Collision between a segment and a triangle

Benjamin Loison

8 march 2018

Contents

1 Opening speech

The modus operandi (way of proceeding) is composed of four steps (if one fails, we can directly declare that there isn't any collision between the segment and the triangle):

1. Check whether or not the straight line which supports the segment intersects the plane which supports the triangle (it is an easy mathematic logical operation). Likewise the direction is checked.
2. Check whether or not the intersection point has a positive scalar factor multiplied by the relative target vector (if we state an endpoint of the segment as the beginning of the target vector and the other endpoint as the end of the target vector). Likewise the way is checked.
3. Check whether or not the intersection point, inside the triangle plane, is inside the triangle. Likewise this checks whether or not the precise intersection point of the line which supports the segment is in the triangle.
4. Check whether or not the distance between the origin (of the target vector) and the intersection point is lower or equal to the distance between the origin and the other endpoint of the segment.

2 Let's dive into mathematics

2.1 Check if there is an intersection point

- The cartesian equation of a line is: $\alpha * x + \beta * y + \gamma * z + \lambda = 0$. Where λ is a constant which depends of α, β and γ.
Where here:
$\begin{cases} \alpha = B_x - A_x \\ \beta = B_y - A_y \\ \gamma = B_z - A_z \end{cases}$
If we consider A and B the two points of the segment.
Likewise: $\lambda = -(\alpha * A_x + \beta * A_y + \gamma * A_z)$.
(If we consider A as the origin point of the segment (the origin can be an arbitrary choice).)
So finally: $\alpha * x + \beta * y + \gamma * z - (\alpha * A_x + \beta * A_y + \gamma * A_z) = 0$
Id est: $\alpha(x - A_x) + \beta(y - A_y) + \gamma(z - A_z) = 0$
Which is equivalent to:

$$(B_x - A_x)(x - A_x) + (B_y - A_y)(y - A_y) + (B_z - A_z)(z - A_z) = 0$$

Ie:

$$\alpha x + \beta y + \gamma z = \alpha A_x + \beta A_y + \gamma A_z$$

Ie:

$$x(B_x - A_x) + y(B_y - A_y) + z(B_z - A_z) = A_x(B_x - A_x) + A_y(B_y - A_y) + A_z(B_z - A_z)$$

- The system of equations of the line is:
$\overrightarrow{AM}$ (with $M(x, y, z)$ a point of the line) and $\overrightarrow{AB}$ are collinears if and only if it exists $k \in \mathbb{Z}$, such as:
$\begin{cases} x - A_x = k * (B_x - A_x) \\ y - A_y = k * (B_y - A_y) \\ z - A_z = k * (B_z - A_z) \end{cases}$

Which involves: $k = \frac{x - A_x}{B_x - A_x}$
- The cartesian equation of the triangle plane is:
$n_x * x + n_y * y + n_z * z + d' = 0$ where $\vec{n}$ is a normal vector to the triangle plane and d' is a constant.
We compute $\vec{n}$ by using the vector product of two vectors of the triangle like $\overrightarrow{CD}$ and $\overrightarrow{CE}$.
$\vec{n} = CD \wedge CE = (CD_y * CE_z - CD_z * CE_y, CE_x * CD_z - CD_x * CE_z, CD_x * CE_y - CE_x * CD_y)$
Then to calculate d' we inject for instance the coordinates of the point C.
$d' = -(n_x * C_x + n_y * C_y + n_z * C_z) = -((CD_y * CE_z - CD_z * CE_y) * C_x + (CE_x * CD_z - CD_x * CE_z) * C_y + (CD_x * CE_y - CE_x * CD_y) * C_z)$
Finally we got:

$$n_x * x + n_y * y + n_z * z - (n_x * C_x + n_y * C_y + n_z * C_z) = 0$$

which is equivalent to:

$$n_x * x + n_y * y + n_z * z = n_x * C_x + n_y * C_y + n_z * C_z$$

Let set z to 1, likewise both equations give:

$$\begin{cases} n_x * x + n_y * y = n_x * C_x + n_y * C_y + n_z * C_z - n_z \\ \alpha x + \beta y = \alpha A_x + \beta A_y + \gamma A_z - \gamma \end{cases}$$

Which is equivalent to:

$$\begin{cases} n_x x + n_y y = n_x C_x + n_y C_y + n_z (C_z - 1) \\ \alpha x + \beta y = \alpha A_x + \beta A_y + \gamma (A_z - 1) \end{cases}$$

Now we use the Cramer's determinant to check whether or not there is a or multiple intersection point:

$\Delta = n_x * \beta - n_y * \alpha$

If Δ is equal to 0, we need to check if the line is in the triangle plane or is parallel to this plane. (easy with coordinates)

Else we got:

$$\begin{cases} x = \frac{(n_x C_x + n_y C_y + n_z (C_z - 1)) * \beta - n_y * (\alpha A_x + \beta A_y + \gamma (A_z - 1))}{\Delta} \\ y = \frac{n_x * (\alpha A_x + \beta A_y + \gamma (A_z - 1)) - \alpha * (n_x C_x + n_y C_y + n_z (C_z - 1))}{\Delta} \end{cases}$$

Listing 1: Implementation in C++ to check (and compute) if there is an intersection point

```
bool isCollisionSegmentTriangle(Vector3D A, Vector3D B, Vector3D C, Vector3D D,Vector3D E)
{
    Vector3D CD = Vector3D(C, D), CE = Vector3D(C, E), n = crossProduct(CD, CE);
    double alpha = B.X - A.X, beta = B.Y - A.Y, gamma = B.Z - A.Z,
           delta = n.X * beta - n.Y * alpha;

    if(delta == 0)
    {
        // check whether or not the triangle plane contains the line or is strictly parallel
        // we check whether or not a point of which defines the line is in the triangle plane
        return n.X * A.X + n.Y * A.Y + n.Z * A.Z == n.X * C.X + n.Y * C.Y + n.Z * C.Z;
    }
    else
    {
        double x = ((n.X * C.X + n.Y * C.Y + n.Z * (C.Z - 1)) * beta
                    - n.Y * (alpha * A.X + beta * A.Y + gamma * (A.Z - 1)) / delta,
               y = (n.X * (alpha * A.X + beta * A.y + gamma * (A.Z - 1))
                    - alpha * (n.X * C.X + n.Y * C.Y + n.Z * (C.Z - 1))) / delta;
        // TODO: to continue
    }
}
```

2.2 Check if the intersection point is in the right way

We need to check if k is superior or equal to 0. (the formula to compute k was given above)

Listing 2: Implementation to check if the intersection point is in the right way

```
double k = (x - A.X) / (B.X - A.X);
if(k < 0) return false;
// TODO: to continue...
```

2.3 For a triangle

I leave it to the reader to complete by doing a simple test if the intersection point is in the triangle (in the triangle plane) by using a triple verification if the point is on the left if we run all the triangle around. Or by using a simple test like decomposing the coordinates of the intersection point with two vectors of the triangle (which define the triangle plane with the coordinates of a point of the triangle). Likewise there is just a test to perform on the components of the new system of coordinates of the intersection point checking if positive and with a linearization to test if it is in the triangle.

2.4 For a rectangle

It is important to deal with a rectangle and not a quadrilateral because we assume that two "random" vectors (of the segments) of the rectangle are perpendicular.

1. By using twice the algorithm defined above for the triangle (reminder: use the four points in the right order)
2. By decomposing like for the triangle modus operandi and checking if the components of the new syste of coordinates are between (with equal meaning) 0 and 1.