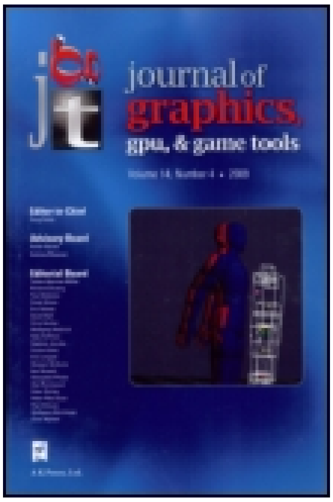

Journal of Graphics Tools
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/ujgt19

A Fast Triangle-Triangle Intersection Test

Tomas Möller [a]
[a] Prosolvia Clarus AB, Chalmers University of Technology , Gardavagen , 8-41250 , Gothenberg , Sweden E-mail:
Published online: 06 Apr 2012.

To cite this article: Tomas Möller (1997) A Fast Triangle-Triangle Intersection Test, Journal of Graphics Tools, 2:2, 25-30, DOI: 10.1080/10867651.1997.10487472

To link to this article: http://dx.doi.org/10.1080/10867651.1997.10487472

Vol. 2, No. 2: 25–30

A Fast Triangle-Triangle Intersection Test

Tomas Möller
Prosolvia Clarus AB

Abstract. This paper presents a method, along with some optimizations, for computing whether or not two triangles intersect. The code, which is shown to be fast, can be used, for example, in collision detection algorithms.

1. Introduction

Most collision detection algorithms, such as OBBTree [Gottschalk 96], sphere hierarchies [Hubbard 96], and BV-trees [Klosowski 97], try to minimize the number of primitive-primitive intersections that have to be computed. Still, a fast and reliable method for computing the primitive-primitive intersection is desired. Since rendering hardware is often targeted for triangles, the primitives in collision detection algorithms are often triangles as well. This paper describes a method for determining if two triangles intersect.

2. Intersection Test Method

Let us denote the two triangles T_1 and T_2; the vertices of T_1 and T_2 by V_0^1, V_1^1, V_2^1, and V_0^2, V_1^2, V_2^2 respectively; and the planes in which the triangles lie by π_1 and π_2.

1086-7651/97 $0.50 per page

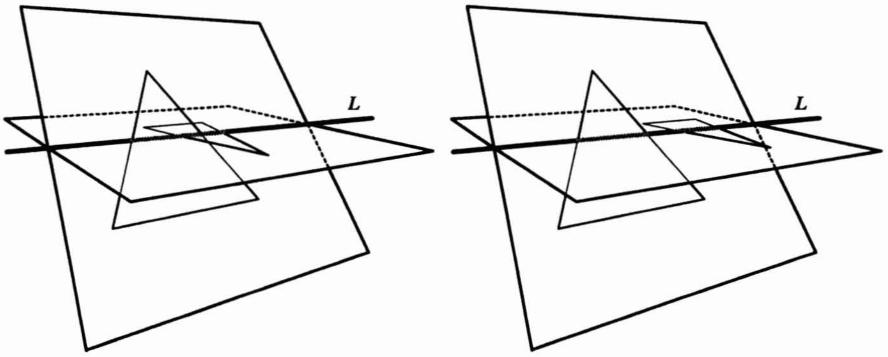

Figure 1. Triangles and the planes in which they lie. Intersection intervals are marked gray in both figures. Left: the intervals along L overlap as well as the triangles. Right: no intersection, the intervals do not overlap.

First, the plane equation $\pi_2 : N_2 \cdot X + d_2 = 0$ (where X is any point on the plane) is computed:

$$\begin{aligned} N_2 &= (V_1^2 - V_0^2) \times (V_2^2 - V_0^2) \\ d_2 &= -N_2 \cdot V_0^2. \end{aligned} \tag{1}$$

Then the signed distances from the vertices of T_1 to π_2 (multiplied by a constant $|N_2|$) are computed by simply inserting the vertices into the plane equation:

$$d_{V_i^1} = N_2 \cdot V_i^1 + d_2, \quad i = 0, 1, 2. \tag{2}$$

Now, if all $d_{V_i^1} \neq 0$, $i = 0, 1$, and 2 (that is, no point is on the plane) and all have the same sign, then T_1 lies on one side of π_2 and the overlap is rejected. The same is done for T_2 and π_1. These two early rejection tests avoid a lot of computations for some triangle pairs. Indeed, for a pair to pass this test there must be some line of direction $N_1 \times N_2$ that meets both.

If all $d_{V_i^1} = 0$, $i = 0, 1$, and 2, then the triangles are co-planar, and this case is handled separately and discussed later. If not, the intersection of π_1 and π_2 is a line, $L = O + tD$, where $D = N_1 \times N_2$ is the direction of the line and O is some point on it. Note that due to our previous calculations and rejections, both triangles are guaranteed to intersect L. These intersections form intervals on L, and if these intervals overlap, the triangles overlap as well. A similar interval test is used in a different context by Laidlaw et al. [Laidlaw 86]. Two situations that can occur are depicted in Figure 1.

