# Fast 3D Triangle-Box Overlap Testing

Tomas Akenine-Möller[*]
Department of Computer Engineering,
Chalmers University of Technology

March 2001, updated June 2001

**Abstract**

A fast routine for testing whether a triangle and a box are overlapping in three dimensions is presented. The test is derived using the separating axis theorem, whereafter the test is simplified and the code is optimized for speed. We show that this approach is 2.3 vs. 3.8 (PC vs. Sun) times faster than previous routines for this. It can be used for faster collision detection and faster voxelization in interactive ray tracers. The code is available online.

## 1 Introduction

Testing whether a triangle overlaps a box is an important routine to have in a graphics programmer's toolbox. For example, the test can be used to voxelize triangle meshes in ray tracers, and it can be used in collision detection algorithms that are based on boxes [3]. Gottschalk et al's collision detection framework only used OBB/OBB tests and triangle-triangle tests. However, it has been noted that both memory and speed can be gained [7] by not having an OBB around each triangle, and instead test a triangle against an OBB.

Previously, Voorhies has presented code for testing a triangle against a unit cube centered at the origin [8]. His test tries to eliminate work by doing some simple acceptance/rejection tests early on, and then testing each triangle edge for intersection with the cube faces. Finally, he checks whether the interior of the triangle is penetrated by the cube. Green and Hatch [4] improve on the efficiency of Voorhies' work and generalize it to handle arbitrary polygons as well. They also use fast acceptance/rejectance tests, but recast the testing of an edge against the cube into testing a point against a skewed rhombic dodecahedron, which is more robust. Finally, they test whether one diagonal of the cube intersect the polygon, which further improves the efficiency.

[*]Previously known as Tomas Möller.

## 2 Derivation and Optimization

Our test is derived from the *separating axis theorem* (SAT) [1, 3, 6]. The theorem states that two convex polyhedra, $A$ and $B$, are disjoint if they can be separated along either an axis parallel to a normal of a face of either $A$ or $B$, or along an axis formed from the cross product of an edge from $A$ with and edge from $B$.

We focus on testing an axis-aligned bounding box (AABB), defined by a center $\mathbf{c}$, and a vector of half lengths, $\mathbf{h}$, against a triangle $\Delta\mathbf{u}_0\mathbf{u}_1\mathbf{u}_2$. To simplify the tests, we first move the triangle so that the box is centered around the origin, i.e., $\mathbf{v}_i = \mathbf{u}_i - \mathbf{c}$, $i \in \{0, 1, 2\}$. To test against an oriented box, we would first rotate the triangle vertices by the inverse box transform, then use the presented test. Based on SAT, we test the following 13 axes:

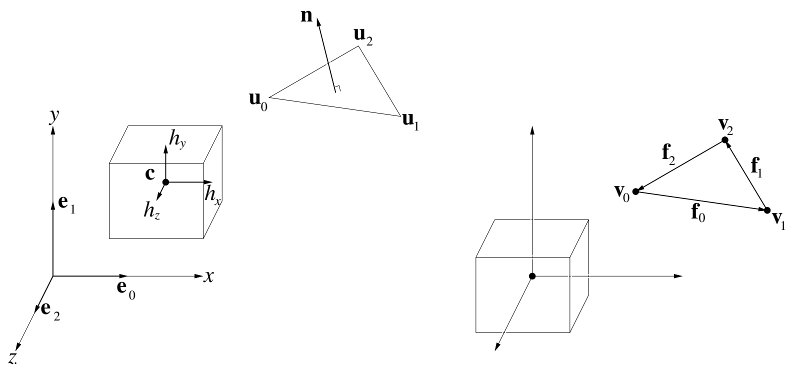

Figure 1: Notation used for the triangle-box overlap test. To the left the inital position of the box and the triangle is shown, while at the right, the box and the triangle has been translated so that the box center coincides with the origin.

1. [3 tests] $\mathbf{e}_0 = (1, 0, 0)$, $\mathbf{e}_1 = (0, 1, 0)$, $\mathbf{e}_2 = (0, 0, 1)$ (the normals of the AABB). Test the AABB against the minimal AABB around the triangle.
2. [1 test] $\mathbf{n}$, the normal of $\Delta$. We use a fast plane/AABB overlap test [5, 6], which only tests the two diagonal vertices, whose direction is most closely aligned to the normal of the triangle.
3. [9 tests] $\mathbf{a}_{ij} = \mathbf{e}_i \times \mathbf{f}_j$, $i, j \in \{0, 1, 2\}$, where $\mathbf{f}_0 = \mathbf{v}_1 - \mathbf{v}_0$, $\mathbf{f}_1 = \mathbf{v}_2 - \mathbf{v}_1$, and $\mathbf{f}_2 = \mathbf{v}_0 - \mathbf{v}_2$. These tests are very similar and we will only show the derivation of the case where $i = 0$ and $j = 0$ (see below).

If all tests pass, i.e., there is no separating axis, then the triangle overlaps the box. Also, as soon as a separating axis is found the the algorithm terminates and returns “no overlap”.

