors in the results of ray intersection calculations.

Plate 1 is a visualization of a typical situation of a numerical intersection of a ray with a surface. Here, the neighborhood of an actual ray intersection is examined at the resolution of machine accuracy. The red sphere is the calculated point of intersection of a ray with an implicit surface $F(x, y, z) = 0$. The white spheres represent points where a floating-point evaluation of F is exactly equal to 0.0. We see that the intersection point is not actually on the zero-set of the function, and we also see that the zero-set is not a well-behaved surface.

This inaccuracy (and the discreteness of hardware floating-point numbers) makes it numerically (and topologically) impossible to realize the ideal regularization of solids discussed above. For example, in Case I of Figure 4, the result of computing the ray intersection with $A \cap B$ is likely to produce a parametric sequence like $(t_1, t_1 + \varepsilon)$, for some small ε.

One approach to this problem is to redefine regular closure in a way that allows from some error tolerance. *Proximity regularization* peels off an ε-thick layer from the surface of a solid, and then adds an ε-thick layer to the result. This is analogous to taking the interior and then closure of a set in regularization in $\mathbf{R}^3$. It is straightforward to define a 1-D proximity regularization in terms of rewrite rules on parametric sequences:

$$(\cdots t_{i-1}, t_i, t_i + \varepsilon, t_{i+1}, \cdots) \tag{7}$$

$$\rightarrow (\cdots t_{i-1}, t_{i+1}, \cdots)$$

Proximity regularization consists of applying this rewrite rule, first to inside intervals (where the t_i value appears in an even-numbered position in the sequence), and then to outside intervals. Some form of proximity regularization has been used in CSG ray tracing and in CSG z-buffer algorithms [Wyvill86, Rossignac86].

We make no recommendation on whether one should apply this 1-D regularization in general. There are many things wrong with proximity regularization.

The maximum value of ε is difficult to determine objectively, and often results from trial-and-error. In addition, this type of regularization tends to remove or "erode" thin objects and thin parts of objects.

However, a particular situation definitely requires some form of regularization—rays originating *on* the surface of objects. Ideally, intersection of such rays with the model should result in parametric sequences that begin with zero (see ray B and C in Figure 1). However, as Plate 1 suggests, they will rarely occur when shadow rays, reflection rays, and refraction rays are cast. Even when the best numerical methods are used, more typical situations are illustrated in Figure 5.
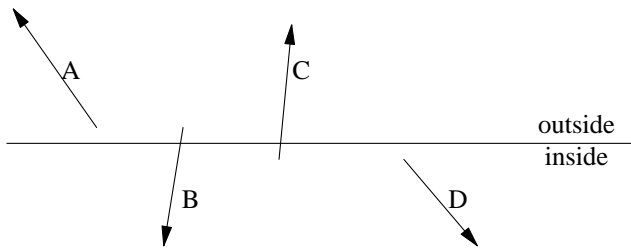

Figure 5. Secondary Rays from Inaccurate Surface-Intersection Points

The upper half of Plate 2 shows the result of performing shadow calculations with no regularization. The black spots on the surface are regions where the shadow rays happen to be like ray C in Figure 5. It intersects the surface that it was suppose to originate from, and so that part of the surface is in shadow.

Similar problems can happen with refraction and reflection. All working ray tracers must have some fix for this problem. Usually, there is a test that rejects intersections that are too close to the origin of a ray. More formally, we would like the rays in Figure 5 to act as if they really originated from *on* the surface. The following special proximity regularization rules can accomplish that:

$$(0, \varepsilon, \cdots) \;\rightarrow\; (0, 0, \cdots) \;\rightarrow\; (\cdots) \qquad (8a)$$

$$(\varepsilon, \cdots) \;\rightarrow\; (0, \cdots) \qquad (8b)$$

Rule (8a) deals with rays exiting a surface (e.g., ray C in Figure 5), such as shadow rays or external reflections. Rule 8b deals with rays penetrating the solid

(e.g., ray B), such as internal reflections or refraction of an external ray into glass. The lower half of Plate 2 shows an image generated using these rules, and the defects have been repaired.

7. Improved Heuristics for Proximity Regularization

The rewrite rules, (8a) and (8b), are adequate to give good behavior for shadows, reflections and refraction so long as the local neighborhood of the ray origin is fairly flat. However, the upper half of Plate 3 shows some bad pixels which occur in the vicinity of a corner inside a block of glass.
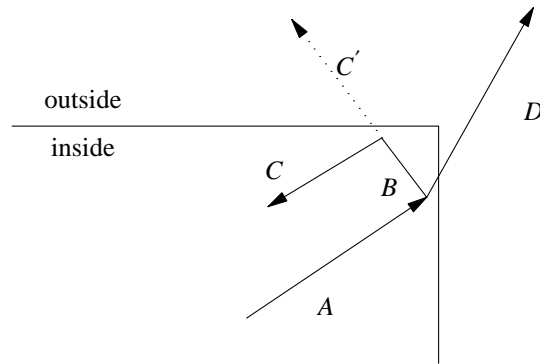

Figure 6. Failure of Proximity Rules Near a Corner

Figure 6 illustrates the problem that is occurring. We want the internal reflection of ray C to occur. But if B is too short, rule (8a) will throw away the intersection with the top face, and some further intersection (if any) in the direction C' will be used instead.