Now, assume that we want to compute a scalar interval (on L) that represents the intersection between T_1 and L, and that, for example, V_0^1 and V_2^1 lie on the same side of π_2 and that V_1^1 lies on the other side (if not, you have

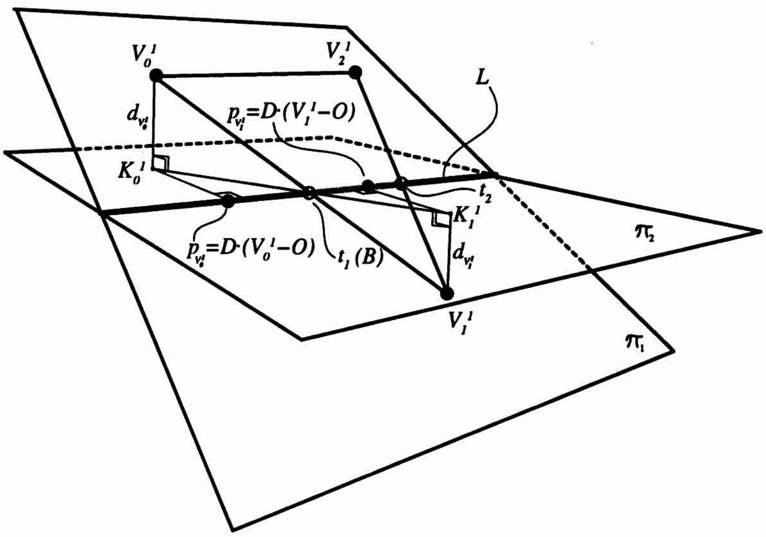

Figure 2. The geometric situation: V_i^1 are the vertices of T_1, π_1 and π_2 are the planes in which T_1 and T_2 lie; $d_{V_i^1}$ are the signed distances from V_i^1 to π_2; K_i^1 are the projections of V_i^1 onto π_2; and $p_{V_i^1}$ are the projections of V_i^1 onto L, which is the line of intersection.

already rejected it). To find scalar values that represent the intersection between the edges $\overline{V_0^1V_1^1}$ and $\overline{V_1^1V_2^1}$ and L, the vertices are first projected onto L:

$$p_{V_i^1} = D \cdot (V_i^1 - O). \tag{3}$$

The geometric situation is shown in Figure 2. Then we want to compute a line parameter value, t_1, for $B = \overline{V_0^1V_1^1} \cap L = O + t_1 D$. Letting K_i^1 denote the projection of V_i^1 onto π_2, we see that $\triangle V_0^1BK_0^1$ and $\triangle V_1^1BK_1^1$ are similar, so

$$t_1 = p_{V_0^1} + (p_{V_1^1} - p_{V_0^1})\frac{d_{V_0^1}}{d_{V_0^1} - d_{V_1^1}}. \tag{4}$$

Similar calculations are done to compute t_2, and an interval for T_2 is computed as well. If these intervals overlap, the triangles intersect.

If the triangles are co-planar, they are projected onto the axis-aligned plane where the areas of the triangles are maximized. Then a simple two-dimensional triangle-triangle overlap test is performed. First, test all closed edges of T_1 for intersection with the edges of T_2. If any intersection is found, then the triangles intersect. Otherwise, we must test if T_1 is totally contained in T_2 or vice versa. This can be done by performing a point-in-triangle test [Haines 94] for one vertex of T_1 against T_2 and vice versa.

2.1. Optimizations

Since the intervals can be translated without altering the result of the interval overlap test, Equation (3) can be simplified into:

$$p_{V_i^1} = D \cdot V_i^1, \quad i = 0, 1, 2. \tag{5}$$

Therefore O does not need to be computed.

Also, the result of the overlap test does not change if we project L onto the coordinate axis with which it is most closely aligned. Therefore Equation (5) can be simplified further:

$$p_{V_i^1} = \begin{cases} V_{ix}^1, & if \ \ |D_x| = \max(|D_x|, |D_y|, |D_z|) \\ V_{iy}^1, & if \ \ |D_y| = \max(|D_x|, |D_y|, |D_z|) \\ V_{iz}^1, & if \ \ |D_z| = \max(|D_x|, |D_y|, |D_z|) \end{cases}, \quad i = 0, 1, 2. \tag{6}$$

Here, V_{0x}^1 represents the x-component of V_0^1 and so on. The same principle was used by Mirtich [Mirtich 96] in order to get a numerically stable simplification of an integral over a polygon's area.

