Analysis of Eberly's Method of Separating Axes Algorithm

Peter Gels, 1536478

18-06-2009

Abstract

In his paper called "'Intersection of Convex Objects: The Method of Separating Axes"', David Eberly [1] proposes a method of separating axes for convex objects. This article examines this for convex polygons in 2D.

1 Separating Axes

The Method of Separating Axes works like this: suppose you have two convex objects c_0 and c_1, which are both collections of points and lines between those points. If you are able to find an axes, so that c_0 in its entirety lies on one side of the axes, and c_1 in its entirety lies on the other side of the axes, then the two objects don't collide. If there is not such an axis, then the two objects do collide. Try to imagine this by having a regular 2D-axis, in which c_0 consists out of points where the horizontal and vertical values are both positive, and the points of c_1 all have positive vertical values, but negative horizontal ones.

Let us first consider the simple case in which the algorithm is applied to two convex polygons in a 2D-environment. In order to implement this, we need to separate three cases in which the two objects could have collided with each other: you can have two of the edges exactly touching each other, you can have one point touching an edge (which is most likely going to happen) and there is the case in which two points of the opposing objects touch each other.

For the implementation, we need two functions. The first, $Perp(x, y)$, says that given an edge (x, y) (with x and y both vertexes), it will return its outward pointing normal. The second, $WhichSide(S, D, P)$ takes as input a set of points S, the direction of an axis D and the place of an axis P. It will return a value less than zero if S is entirely on the left side of D, it will return a value greater than zero if S is entirely on the right side of D and otherwise it will return a 0, meaning that D intersects the polygon in S.

The pseudo-code for the algorithm is as follows:

```
bool TestIntersection2D (ConvexPolygon C0, ConvexPolygon C1) {
    // Test edges of C0 for separation. Only try to determine
    // if C1 is on the 'positive' side of the line.
    for (i0 = 0, i1 = C0.N-1; i0 < C0.N; i1 = i0, i0++) {
        D = Perp(C0.V(i0) - C0.V(i1));
        if ( WhichSide(C1.V,D,C0.V(i0)) > 0 ) {
            // C1 is entirely on 'positive' side of the line
            return false;
        }
    }

    // Test edges of C1 for separation. Only try to determine
    // if C0 is on the 'positive' side of the line.
    for (i0 = 0, i1 = C1.N-1; i0 < C1.N; i1 = i0, i0++){
        D = Perp(C1.V(i0) - C1.V(i1));
        if ( WhichSide(C0.V,D,C1.V(i0)) > 0 ) {
            // C0 is entirely on 'positive' side of the line
            return false;
        }
    }

    return true;
}
```

What happens in the above piece of code is the following: you have two convex polygons C_0 and C_1, in which all of the edges are sorted counter-clockwise, and you want to determine whether or not the two collide with each other. Then for C_0, you check for every edge of C_0 whether or not the axes that is formed by constructing the normal of the vector between that edge and the opposing edge of C_1 with one of the points on the edge as base separates all of the vertices of C_1 on one side of this axes. If this is the case, then false is returned, indicating that the polygons are not intersecting. The same process is repeated for C_1.

One reason why this algorithm is efficient is that you can simply quit once you've found an axes that separates both the polygons, since the algorithm says that if you can find such an axis, then the two polygons in question don't intersect. This can often save a lot of time when the polygons aren't intersecting, which is usually the case.

References

[1] David Eberly. Method of separating axes. *www.geometrictools.com/Documentation/MethodOfSeparatingAxes.pdf*, 2001.