Next, we derive one of the nine tests, where $i = 0$ and $j = 0$, in bullet 3 above. This means that $\mathbf{a}_{00} = \mathbf{e}_0 \times \mathbf{f}_0 = (0, -f_{0z}, f_{0y})$. So, now we need to project the triangle vertices onto $\mathbf{a}_{00}$ (hereafter called $\mathbf{a}$):

$$\begin{aligned} p_0 &= \mathbf{a} \cdot \mathbf{v}_0 = (0, -f_{0z}, f_{0y}) \cdot \mathbf{v}_0 = v_{0z}v_{1y} - v_{0y}v_{1z} \\ p_1 &= \mathbf{a} \cdot \mathbf{v}_1 = (0, -f_{0z}, f_{0y}) \cdot \mathbf{v}_1 = v_{0z}v_{1y} - v_{0y}v_{1z} = p_0 \\ p_2 &= \mathbf{a} \cdot \mathbf{v}_2 = (0, -f_{0z}, f_{0y}) \cdot \mathbf{v}_2 = (v_{1y} - v_{0y})v_{2z} - (v_{1z} - v_{0z})v_{2y} \end{aligned} \tag{1}$$

Normally, we would have had to find $\min(p_0, p_1, p_2)$ and $\max(p_0, p_1, p_2)$, but fortunately $p_0 = p_1$, which simplify the computations a lot. Now we only need to find $\min(p_0, p_2)$ and $\max(p_0, p_2)$, which is significantly faster because conditional statements are expensive on modern CPUs.

After the projection of the triangle onto $\mathbf{a}$, we need to project the box onto $\mathbf{a}$ as well. We compute a "radius", called $r$, of the box projected on $\mathbf{a}$ as

$$r = h_x|a_x| + h_y|a_y| + h_z|a_z| = h_y|a_y| + h_z|a_z| \tag{2}$$

where the last step comes from that $a_x = 0$ for this particular axis. Then this axis test becomes:

$$\texttt{if(}\ \min(p_0, p_2) > r\ \texttt{or}\ \max(p_0, p_2) < -r)\ \texttt{return false;} \tag{3}$$

Now, if all these 13 tests pass, then the triangle overlaps the box.

## 3 Performance Evaluation

To evaluate performance, we used the same test as Voorhies [8], i.e., we randomly select the triangle vertices inside a $4 \times 4 \times 4$ cube centered around the origin and the AABB is the unit cube: from $(-0.5, -0.5, -0.5)$ to $(0.5, 0.5, 0.5)$. To get accurate timings we randomly selected $100,000$ triangles and tested these in a sequence 100 times. We verified that our code generated the same result as Green and Hatch [4], and compared runtimes (we did not test against Voorhies code since that was found to be incorrect [2]).

On a Sun Sparc Ultra 10 at 333 MHz, the presented code was 3.8 times faster on average in this test[1]. On a Linux PC with a 1333 MHz AMD Athlon, the speed up was found to be 2.3[2]. Also, the best order to perform the tests on the Sun was found to be: 3, 1, and finally 2 (the most expensive). On the PC, the order did not matter significantly.

Note that Green and Hatch's code handles the more general case of testing a general polygon against a cube, while we only test a triangle against a cube, and hence we can expect some degradation in performance due to this.

The only place, where there is a robustness issue, is when the normal of the triangle is computed; $\mathbf{n} = \mathbf{f}_0 \times \mathbf{f}_1$. If the triangle has an area close to zero, then the normal calculation is not robust, and our code does not solve that problem. However, in most applications thin long triangles are best avoided.

We have used the code for fast voxelization in a ray tracer, and it has been used in a 3D engine [7].

[1]Our code was compiled using `gcc`, and Green and Hatch's code was compiled using Sun's `cc`, because the runtimes were best for the different routines like that.

[2]Compiled with `gcc -O9 -fomit-frame-pointer -funroll-loops -march=athlon`.

## 4 Acknowledgement

Thanks to Pierre Terdiman for suggesting different ways to optimize the code, and for trying the code in his game engine. Thanks to Peter Rundberg for letting me use his PC for timings.

Code is available at: `http://www.acm.org/jgt/AkenineMoller01/`

## References

[1] Eberly, David, *3D Game Engine Design: A Practical Approach to Real-Time Computer Graphics*, Morgan Kaufmann Publishers Inc., San Francisco, 2000. `http://www.magic-software.com/`

[2] *Graphics Gems III Errata Listing*, `http://www.graphicsgems.org/`

[3] Gottschalk, S., M.C. Lin, and D. Manocha, "OBBTree: A Hierarchical Structure for Rapid Interference Detection," *Computer Graphics (SIGGRAPH '96 Proceedings)*, pp. 171–180, August, 1996. `http://www.cs.unc.edu/~geom/OBB/OBBT.html`

[4] Green, D. and D. Hatch, "Fast Polygon-Cube Intersection Testing," in Alan Paeth, ed., *Graphics Gems V*, AP Professional, Boston, pp. 375–379, 1995. `http://www.graphicsgems.org/`

[5] Haines, Eric, and John Wallace, "Shaft Culling for Efficient Ray-Traced Radiosity," in P. Brunet and F.W. Jansen, eds., *Photorealistic Rendering in Computer Graphics (Proceedings of the Second Eurographics Workshop on Rendering)*, Springer-Verlag, New York, pp. 122–138, 1994. `http://www.acm.org/tog/editors/erich/`

[6] Möller, Tomas, and Eric Haines, *Real-Time Rendering*, AK Peters Ltd., Natick, MA, 1999. `http://www.realtimerendering.com/`

[7] Terdiman, Pierre, Personal communication, 2001.

[8] Voorhies, Douglas, "Triangle-Cube Intersection," in David Kirk, ed., *Graphics Gems III*, AP Professional, Boston, pp. 236–239, 1992. `http://www.graphicsgems.org/`