ALGORITHMS TO TEST RAY-TRIANGLE INTERSECTION. COMPARATIVE STUDY

Rafael J. Segura[1], **Francisco R. Feito**

Departamento de Informática
Universidad de Jaén
Escuela Politécnica Superior
Avda. Madrid, 35, 23071, Jaén
Spain
{rsegura,ffeito@ujaen.es}

ABSTRACT

In this article we present an algorithm to determine the intersection between rays and triangles based on the idea of the study of signs with respect to triangles. One of the advantages of this approach is its robustness due to its lack of trigonometric operations or complex divisions which might alter the result of the calculations. The algorithm is similar (or even better) in time to other existing algorithms, but it is based exclusively on the study of signs, so that the results obtained are more precise. A comparative study of times between the algorithm and other similar algorithms is presented.

Keywords: algorithm complexity, computer graphics, triangle-meshes, optimal algorithm, geometric algorithms.

1. INTRODUCTION

The problem of the intersection of a ray (segment or line) with a triangle is one of the classical problems in the field of Computer Graphics. The calculation for such intersection is fundamental for many problems: ray casting, ray-tracing, detection of collisions between objects, inclusion test, operation between solids, etc.., especially having in mind that most methods used today work on triangulated polygons. Several authors [Agrawal94] have warned of some problems presented in the field of Computational Geometry and Geometric Modellling: first of all, numerical errors when determining real values, and secondly, logical errors when these real values are compared. In this article we propose a robust and efficient algorithm of detection of intersections, which uses exclusively the study of sign of triangles, being these operations in the which no precision errors are involved.

Different solutions exist for the study of intersection between rays and triangles, although the most important ones, thanks to their simplicity and efficiency, are the algorithms proposed by Snyder [Snyder87], Badouel [Badouel90] and Moller [Moller97].

These algorithms are focused especially on the solution of visualization problems, so that the segment is established in its parametric form. The algorithm that we propose does not use this notion due to the fact that it is basically focused on the calculation of operations between objects; for this reason, the segment is given in the shape *QQ'*, instead of a point *Q* and a direction *R*.

The algorithm proposed by Snyder [Snyder87] is worse (in time) than Badouel's; so we will concentrate exclusively on the study of Badouel and Moller's algorithms, to compare the new solution with them

In the first section, we present the algorithm proposed by Badouel and Moller. In the next one we present a summary of the theoretical fundamentals of

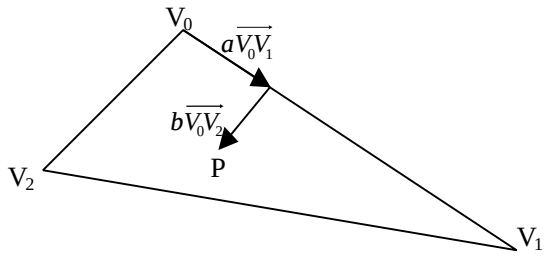
[1] Author for correspondence

the new algorithm [Segura98]. Then, we will present the algorithm and finally we will make a comparative study of times of the three algorithms.

2. BADOUEL'S ALGORITHM

Badouel's algorithm is based on the study of barycentrics coordinates, following the line of Snyder's algorithm. Let $V_0V_1V_2$ be a triangle. The position of point P inside the triangle can be expressed as shown in fig.1, or either in equation 1.

$$\overrightarrow{V_0P} = a\overrightarrow{V_0V_1} + b\overrightarrow{V_0V_2} \tag{1}$$



Barycentrics coordinates of point P
Figure 1.

