How to calculate the minimum distance between two polygons

> ***Question 1***
>
> Do you know how to calculate the minimum distance between two polygons if they don't overlap?

Well, this is rather complex. I start with the basics (don't know if you already know). I use UPPER letters for vectors and lower letters for numbers. Well, and sorry that I repeat some things, I wrote the text and realized I forgot something you have to know for a later step, so I went back and added it, I didn't write this text line by line. If something isn't clear to you, I recommend to make some small drawings, I had to do many, too.

A plane is defined by `(X - P) * N = 0` where *P* is a vector to any point in your plane and *N* is the normal vector of your plane. Sometimes another definition is used, which is easier to gain if you have the corners of a polygon: `X = P + a * A + b * B` (*a*, *b* are any real numbers). If you know 3 points *X1*, *X2*, *X3* of the plane (3 corners of the polygon), you can get *A* and *B* by `A = X2 - X1` (subtract each component of the vector from the same component of the other vector) and `B = X3 - X1` (and you can use `P = X1`).

Unfortunately this definition is not good to calculate distances, so you have to get *N* out of *A* and *B*. `A * N` must be `0` and `B * N` must be `0` (which means `a1 * n1 + a2 * n2 + a3 * n3 = 0` and `b1 * n1 + b2 * n2 + b3 * n3 = 0`). Sorry I cannot remember how to do this, but you have 2 equations with three unknown variables, so you can choose one of them as you want (just be careful with `0` and not `0`), the only difference is that the resulting *N* differs in its length.

A line is defined by `X = P + v * V` where *P* is any one point of your line and *V* is the line's direction (like *A* and *B* of the plane). Again if you know two points *X1* and *X2* of your line you get *V* by `V = X2 - X1` (and you can use `P = X1`).

The length of a vector `V = (v1, v2, v3)` is `length = sqrt(sqr(v1) + sqr(v2) + sqr(v3))` (just 3-dimensional Pythagoras).

You add two vectors `A = (a1, a2, a3)` and `B = (b1, b2, b3)` like that: `A + B = (a1 + b1, a2 + b2, a3 + b3)` (which is a vector again).

You multiply two vectors `A = (a1, a2, a3)` and `B = (b1, b2, b3)` like that: `A * B = (a1 * b1 + a2 *b2 + a3 * b3)` (which is a number).

Use following formula to get the distance between any one point *X* and a plane: `dist = 1 / n * (X - P) * N` where *X* is a vector to the point you want to examine and *n* is the length of *N* (you don't need `1/n` if *N* has already length of `1`). If *X* is `(x1, x2, x3)` this is

```
dist = 1 / n * ((x1 - p1) * n1 + (x2 - p2) * n2 + (x3 - p3) * n3)
n = sqrt(sqr(n1) + sqr(n2) + sqr(n3))
```

Now the distance between two polygons isn't that simple, because there are many different cases (and a polygon is more than just a simple plane, even if it's size is smaller).

What you also need is to calculate the distance between two lines. At first you need a plane, that is parallel to both lines and includes one of the lines:

```
P(plane) = P(line1)
A(plane) = V(line1)
B(plane) = V(line2)
```

where *A* and *B* are two vectors in the plane `(N * A = 0 and N * B = 0)`, *V* is a vector in your line (for polygons, you can take `V = X2 - X1` where *X2* and *X1* are two corners). Now calculate the distance between this plane and any one point of *line2* using the formula above (any point because ALL points have the same distance to a plane that's parallel to the line - nice trick, isn't it?

The last thing we need is not only the minimum distance between two lines, but the points of the lines, that have minimum distance. You can do this (for the point of *line1 M1* with minimum distance to *line2*) by calculating a plane again with

```
P(secondplane) = P(line2)
A(secondplane) = V(line2)
B(secondplane) = N(plane) <-- the plane we calculated above
```

The second plane includes *line2* and the point of line1 with the minimum distance to *line2*. To get this point of minimum distance, set `P(line1) + v * V(line1) = P(secondplane) + a * A(secondplane) + b * B(secondplane)`. Solve this, you should get the *v* and when you set this *v* into `X = P + v * V` of *line1* you have the point *X* (=*M1*) of minimum distance.

The bad news: To get *M2*, you have to repeat this for *line2*. Another way would be to take the distance between the lines (I call it *d*) and do following: `M2 = M1 + d * 1 / n * N(plane)` (or `M2 = M1 - d * 1 / n * N(plane)`, depends on the direction of *N*). The distance between two points *X1* and *X2* equals the length of the vector `X2 - X1`.

Okay, these were the basics. Now the different cases, you have to cope with:

1) The planes of both polygons are parallel `(N1 = x * N2)`:

Transform both polygons the following way: `X' = X - P` for each corner of the polygon (where *P* is any point in your plane). The new polygons should now be in the same plane. Test whether both polygons overlap (is not as simple as it sounds, to be honest I don't know how to do that).

1a) They overlap:

The minimum distance is the distance of the two planes (take any one point of one plane and use the formula above to get the distance to the other plane).

1b) They do not overlap:

Use 2) to calculate the minimum distance.

2) The planes are not parallel (or case 1b):

Calculate the minimum distance of one line of one polygon and one line of the other polygon. Calculate the points of minimum distance from the lines. The edges of the polygons do not have infinite length (the lines do have), so check whether the points of minimum distance are within the polygons (I'd better say: within the edges of the poygons).

2a) Both points are within the polygons:

Store the minimum distance from the lines.

2b) One point or both points are not within the polygon:

Take the corner(s) of the polygon(s) within the line(s) you checked next to the point(s) of minimum distance. Calculate the distance between these points and store it. Now repeat this for each pair of lines (if you have 2 triangles you get 9 combinations (3 times 3). When you are ready compare all the minimum distances and take the smallest one.

Okay, this is quite much to do (realtime? difficult. perhaps if you don't have many polygons) and there are several problems (a vector `(x1, x2, x3)` and a vector `(x1, x2, 0)` may have to be treated different, for example when you try to get *N* out of *A* and *B*). If you really need the minimum distance, try it, but perhaps you find an easier way, that is not that exact (take the distance between the center of each polygon would be least exact, but very much easier).

I want to add, that I don't know of other solutions, perhaps there are better ones, and that I don't know if everything I told you is right, I haven't tested it, everything is just theoretically.

Original resource:	The Delphi Pool
Author:	Jens Gruschel
Added:	2013-04-09
Last updated:	2013-04-09