However, in the case of ray D, the same proximity rule was beneficial and allowed a refracting ray to exit the object as we would wish. The difference between ray B and ray D is that we "expect" ray D to give a parametric sequence of the form $(0, 0, \cdots)$ because we know that it is starting on a surface and heading away from the solid. On the other hand, ray B is an internal reflection and we expect a parametric sequence of the form $(0, \cdots)$, where some nonzero intersections may follow the first element in the sequence.

The proximity rule (8a) should only be applied when we expect the ideal parametric sequence to be $(0, 0, \cdots)$, and rule (8b) should only be applied when we expect $(0, \cdots)$. For shadow probes, reflections, and refractions, our expected result depends on whether the parent ray is on the inside of the solid

(like ray *A* in figure 6) or on the outside.

Given	Expected		
parent	shadow	reflection	refraction
outside	$(0, 0, \cdots)$	$(0, 0, \cdots)$	$(0, \cdots)$
inside	$(0, 0, \cdots)$	$(0, \cdots)$	$(0, 0, \cdots)$

Table 2. Expected Form of Secondary-Ray Intersections

It may not seem to make sense to have a shadow ray associated with an internal parent ray, but this is required to correctly compute transmission highlights (an extension to Whitted's model introduced by Hall and Greenberg [Hall83]).

We have found that it is sufficient to execute the proximity rules only after intersection with the primitive solids at the leaves of the CSG tree. It is important to keep track of set-difference operations. If an odd number of set-difference operators are above a primitive, then the inside/outside sense of the parent ray must be reversed.

The lower half of Plate 3 shows how these new proximity rules eliminate the problem near the corner of the glass block.

8. Conclusions

We have taken another look at some fundamental topological and numerical issues in ray tracing. Parametric-sequence notation is introduced to facilitate the discussion of these issues in the context of ray tracing.

We agree with Roth, that full 3-D regularization with neighborhood representations is unnecessary in ray tracers. A 1-D version of proximity regularization can partially simulate true regularization, but we are not sure if the side effects of this approximation are better than the benefits.

It is necessary to apply some form of regularization to the case of rays originating on the surface of objects. This was the one regularization issue that caused real problems for us in the recent development of a ray tracing system. We have found a more robust form of proximity regularization for this special case. This has allowed us to make good images using only single-precision arithmetic.

9. References

[Goldstein71] Goldstein, Robert A. and Roger Nagel, "3-D Visual simulation", *Simulation,* Vol. 16, No. 1, January, 1971, pp 25-31.

[Hall83] Hall, Roy and Donald P. Greenberg, "A Testbed for Realistic Image Synthesis", *IEEE CG&A,* Vol. 3, No. 8, November 1983, pp 10-20.

[Requicha80] Requicha, Aristides A. G., "Representations for Rigid Solids: Theory, Methods, and Systems", *ACM Computing Surveys,* Vol. 12, No. 4, December, 1980, pp 437-464.

[Requicha85] Requicha, Aristides A. G. and H. B. Voelcker, "Boolean Operations in Solid Modeling: Boundary Evaluation and Merging Algorithms", *Proc. IEEE,* Vol. 73, No. 1, January, 1985, pp 30-44.

[Rossignac86] Rossignac, Jaroslaw R., Aristides A. G. Requicha, "Depth-Buffering Display Techniques for Constructive Solid Geometry", *IEEE Computer Graphics and Applications,* September, 1986, pp 29-39.

[Roth82] Roth, Scott D., "Ray Casting for Modeling Solids", *Computer Graphics and Image Processing,* Vol. 18, No. 2, February, 1982, pp 109-144.

[Tilove80] Tilove, Robert Bruce, "Set Membership Classification: A Unified Approach to Geometric Intersection Problems", *IEEE Trans. Computers,* Vol. C-29, No. 10, October 1980, pp 874-883.

[Whitted80] Whitted, Turner, "An improved illumination model for shaded display", *Comm. ACM,* vol. 23, No. 6, June 1980, pp 343-349.

[Wyvill86] Wyvill, Geoff, *et al,* "Space Division for Ray Tracing in CSG", *IEEE Computer Graphics and Applications,* April, 1986, pp 28-34.