It is said that point P is inside the triangle if it is true that

$$a \geq 0, \quad b \geq 0, \quad a + b \leq 1$$

Equation 1 is decomposed in a system of three equations, as shown in equation 2.

$$\begin{cases} x_p - x_0 = a(x_1 - x_0) + b(x_2 - x_0) \\ y_p - y_0 = a(y_1 - y_0) + b(y_2 - y_0) \\ z_p - z_0 = a(z_1 - z_0) + b(z_2 - z_0) \end{cases} \tag{2}$$

Then, the triangle is projected into one of the planes xy, xz, or yz, finding for it the largest value of coefficients in the equation of the plane on which the triangle is to be found. Let i_o be the coordinate which is to be eliminated when the triangle is projected, and i_1 and i_2 be the other coordinates. Then we define

$$u_0 = P_{i_1} - V_{0_{i_1}} \quad u_1 = V_{1_{i_1}} - V_{0_{i_1}} \quad u_2 = V_{2_{i_1}} - V_{0_{i_1}}$$
$$v_0 = P_{i_2} - V_{0_{i_2}} \quad v_1 = V_{1_{i_2}} - V_{0_{i_2}} \quad v_2 = V_{1_{i_2}} - V_{0_{i_2}}$$

Substituting in equation (2) we obtain

$$\begin{cases} u_0 = a \cdot u_1 + b \cdot u_2 \\ v_0 = a \cdot v_1 + b \cdot v_2 \end{cases} \tag{3}$$

For the implementation of Badouel's algorithm the code written in C language shown in [Badouel90] has been used. So that the execution time of this algorithm is not penalized, it has not been taken into account when recording the times of precomputing the normal vector of the plane on which the triangle lies.

3. MOLLER'S ALGORITHM

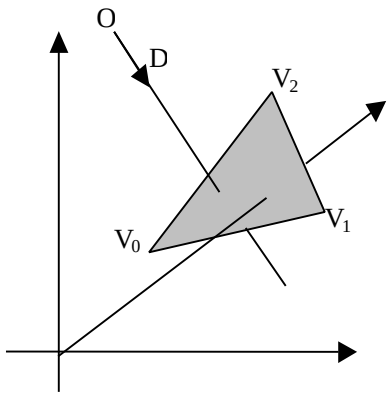
The modifications of Moller's algorithm [Moller97] compared to Badouel's are specially concentrated on creating a series of transformations to the triangle. The objective is that two of the edges will be aligned with the axis of coordinates (see fig.2), being M=[-D V_1-V_0 V_2-V_0], and *u*, *v* being the barycentric coordinates of the point (as we saw in the previous section), and *t* the parameter of the ray equation. In this way, the calculations are notably simplified, resulting in equation 4.

$$\begin{bmatrix} t \\ u \\ v \end{bmatrix} = \frac{1}{(D \times E_2) \cdot D_1} \begin{bmatrix} (T \times E_1) \cdot E_2 \\ (D \times E_2) \cdot T \\ (T \times E_1) \cdot D \end{bmatrix} = \frac{1}{P \cdot E_1} \begin{bmatrix} Q \cdot E \\ P \cdot T \\ Q \cdot D \end{bmatrix} \tag{4}$$

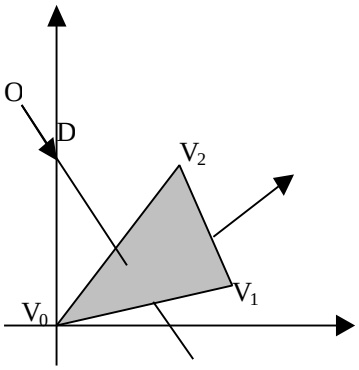
being $E_1=V_1-V_0$, $E_2=V_2-V_0$, $T=O-V_0$, $P=(D\times E_2)$, $Q=(T\times E_1)$

For the implementation of Moller's algorithm, the code written in C language which appears in reference [Moller97] has been used.

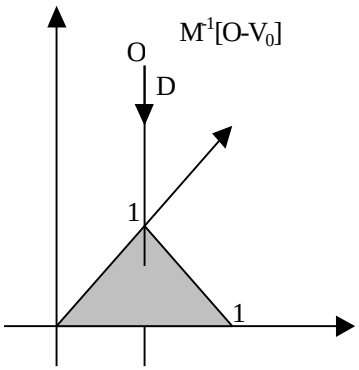
The advantages presented by Moller's algorithm compared to Badouel's are particularly focused on the reduction of the storage of times and spaces, as it is not necessary to store the normal vector of the plane on which the triangle is. The time of this algorithm is better than Badouel's, but for solids with many triangles the time of both algorithms is similar.



a)



b)



c)

Transformations and base change of ray in Moller's algorithm: a) Initial position; b) Translation to origin; c) Rotation and scaling to determine the barycentric coordinate.
Figure 2.