3. Implementation and Performance

To summarize, the steps of the algorithm are as follows (complete C code is available at http://www.acm.org/jgt/papers/Moller97/):

1. Compute the plane equation of Triangle 2.
2. Reject as trivial if all points of Triangle 1 are on the same side.
3. Compute the plane equation of Triangle 1.
4. Reject as trivial if all points of Triangle 2 are on same side.
5. Compute the intersection line and project onto the largest axis.
6. Compute the intervals for each triangle.
7. Intersect the intervals.

Note that after Step 2, there is enough information to immediately test whether the triangles are co-planar, but because this is a rare occurrence, the test is deferred until after several more frequently occurring rejection tests have been performed.

Robustness problems may arise when the triangles are nearly co-planar or when an edge is nearly co-planar to the other triangle (especially when the edge is close to an edge of the other triangle). To handle these cases in a reasonable way the source code provides a constant **EPSILON** (ϵ) which the user defines. As a result, if any $|d_{V_i^k}| < \epsilon$, they are reset so that $d_{V_i^k} = 0$. Geometrically, this means that if a point is "close enough" to the other triangle's plane, it is considered as being on the plane. The source code does not handle degenerate triangles (i.e., lines and points). If it did, then those cases would have to be detected first and then handled as special cases.

Performance was measured for several different scenarios in a collision detection program [RAPID 97]. Both our method and the method from ERIT [Held 97] were found to be faster than the brute-force method,[1] which took between 1.3 and 1.6 times longer to execute. Our method was found to be slightly faster (in RAPID) in most cases when compared to the author's implementation of ERIT's method. We also tested the performance exactly like ERIT and found that our method was approximately 7 percent faster on an SGI Impact. However, on the SGI O2, we found that our method had a slight tendency to be faster for lower collision ratios (i.e. the number of triangle-triangle collisions divided by the number of tests), and the break even point was around 45 percent. Therefore, having worse bounding volumes in a collision detection program implies that our method should perform better than ERIT and vice versa.

Our method is also used in an in-house collision detection package for a commercial VR-platform [Oxygen 97].

Acknowledgements. A special thanks to Stefan Gottschalk (UNC) for many valuable discussions on intersection tests. Constructive criticism and testing on an early version of this work came from Jacob Ström, Per Carleberg, Pär Oscarsson, Patric Fredin, and Ola Liljedahl. Finally, thanks to both John F. Hughes and Ronen Barzel for helping me improve the contents of this paper.

References

[Gottschalk 96] S. Gottschalk, M.C. Lin, and D. Manocha. "OBBTree: A Hierarchical Structure for Rapid Interference Detection." *Computer Graphics (Proc. SIGGRAPH 96)*, pp. 171–180(August 1996).

[Haines 94] Eric Haines. "Point in Polygon Strategies." In *Graphics Gems IV*, edited by Paul S. Heckbert, pp. 24–46. Cambridge, MA: Academic Press Professional, 1994.

[1]Here, each closed edge of each triangle is tested for intersection with the other triangle and if, at any time, an intersection occurs, then the triangles intersect. However, this method does not (easily) handle cases where one or three edges are parallel to the plane of the other triangle.

[Held 97] Martin Held. "ERIT—A Collection of Efficient and Reliable Intersection Tests." Submitted to *journal of graphics tools*, 1997.

[Hubbard 96] Philip M. Hubbard. "Approximating Polyhedra with Spheres for Time-Critical Collision Detection." *ACM Transactions on Graphics*, 15(3):179–210(1996).

[Klosowski 97] James T. Klosowski, Martin Held, Joseph S.B. Mitchell, Henry Sowizral, and Karel Zikan. "Efficient Collision Detection Using Bounding Volume Hierarchies of k-DOPs." Submitted for publication, *IEEE Transactions on Visualization and Computer Graphics*, 1997.

[Laidlaw 86] David H. Laidlaw, W. Benjamin Trumbore, and John F. Hughes. "Constructive Solid Geometry for Polyhedral Objects." *Computer Graphics (Proc. SIGGRAPH 86)*, 20(4):161–168(1986).

[Mirtich 96] Brian Mirtich. "Fast and Accurate Computation of Polyhedral Mass Properties." *journal of graphics tools*, 1(2):31–50(1996).

[Oxygen 97] "Oxygen Base User's Guide 1.0." Gothenberg, Sweden: Prosolvia Clarus AB, 1997.

[RAPID 97] Source code for collision detection by the Research Group on Modeling, Physically-Based Simulation and Applications at University of North Chapel Hill, "RAPID—Robust and Accurate Polygon Interference Detection." Available at http://www.cs.unc.edu/g̃eom/OBB/OBBT.html, 1997.

Web Information:

Complete C source code is available at http://www/acm/org/jgt/papers/Moller97/.

Tomas Möller, Prosolvia Clarus AB, Chalmers University of Technology, Gårdavägen, S-41250, Gothenberg, Sweden (tompa@clarus.se)

Received June 2, 1997; accepted in revised form October 21, 1997