4. THEORETICAL FUNDAMENTALS.

Definition 1. [Orour94]
Let A, B, C and D be four points in R^3. The signed volume of the tetrahedron of vertices D, A, B, and C, denoted by [DABC], is defined as follows:

$$[DABC] = \frac{1}{6} * \begin{vmatrix} x_a - x_d & y_a - y_d & z_a - z_d \\ x_b - x_d & y_b - y_d & z_b - z_d \\ x_c - x_d & y_c - y_d & z_c - z_d \end{vmatrix} = \frac{1}{6} * \begin{vmatrix} x_a & y_a & z_a & 1 \\ x_b & y_b & z_b & 1 \\ x_c & y_c & z_c & 1 \\ x_d & y_d & z_d & 1 \end{vmatrix} \quad (5)$$

being D= (x_d, y_d, z_d), A=(x_a, y_a, z_a), B=(x_b, y_b, z_b) and C=(x_c, y_c, z_c). It is easy to demonstrate that the tetrahedron has a positive orientation (that is, the rest of vertices are seen anticlockwise from the opposite side of a point) if the signed volume is positive.

The definition of the signed volume allows us to determine easily whether a segment cuts or not a triangle in space. Previously it is assumed that the extremes of the segment are in opposite sides of the plane determined by the vertices of the triangles. In order to do it, it is only necessary that the points have signed distance to the plane of different sign. Only in that case it is possible that an intersection between the segment and the polygon exists. In the case that both points are coplanar with the polygon (thta is, the signed distance to the plane is 0), the problem will be reduced to the 2D case.

Lemma. [Segura98]
Let ABC be a triangle in R^3 and Q and Q' be two points that determine a segment in R^3, placed on different sides of the plane determined by ABC, and ordered so that the tetrahedron QABC has positive orientation (i.e., the signed volume of QABC is positive). Then segment QQ' cuts triangle ABC if and only if:

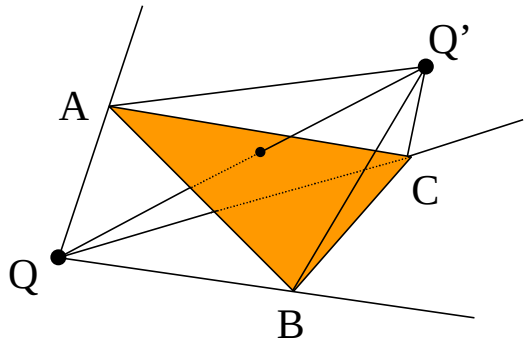
$$sign([Q'ABQ]) \geq 0 \wedge sign([Q'CBQ]) \geq 0 \wedge sign([Q'ACQ]) \geq 0$$

where

$$sign(x) = \begin{cases} 1 & if \quad x > 0 \\ 0 & if \quad x = 0 \\ -1 & if \quad x < 0 \end{cases}$$

Proof.
Segment QQ' will cut triangle ABC if and only if Q' is contained in the trihedral of edges QA, QB, and QC. This is equivalent to say that Q' sees triangles ABQ, BCQ, and CAQ clockwise; this is equivalent to say that tetrahedra Q'AQB, Q'CBQ and Q'ACQ have positive orientation and so their volumes have positive sign (see figure 3). If one of the signs is zero it means that Q' is coplanar with the respective triangle, which indicates that QQ' will cut the

triangle in the respective edges. In the case that two of the signs are zero, it will mean that the intersection between segment QQ' and triangle ABC will take place in one of the vertices of that triangle.



Study of point Q' with respect to the trihedral QABC
Figure 3.

5. ALGORITHM TO TEST SEGMENT-TRIANGLE INTERSECTION IN 3D.

Taking into account the previous definitions, an algorithm to detect when an intersection between a segment defined by points Q and Q' and a triangle T=ABC takes place, can be formulated. The algorithm is shown in figure 4. The focus followed to obtain the algorithm is based on solid modelling by simplicial coverings proposed by Feito [Feito97,Feito98]. This scheme of representation presents many advantages compared to other focuses, though the most outstanding is its simplicity.

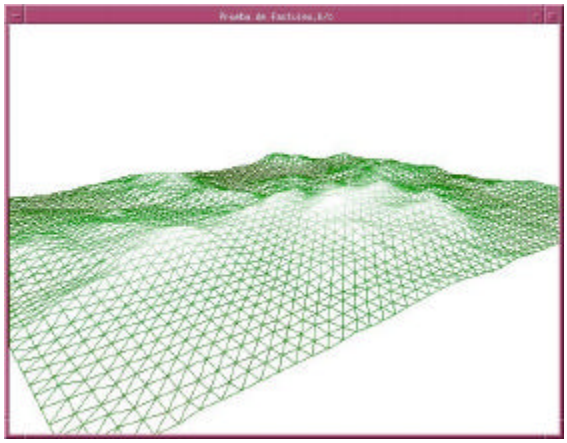
```
int testIntersectSegment(point Q,point Q'){
int i=sign3D(Q,Q',A,C);
int j=sign3D(Q,Q',B,C);
int k=sign3D(Q,Q',A,B);
if (((i==0)&&(j==0))||// Intersects in C
    ((i==0)&&(k==0))||// Intersects in A
    ((j==0)&&(k==0))) // Intersects in B
      return VERTEX;
if ((i==0) && (i==k)) // Intersects in AC
      return EDGE_AC;
if ((j==0) && (i==k)) // Intersects in BC
      return EDGE_BC;
if ((k==0) && (i==j)) // Intersects in AB
      return EDGE_AB;
if ((i==j)&&(j==k))   // Intersects inside
      return IN;
return OUT;           // Does not intersect
}
```

Algorithm to test intersection segment-triangle. Function *sign3D* is defined as the signed volume of the tetrahedron formed by the four points.
Figure 4.

As can be seen, the algorithm proposed only detects whether an intersection between the ray and the triangle exists. In the case that the value of the point of intersection wants to be obtained, it would be necessary to calculate it by the calculation of intersection between the ray and the plane which contains the triangle.

6. COMPARING WITH OTHER METHODS OF CALCULATION OF INTERSECTION.

The previous algorithm has been implemented using C language. Once implemented, a study of the times obtained by the algorithm has been carried out, and it has been compared with implementations of the algorithms proposed by Moller and Badouel, obtaining the results which can be seen in table 1 and in figure 6. The results have been carried out over a terrain generated by [Conde97], over a mesh of more than 500,000 triangles (fig.5). Due to the difficulty of the mesh, we can consider that in that the amount of triangles coincides equally with the best and worst cases of the three studied algorithms. Besides, we have generated 5,000 random rays and we have done the trials with them (every random ray has been tested with every triangle of the mesh using the three algortihms). The C compiler has been used, with all options of optimization. The trials have been carried out in a Silicon Graphics Indy, with 32 MB and using Irix 5.3 as operating system. The time is measured in seconds, but as the time is very little, we have done every test 100 times.



Triangles-mesh used for the study of time of the algorithms
Figure 5.

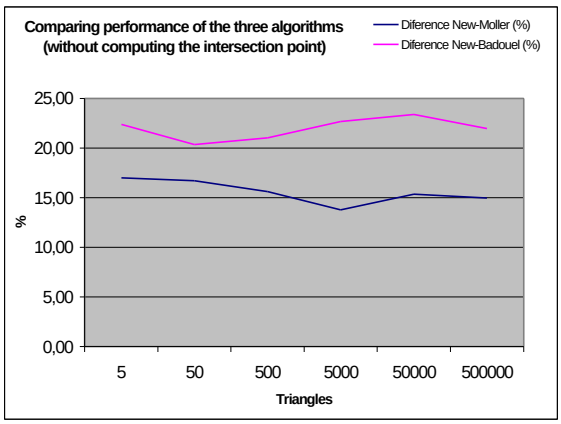
For the calculation of function *sign3D* the method proposed by Yamaguchi [Yamagu90] has been used, which reduces the number of operations needed to resolve a 4x4 determinant.

The comparative study has been done excluding for every algorithm tested the time necessary to obtain the input of the algorithm. So, the cost to obtain the plane of the triangle in Badouel's algorithm has not been considered. Also, the time needed to obtain the parameter of the ray (origin and direction vector in Moller's and Badouel's algoritthm, or origin and end in the proposed algorithm) is not considered.

From table 1, it can be seen that the algorithm is better than Moller's, and also better than Badouel's. In the second column, the number of successful intersection is shown. In the last column of the table, differences between the new algorithm and Moller's is shown: the new algorithm is 15% better than Moller's, and also is better than Badouel's method.

Triangles	Inters.	New	Moller	Badouel	Diference (%)
5	1	0,0079	0,0095	0,0102	16,99
50	3410	0,0832	0,0998	0,1044	16,70
500	42922	0,8135	0,9639	1,0300	15,60
5000	499085	7,9306	9,1963	10,2528	13,76
50000	6324496	78,3664	92,5751	102,2474	15,35
500000	66707323	832,2615	978,7353	1066,4127	14,97

Time in seconds of the three algorithms
Table 1.



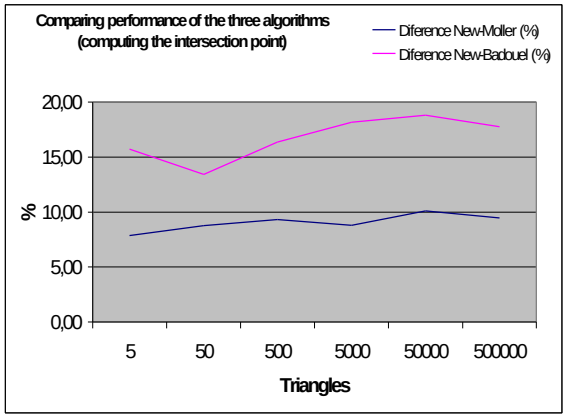
Study of time of the three algorithms.
Figure 6.

In the results obtained in table 1 it is not considered the time necessary to obtain the intersection point in the new algorithm. If we consider this time (only when intersection exists), then the time obtained is slighty better than Moller's (about 8% better) as you can see in table 2. So, the difference is shorter but still better for the new algorithm.

Despite that, the new algorithm presents some advantages: we make no use of divisions or other operations in which precision errors play an important part. In fact, the algorithm is based on the study of the sign of the values obtained, so, operations on bit level could be used for their study. Another advantage is that it is only necessary to calculate the point of intersection between the segment and the triangle in those cases in which this intersection exists, so that, again, we are eliminating possible errors in the calculations of these intersections due to precision errors.

Triangles	Inters.	New	Moller	Badouel	Diference (%)
5	1	0,0082	0,0089	0,0133	7,87
50	3658	0,0915	0,1003	0,1057	8,76
500	43657	0,8657	0,9548	1,0349	9,33
5000	501369	8,3942	9,2040	10,2567	8,80
50000	6174496	83,1855	92,5242	102,4570	10,09
500000	676857418	880,6022	972,7677	1070,8090	9,47

Time in seconds of the three algorithms, considering the cost of computint the intersection point.
Table 2.



Study of time of the three algorithms considering the cost of computing the intersection point
Figure 7.

In inclusion test problem or similar ones, it is not necessary to compute the intersection point between the segment ant the triangle: it is only necessary to know whether intersection exists or not. So, we think that the new algorithm is better to be used in that kind of problems. In [Feito00], the new algorithm has been used to solve the problem of inclusion-test in triangles meshes using the ray-casting solution, i.e., counting the number of intersections. The time obtained with the new algorithm is better than Moller's algorithm, because the new algorithm returns whether intersection exists or not, and in the case that intersection exists, the

algorithm returns where it is (inside, an edge or a vertex). And so, the study of special cases (necessary to make if we use Moller's algorithm) is reduced.

Obviously, for dealing with large triangular meshes we need to use some techniques to reduce the number of intersection, for example hierarchical testing (BSP, octrees, ...) or others. But in this paper we only consider the problem of test the intersection. This solution is valid also when some of those techniques is applied. And the solution could be improved if we use triangles strip because we can use some results from one triangle to the following one. For it, we can notice that the determinants for one of the edge shared for two (or more) triangles does not change and we can conserve this result when the following triangle of the strip is studied.

7. CONCLUSIONS

We have just presented an algorithm to determine when the intersection between the rays and triangles takes place. The validity of the algorithm has been demonstrated, and a comparative study of the obtained times with other algorithms has been carried out, resulting to be better than Moller and Badouel's algorithm.

The algorithm is more robust than the other two algorithms mentioned in the article due to the absence of complex operations. Also, the speed of the algorithm could be increased resolving the operations of sign on bit level of the processor.

The main problem of the solution proposed is that the intersection point is not computed: the algorithm computes only whether intersection exists or not. So, we must compute the point of intersection if we need it. In order to solve the ray-casting problem or similar ones, the new algorithm will be better if the number of intersection is low. But for problems like inclusion test or similar ones, it is only necessary to know whether intersection exists or not, because we only need to count the number of intersections. And so, the algorithm proposed is better than other algorithms because if intersection exists, we know where the point of intersection is, and so, the treatment of special cases (intersection in edges or vertices) improves.

The authors wish to thank to the anonymous reviewers for their constructive comments which led to several improve ments in the final version of this paper.

REFERENCES

[Agrawal94] Agrawal, A., Requicha, A.: A paradigm for the robust design of algorithms for geometric modelling, *Eurographics'94, Computer Graphics Forum*, Vol.13, No.3, pp.33-44, 1994.

[Snyder87] Snyder, M., Barr, A.H.: Raytracing complex models containing surface tesselations, *Proceedings of the 14th annual conference on Computer Graphics*, 1987, Vol.21, No.4, pp.119-128, 1987.

[Badouel90] Badouel, F.: An efficient Ray-Polygon intersection, *Graphic Gems*, *Academic Press*, pp:390-393, 1990.

[Moller97] Moller, T., Trumbore, B.: Fast, minimun storage ray-triangle intersection, *Journal on Graphic Tools*, Vol.2, No.1, pp.21-28, 1997.

[ORour94] O'Rourque, J.: Computational Geometry in C, *Cambridge University Press*, 1994.

[Segura98] Segura, R.J., Feito, F.R.: An algorithm for determining intersection segment-polygon in 3D, *Computer & Graphics*, Vol.22, No.5, pp.587-592, 1998.

[Feito97] Feito, F.R., Torres, J.C., Boundary representation of polyhedral heterogeneus solids in the context of a graphic object algebra, *The Visual Computer*, No. 13, 1997.

[Feito98] Feito, F.R., Segura, R.J., Torres, J.C., Representing polyhedral solids by simplicial coverings, *CSG'98, Set-Theoretic Solid Modelling. Techniques and applications*, Informations Geometers Ltd., UK., pp:203-219, 1998.

[Yamagu90] Yamaguchi, F., Niizeki, M., Fukunaga, H., Two robust point-in-polygon tests based on the 4×4 determinant method. *Advanced on degign automation (ASME)*, Vol. 23, No. 1, pp:89-95, 1990.

[Conde97] Conde, F.A., Feito, F.R., Montejo, A., Rojas, J., Visualización realista del terreno a partir de datos de elevación digitales, fotos de satélite y datos vectoriales, *Mapping, Spain*, No. 36, 1997.

[Feito00] Feito, F.R., Ruiz, J., Segura, R.J., Torres, J.C., Point membership clsssification respect triangle meshes: an aptimized algorithm, *Technical Report, Departamento de Informática, Universidad de Jaén, Spain*, 2000.